2016

Multidimensional Projection Visualization: Control-points Selection and Inverse Projection Exploration

Portes dos Santos Amorim, Elisa

Portes dos Santos Amorim, E. (2016). Multidimensional Projection Visualization: Control-points Selection and Inverse Projection Exploration (Unpublished doctoral thesis). University of Calgary, Calgary, AB. doi:10.11575/PRISM/27026

http://hdl.handle.net/11023/3480

doctoral thesis

UNIVERSITY OF CALGARY

Multidimensional Projection Visualization:

Control-points Selection and Inverse Projection Exploration

by

Elisa Portes dos Santos Amorim

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN COMPUTER SCIENCE

CALGARY, ALBERTA

DECEMBER, 2016

Abstract

The task of interpreting multidimensional data is as important as it is challenging. The importance comes from the fact that virtually every data worth analyzing is multidimensional, while the challenge comes from the very nature of these data sets, as the multiple features describing each instance can quickly overwhelm our visual perception system, thus making it difficult to observe meaningful information. Visualization techniques play an essential role in simplifying this task, by preprocessing the data to extract critical features and displaying them effectively, by using visual metaphors that can be easily understood. Multidimensional Projection (MP) is one of such techniques, whose fundamental goal is to present an overview of the data distribution in the form of a 2D scatterplot graph. It does so by reducing the dimensionality of the dataset in such a way that distances are preserved as much as possible. MP approaches, along with most visualizations, are shifting from a static display to a more interactive one, allowing human intervention to modify the layout and facilitate exploration and understanding of the data. In this thesis, I present contributions that specifically relate to interactive aspects of multidimensional projection. First, I propose a computational framework and methodology for control points selection. Control points are a particular set of projected points used to steer and rearrange the projection layout. I demonstrate the proposed method can improve the projection quality while requiring only a small amount of control points. Second, I introduce inverse projection, a novel paradigm to create multidimensional points exclusively through 2D interactions. The projection space is transformed into a canvas, where new points can be added. These new points are then mapped into the original multidimensional space, i.e., they become unique multidimensional instances themselves. Lastly, I present the usability of the inverse projection framework in two demonstration examples. (1) A parameter exploration prototype system for optimization with multiple minima. (2) A face-synthesis application, where new face models are generated on the fly.

Acknowledgements

What a journey this has been! Fortunately, I had much-needed support from life-long friends, and others I made along the way, whom I would like to thank with all my heart. My dear husband Ronan, without him I wouldn't even have started this journey. We have been through Bachelors, Masters and PhD together. So come what may, I can face it with him! I am so glad for his unconditional love and support. My beloved mother, father, brothers, sister, and extended family, who never doubted I could achieve all my wildest dreams and ambitions. My dear Prof. Rodrigo Weber dos Santos, who so passionately mentored me through my early research years, and had a great impact in my life. All the professors and friends I made throughout my academic years, from early to higher education, who helped shape my character and personality. My supervisor, Prof. Mario Costa Sousa, who gave me the great opportunity to pursue this degree in such a wonderful university and country, and his family who welcomed us with open arms. My co-supervisor, Prof. Faramarz Samavati, for the support, guidance and friendship. Emilio Vital Brazil, for always believing in me and making sure to reassure me, whenever I needed. His friendship and collaboration are a few of the many good memories I have of this period of my life. Prof. Luis Gustavo Nonato, who introduced me to multidimensional projection and ignited the idea of inverse projection. All the professors in my committee, for the valuable revisions and suggestions. All my lab mates, who were true friends and supporters, whom I cherish the most. Lastly, to my Heavenly Father, who gave me my life and the strength to press forward.

Table of Contents

List of Tables

List of Figures and Illustrations

List of Symbols, Abbreviations and Nomenclature

Symbol	Definition
U of C	University of Calgary
MP	Multidimensional Projection
PCA	Principal Component Analysis
MDS	Multidimensional Scaling
CP	Control Point
RBF	Radial Basis Function
iLAMP	Inverse Local Affine Multidimensional Projection

Chapter 1

Introduction

Visualization algorithms and methods are fundamental to data analysis and communication of information. The representation of data as images has long proven to be a powerful mean to convey information and reveal patterns otherwise hard to identify. Whether the dataset presents itself in a simple form, such as a table correlating two variables, or in a more complex structure, like a graph, visualization can help us gain insights about the data, leveraging the efficiency of humans' visual perception [Few, 2013].

Multidimensional data is a particular type of data that poses many challenges for interpretation. A dataset is said to be multidimensional when each *instance* (or data sample) is represented by three or more *attributes*. Such a representation of data is ubiquitous and is present in almost all real-world applications. Music and video [Joia et al., 2011], text documents [Chalmers, 1993] and vector fields [Daniels II et al., 2010] are just a few examples of data that have been represented multidimensionaly.

Figure 1.1 illustrates four additional examples. Figure 1.1-(a) The *wordle* [1] representation of this chapter. Text documents can be represented as a multidimensional feature vector, each dimension accounting for the number of times a given word appears [Paulovich and Minghim, 2006]. Figure 1.1-(b) Computer simulation of various phenomena, such as fluid flow models of petroleum reservoirs, use grids for numerical simulations over a discrete space. Each model often contains thousands or even millions of grid blocks, each of which with dozens of simulated properties and parameters [Hajizadeh et al., 2012]. Figure 1.1-(c) 3D objects, such as face models, are often composed of a mesh with many points and a corresponding texture layer, whose values may be arranged vector-wise to represent

[1] Wordle is a word cloud visualization that gives greater prominence to words that appear more frequently in a text document.

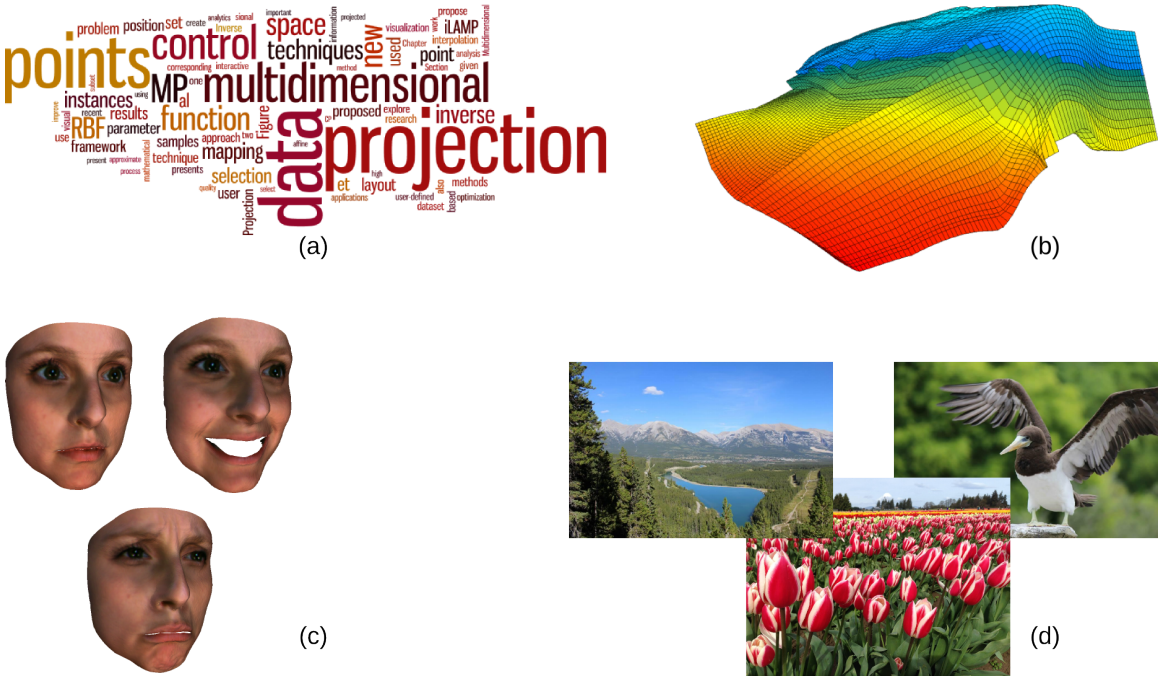


Figure 1.1: Examples of multidimensional data. (a) The wordle representation of this chapter. (b) Computer simulation of natural phenomena. (c) 3D objects, such as face models. (d) multidimensional representation of images

an object as a multidimensional feature vector [Amorim et al., 2015]. Figure 1.1-(d) Images can also have a multidimensional representation, by for example extracting the RGB values and other attributes pixel by pixel like the G-buffer approach [Saito and Takahashi, 1990] or through other techniques, such as *bag-of-visual features* [Yang et al., 2007].

The ever increasing computing power and technological advances, paired up with decreasing of the cost of data storage, have substantially augmented the size, dimension and complexity of data across several domains. This overwhelming data size and dimensions have driven a recent influx of research and development work dedicated to techniques and tools that aid in understanding multidimensional datasets. Typical methods for visualizing multidimensional data include: (1) *non-projective mappings* - e.g., parallel coordinates [Inselberg and Dimsdale, 1990a] and star coordinates [Kandogan, 2000]; (2) *scatterplots* – e.g., primarily scatterplot matrices; and (3) *projective mappings*, such as dimensionality reduction techniques, also known as *multidimensional projection* (MP) in the visualization

community.

The goal of MP methods is to create a 2D representation for the data and display it as a scatterplot graph. In this graph, each point represents one instance of the dataset and, in general, every instance is represented by one point in the graph. The scatterplot, or *projection*, is organized in such a way that the distance between two points reflects, as best as possible, the dissimilarity between the corresponding multidimensional instances. In other words, points close together suggest that the corresponding instances are similar, while points far apart indicate the opposite. Such an arrangement provides an important panorama of the data structure, enabling to reveal potential clusters, patterns or outliers.

While there are multiple techniques designed to compute the projection of a multidimensional dataset, few of them provide flexible mechanisms to visualize and interactively explore the data simultaneously. In this thesis, we focus on the *visual analytics*[2] aspects of multidimensional datasets through MP. In particular, I propose the inverse projection workflow, which transforms the 2D projection space into an interactive medium where new 2D points can be created. These points are then mapped to the original dimensionality of the dataset. Inverse projection allows creating new multidimensional data in a 2D environment, entirely removing the complexity of dealing with multiple dimensions. Furthermore, I propose a technique for the selection of control points, which are unique instances used as a "steering" mechanism to rearrange the projection layout.

In the next Section, I present the objectives of this research. Section 1.2 introduces the methodology adopted in the proposed techniques. A description of the main contributions of this thesis is given in Section 1.3. Finally, Section 1.4 presents a roadmap for the organization of this thesis.

[2]Visual analytics is an outgrowth of the fields of information and scientific visualization, focusing on analytical reasoning through interactive visual interfaces [Wong and Thomas, 2004, Sun et al., 2013].

1.1 Objectives

The primary goal of this work is to develop and improve interactive aspects of MP by providing a means to explore and interact with the projection layout more fully. While a 2D static view of a multidimensional data is capable of providing insights on the underlying data structure, so much more can be accomplished when the visualization medium empowers the user with the ability to manipulate and actively explore the data.

A critical approach to a more interactive MP environment is the use of *control points* (CPs). Several MP techniques incorporate CPs as part of their pipeline. This pipeline is divided into three main stages, as illustrated in Figure 1.2. In the first stage, the CPs are selected as a subset of samples from the multidimensional dataset. The CPs are then projected into the 2D space through a traditional dimensionality reduction technique. Finally, the remaining instances are projected into 2D through a mapping function that is designed to approximate, or interpolate, the position of the CPs. The repositioning of a CP in the projection space is followed by a change in the mapping function which, in turn, causes the position of the remaining instances to change accordingly. Recent MP techniques introduce the use of CPs as a steering mechanism, used to incorporate user input into the projection layout interactively [Joia et al., 2011]. In this scenario, the data projection is presented to the user, who is given the option to rearrange the position of the control points, which is subsequently followed by an update of the projection results.

Even though control points have been established as a steering mechanism embedded in MP, limited effort has been made towards techniques to select which data samples to play their role. Part of this thesis focuses on developing a mathematical methodology for the selection of control points. Since CPs are used to build a *mapping* that approximates the projection of the data, our goal is to select the subset of points that best represent the multidimensional dataset, resulting in an *accurate* mapping – i.e., the projection preserves multidimensional distances as much as possible. Furthermore, given that CPs are also used

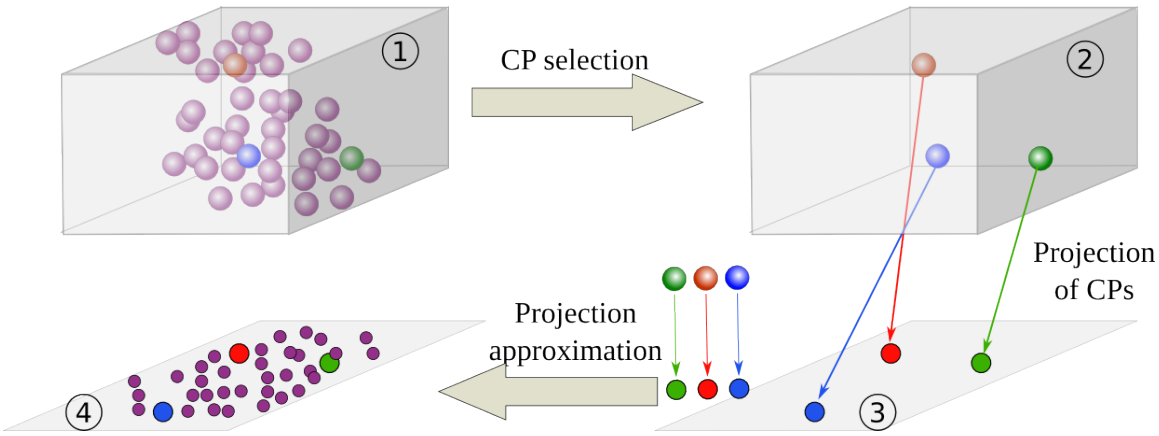


Figure 1.2: General pipeline of MP techniques with control points. (1) Given a multidimensional dataset, (2) a subset of samples, the control points, is selected; (3) the control points are projected into the 2D space through some standard dimensionality reduction technique. The correspondence between the high and low-dimensional position of control points is used to create a mapping function which will then (4) approximate the position of the remaining instances in the 2D projection space.

to steer the projection layout, another important goal is to select a concise subset, as to simplify the projection's interface.

I also propose a new workflow that takes full advantage of the MP layout, allowing users to generate new *multidimensional* points in the same environment used for data exploration. Some real-world applications go beyond exploring *existing* data but require the creation and evaluation of *new* multidimensional instances. For example, consider the problem of encountering input parameter combinations that result in a particular system output. This case is a common problem in science and is usually modeled as an optimization problem. Often, the parameter space that defines the regions of feasible solutions is large and difficult to explore or navigate visually. Automatic optimization methods are usually employed but fail on incorporating the user expertise and intuition into the process. Moreover, the solution to the problem is often non-unique, i.e., several different parameter combinations suit the required restrictions, making it necessary to perform an active exploration of the parameter space. The proposed *inverse projection* framework can be "plugged" into such applications, permitting an interactive visual exploration of the parameter space through

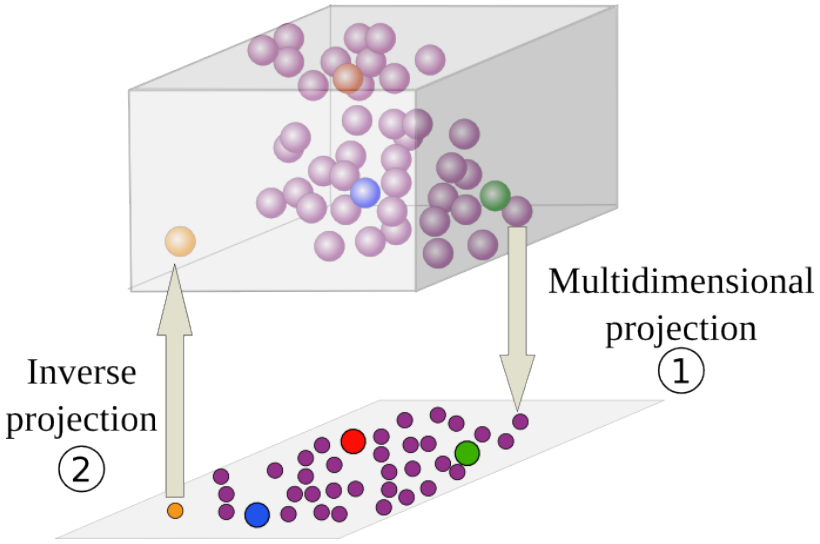


Figure 1.3: Inverse Projection framework: MP is calculated for a multidimensional dataset. The user defines a point in the MP layout (orange circle in the projection), and this point is transformed into a multidimensional vector (orange sphere in the multidimensional space).

data extrapolation, i.e., the creation of new multidimensional data instances by creating new samples in the 2D projection space, as illustrated in Figure 1.3.

1.2 Methodology

In this section, I present an overview of the proposed methodology for the CP selection approach, as well as the inverse projection framework.

1.2.1 Selection of Control Points

As shown in the previous section, control points play an essential role in the workflow of several MP techniques. The correspondence between their high- *and* low-dimensional positions is used to create mapping functions that calculate the projection of the remaining instances. Assuming the task of selecting a suitable set of control points is intrinsically linked to the task of creating the mapping function itself, I developed an MP technique with a novel definition for the mapping function, with CP selection in mind.

The problem of finding a mapping function, given the correspondence between the high

and low-dimensional position of control points, can be modeled as an interpolation problem. There are several mathematical theories and models devoted to the solution of interpolation problems, but Radial Basis Functions (RBF) stands out for our multidimensional scenario. RBF presents a well-established mathematical formulation, which has been used in diverse approximation applications. RBF is known for behaving well in the interpolation of multidimensional samples [Buhmann, 2003].

An advantage of the proposed RBF formulation is the existence of different works dedicated to center selection, where centers are the data samples used to derive the RBF interpolation function. One of these approaches is based on the solution of Orthogonal Least Squares problems to select a subset of samples that satisfactorily represents the dataset [Chen et al., 1991]. We incorporate this technique into our MP framework as a means to perform control points selection and to improve the final projection results. This approach automatically determines a good number of control points; thus, the user is not required to provide this important parameter. We evaluate our technique using various datasets, and we compare the projection quality against that of other well-established MP techniques.

1.2.2 Inverse Projection

Given a multidimensional dataset and its corresponding 2D projection, the goal of inverse projection is to find a multidimensional representation for a new user-defined 2D point. We call this process *inverse* projection, since it operates in the opposite direction of the MP workflow, as illustrated in Figure 1.3. In the proposed framework, the user is presented with the projection layout to interact with, and he/she can create new points in regions of interest inside the projection layout itself.

The intrinsic neighborhood information conveyed by MP makes it the ideal visualization environment to incorporate such an extrapolation tool. Since the proximity of points in 2D indicates the similarity between their corresponding instances in the multidimensional space, it is natural to interpret a new user-defined 2D point with respect to its distances

from the projected points.

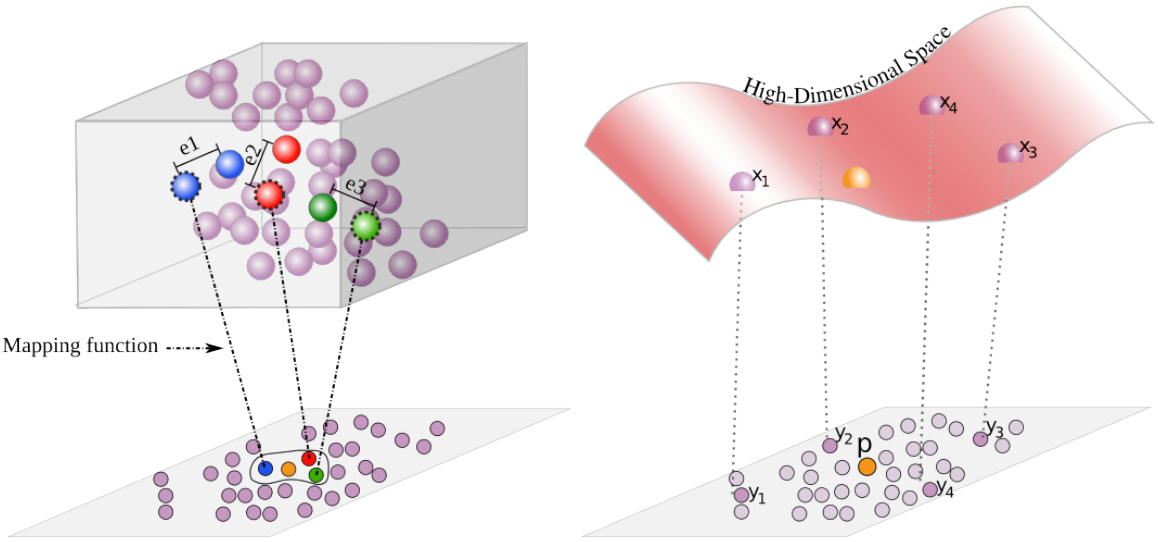
The problem of inverse projection is addressed as one of constructing a mapping function that maps a 2D point to the original parameter space, in such a way that user-defined 2D distances are preserved in the multidimensional space. I tackle this problem using two different approaches: a linear mapping function, based on local affine mappings, and a non-linear mapping function, based on the aforementioned RBF interpolation theory.

1.2.2.1 Linear Approach

The proposed linear approach for inverse projection is based upon the Local Affine Multidimensional Projection (LAMP) formulation [Joia et al., 2011], and we name it *iLAMP* (*inverse* LAMP). For each user-defined point p, a local affine mapping function is constructed, taking into consideration the distance between p and p's k-closest neighbors in the projection layout.

In general terms, given k 2D points $\mathbf{y}_i$ and their k corresponding multidimensional points $\mathbf{x}_i$, we seek for an affine transformation f that maps $\mathbf{y}_i$ into $\mathbf{x}_i$ as best as possible, assigning more weight to $\mathbf{y}_i$s that are closest to the user-defined point p. This function is then used to map p into the multidimensional space. Figure 1.4 (a) illustrates the construction of the iLAMP mapping function, as follows. Given a user-defined point (orange) in the projection space, the proposed method finds the k-nearest neighbors (red, green and blue, in this example $k = 3$) and constructs an affine mapping function. In the multidimensional space – spheres in red, green and blue represent the k-nearest neighbors correct position and dashed indicate the position *after* applying the mapping function, which should minimize the sum of errors $e_1 + e_2 + e_3$;

We demonstrate the usability of the iLAMP technique in an optimization problem, where iLAMP is used to explore the multidimensional parameter space in a scenario with multiple local minima. The method assists the user during the interactive exploration of the parameter space allowing different local minima to be rapidly discovered.



(a) iLAMP (b) iRBF

Figure 1.4: Inverse projection mapping Functions proposed in this thesis. Please refer to the main text.

When control points are drastically rearranged in the projection space, iLAMP may not be a suitable inverse projection choice, since its results can become exceedingly distorted. Furthermore, iLAMP does not provide a smooth and continuous mapping function, which may be desired in certain applications. Therefore, I also propose an alternative approach to iLAMP, which can be defined as a non-linear inverse projection technique based on the Radial Basis Function theory. This second approach is presented in the next section.

1.2.2.2 Non-linear Approach

The inverse projection with RBF, or iRBF, differs from the iLAMP approach in the way the mapping function is defined. In contrast to a local affine mapping, which *approximates* the position of k-closest neighbors, the iRBF method constructs a mapping function that *interpolates* the position of the projected points and their multidimensional counterparts. This mapping function is formed by a linear combination of non-linear radial basis functions, which are special basis functions that take as argument the distance between a point and the so-called *RBF centers*.

Figure 1.4 (b) illustrates the behavior of the RBF mapping function in the inverse projection framework. Note we are only using 4 RBF centers to simplify and facilitate understanding). Given RBF centers ($y_1..y_4$) and their corresponding multidimensional counterparts ($x_1..x_4$), the proposed iRBF mapping function interpolates exactly the given points and provides a mean to approximate a user defined point **p**. The RBF function used to approximate a new user-generated point p is continuous and interpolates the data samples exactly.

The proposed iRBF technique is demonstrated by a face-synthesis application, in which a 3D human-faces dataset is used as input, and iRBF is used to generate new 3D faces interactively. The results show the simplicity, robustness, and efficiency of the proposed approach to creating new face models from a structured dataset, a task that would typically require the manipulation of hundreds of parameters.

1.2.3 Software Prototype System

Both the inverse projection and CP selection workflows involve the design and implementation of mathematical formulations. The implementations are integrated into one interactive software prototype system, with five main capabilities: (1) Loading a multidimensional dataset; (2) Performing the CP selection using the proposed mechanism; (3) Calculating the 2D projection layout and displaying it in the projection space; (4) Allowig the user to manipulate control points to rearrange the projection; and (5) Permitting the user to create new multidimensional instances using inverse projection. Figure 1.5 presents the user interface layout of the prototype system.

1.3 Contributions

The contributions of this thesis can be divided into two main parts. (1) proposing a new form of control points selection, showing it can improve the projection quality and re-

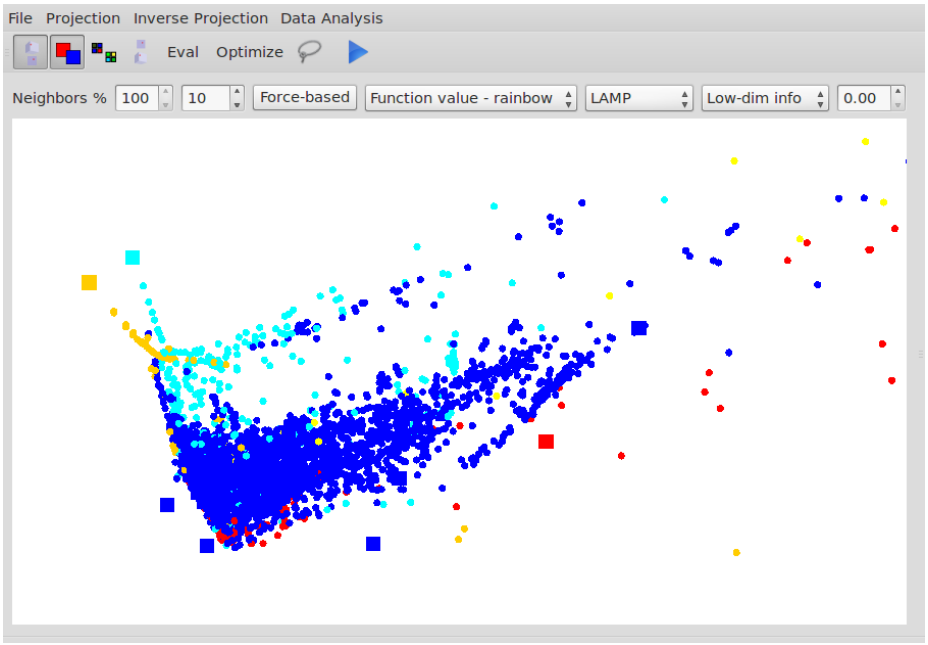


Figure 1.5: Software prototype system developed to incorporate and demonstrate the proposed MP workflows. The system can load any multidimensional dataset from an input file, select and position control points (squares on the plot), which can be interactively rearranged with real-time projection update. The system is also capable of processing inverse projections.

duce redundancy in the set of control points. (2) Creating a new workflow named *inverse projection*, incorporating an interactive exploration and extrapolation component into the multidimensional projection framework.

1.3.1 Selection of Control Points in Multidimensional Projection

As part of this research, I propose a new multidimensional projection technique with a built-in approach for control points selection. The method is built upon the radial basis function (RBF) interpolation theory; it is demonstrated that the CP selection strategy can generally improve the projection quality, while requiring a small number of CPs in comparison to other MP methods. I present an evaluation of the CP selection when incorporated into other MP methods, and we observe an improvement in projection quality when compared to random CP selection approaches. The results of this part of my research were published in [Amorim et al., 2014].

1.3.2 Inverse Projection

Taking advantage of the neighborhood information captured by multidimensional projection, I also propose an *inverse projection* framework to allow users to create new high-dimensional instances of data by creating reference 2D points in the projection space. The inverse projection approach is designed to assist in tasks where data analysis is necessary, but data extrapolation is also desirable. In the proposed framework, the projection space is used as a mechanism to inspect the dataset and to identify potential regions of interest yet to be sampled. Inverse projection operates in reverse to traditional projection mappings by transforming user-input 2D information into a high-dimensional space.

I demonstrate the usability of inverse projection in optimization problems with functions of various parameters, and in a 3D face synthesis application. I also propose a linear and a non-linear alternative to map data from low to high dimensions and discuss the advantages of one over the other. The results of the inverse projection research have been published in [Amorim et al., 2012] and [Amorim et al., 2015]. Also, in [Hajizadeh et al., 2012] we demonstrate how MP techniques can be used to give insight into the parameter exploration and exploitation in optimization problems, more specifically the history matching problem defined in Petroleum Engineering.

1.4 Organization of Thesis

This thesis is organized as follows: Chapter 2 presents a background on multidimensional data visualization, more specifically MP, with the mathematical details of the most important techniques in this area. The usage of control points in MP is also discussed, along with current applications of MP. Chapter 3 introduces the method based on radial basis functions, which contains a built-in mechanism for control points selection. This chapter presents results and comparisons with the most recent techniques, and also demonstrates how the proposed control points selection mechanism can be successfully applied in other

MP techniques. Chapter 4 presents the general inverse projection framework and software prototype system developed as part of this thesis. Chapter 5 introduces the inverse projection framework based on the iLAMP methodology. A detailed description of the mathematical formulations of iLAMP is presented, as well as an application designed to demonstrate examples for exploring multidimensional parameter spaces in an optimization problem. Chapter 6 presents the inverse projection framework based on the RBF methodology. I describe comparisons between RBF and iLAMP and introduce a new facial-synthesis application using the proposed method. Finally, Chapter 7 presents the conclusion and remarks on future work.

Chapter 2

Background

The goal of this chapter is to provide a background in multidimensional data visualization, with a special focus on MP methods and most recent trends in MP research. We also present a brief introduction to Radial Basis Function interpolation theory in Section 2.6, the foundation of two of the methods proposed in this work.

2.1 Multidimensional Data Visualization

Virtually every dataset can be represented as a multidimensional dataset. Examples include music and video [Joia et al., 2011], text documents [Chalmers, 1993], vector fields [Daniels II et al., 2010], geometrical shapes [Amorim et al., 2015] and flowers [Fisher, 1936]. In data visualization, there are techniques specially designed to visualize multidimensional data and, in this section, we present an overview of them.

Multidimensional datasets are commonly organized into tables, where each row accounts for one instance of data, and each column accounts for one attribute. As an example, refer to Table 2.1 which represents the *Iris* flower dataset [1] in table format, which catalogs 150 flowers with four parameters each: sepal length, sepal width, petal length and petal width.

Common tasks during analysis of multidimensional datasets include the identification of correlations between attributes, outliers, clusters and patterns. Statistical and data mining techniques play important roles in the accomplishment of these tasks, but they alone lack the visual representation and appeal we have discussed in the previous chapter. Thus, visualization plays an important role as a supporting analysis tool that aid users to make

[1] The iris dataset [Fisher, 1936] is popularly used in data classification and clustering as a primary example, and will be used throughout this section to demonstrate some multidimensional visualization techniques.

Exemplar	Sepal Length	Sepal Width	Petal Length	Petal Width	Species
1	5.1	3.5	1.4	0.2	I. setosa
⋮	⋮	⋮	⋮	⋮	⋮
51	7.0	3.2	4.7	1.4	I. versicolor
⋮	⋮	⋮	⋮	⋮	⋮
150	5.9	3.0	5.1	1.8	I. virginica

Table 2.1: The *Iris* flower dataset, containing observations of 150 exemplars of iris flowers, of three different species (Iris setosa, Iris versicolor and Iris virginica). Four attributes describe this dataset: sepal length, sepal width, petal length and petal width. The complete dataset can be found in the UCI Machine Learning Repository [A. Asuncion, 2007].

sense of such complex datasets and draw their interpretation of the results.

The common objective underlying multidimensional visualization techniques is to somehow map high-dimensional data into a 1- or 2-dimensional visual space. For instance, *parallel coordinates* [Inselberg and Dimsdale, 1990b] is a traditional technique to visualize and interactively explore multivariate data. Figure 2.1 presents the representation of the *Iris* dataset by parallel coordinates. Each dimension of the data is represented as a vertical axis with the value range of a particular parameter (i.e. its coordinate space); each of these axes is arranged in parallel to each other, describing a non-projective mapping of the *N*-dimensional space to the plane. A point in the N-dimensional dataset is represented as a polyline in the parallel coordinates, connecting its value for each data dimension.

Techniques like *Star Coordinates* are variations of parallel coordinates in which the axis of each dimension share a common origin [Kandogan, 2000]. This group of methods is often called *non-projective mappings*.

In contrast to non-projective mappings, *scatter plots* describe a simplistic projective view of the high-dimensional data [Cleveland and McGill, 1988]. A scatter plot is a standard plot, considering two variables of the high-dimensional data. Many visualization toolkits provide interactive scatter plot capabilities, including Tableau/Polaris [Stolte et al., 2002] and GGobi [Swayne et al., 2003]. Because scatter plots are limited in the number of dimensions they visualize in comparison to the size of most datasets, multiple plots are

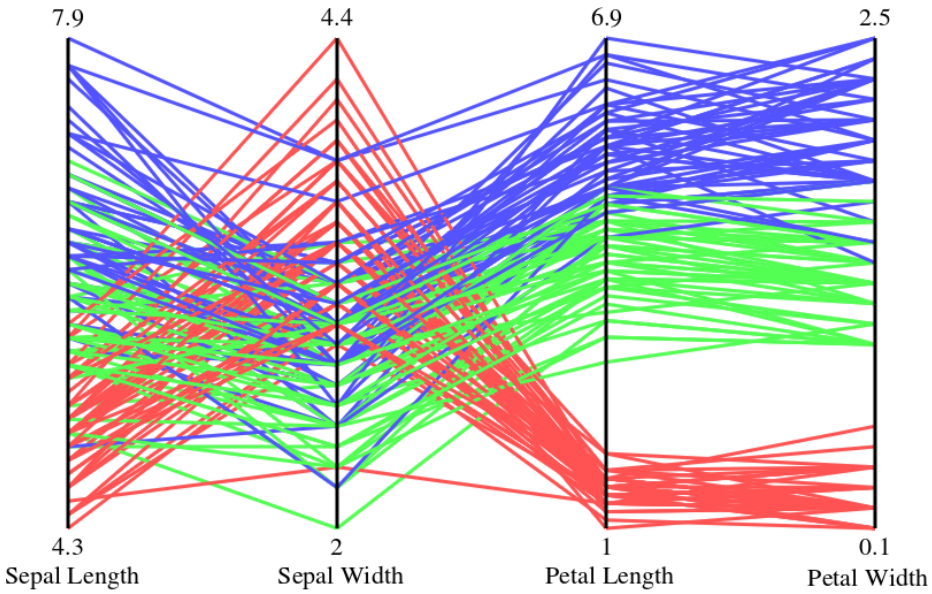


Figure 2.1: Representation of *iris* dataset with parallel coordinates. Each of the four parameters is represented as parallel lines mapping its value range (or coordinate space), and each instance of data is represented by a polyline connecting the values of each parameter. The colors indicate to which species the instance belongs to (red - Iris setosa, green - Iris versicolor and blue - Iris virginica).

typically arranged in rows and columns forming *scatter plot matrices*. Figure 2.2 illustrates the scatter plot matrix representation for the *iris* dataset. Scatterplot representations of data are classified as *simple-projective mappings*.

Both non-projective and simple-projective mappings are valuable tools to understand the correlation between pairs of parameters of the dataset. However, there is a clear limitation in the number of attributes that can be represented simultaneously, making it challenging to use such techniques for high-dimensional data. Moreover, these techniques fail to provide a clear representation of proximity information in the dataset, an important aspect in the identification of clusters, outliers and patterns. *Dimensionality reduction* solutions, in turn, further extend scatter plots by encoding all dimensions of the original data within the 2D visualization. In the visualization community, dimensionality reduction techniques are often called *multidimensional projection*, and we discuss more in-depth about this set of methods in the next section.

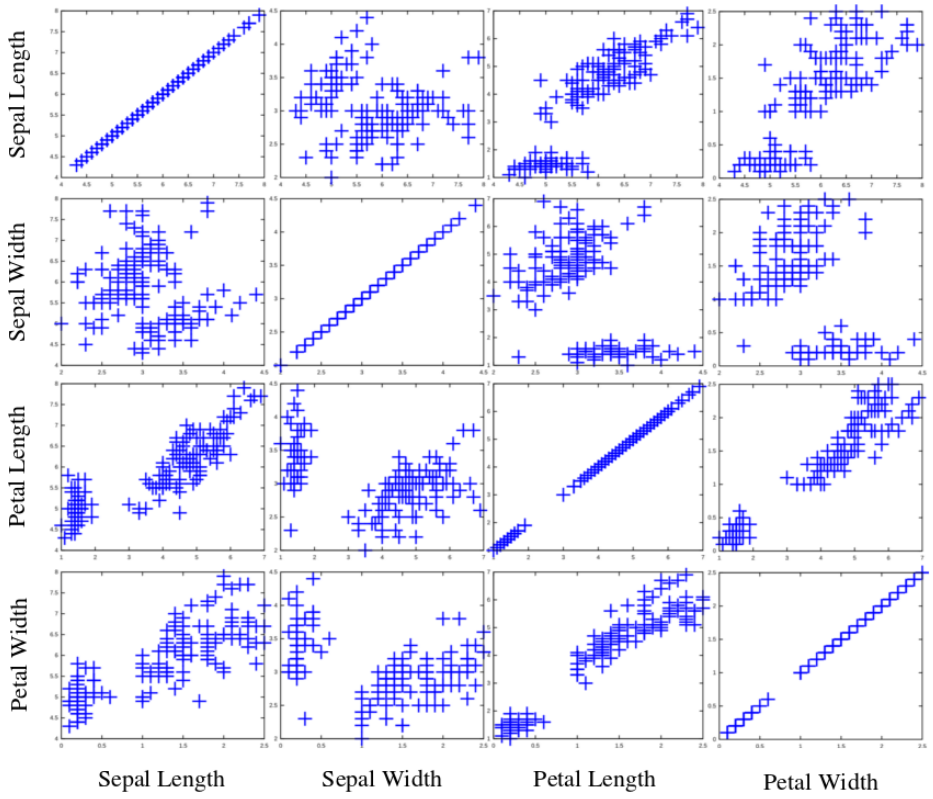


Figure 2.2: Representation of *iris* dataset with Scatterplot matrix. For each pair of attributes, a scatter plot graph is constructed.

2.2 Multidimensional Projection

Techniques for dimensionality reduction are commonly used in machine learning solutions, for example, to decrease the number of attributes and consequently reduce the complexity of the problem [Fukumizu et al., 2004]. In visualization, these techniques are used to transform a multi-dimensional dataset into a 2-dimensional one [2], permitting the data to be represented as a scatter plot.

Multidimensional projection methods have been the foundation of several high-dimensional exploration and visualization tools. Examples include: feature exploration in multivariate scalar fields [Janicke et al., 2008], vector fields [Daniels II et al., 2010], text mining [Chen et al., 2009, Paulovich and Minghim, 2006], finances [Deboeck and Kohonen, 2010] and

[2] 3D mappings are also used in some cases [Poco et al., 2011]. The choice between 2D or 3D mappings is driven by which space is likely to improve user perception of data groups. In this thesis, we are considering 2D mappings only.

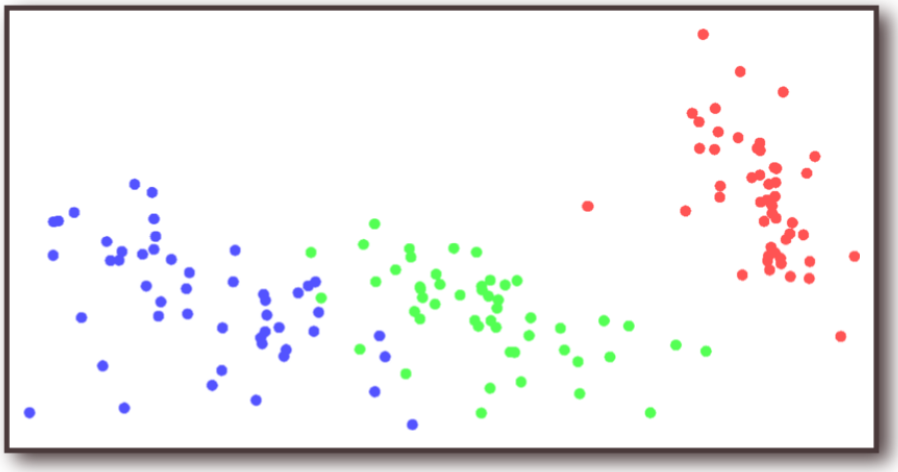


Figure 2.3: Representation of *iris* dataset with multidimensional projection. The colors indicate to which species the instance belongs to (red - Iris setosa, green - Iris versicolor and blue - Iris virginica). MP facilitates the interpretation of clusters and patterns, as opposed to parallel-coordinates and scatterplot matrices.

psychology [Bennett and Hays, 1960] rely on the clustering nature of these projection techniques.

In general terms, a multidimensional projection solution attempts to find a 2-dimensional representation of the dataset in such a way that distances match, as well as possible, the original dissimilarities. In other words, instances that are similar to each other should be plotted close together, while dissimilar instances should be plotted far apart. When this is the case, the resulting scatter plots become extremely useful for visual analysis and exploration within the plane of similarities hidden in the high-dimensional data.

Consider for example the representation of the *iris* dataset in terms of multidimensional projection in Figure 2.3. Each point in the plot represents one instance of the data, i.e., one exemplar of the iris flower. Colors once again are used to differentiate between the three observed species of iris. The distance between instances in this 2D representation, or *projection space*, should match, as well as possible, the original distances in the multidimensional space. Thus, similar instances are rendered close together, facilitating the interpretation of clusters and patterns. In fact, in Figure 2.3 we can more explicitly identify

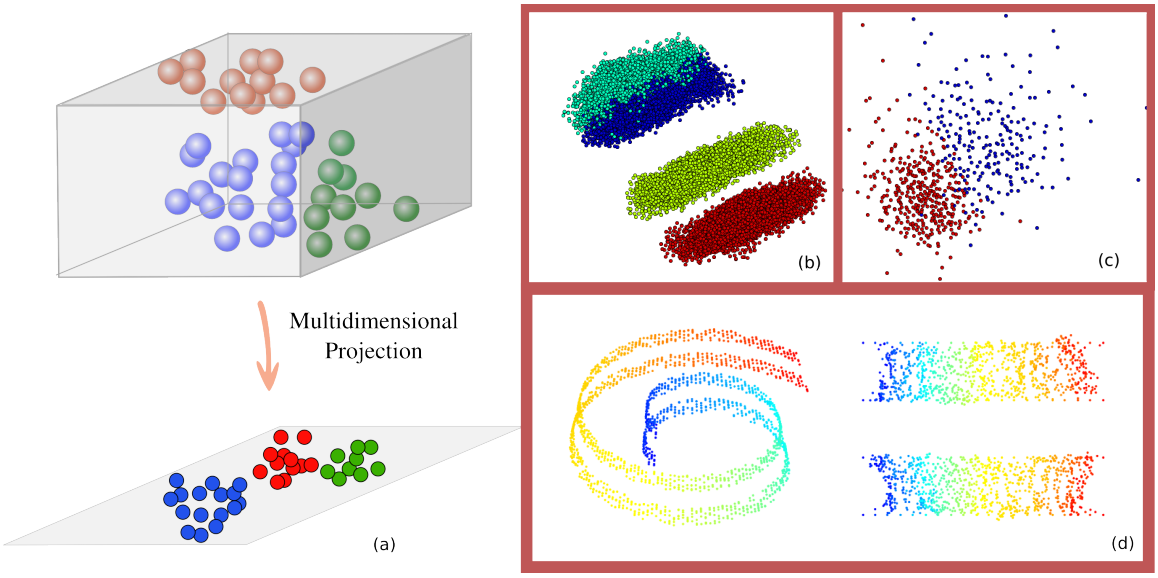


Figure 2.4: (a) Graphical representation of multidimensional projection process; (b) Projection of mammals dataset – 20,000 instances with 4 different quadruped mammal species: dogs (blue), cats (green), horses (yellow) and giraffes (red); (c) Projection of "Breast cancer Winsconsin" dataset – 699 instances with 2 different groups: malign (red) and benign (blue) tumors; (d) Parallel 3D swiss rolls projected into 2D through MP. Colors are used as a gradient from red to blue to facilitate correspondence between 3D to 2D.

the clusters of the dataset in contrast with the other representations where this information is not as clearly displayed. Furthermore, with such a representation one can comprehend which iris species are more similar (or dissimilar) to each other based on the distances between the three point clouds.

Figure 2.4 illustrates the MP process and presents some examples of MP results. Figure 2.4-(a) shows the general multidimensional projection process, which consists of mapping a multidimensional dataset into a 2D plane, or the projection space. Figures 2.4-(b) and (c) presents the projection results of the *mammals* and the *Breast Cancer Winsconsin* datasets, respectively. Both datasets are part of the UCI Machine Learning repository [A. Asuncion, 2007]. The mammal's dataset consists of 20,000 instances and 47 attributes, and it contains four different quadruped animal species: dogs, cats, horses and giraffes. The features in this dataset describe the height, radius and position of eight components: neck, four legs,

torso, head and tail. MP can neatly separate these four clusters, as can be seen in Figure 2.4-(b). The *Breast Cancer Winsconsin* dataset, in turn, consists of 699 instances divided into two groups: malign and benign tumors. Each instance is described by ten attributes, such as clump thickness and uniformity of cell size. MP gives an overview of how these two groups are distributed, as seen in Figure 2.4-(c). Finally, Figure 2.4-(d) presents an example of using MP as a manifold learning technique [3], where 3D swiss rolls are "unfolded" into 2D through multidimensional projection.

In mathematical terms, consider an m-dimensional dataset $X = \{x_1, \ldots, x_N\}, x_i \in \mathbb{R}^m$ and let $Y = \{y_1, \ldots, y_N\}, y_i \in \mathbb{R}^2$ be its 2D counterpart, i.e., y_i is the 2D representation of x_i. Let δ_{ij} be a measure of dissimilarity between instances x_i and x_j (e.g., the distance in $\mathbb{R}^m$ between the two instances), and d_{ij} be the Euclidean distance between 2D points y_i and y_j. A common metric that measures the quality of MP results is the *stress* $s(X,Y)$, written as

$$s(X,Y) = \frac{\sum_{i,j=1}^{n} (\delta_{ij} - d_{ij})^2}{\sum_{i,j=1}^{n} \delta_{ij}^2}. \tag{2.1}$$

The stress function indicates how well dissimilarities of the original dataset were mapped into the 2D projection space. A stress value of 0 indicates that there is no loss of information during the projection process, while higher stress values indicate poorer MP mappings.

Dissimilarity metrics in the high dimensional space can be user-defined, and choice will depend on the problem at hand. Common dissimilarity metrics for quantitative data are the

[3] In simple terms, a manifold can be defined as any object that is nearly flat on small scales - a more formal definition can be found in [Lee and Verleysen, 2007]. For example, general non-linear surfaces are considered manifolds. Manifold learning is the term used when dimensionality reduction is applied to find a lower-dimensional representation of data that is originally embedded in a high-dimensional space while preserving the topological structure.

p-norms, i.e.,

$$\delta_{ij} = \left(\sum_{k=1}^{m} (x_i[k] - x_j[k])^p \right)^{\frac{1}{p}}, \qquad p \geq 1. \tag{2.2}$$

When $p = 2$ we have the Euclidean distance, which we often use in the experiments presented in this work.

In the next section, we discuss some classic dimensionality reduction techniques, used as MP methods. Later on, in Section 2.5 we present some recent trends in MP research, more specifically the usage of control points as means to speed up the process and give users some exploration power.

2.3 Classic MP techniques

Principal component analysis (PCA) [Jollife, 1986] and *multidimensional scaling* (MDS) [Cox and Cox, 2000] are traditionally used as MP techniques. Both techniques were initially designed as mathematical tools to transform an m-dimensional dataset into a p-dimensional one, with $p \ll m$. It didn't take long for their data visualization potential to be realized and explored. Next, we present the mathematical description of these methods.

2.3.1 Principal Component Analysis

The development of PCA is mainly attributed to [Pearson, 1901], more than a century ago. The central idea of PCA is to reduce the dimensionality of a high-dimensional dataset with interrelated variables while retaining as much as possible of the variation present in the dataset. This dimensionality reduction is made by transforming the data into a new set of uncorrelated variables, the *principal components* (PCs). The PCs are ordered such that the first retains the most variation of the original variables, the second retains the second most variation possible but subjected to the constraint of being uncorrelated to the first

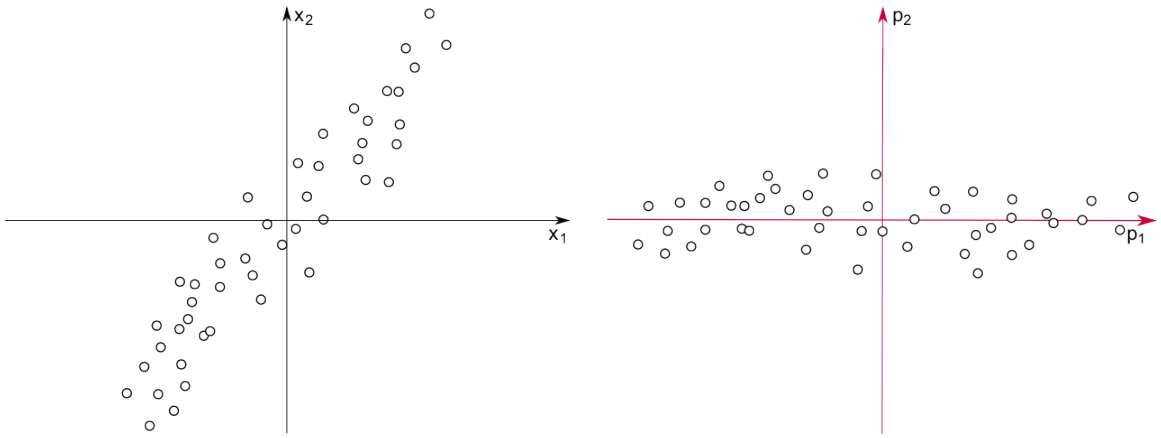


Figure 2.5: Application of Principal Component Analysis in a simple 2D example. The plot on the left contains the dataset represented by its original variables x_1 and x_2. On the right, the variables have been transformed into the principal components p_1 and p_2. It is clear that p_1 retains most of the data variation and, in this very simple case, it could be used to represented the data in a 1-dimensional manner, ignoring p_2 completely. The same principle can be applied in multivariate datasets.

one, and so on. A very detailed presentation of PCA and its uses is encountered in the book of [Jollife, 1986] and in this thesis we present the basic mathematical concepts of this technique.

There are different ways to derive the PCA solution. Refer to [Shlens, 2005] for a comprehensive representation of linear algebra approaches. We show how the PCs can be found by decomposing the covariance matrix of the dataset into eigenvectors and eigenvalues. Consider the dataset X introduced in Section 2.2, but this time represented as an $N \times m$ matrix, i.e., rows of matrix X are data instances and columns are the attributes. The first step in PCA is to make each attribute of X mean-centered, i.e., subtract each column by the mean of the column, yielding matrix $\bar{X}$. The next step is to calculate the covariance matrix [4] Σ of $\bar{X}$, which is given by the simple matrix-matrix multiplication

$$\Sigma = \frac{1}{N-1}(\bar{X}^T \bar{X}). \tag{2.3}$$

[4]Covariance is a statistical metric that indicates the linear relationship between two random variables. Here, the covariance matrix captures the covariance between each pair of attributes present in the dataset.

Matrix Σ is then decomposed regarding eigenvalues and eigenvectors. Because Σ is symmetric, its eigenvector decomposition is written as

$$\Sigma = EDE^T, \tag{2.4}$$

where D is a diagonal matrix, with d_{ii} the i-th eigenvalue of matrix Σ, and $E = \{e_1, \ldots, e_m\}$ is an orthogonal matrix of eigenvectors of Σ arranged into columns. The measure of how much variation of the original dataset X is preserved by vector e_i is given by its corresponding eigenvalue d_{ii}: the larger its value, the more variation is preserved by e_i.

Figure 2.5 illustrates the PCA method in a simple 2D example. Consider that the basis E is ordered according to variance preservation, i.e., $d_{11} \geq d_{22} \geq \ldots \geq d_{mm}$. The vectors e_i will form a basis to transform the original dataset X into a set of uncorrelated variables. If we make $E^T X^T$ we find the representation of X in terms of the orthogonal basis E, but nothing is gained regarding dimensionality reduction. If we take, however, only the first $k < m$ most important eigenvectors to form a basis, i.e., $E_k = \{e_1, \ldots, e_k\}$ and apply $E_k^T X^T$, the dimensionality of X is reduced from m to k. For visualization purposes, $k = 2$ is used to encounter a 2D representation of dataset X.

2.3.2 Multidimensional Scaling

Multidimensional Scaling (MDS) comprises a group of techniques that operates upon dissimilarities directly, i.e., an Euclidean representation of the data is not required as long as the dissimilarities δ_{ij} between each pair of instances is known. As defined by [Cox and Cox, 2000], MDS is the search for a low-dimensional space in which points in the space represent the instances of data, one point representing one instance, and such that distances

between the points in the space match as well as possible the original dissimilarities. There are various MDS methods, but the best-known are *classical scaling*, *metric least squares scaling* and *force schemes*, as described next.

Classical scaling treats dissimilarities δ_{ij} as Euclidean distances and makes use of a spectral decomposition of the doubly-centered matrix of dissimilarities. It developed from the work of [Young and Householder, 1938] that demonstrated how to recover the original coordinates of N points given only the Euclidean distances between pairs of the original points. The spectral decomposition used in classical scaling is the same mathematical tool used to derive the PCA formulation, as presented before. In fact, classical scaling yields the same results as PCA when the dissimilarities used are exactly the Euclidean distances, as demonstrated in the book of [Cox and Cox, 2000]. As in PCA, classical scaling makes use of the first k "principal coordinates" (equivalent to the principal components in PCA) calculated in the eigenvalue decomposition to represent the data in a lower-dimensional space.

Metric least squares scaling, in turn, finds a configuration matching d_{ij} to δ_{ij} by minimizing a loss function S. Sammon's mapping, proposed by [Sammon, 1969], is a well-known MDS technique that suggests the loss function

$$S = \frac{1}{\sum_{i<j}^{N} \delta_{ij}} \sum_{i<j}^{N} \frac{(d_{ij} - \delta_{ij})^2}{\delta_{ij}}, \tag{2.5}$$

i.e., function S measures the error between original dissimilarities δ_{ij} and distances d_{ij} between 2D y_i and y_j. The dissimilarities δ_{ij} are known, and a steepest descent optimization algorithm is applied to find the coordinates of the 2D points that minimizes S, i.e., making d_{ij} as close as possible to δ_{ij}. This approach is, as previously discussed, the goal of most MP methods. In fact, the stress function presented in Equation (2.1) is very similar to the Sammon's mapping loss function. As we can see from the denominator, such loss function

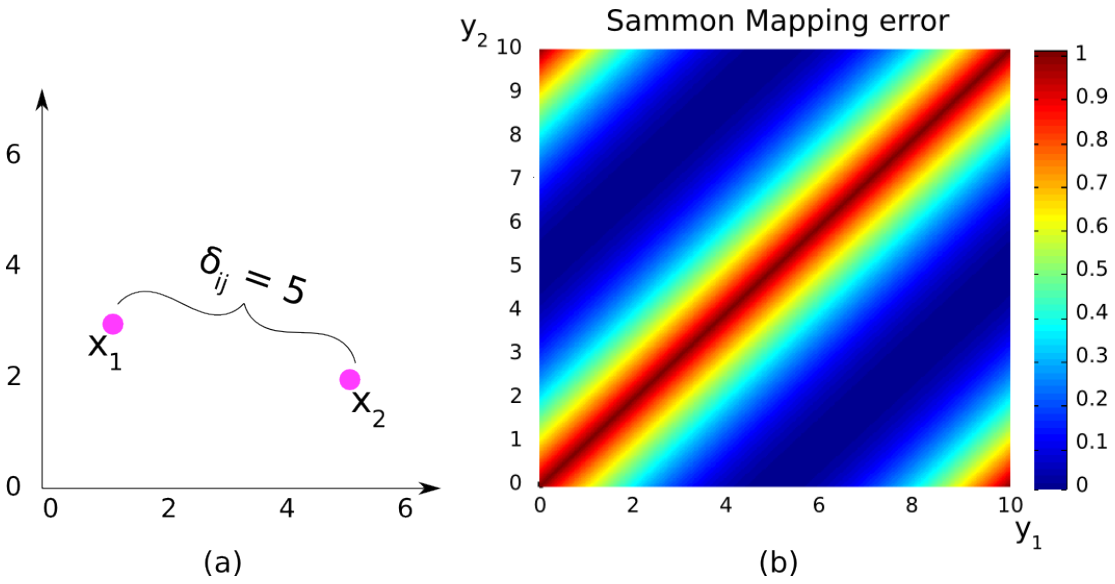


Figure 2.6: A simple example of the Sammon's mapping loss function applied in a 2D to 1D mapping. (a) Given two 2D points x_1, x_2 with dissimilarity of 5 units, a 1D representation of the data is sought in such a way that the loss function (2.5) is minimized. In such a simple example, there are infinite pairwise combinations of 1D points that will make the loss function zero, i.e., whose distance equals 5. The loss function in this scenario is illustrated in (b) - the horizontal and vertical axes represent the $y_1, y_2 \in \mathbb{R}$. In real world datasets, the number of instances and dimensions is much larger. The loss function may become very complex and no guarantees can be offered regarding finding the global optimum of the loss function.

gives greater weight to smaller dissimilarities, i.e., it works better to maintain similar pairs close together than dissimilar pairs far apart. Different functions can be defined to work the other way around or to provide a better balance between smaller or bigger dissimilarities, such as the one presented in the work of [Demartines and Herault, 1997].

Force schemes use a spring model to try to minimize the difference between the distances in the projection space and in the multidimensional space. The basis of these methods come mainly from work developed by [Eades, 1984] in graph visualization, which makes an analogy between stress minimization and mass-spring systems. Force-based schemes rely on spring forces, similar to those in Hooke's law, where repulsive and attractive forces between the points in the multidimensional space dictate the final projection layout. The springs generate forces in directions which tend to pull distant but similar

objects towards each other, and push close but dissimilar ones apart [Tejada et al., 2003].

In the next section, we discuss MP techniques in terms of the linearity of their mathematical methodology.

2.4 Linear and nonlinear techniques

The classification of linear and nonlinear accounts for the kind of transformation applied to the instances of the original dataset. Linear mappings are designed to operate when the submanifold is embedded linearly, or almost linearly in the observation space [de Silva and Tenenbaum, 2002]. However, they cannot capture nonlinear relationships between data instances, which are usually accomplished by nonlinear transformations. Some examples of linear projection methods are the aforementioned principal component analysis (PCA) [Jolliffee, 2002] and classic multidimensional scaling [Cox and Cox, 2000].

Least-squares scaling methods, such as the Sammon's mapping [Sammon, 1969], are examples of nonlinear mappings. Nonlinear techniques attempt to minimize a function of the information loss caused by the projection. Such a function measures the error between dissimilarities in the original and projected spaces. Sammon's mapping and many others that derive from it applies a steepest-descent procedure to solve the optimization problem. One of the disadvantages of least-squares techniques is that gradient-based methods do not guarantee convergence to the global minimum of the function. Consequently, the final projection layout may not be a good representation of the original dataset. Roweis and Saul [Roweis and Saul, 2000b] proposed a method called Locally Linear Embedding (LLE) that uses local information to achieve an optimization without local minima. There are other examples of nonlinear projection methods. (a) Curvilinear Component Analysis (CCA) [Demartines and Herault, 1997], which presents a variation on the loss function proposed by Sammon. (b) Isomap [Tenenbaum et al., 2000] which applies the the geodesic distance information to compute dissimilarities. (c) Least Square Projection (LSP), intro-

duced by [Paulovich et al., 2008a].

Some recent methods propose the combination of linear and nonlinear transformations. For instance, the aforementioned LLE [Roweis and Saul, 2000b] computes some weights and vectors linearly, but the overall process is nonlinear. Also, Part-linear multidimensional projection (PLMP) [Paulovich et al., 2010] and LAMP [Joia et al., 2011] makes use of a subset of samples (*control points*) initially positioned in the projection space through a nonlinear technique. The remaining instances are subjected to a linear transformation constructed based on the final position of the CPs in the projection space.

As described in Chapter 1, the use of CPs have gained some popularity in MP techniques, and we present a more in-depth discussion about them in the next section.

2.5 Multidimensional Projection with Control Points

Recently, control points have become an essential part of the workflow of many MP techniques. Generally, control points are a set of multidimensional points $X_S = \{x_{s1}, \ldots, x_{sk}\}$ formed by a reduced number of samples of the original dataset X, i.e., $X_S \subset X$, with $k << N$. In a preprocessing step, the set of control points X_S is projected into the 2D space with standard techniques such as MDS, PCA or force-based. Once the 2D correspondence $Y_S \in \mathbb{R}^2$ of X_S is known, it is used to construct a mapping function $f : \mathbb{R}^m \rightarrow \mathbb{R}^2$ that approximates the 2D position of the remaining instances of X. This workflow is illustrated in Figure 2.7.

Control points were first introduced in multidimensional projection as a means to speed up the projection process, since traditional techniques can become computationally prohibitive as N increases. The process indeed is faster by using a reduced number of points $k << N$ and a mapping function f to approximate the 2D position of the remaining points. [Pekalska et al., 1999] were the first to introduce the use control points in multidimensional projection with the goal of specifically speeding up the Sammon's mapping algorithm while preserving the projection quality. The mapping function f created in their work is a linear

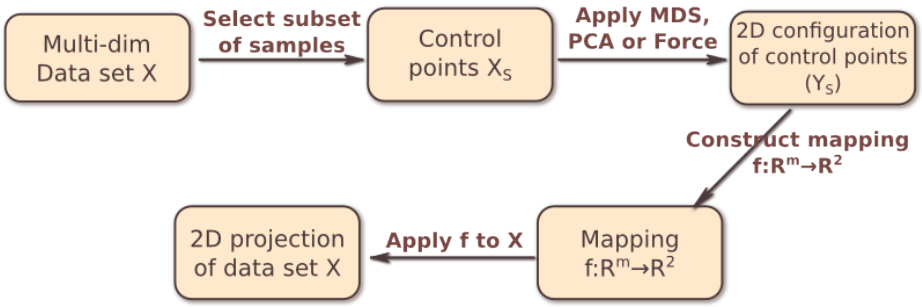


Figure 2.7: General workflow of techniques with control points. Given a multidimensional dataset, a subset of samples is selected and projected in a 2D space through a standard technique. A mapping function f is constructed based on this subset high to low dimensional correspondence. The remaining instances are projected to 2D through f.

transformation that respects the high to low dimensional mapping of the control points.

Since Pekalska *et al.*'s work [1999], different techniques based on control points have been proposed. Exampled include L-Isomap [de Silva and Tenenbaum, 2002], L-MDS [de Silva and Tenenbaum, 2004], LSP [Paulovich et al., 2008a], PLMP [Paulovich et al., 2010], LAMP [Joia et al., 2011] and PLP [Paulovich et al., 2011]. The technique used to project the control points and the type of mapping function f are what make each method unique. A comparison of the methods is presented in Table 2.2.

Method	Projects CPs with...	Mapping function
Pekalska	Sammon's mapping	Linear transformation
L-MDS	Classical MDS	Linear triangulation procedure
L-ISOMAP	ISOMAP	Linear triangulation procedure
LSP	MDS	Laplace operator
PLMP	Force-based	Linear
LAMP	Force-based	Local affine
PLP	Force-based	Local Laplace operator

Table 2.2: Comparison between various methods that use control points, regarding projection technique and mapping function.

2.5.1 Control Points and User Interaction

In recent years, control points have gained even more significance in visualization applications, as they have been leveraged as a way for users to control, to some extent, the

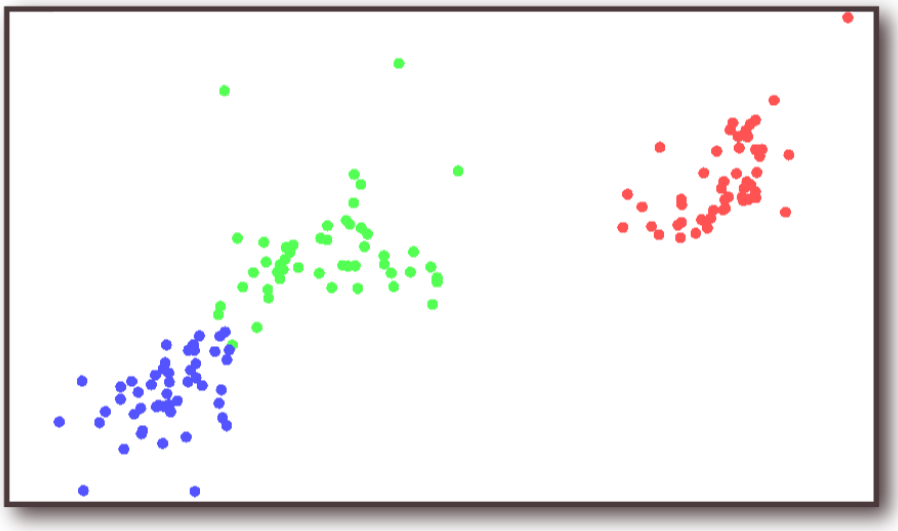


Figure 2.8: Modification of the iris dataset projection layout, by means of control points manipulation. By rearranging control points one is able to better separate clusters that would be cluttered otherwise. Compare this layout with the one displayed in Figure 2.3, where the green and blue clusters are more overlapped.

projection results. In such applications, the user can move control points and rearrange their positions in the projection space. Since the mapping function f is created based on the 2D position of the control points, these rearrangements are reflected in the final 2D projection of the remaining points.

This use of control points is demonstrated to be useful in different scenarios. For example, it can be utilized as a means to incorporate a priori knowledge of the dataset into the projection or to better separate clusters in the visual space, as illustrated for the Iris dataset in Figure 2.8. It can also be a valuable tool to unveil points that can become hidden due to cluttering in the projection space, as presented in Figure 2.9. A more in-depth exploration of control points rearrangement can be seen in [Joia et al., 2011], where it is used to create correlations between different datasets in the projection space.

One of the contributions of this work is a mechanism for CP selection, which is built upon the Radial Basis Functions (RBF) interpolation theory, as will be presented in the next chapter. Since RBF is also the foundation of one of our proposed inverse projection

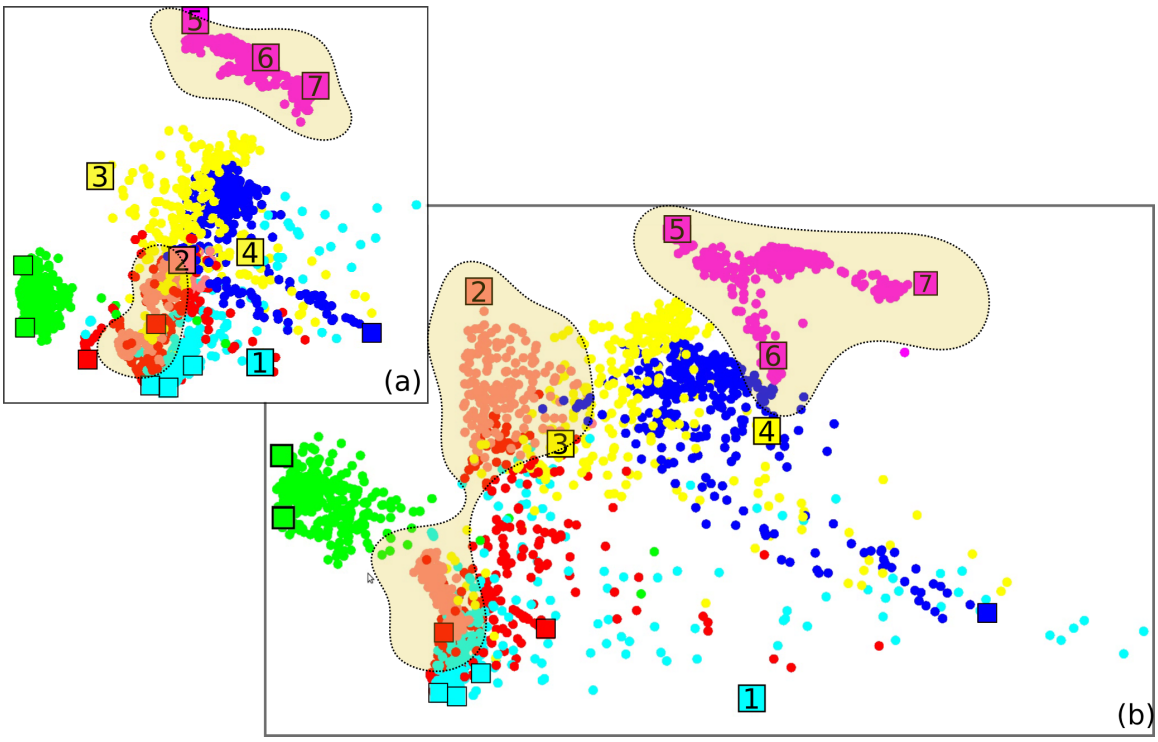


Figure 2.9: Control points manipulation can unveil points cluttered in the projection. In this example, we used a dataset called *Segmentation*, comprised of 2310 instances and 19 attributes extracted from seven outdoor images. Points with the same color indicate instances that belong to the same image [A. Asuncion, 2007]. (a) Initial projection; control points are represented by squares. The numbered control points $(1-7)$ are repositioned in (b), and the projection layout changes substantially, revealing new instances mainly in the highlighted areas.

formulations (Chapter), we present an overview of its theory below.

2.6 Radial Basis Functions

The problem of reconstructing an unknown function f from a finite set of discrete data is very common in several applications from various domains [Wendland, 2004]. The available data are usually formed by data sites $\Xi = \{\xi_1, \ldots, \xi_N\} \subset \mathbb{R}^m$ and data values $f_i = f(\xi_i) \in \mathbb{R}, 1 \leq i \leq N$ and the reconstruction of f consists on finding a function s that approximates the data values at the data sites. The approximant function s should either interpolate the data, i.e. $s(\xi_i) = f_i$, or at least approximate it, i.e. $s(\xi_i) \approx f_i$. When data

sites Ξ are multivariate, one of the best methods for reconstruction is *radial basis functions* (RBF).

In RBF, the approximant s takes the form of a linear combination of radial basis kernels ϕ centered at the data sites:

$$s(x) = \sum_{i=1}^{N} \lambda_i \phi(\|\xi_i - x\|) \tag{2.6}$$

where λ_i are real coefficients; $\phi : \mathbb{R}^+ \to \mathbb{R}$ are given, continuous functions, called radial basis kernels centered at ξ_i; and ξ_i are the data sites, here called *RBF centers*. The function ϕ and centers ξ_i are given, while the real coefficients λ_i need to be calculated. Figure 2.10 illustrates a 1-dimensional RBF interpolation using five data samples and Gaussian kernel.

In this thesis, we are interested in a function s that interpolates the available data, thus λ_i's are sought so as to make $s(\xi_j) = f_j, \forall \xi_j \in \Xi$:

$$\begin{cases} s(\xi_1) = \sum_{i=1}^{N} \lambda_i \phi(\|\xi_1 - \xi_i\|) = f_1 \\ \dots \\ s(\xi_N) = \sum_{i=1}^{N} \lambda_i \phi(\|\xi_N - \xi_i\|) = f_N. \end{cases} \tag{2.7}$$

The solution of coefficients λ_i's comes down to the solution of a linear system

$$\boldsymbol{A}\boldsymbol{\lambda} = \boldsymbol{f}, \tag{2.8}$$

where $\boldsymbol{A}$ is the $N \times N$ interpolation matrix, $\boldsymbol{\lambda}$ is an N-dimensional vector of unknowns and

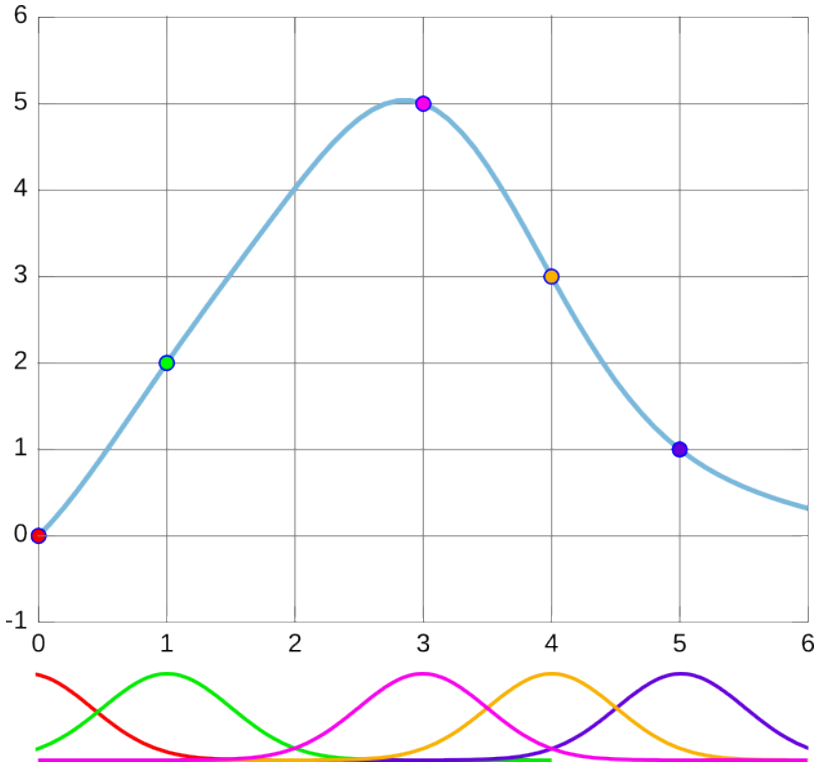


Figure 2.10: Radial Basis Function interpolation with five data samples $X = \{0,1,3,4,5\}$ and function values $Y = \{0,2,5,3,1\}$. The data samples are represented as colored points in the graph. The blue curve is the function obtained with RBF interpolating between the data samples. Below the graph, the Gaussian radial basis functions $\phi(r) = e^{-(\varepsilon r)^2}$ are represented, where r is the distance between a point and the data sample, with $\varepsilon^2 = 0.5$. The colors of the curves make the correspondence to the data samples.

$\boldsymbol{f}$ is an N-dimensional vector. Let $\phi_{ij} = \phi(\|\xi_i - \xi_j\|)$, Equation (2.8) can be written as

$$\begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1N} \\ \vdots & \vdots & \vdots & \vdots \\ \phi_{N1} & \phi_{N2} & \cdots & \phi_{NN} \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_N \end{bmatrix} = \begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix}. \tag{2.9}$$

Once the scalar λ_i's are calculated, function s can be used to approximate the value of any given point $x \in \mathbb{R}^m$.

2.6.1 RBF kernels

We showed in the previous section that the solution of the radial basis function interpolation problem reduces to the solution of linear system (2.8). Thus, a fundamental matter in RBF

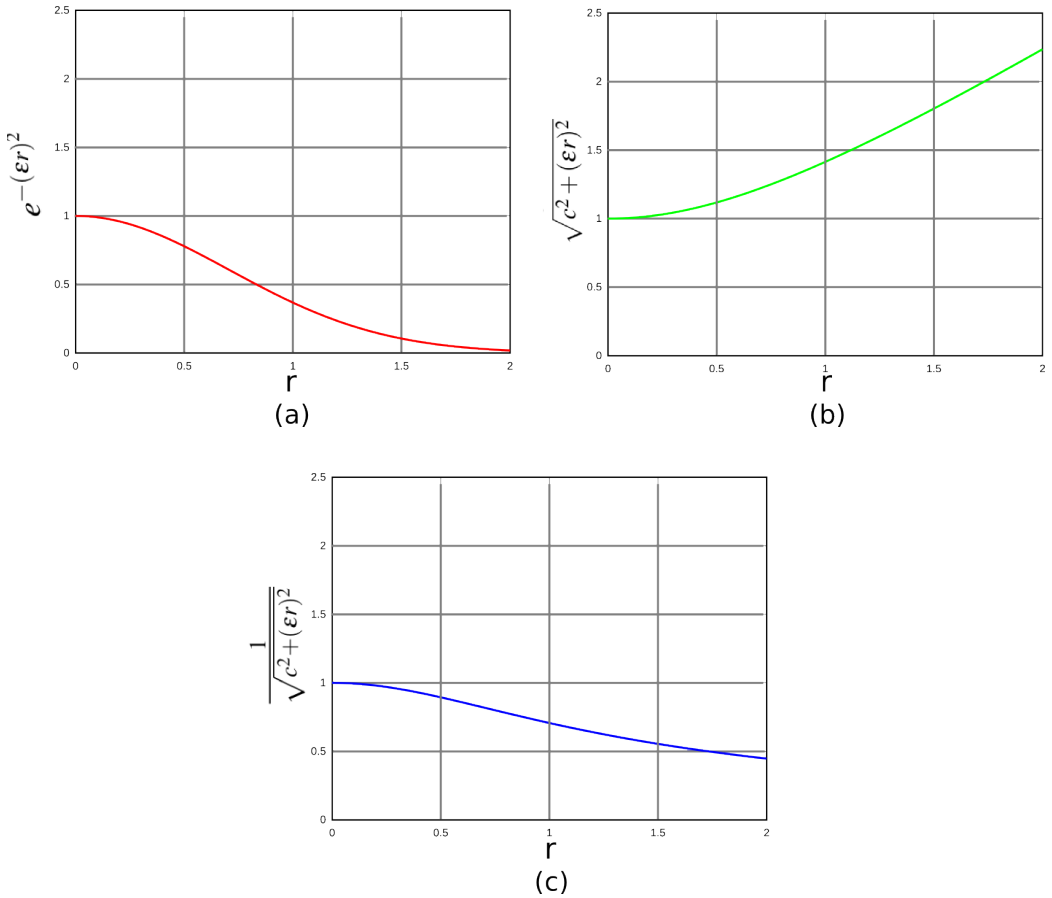


Figure 2.11: (a) Gaussian, (b) multiquadrics and (c) inverse multiquadrics functions in one dimension. In this example all of the functions have $\varepsilon = 1$ and $c = 1$, when applicable.

approximation is to ensure the non-singularity of the interpolation matrix $\boldsymbol{A}$. Remember from Equation (2.9), that the entries of $\boldsymbol{A}$ are determined by the RBF centers ξ_i and the kernel function ϕ. Therefore, the kernel function ϕ is crucial and cannot be arbitrarily chosen, as it plays an essential role in the form the interpolation matrix assumes. In fact, much of the research done in the area of RBF approximation focus precisely in encountering kernel functions that produce non-singular interpolation matrices with the reasonable assumption the RBF centers are all unique.

Three of the most common kernels that guarantee the non-singularity of matrix $\boldsymbol{A}$ are called *Gaussian*, *Multiquadrics* and *Inverse Multiquadrics* and their definitions are given in Table 2.3. Figure 2.11 presents the graph of these three functions with respect to the

Name	Definition of $\phi(r)$
Gaussian	$e^{-(\varepsilon r)^2}$
Multiquadrics	$\sqrt{c^2+(\varepsilon r)^2}$
Inverse Multiquadrics	$\frac{1}{\sqrt{c^2+(\varepsilon r)^2}}$

Table 2.3: Commonly used kernels in RBF. r is the distance argument of ϕ; ε and c are positive parameters also referred to as *shape parameters* [Mongillo, 2011].

distance argument r. Note that RBF kernels have as argument a positive real number that measures the distance between the point $x \in \mathbb{R}^m$ and the RBF center. Euclidean distance is typically used, in which case the kernel is radially symmetric.

In Chapter 3, we present how we apply the RBF theory to create a new MP method with control points. We also propose an embedded CP selection mechanism that employs ROLS, a well-established RBF center selection strategy, for the selection of control points. We demonstrate that the proposed CP selection mechanism significantly improves the projection quality, measured by the stress, as shown in Equation (2.1), while requiring a reduced amount of control points when compared to other techniques.

Chapter 3

Control Points Selection for Multidimensional Projection

As presented in Section 2.5, the use of control points in multidimensional projection has become a valuable asset to both speeding up the projection process and providing user control for interactive data exploration. Even though the control points paradigm is gaining attention and being incorporated in many new MP techniques [Joia et al., 2011, Paulovich et al., 2011], the selection of instances that should be used as control points has not been thoroughly investigated. And yet, the set of control points play a central role in the quality of the final projection, as illustrated in Figure 3.1.

Recall from Figure 2.7, how the set of control points, together with their corresponding 2D position in the projection space, is used to approximate the projection of the remaining instances. A satisfactory approximation can only be achieved if the set of control points is composed of good representative samples of the entire dataset.

The works of [Pekalska et al., 1999], [de Silva and Tenenbaum, 2002] (L-Isomap), [de Silva and Tenenbaum, 2004] (L-MDS), [Paulovich et al., 2010] (PLMP) and [Joia et al., 2011] (LAMP), suggest to select control points randomly. On the other hand, works from [Paulovich et al., 2008a] (LSP) and [Paulovich et al., 2011] (PLP) make use of clustering techniques to divide the dataset into regions, and select one or more representative instances of each region as control points. When the dataset is uniformly distributed in the multidimensional space, a random selection of control points is a viable option. If that is not the case, i.e., the data distribution is not uniform, the random approach will likely fail to select a subset of samples that is a good representative of the entire dataset, resulting in poor-quality mappings. The clustering approach avoids this issue by introducing an additional step to the control point selection process. The dataset is first divided into clusters and, sub-

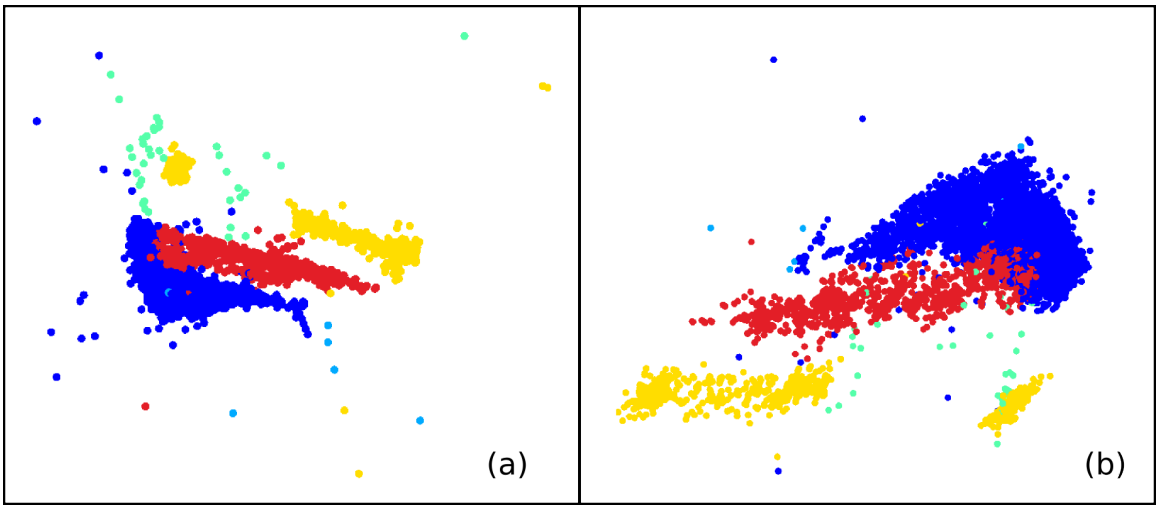


Figure 3.1: Shuttle dataset (see Table 3.1 for description) used to exemplify the influence of the set of control points in the final layout and quality of the projection. Both (a) and (b) depict the projection results achieved through PLMP technique [Paulovich et al., 2010]. The only difference between the two examples is the set of control points used in the approximation step. Two distinct sets with 20 control points each are used to calculate the projections. Projection (a) has a stress value of 0.8, while the stress of (b) is only 0.17. We can also observe how their layout differs, with (b) presenting a much more separated cloud of points than (a).

sequently, one or more instances from each cluster are randomly selected to form the set of control points. This strategy guarantees a better distribution of control points in comparison to the random approach, but it presents challenges of its own. It is not always clear what is the number of clusters in a dataset and most clustering techniques, like k-means, rely on this parameter to divide the dataset. Also, the clustering process can become computationally expensive. Furthermore, both approaches present the drawback of requiring the user to determine the number of samples to compose the set of control points, which may not be an obvious parameter to choose. Of course, one could apply a brute-force approach, and iterate between various subset sizes to decide which is ideal, in an automated fashion. This process, however, would prove very computationally expensive, as one would need to compute a multidimensional projection for each of the subset of control points, and also its stress to evaluate the projection quality.

Besides these limitations, imposed by the CP-selection strategy, most MP techniques require a minimum number of control points to ensure better projection quality. For example, the methods PLP, PLMP and LSP indicate that a minimum of $\sqrt{n}$ control points are necessary to produce a projection with low stress, n being the number of instances in the dataset. This limitation is mainly due to a lack of CP-selection strategy to better define a compact subset of samples capable of approximating well the overall dataset. A large number of control points create a less efficient high to low-dimensional mapping, and may not be ideal for applications with user intervention, as shown by [Joia et al., 2011].

In this thesis, I propose a novel multidimensional projection technique, built upon Radial Basis Function (RBF), with a built-in mechanism for control points selection. RBF presents a well-established mathematical formulation, which has been used in diverse approximation applications [Buhmann, 2003], and has been introduced in Section 2.6. I also propose to apply RBF to create an interpolation function that respects the low-dimensional position of previously projected control points and use that function to approximate the projection of the remaining instances. This method provides an explicit mapping from high to low dimensions, and allows one to incorporate new data in real time with little computational effort. The multidimensional projection technique introduced by [Pekalska et al., 1999] is a particular case of RBF. I generalize this traditional method and improve it by reducing the number of required control points.

In the proposed multidimensional projection, the problem of selecting CPs becomes the problem of selecting instances that will act as RBF centers. Various methods have been proposed to systematically choose a subset of samples to serve as RBF centers. One of such methods is based on the solution of Orthogonal Least Squares problems [Chen et al., 1991]. I incorporate this technique into the proposed Multidimensional Projection framework as a means to perform control points selection and to improve the final projection results. This approach automatically determines a good number of control points; thus, the user

is not required to provide this important parameter. Furthermore, the proposed technique can produce good-quality results with a reduced number of control points, which improves efficiency as well as favors user interactivity [Joia et al., 2011].

3.1 Multidimensional Projection with RBF

In this thesis, we propose a novel MP method that uses control points along with RBF approximation to create the mapping function $s(x)$. The main advantage of the proposed technique over the existing ones is the fact that a built-in mechanism for control points selection is provided, which improves the projection quality with respect to stress (Equation 2.1) while reducing the number of required control points. Recall that the stress is a metric that indicates how well original distances are preserved in the projection results. I will discuss the CP selection methodology later on, but first let us discuss some technical aspects of the proposed MP technique.

A natural way to formulate the MP problem using RBF is to consider control points as the RBF centers and their 2D position as the function values we want to interpolate (read Section 2.6 for a description of the RBF theory). The illustration presented in Figure 1.2 is a good reference to understand how these elements come together in the MP pipeline. Let $X_S = \{x_{s_1}, \ldots, x_{s_i}, \ldots, x_{s_k}\} \in \mathbb{R}^m$ be the set of control points and $Y_S = \{y_{s_1}, \ldots, y_{s_i}, \ldots, y_{s_k}\} \in \mathbb{R}^2$ be its 2D counterpart, i.e., y_{s_i} is the known 2D position of control point x_{s_i}. As discussed in Section 2.6, the interpolation with RBF functions requires data sites, or RBF centers, ξ_i and data values f_i. For the MP formulation, x_{s_i} are RBF centers and y_{s_i} are data values. The difference, in this case, is that the function values y_{si} are 2-dimensional vectors instead of scalar values. Thus, the proposed formulation presents two separate RBF functions, one

for each output dimension:

$$
\begin{aligned}
f_1(x) &= \sum_{i=1}^{k} \lambda_{1i} \phi(\|x - x_{s_i}\|), \\
f_2(x) &= \sum_{i=1}^{k} \lambda_{2i} \phi(\|x - x_{s_i}\|),
\end{aligned}
\tag{3.1}
$$

where λ_i are real coefficients; $\phi : \mathbb{R}^+ \to \mathbb{R}$ is the radial basis kernels. Note, however, that in this scenario we can solve the interpolation problem component-wise, as the interpolation matrices of both functions are the same. A linear system equivalent to the one in Equation (2.9) is written as

$$
\Phi \Lambda = Y, \tag{3.2}
$$

where

$$
\Phi = \begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1k} \\ \vdots & \vdots & \vdots & \vdots \\ \phi_{k1} & \phi_{k2} & \cdots & \phi_{kk} \end{bmatrix}, \qquad \Lambda = \begin{bmatrix} \lambda_{11} & \lambda_{21} \\ \vdots & \vdots \\ \lambda_{1k} & \lambda_{2k} \end{bmatrix} \qquad \text{and} \qquad Y = \begin{bmatrix} y_{s11} & y_{s21} \\ \vdots & \vdots \\ y_{s1k} & y_{s2k} \end{bmatrix},
$$

,

where $\phi_{ij} = \phi(\|x_{si} - x_{sj}\|)$. We solve the linear system only once in the process by factorizing Φ. There is a vast array of techniques designed to solve linear systems. The interpolation matrix Φ is always symmetric (since $\phi_{ij} = \phi_{ji}$) and, depending on the choice of kernel ϕ, it can be positive-definite. In such cases, the Cholesky factorization is a good choice. Otherwise, general factorizations such as LU or QR can be used [1] [Golub and Van Loan, 2012].

The proposed RBF methodology is illustrated in Figure 3.2 through a simpler 2D to 1D projection example.

[1] The linear solvers used in this thesis are from the LAPACK library [Anderson et al., 1990].

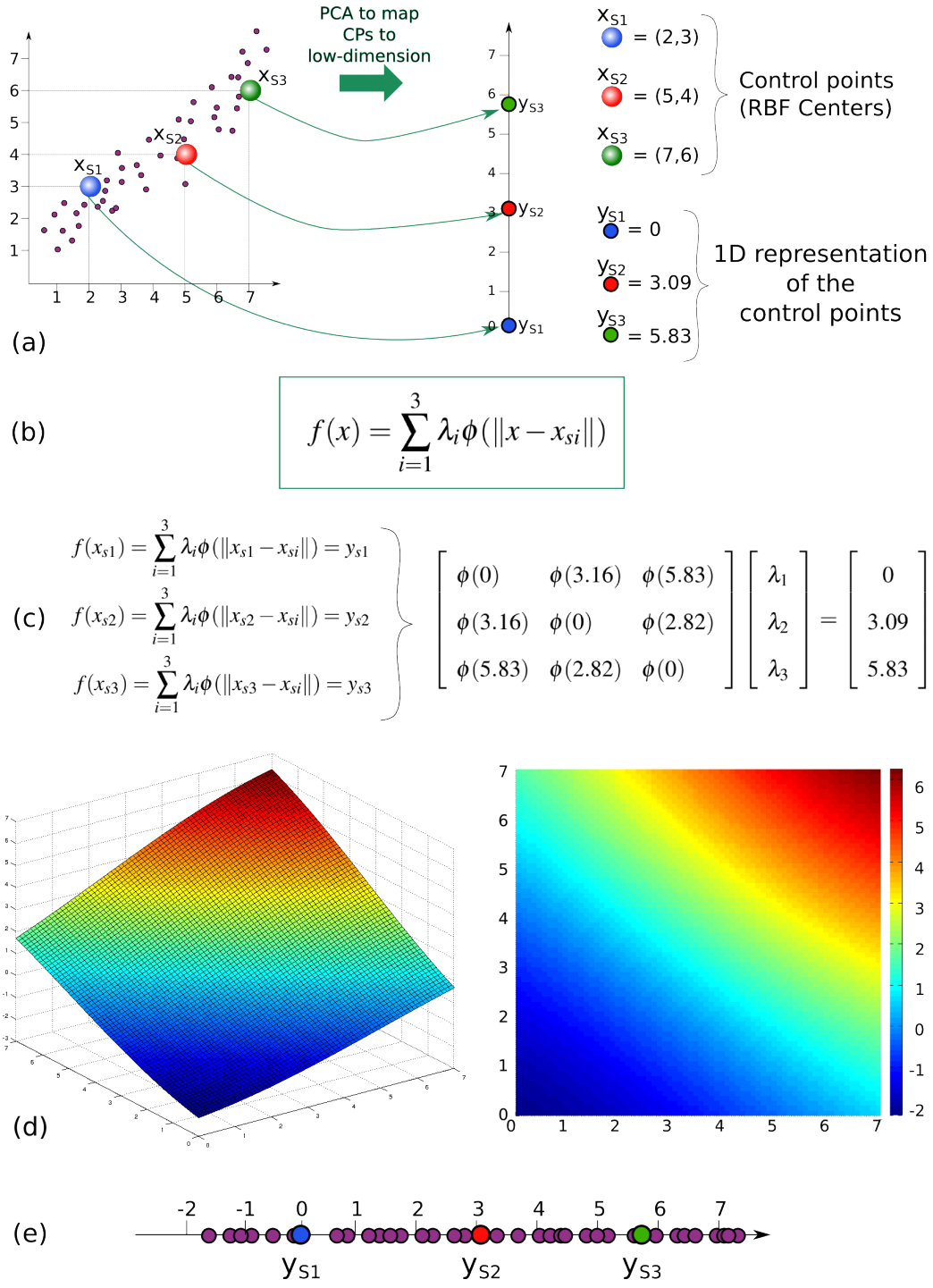


Figure 3.2: RBF Projection example. (a) A 2D dataset is given, and three control points are chosen; the CPs are projected into 1D through PCA (see Section 2.3.1). (b) An RBF function is sought to approximate the projection of the remaining instances. (c) The RBF coefficients λs are determined by solving a linear system, in such a way that the CPs output positions are interpolated. (d) The RBF function for this example, using a Gaussian kernel $\phi(r) = e^{-(\varepsilon r)^2}$, with $\varepsilon = 0.1$. (e) Applying the function to every instance in the dataset approximates their position in the projection space.

Some aspects of RBF makes it a suitable and appealing interpolation method for multi-dimensional projection:

1. ***A good approximation of multivariate functions.*** As stated by Martin Buhmman in his book entitled *Radial Basis Functions* [Buhmann, 2003], the RBF approach is well suited for cases with large datasets with multiple attributes.

2. ***Operation upon the distance between a point in the domain and the RBF center directly.*** Even though the most commonly used distance metric is the Euclidean, in practice, one could use different dissimilarity measurements not necessarily defined in a Cartesian space. For instance, [Cox and Cox, 2000] presents an example where the dataset consists of 10 whisky bottles from different distilleries. The dissimilarities between each pair of them is an integer score between zero and ten given by an expert judge. Note that, in this specific example, instances are not necessarily defined in the Cartesian space, but dissimilarities are provided. The proposed technique can handle such scenarios, making it more flexible than recent multidimensional projection methods, which require the original dataset to be embedded in a Cartesian space [Paulovich et al., 2010, Joia et al., 2011].

3. ***No constraints over number of control points.*** The number of centers is not a restriction in RBF, i.e., the MP mapping function can be created with a very small-sized set of control points. This characteristic is an advantage over some MP techniques, for instance, the PLMP method [Paulovich et al., 2010], that requires the number of control points to be larger than the number of parameters in the dataset. It was previously discussed the importance of a reduced number of control points in applications with user interaction.

4. ***Center selection.*** This aspect is an important topic in RBF research, and different approaches have been proposed to determine a suitable subset of representative samples to be used as centers, the correspondent to our control points. Techniques based on cluster-

ing [Uykan and Guzelis, 1997], genetic algorithms [Zhao et al., 2002] and Orthogonal Least Squares [Chen et al., 1991, Chen et al., 1996] are among the most popular. The methodology applied in this work is based on the former, and we believe it provides a good starting point to investigate the impact that a careful selection of CPs can have in the quality of MP results.

In the next section, I discuss the Regularized Orthogonal Least Squares (ROLS), the method used to select representative control points.

3.2 Control Points Selection through Regularized Orthogonal Least Squares

The set of control points is an essential part of the RBF technique, having a significant impact on the quality of the RBF approximation and, in the case of this thesis, in the quality of the final projection. Ideally, the set of control points should represent well the entire dataset domain and still be small sized and with low redundancy. In this thesis, I propose to employ a method based on Orthogonal Least Squares (OLS), introduced by [Chen et al., 1991] for center selection in RBF. OLS is a well-established methodology, and its mathematical formulation is derived exclusively from RBF approximation/interpolation. It is a deterministic approach, unlike other methodologies such as the one presented in [Zhao et al., 2002], and it was the natural choice to start exploring with control points selection. In general terms, the proposed CP selection mechanism applies an iterative forward-selection strategy, whose process starts with an empty set and, at each iteration, the instance that reduces the RBF approximation error the most is added to the set. This process is repeated until stop criteria are met. Figure 3.3 illustrates the pipeline of the proposed RBF methodology with embedded control points selection.

To understand how OLS works for control points selection, it is important to view RBF

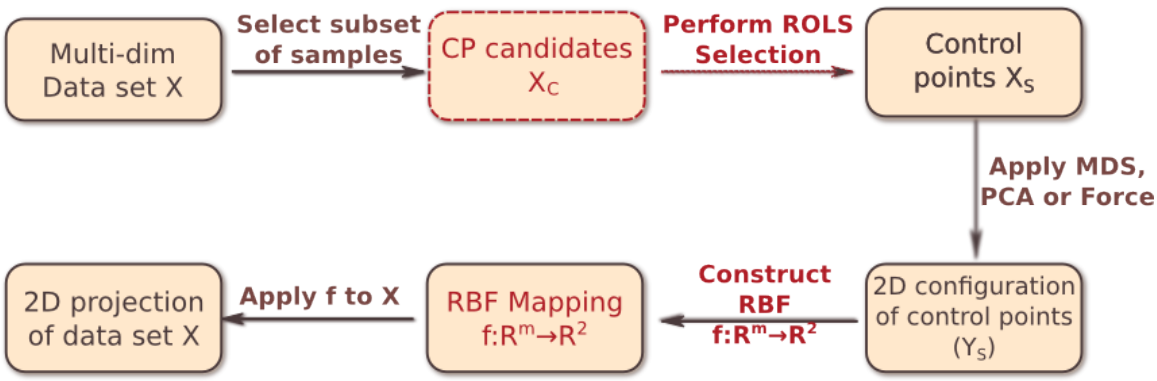


Figure 3.3: RBF projection with embedded CP selection. This scheme highlights in red the extra steps when compared to the general pipeline presented in Figure 2.7.

as a linear regression [2] model [Rencher and Christensen, 2012]. Let us assume we have K control points *candidates* $\{x_i, y_i\}_{i=1}^{K}$, where y_i is the output corresponding to control point x_i. If all x_i are used as control points, Equation (2.6) can be rewritten as:

$$s(x_t) = \sum_{i=1}^{K} \lambda_i \phi(\|x_t - x_i\|), 1 \leq t \leq K. \tag{3.3}$$

Let

$$\phi_i(t) = \phi(\|x_t - x_i\|), \tag{3.4}$$

we can express the desired output y_t as

$$y_t = \sum_{i=1}^{K} \lambda_i \phi_i(t) + e_t, 1 \leq t \leq K, \tag{3.5}$$

where $e(t)$ is the error between the desired output y_t and the approximated output $s(x_t)$, i.e., $e(t) = y_t - s(x_t)$. Note $e(t)$ will be zero when all candidates are used as control points, but the goal of the method is to reduce this set. Finally, we can write Equation (3.5) in matrix form as

$$\mathbf{y} = \Phi\lambda + \mathbf{e}, \tag{3.6}$$

[2] In general terms, linear regression is an approach to model the relationship between the output and the data samples in a linear fashion.

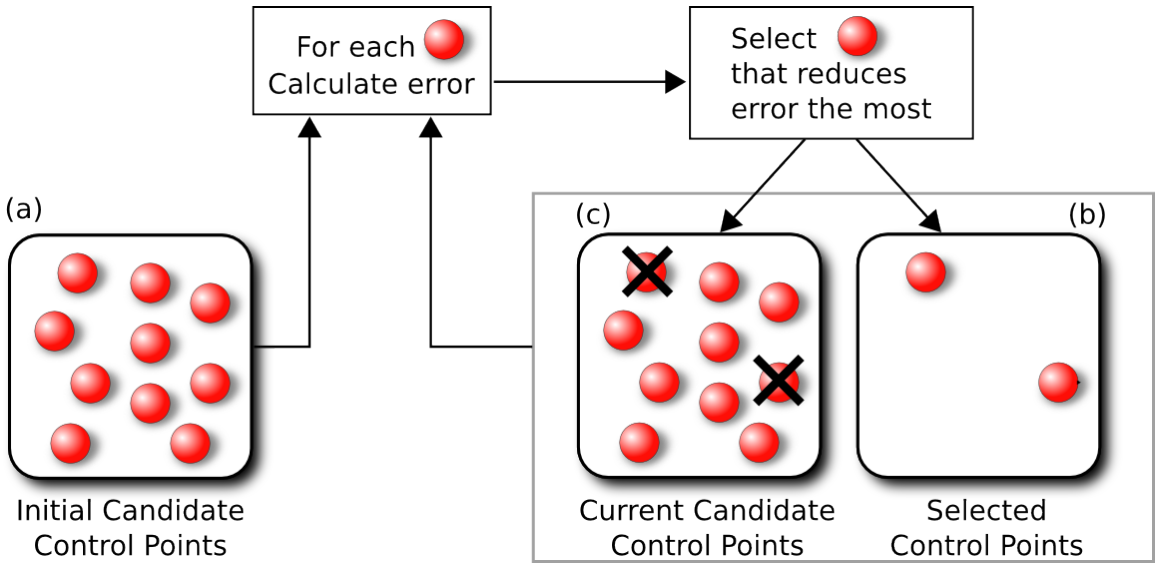


Figure 3.4: Overview of the CP selection mechanism. Starting with a set of candidate control points (a), calculate the approximation error, assuming each CP alone is used as RBF center. The candidate that reduces the error the most is included in the set of control points (b) and removed from the set of candidates (c). In this diagram, we show two complete iterations, where two CPs have been selected. This process is repeated until the stop criteria are met.

.

where $\mathbf{y} = [y_1 \dots y_K]^T$, $\Phi = [\phi_1 \dots \phi_K]$, $\phi_i = [\phi_i(1) \dots \phi_i(K)]^T$, $\lambda = [\lambda_1 \dots \lambda_K]$ and $\mathbf{e} = [e_1 \dots e_K]$.

Equation (3.6) has the form of a linear regression model and the vectors ϕ_i can be referred to as regressors. Thus, the question of how to select control points can be translated into the problem of selecting significant regressors. The adopted technique is based on a "forward selection", i.e., the process starts with an empty set of regressors and one regressor from the set of candidates is selected at a time. Each selection is made in such a way to maximally decrease the squared error $\mathbf{e}^T\mathbf{e}$.

Since the regressors are usually correlated, it is not clear how to measure their individual contributions to the error decrement. Applying the concept of the OLS method, which transforms the set of ϕ_i into a set of orthogonal basis vectors, it is possible to *isolate* the re-

gressors and calculate their individual contributions. Using QR factorization, the regression matrix Φ can be decomposed as

$$\Phi = WA, \tag{3.7}$$

where A is an upper-triangular matrix with diagonal 1 and $W = [w_1 \dots w_K]$ with orthogonal columns that satisfy $w_i^T w_j = 0$, if $i \neq j$. The model (3.6) can now be written as

$$\mathbf{y} = Wg + \mathbf{e} \tag{3.8}$$

with $A\lambda = g$.

However, as discussed by [Chen et al., 1996], the minimization of the squared error $\mathbf{e}^T\mathbf{e}$ alone may become prone to overfitting, i.e., although the approximant $s(x)$ interpolates the control points candidates, it may not be able to generalize well to the remaining instances of the dataset - what could result in poor-quality projection results. To prevent this problem, a regularization term is added to the error, which penalizes large λ values. Observe that, since $A\lambda = g$, penalizing λ is equivalent to penalizing g; thus, the final formulation for the error we aim to minimize is:

$$\mathbf{e}^T\mathbf{e} + \beta g^T g, \tag{3.9}$$

where $\beta \geq 0$ is the regularization parameter. This error formulation renames the technique to *Regularized* Orthogonal Least Squares (ROLS). Equation (3.9) can be rewritten as

$$\mathbf{e}^T\mathbf{e} + \beta g^T g = \mathbf{y}^T\mathbf{y} - \sum_{i=1}^{K} (w_i^T w_i + \beta) g_i^2. \tag{3.10}$$

Dividing (3.10) by $\mathbf{y}^T\mathbf{y}$ we have

$$\frac{(\mathbf{e}^T\mathbf{e} + \beta g^T g)}{\mathbf{y}^T\mathbf{y}} = 1 - \frac{\sum_{i=1}^{K} (w_i^T w_i + \beta) g_i^2}{\mathbf{y}^T\mathbf{y}}, \tag{3.11}$$

and the regularized error reduction ratio due to w_i is defined as

$$[rerr]_i = \frac{\sum_{i=1}^{K}(w_i^T w_i + \beta)g_i^2}{\mathbf{y}^T\mathbf{y}}. \tag{3.12}$$

At each step of the selection, the control point x_i associated with vector w_i and maximum *rerr* is included in the control points' set.

In this work, I also propose the inclusion of an extra step, which is the calculation of the stress metric, given by Equation (2.1), of the remaining CP candidates. It is clear that using all candidates as control points reduces the error *rerr* (3.12) at most, but not necessarily the stress. The goal is to select a limited amount of points that better explains the entire dataset and potentially reduces the stress. This objective is achieved by introducing a stop criterion in the selection process: (1) Akaike-type criteria reaches a minimum, as suggested in [Chen et al., 1991] and (2) a maximum number of control points is reached. At the end of the selection process, the iteration with minimum stress is identified. As a final step, we seek a trade-off between the minimum stress and the reduced number of control points, by finding an iteration with fewer control points than the one with minimum stress, but with stress slightly higher. Heuristically, I found in the experiments that stress values up to 5% higher yield a good trade-off between number of control points and projection quality. This threshold was defined through a trial-and-error process, where various thresholds were tested with regards to its final stress and size of control points' set.

To prevent ill-conditioning [3], a simple check can be built into the procedure. The relation $w_i^T w_i = 0$ implies that ϕ_i is a linear combination of the previously selected control points. Thus, if $w_i^T w_i$ is less than a preset threshold γ, ϕ_i will not be selected as a control point.

The ROLS technique for control points selection is summarized in Algorithm 1. Note

[3] An ill-conditioned matrix is one that has a very large conditional number, which indicates the matrix is almost singular. In such case, the solution of the linear system is prone to significant numerical errors.

Algorithm 1 Control points selection with ROLS

Given $X_c = x_1, \ldots x_K$ (set of control points candidates);
Given $Y_c = y_1, \ldots y_K$ (candidates' position in the projection space);
Construct matrix $\Phi = [\phi_1 \ldots \phi_K]$, $\phi_i = [\phi_i(1) \ldots \phi_i(K)]^T$ (Equation 3.4);
$W = \Phi$
$X_0 = \varnothing$
$it = 1$
while Stop-criterion not met **do**
    for each candidate i where $w_i^T w_i > \gamma$ **do**
        Calculate $[rerr]_i(w_i)$ (Equation (3.12))
    end for
    $w_k = \arg\max([rerr]_i)$
    $X_{it} = X_{it-1} \cup \{x_k\}$ (Select x_k as a new control point)
    Remove w_k from W
    Calculate stress for instances in W
    for each $w_i \in W$ **do**
        $w_i =$ Orthogonalization of w_i with respect to w_k through Modified Gram-Schmidt [Chen et al., 1996]
    end for
    $it = it + 1$
end while
Find iteration it with minimum stress (s_{min})
$X = X_i$, with i the iteration with less CPs and $stress < 1.05 * s_{min}$
Return X

that the orthogonalization of matrix Φ is done step by step until a stop criterion is met and the selection process is terminated. The orthogonalization process is acquired through the *Modified Gram-Schmidt* algorithm [Chen et al., 1996].

Even though the first step in the proposed process is to select K candidates for control points randomly and use a force-based technique to calculate their low-dimensional positioning, which an extra computational overhead, as soon as the number of control points is reduced the RBF becomes acutely simple. Moreover, we attested that, in most cases, the projection quality indicated by the stress (Equation (2.1)) significantly improves when this careful selection process is adopted. In fact, Figure 3.5 experiments with three datasets: Page Blocks, Ionosphere and Yeast; with RBF using a Multiquadrics kernel ($c = 1$ and $\varepsilon = 30$). The green boxplots present the stress for random selection of control points, while the pink boxplots present the results for ROLS selection. Since there is a randomness associated with the experiment, each test was performed 100 times, and the stress results were

calculated for each of them.

The ROLS parameters used are: 30 maximum number of final control points and $\gamma = 1.e-5$, selected heuristically. The average number of control points selected is shown in the row *#FCP*. Observe that the stress achieved with ROLS selection with a few control points (less than 30) is lower than the ones obtained with a larger number of randomly selected control points. Another interesting fact to observe is that, increasing the number of candidates N, the chances of picking more meaningful control points are increased, and the stress is improved, but there is a tradeoff between projection quality and computational time. In the next Section, I present an evaluation of the proposed RBF projection and comparison with other techniques.

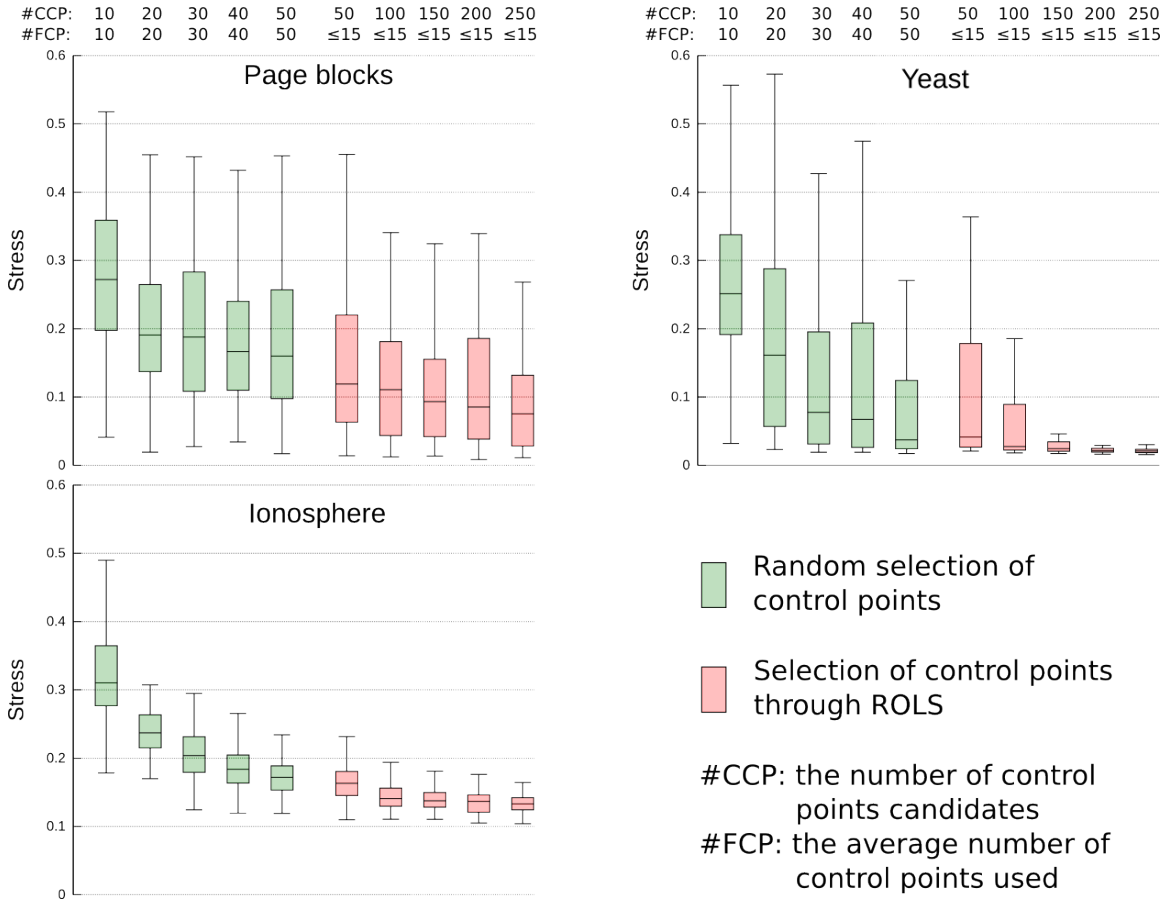


Figure 3.5: Impact of CP selection with ROLS in the MP stress (Equation (2.1)), for three datasets.[4]In ROLS results, *#FCP* indicates the average number of control points selected in the experiments. Each test was executed 100 times, and the stress distribution is represented through boxplots. Note how the average stress value of the experiments with ROLS CP selection (pink boxplots) is smaller than the ones that use pure random selection, even when the number of CPs used is considerably less than in the baseline experiments (green boxplots).

[4]Box plots are used throughout this thesis to provide a visualization of the distribution of metric values for a given set of experiments. To provide guidance for their interpretation, the description of the graphs used in this thesis are presented below. The tops and bottoms of each box are the 25th and 75th percentiles of the data, respectively. The line in the middle of each box is the median of the data. The lines extending above and below each box extend up to the maximum and minimum data value, respectively. In case of outliers, they are represented as circles above/below these lines.

3.3 Evaluation of Technique

The experiments presented in this section were executed in a 2.80 GHz Intel Core i7 CPU 860 with 12 GB of RAM. I compare the proposed technique to four other methods, namely LAMP [Joia et al., 2011], PLMP [Paulovich et al., 2010], Pekalska [Pekalska et al., 1999] and Fastmap [Faloutsos and Lin, 1995]. The first three techniques are control-points-based. All of these techniques propose a random strategy to select control points and, with exception of LAMP, suggest a minimum number of control points to ensure projection quality, based on experimentation. The Fastmap technique, in turn, is known for its computational efficiency as its methodology is more focused on computational speed than in projection accuracy.

For the RBF setup in these experiments, I adopted the Multiquadrics kernel, with $c = \varepsilon = 1$. For the control points selection process, I used the following parameters: number of candidates $N = 150$, the maximum number of control points 30 and $\gamma = 1.e-5$ to avoid matrix ill-conditioning. The datasets used in the experiments are described in Table 3.1.

Dataset	**# Instances**	**# Dimensions**
Ionosphere	350	35
Pima indians diabetes	768	8
Yeast	1152	8
WDBC	569	30
Segmentation	2085	16
Wine Quality	3961	11
Page blocks	5405	10
Letter Recognition	18667	16
Shuttle	42365	8

Table 3.1: Datasets used in the experiments, downloaded from the UCI Machine Learning Repository [Bache and Lichman, 2013].

For each dataset 100 tests were executed, to account for the variance introduced in the results due to the random factors of the techniques (except for Fastmap, which is a deterministic approach). The methods were compared in terms of mapping quality (stress - Equation (2.1)) and execution time. The results are shown in Figure 3.6, and the experi-

ments are discussed in Section 3.4.

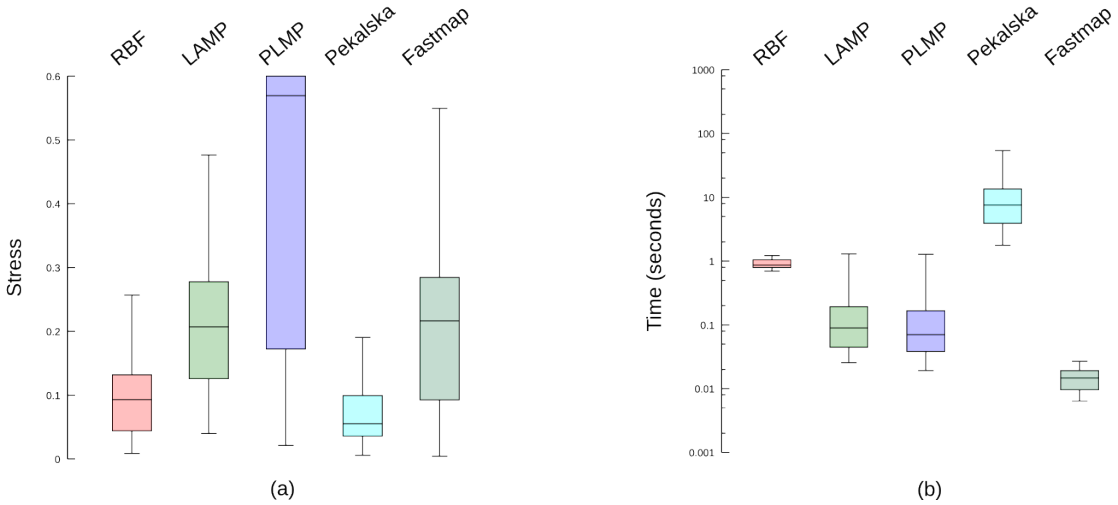


Figure 3.6: Stress and time comparison for RBF, LAMP, PLMP, Pekalska and Fastmap. The proposed RBF technique was only outperformed by Pekalska in terms of stress, but it performs better in terms of computational efficiency.

3.4 Discussion

Figure 3.6(a) presents the comparison of projection quality between the RBF approach and the other four techniques. The y-axis was truncated for $stress = 0.6$. We can see that the stress achieved by the proposed method is very low and outperforms all the other methods, except Pekalska, which present stress only slightly smaller. Notice, however, that Pekalska requires a large number of control points $\sqrt{n}$(where n is the number of instances in the dataset) to produce good-quality results, On the other hand, the proposed method used a maximum of 30 control points in every test case, and an average of 15 control points for the datasets presented in Table 3.1.

Figure 3.6(b), in turn, shows the boxplots with the time variance for each technique. The results indicate that the overhead created by the procedure for control points' selection gives a small disadvantage to RBF compared with LAMP, PLMP and Fastmap. However, it was observed that the execution time was around 1 second for the current experiments,

which is small in practical terms. Moreover, the created overhead is only caused by the control points' selection process, which depends on the number of candidates N chosen by the user. The experiments indicate that 150 control points candidates are usually enough to yield good quality projection results, independent of the size of the dataset. Furthermore, this only creates a lower bound for computational time, since the control points selection is accomplished in a pre-processing step, and the final RBF process with only a few control points is extremely fast.

Regarding ROLS for control points selection, the results achieved with this technique were very satisfactory. The proposed CP selection strategy was applied to all 9 datasets presented in Table 3.1 (a reduced number of results was presented in Figure 3.5) and the stress produced by less than 30 control points was always equal or better than the ones produced by 50 randomly selected control points. Figure 3.7 presents an illustration of how the ROLS-based control points selection work in the *Mammals* dataset, which contains data that characterizes dogs, cats, horses and giraffes, forming four well-separated clusters [A. Asuncion, 2007]. Figures 3.7(a) and 3.7(b) present 150 randomly selected and 15 ROLS-selected control points, respectively. We can observe that the method automatically selects representatives of each cluster, maintaining the general behavior of the projection (results in the top row).

3.5 Control Points Selection Applied to Other MP Techniques

To investigate even further the effects of control points selection, I have applied ROLS to other MP techniques. The methodology for control points selection proposed in this thesis is intrinsically connected to the RBF formulation. However, the basic idea of ROLS is to find points in the domain that better represent the entire dataset, and this selection could benefit and improve the solution of other techniques. In fact, we have evaluated the performances of LAMP and PLMP with ROLS-based control points selection and compared

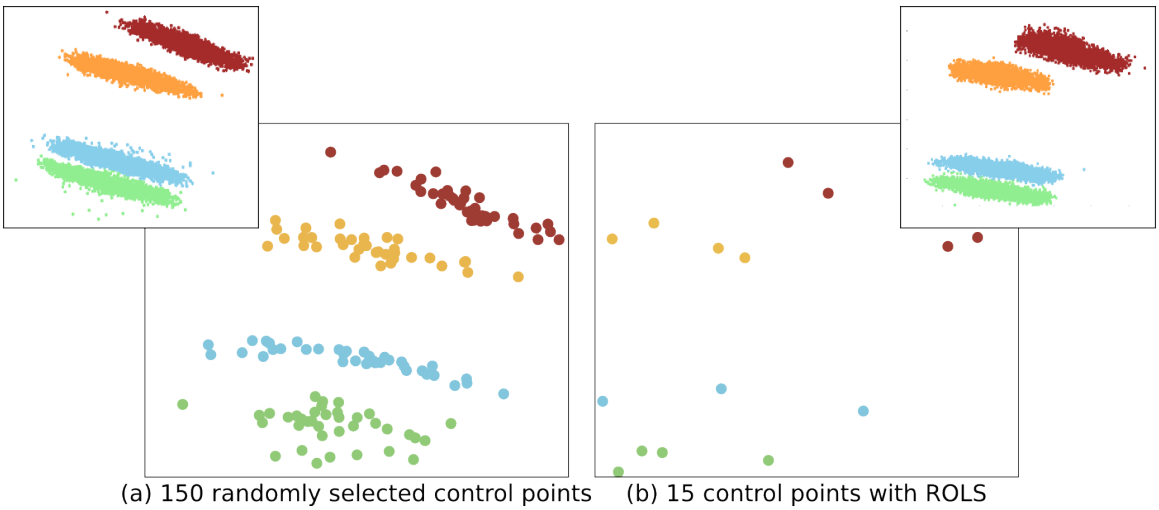


Figure 3.7: Example of control points selection through ROLS in the Mammals dataset (20,000 instances and 47 dimensions). This dataset contains four well-defined clusters, indicated by the different colors in the projection. Figures on the bottom present only the control points used to produce the projection, depicted on the top.

them against their performances with random selection. Preliminary results are encouraging and indicate that ROLS-based control points selection can improve the projection quality of these techniques.

Similarly to the RBF projection experiments (Figure 3.5) LAMP and PLMP were both executed with 10, 20, 30, 40 and 50 randomly selected control points for each dataset; and we have employed control points selection using 50, 100, 150, 200 and 250 initial CP candidates. Each experiment was run 100 times, to account for the stochasticity of the method.

Some of the results evaluating the projection stress of LAMP are presented in Figure 3.8. It is very clear from the graphs that LAMP can have its projection quality substantially improved when combined with the proposed control points selection mechanism. Note that ROLS returned no more than 15 control points for each of the test cases I present. Even using such a small amount of control points, the final projection stress is lower than the ones that resulted from using a larger number of control points chosen randomly. In fact, we can observe that using ROLS-selection the stress median is almost always smaller that

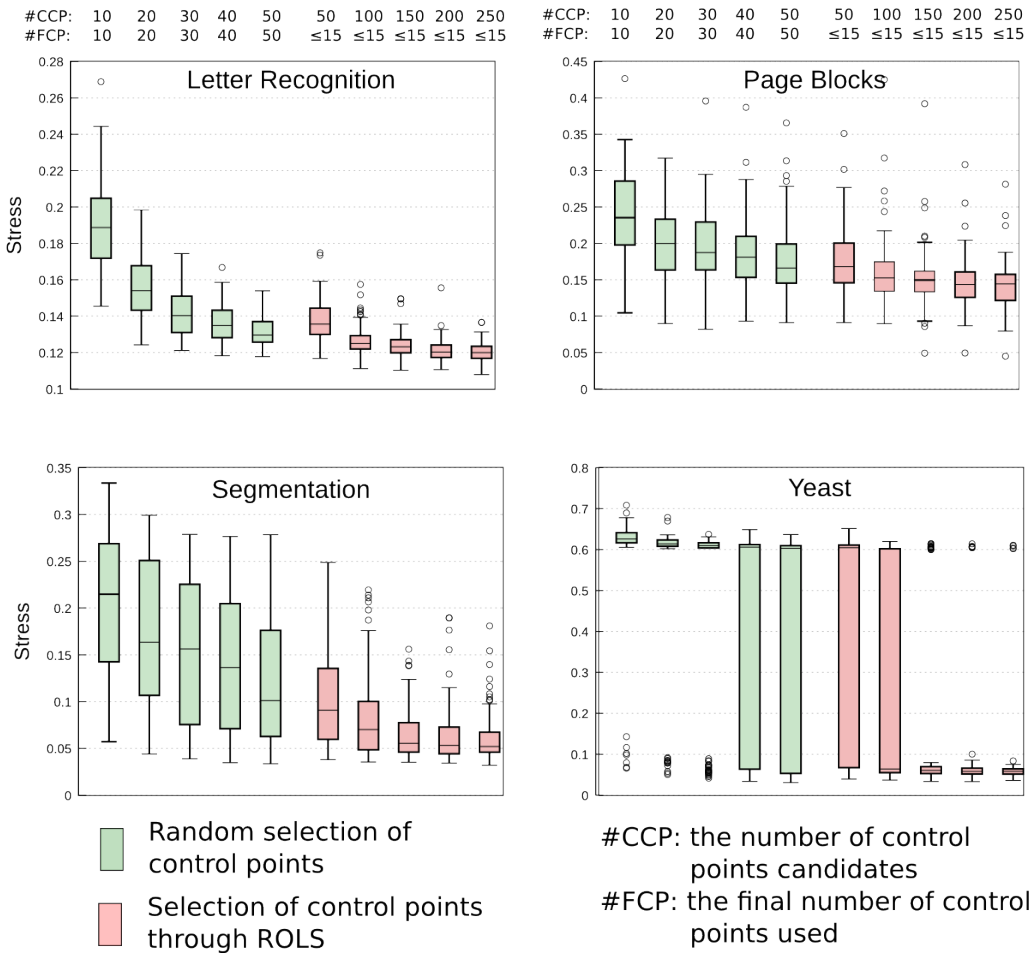


Figure 3.8: Applying ROLS-based control points selection to LAMP. It is evident from the graphs that a careful selection of control points can significantly improve the iLAMP projection stress.

this value using 10-50 randomly selected control points. Notably, a drastic improvement in projection quality can be observed in the *Segmentation* and *Yeast* datasets.

Note that, in the *Segmentation* test case in Figure 3.8, using 15 or fewer control points selected from 150 candidates reduced the stress median close to 50% when compared to 50 random control points. In the *Yeast* experiment, in turn, it was observed that random selection produced projections with very high stress, with median greater than 0.6 in most of the tested cases. Using ROLS-selection with 100 candidates we already see a significant reduction in the stress median to less than 0.1. Using less than 15 control points selected from 100, 200 or 250 candidates, this very small stress level was maintained for all 100

runs, except for a few outliers.

In the PLMP experiments, we can observe a similar behavior since ROLS-selection helps to reduce the overall stress of the projection while keeping the number of control points low. There are a few things to consider about PLMP before discussing the results: (1) Due to its formulation, PLMP technique requires the number of control points to be greater than the number of parameters m of the dataset. Thus, in the experiments I ensured that ROLS would return at least $m+1$ control points when coupled with PLMP; and (2) in the paper describing PLMP [Paulovich et al., 2010], the performance in terms of stress was evaluated using a clustering-based and a random-based control points selection approach. Through these experiments, it was concluded that the clustering-based approach only outperformed the random method when the number of control points was low. However, to achieve satisfactory stress values with either approach, the number of control points has to be increased (the authors suggest $\sqrt{N}$ control points). In such cases, clustering and random yielded similar stress results, which justified the use of random selection in their work.

Figure 3.9 presents the results of *Yeast* and *Page Blocks* datasets obtained when incorporating ROLS-based selection into the PLMP workflow. In order to compare the stress value to the state-of-the-art of the method, we have also included the results achieved with $\sqrt{N}$ random control points as suggested by PLMP. In both test cases, we can observe a considerable reduction in stress when using ROLS-selection.

Note, for instance, the drastic decrease in stress achieved by selecting an average of 23 control points out of 150 or 200 candidates in the *Yeast* example. The projection quality outperforms, by far, even the experiments using the suggested $\sqrt{N}$ (in this case 40) control points. Similarly, the stress in the *Page Blocks* experiment is much reduced when using the proposed selection mechanism. Note that using 25 control points selected out of 200 candidates; the overall stress also outperforms the stress using $\sqrt{N}$ (in this case 75) control points.

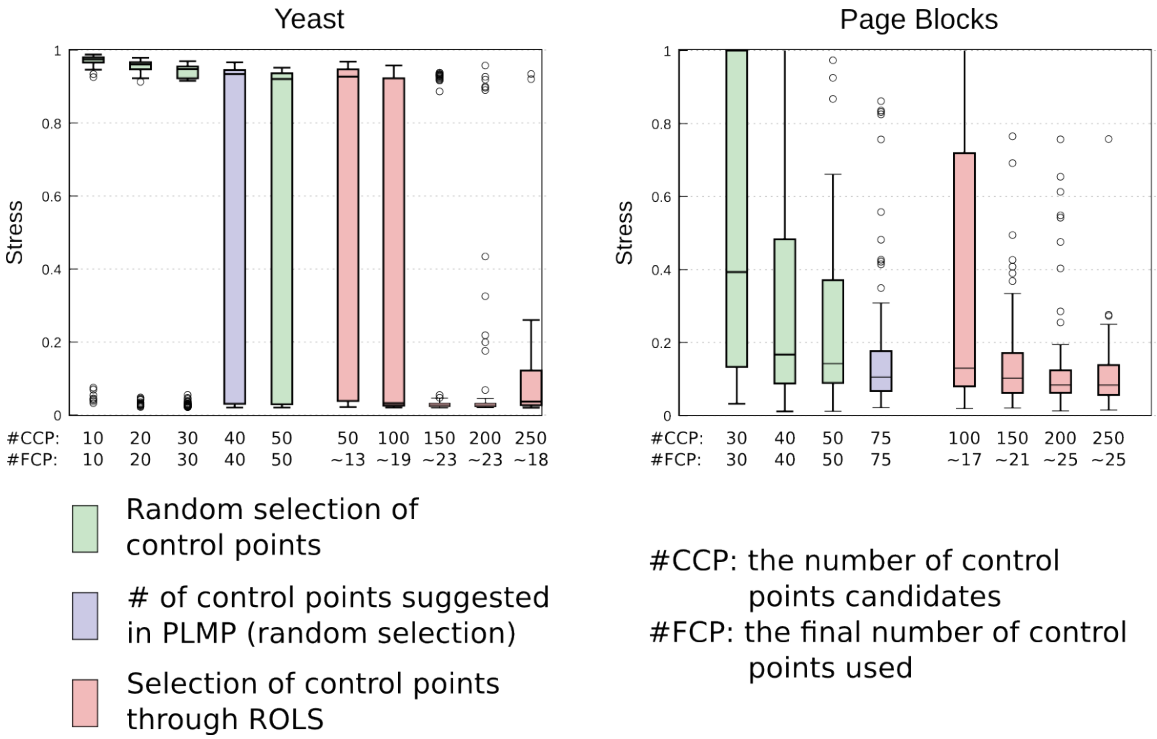


Figure 3.9: Applying ROLS-based control points selection to PLMP. These results indicate that PLMP can be greatly improved when combined with a careful CP selection mechanism, such as the proposed ROLS.

In summary, this chapter presented a novel multidimensional projection technique, based on RBF. The main advantage of the proposed method is its ability to perform control points selection, a step generally not present in most MP techniques. Experiments indicate that the ROLS-selection method offers a good trade-off between time and stress while reducing the number of control points. It was also shown that the proposed control points selection strategy could benefit techniques other than RBF-based, such as LAMP and PLMP.

Chapter 4

Inverse Projection

So far, I have focused on the classic definition of multidimensional projection (MP) and its application for data analysis and visualization. In Chapter 3, I presented a new multidimensional projection technique with a built-in mechanism for control points selection and demonstrated that this method could improve the projection quality even with a low number of control points. Another significant contribution of this thesis, however, lies beyond this traditional usage of MP techniques, in a process called *inverse projection*. Inverse projection was designed with the goal of providing a simple and easy-to-use way of creating new multidimensional data samples. This work was mainly inspired by the possibility of establishing a 2D point and having a tool to automatically infer the position of this point in the multidimensional space. I propose to use inverse projection to achieve this goal, by using the MP layout as a canvas, on which new points can be created amongst an existing projection scatter plot. Figure 4.1 illustrates the inverse projection framework.

The MP layout space provides a good medium on which to build the proposed framework: the neighborhood structure of the original dataset is preserved in the 2D projection space, and we know the exact correspondence between projected 2D points and multidimensional instances. This information can be leveraged to derive a reasonable assumption about the multidimensional position of a newly-defined 2D point. Generating a multidimensional sample can be a cumbersome task, due to either the large number of pa-rameters or to the lack of knowledge about them. On the other hand, with the inverse projection approach, the parameter information can be presented in an abstract fashion to the user, who would only need to decide in 2D the neighborhood in which he/she wants the new sample to be located.

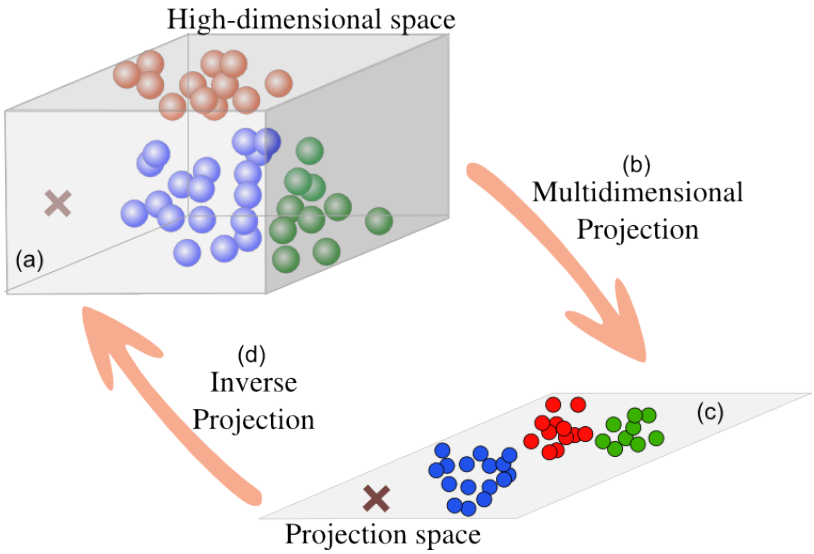


Figure 4.1: Given a multidimensional dataset (illustrated by the colored circles inside the cube), a multidimensional projection maps the data into a projection space. Inverse projection operates in the other direction. With any input device, the user interactively defines a new 2D point in the projection space (illustrated by the cross), which is then mapped back into the multidimensional space (cross inside the cube). MP provides two essential data for inverse projection: the neighborhood structure preserved in the projection space and the correspondence between 2D projected points and the multidimensional instances.

Inverse projection can be used in applications where data analysis is necessary, but creating new samples, i.e., extrapolating the actual samples of a given dataset, is also desirable. In this thesis, I present two such applications with demonstration examples: (1) A parameter space exploration using an optimization problem (Chapter 5); (2) A face-synthesis application (Chapter 6).

The next section presents an overview of the existing methods in multidimensional parameter exploration and how they compare to the proposed inverse projection framework.

4.1 Multidimensional Parameter Space Exploration

The exploration of multidimensional parameter spaces has been a recurring research topic in visualization in the past few years. For instance, systems like the ones presented by [Bruckner and Möller, 2010] and [Pretorius et al., 2011] have been designed to assist the

user to perform this task. [Bruckner and Möller, 2010], introduced a system that allows the user to explore simulation results for special effects. The focus of their system is not to provide a complete understanding of the parameter space but to enable users to achieve the desired animation sequence without the hurdle of parameter tuning. Therefore, the system is not designed to support resampling of the parameter space. Likewise, in [Pretorius et al., 2011], the authors proposed an interactive visualization technique that enables users to analyze the relationship between sampled input parameters and corresponding outputs. Their system is designed to assist users in the task of parameter tuning for image analysis. The first step in their pipeline is to sample the parameter space uniformly. The outputs for each parameter combination are calculated and displayed in such a way that users can understand the input-output relationship. Their system also does not permit resampling the parameter space during the process. In fact, just a few techniques and systems allow for resampling as part of the data exploration process. [Wang et al., 2013], propose a system that allows the user to create and edit multidimensional data points from an existing dataset or from scratch. They use traditional visualization tools, as parallel-coordinates and scatterplot matrices, to create an interface that permits the user to sketch the distribution of new points. In another work, [Torsney-Weir et al., 2011] present a system that assists the user in resampling a continuous parameter space. Their system is designed to help users to find a suitable parameter combination for segmentation methods. They start by automatically sampling the parameter space, and then they evaluate each parameter combination based on multiple-objective optimization functions. Scatterplot matrices and Hyperslice are used to display the samples and the corresponding function values. Their interface guides users in the search for regions of interest of the parameter space while permitting to resample user-defined areas. The resampling mechanism, however, requires the user to set each parameter manually and the system is not designed to deal with a high number of parameters. [Igarashi et al., 2005] propose a system that supports performance-driven animation by

blending input poses of an object using RBF interpolation. In their application, the user defines poses and associates each of them with a point in the 3D space, and a user-defined motion path in 3D creates a corresponding character animation. Their system is specifically designed for animation of 3D characters, while my proposed inverse projection is a general framework that can be applied to any multidimensional dataset. Furthermore, multidimensional projection is a key component of my framework, as it automatically provides a coherent 2D correspondence for each of the objects. This renders our framework scalable with regards to the number of objects in the dataset, since the user is not required to manually create these correspondences in the 2D space.

In inverse projection, since data analysis and extrapolation are carried out in the 2D projection space, the user is not necessarily aware of the number of parameters in hand. The idea is to use the projection of the data to identify regions of interest and generate new samples in these areas. The basic premise of inverse projection is that both a multidimensional dataset and its 2D projection are provided.

In the next section, I will present an overview of inverse projection and the main components necessary for its application.

4.2 Interactive Inverse Projection Framework

The proposed inverse projection is an interactive methodology that allows users to create new multidimensional instances in a straightforward manner. It all starts with a multidimensional dataset, which is projected into a 2D projection space. Besides being the user interface where data analysis takes place, the projection space also provides the interactive medium over which the user can create new multidimensional samples. The user creates 2D points with any pointing device (e.g., mouse, stylus) in regions of interest over the projection space. This area may be either under-sampled, i.e., few or none projected points, or sampled with some projected points of interest. In the latter, the user can further explore

the dataset and generate additional instances sharing similarities with the existing ones. For each user-defined point, the inverse projection finds a multidimensional representation for it using a mapping function.

One may generalize the main components of the interactive inverse projection framework as follows. (a) A multidimensional dataset used as the basis for parameter exploration. (b) A mapping function used to project these multivariate samples onto a projection space. (c) An interactive projection area allowing the user to rearrange the projected data and to create points in it. (d) An inverse mapping function used to map user-created 2D points into the multidimensional space. This framework is illustrated in Figure 4.1.

Two basic premises need to hold true when using the proposed framework for a given dataset: (1) The instances of the dataset can be represented as an n-dimensional feature vectors. (2) There is a mechanism that transforms an n-dimensional feature vector into an object of the same type as the ones in the original dataset. Both points will be addressed and exemplified next.

For the first premise, the numerical representation of the dataset is the information used to find the corresponding 2D projection. It is true that some multidimensional projection techniques like multidimensional scaling (MDS) [Cox and Cox, 2000] only require the dissimilarity information between pairs of instances. However, when the inverse projection takes place, it is the direct correspondence between 2D points and their multidimensional counterparts that permit the creation of a mapping function. The way to find a good feature-vector representation for a dataset is very case-dependent. In some cases, the object itself is the feature-vector. For example, in the optimization application presented in the next chapter, the data instances are multidimensional points in the parameter space. These points are directly used as the feature-vector representation required for the inverse-projection framework. In other cases, this relationship is not as explicit. For instance, the dataset of 3D humans faces (Chapter 6) requires a proper definition of how to extract feature vectors

that represent each face. The definition of feature vectors in this specific scenario will be described in Section 6.3.1. Once each instance in the dataset is represented by feature vectors of same dimension and semantic, one is ready to start using the inverse-projection framework.

Regarding the second premise, the output of inverse-projection is a multidimensional point with the same dimension as the feature vectors describing each instance of the data. A mechanism is required to transform this point into an object of the same class as the ones from the given data set. This mechanism would allow users to comprehend a newly created point entirely and to move forward with the interactive exploration process. For example, in the proposed face-synthesis demonstration application, a multidimensional feature vector is transformed into a 3D face model.

4.3 Inverse Projection Mapping Function

The inverse projection transformation from 2D to multi-dimensions is carried over by a mapping function $f : \mathbb{R}^2 \rightarrow \mathbb{R}^m$. This function is essential in the inverse projection framework, and it should work in such a way that the position of the multidimensional output, with respect to the multidimensional points of the original dataset, is coherent with the 2D neighborhood as defined in the projection layout. In other words, the new multidimensional point should be similar to the points that are in its neighborhood in 2D. The MP components allow us to create such a mapping.

The general idea is very much like the process of establishing a mapping function for MP with control points. Recall from Chapter 3 that the correspondence between multidimensional control points and their 2D position is used to create an interpolation function that approximates the 2D projection of the remaining instances. In inverse projection, a similar function is desired, but one that maps a point from 2D to the original multidimensional space of the dataset. The inverse projection problem does not have a unique solution:

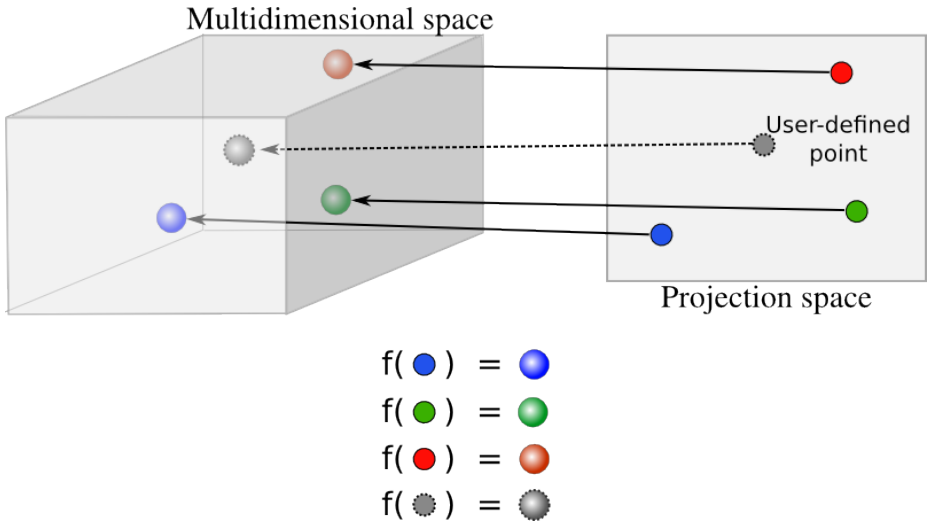


Figure 4.2: In the left, a simple multidimensional dataset with 3 instances is illustrated; and in the right, we present its 2D projection. A function $f : \mathbb{R}^2 \rightarrow \mathbb{R}^m$ is sought in such a way that it respects the known 2D-to-multidimensional correspondence of the data samples. When a new 2D point is created (gray circle to the right), this function is used to approximate its multidimensional position.

there are multiple ways to map a 2D point into a multidimensional space. Using the correspondence between 2D and multidimensional points, one can restrict the solution space to make it unique. In this case, the correspondence between each 2D position and their multidimensional feature vector is given by the MP framework. We propose to use this information to create a mapping function that maps an existing 2D point exactly back to its feature vector. This function can later be used to approximate the multidimensional position of any user-given 2D point. Figure 4.2 illustrates the different components of the inverse projection framework, emphasizing the role of the mapping function.

The mapping function defines important properties that will make each inverse projection method unique. In this thesis, we propose two different mapping functions for inverse projection. One is based on the Local Affine Multidimensional Projection (LAMP) [Joia et al., 2011] methodology. We use the same mathematical formulation proposed in LAMP to approximate the projection with the use of control points, but this time to infer the multidimensional position of 2D samples. The other one is based on the RBF formulation

presented in the previous chapter. Again, the same formulation and logic are used but to create a mapping from low to high-dimension instead.

Before moving on to the inverse projection formulations (Chapters 5 and 6), I will present an overview of the system developed in the course of this research to integrate MP and inverse projection.

4.4 System Overview

As presented in this chapter, inverse projection is a framework that allows users to interactively create new multidimensional points. As such, it requires an interactive system platform to support close to real-time interaction. Part of this research is to design and develop such an interactive system.

The implementation of inverse projection requires a user interface which integrates MP with the inverse projection functionalities. To this end, I have developed a system using the C++ programming language, Qt API to create the GUI and OpenGL API to handle the graphics and projection/inverse projection rendering.

When the system in launched, the user can load a file that describes a multidimensional dataset. The file format consists of a header that contains the number of instances and parameters in the dataset, besides the optional names of each parameter. An extra column can also be included, providing a value associated with the instance. If present, this information is used to assign a color (based on a colormap) to each instance in the projection space. The same input file formatting used in the *pex* toolkit for multidimensional projection [Paulovich et al., 2008b] was adopted for this prototype.

Once the dataset is loaded into the system, an MP technique is chosen to project the data. Three CP-based techniques are implemented and integrated into the prototype system, namely, LAMP [Joia et al., 2011], PLMP [Paulovich et al., 2010] and my proposed RBF method presented in the previous chapter. The user is also given the option to choose

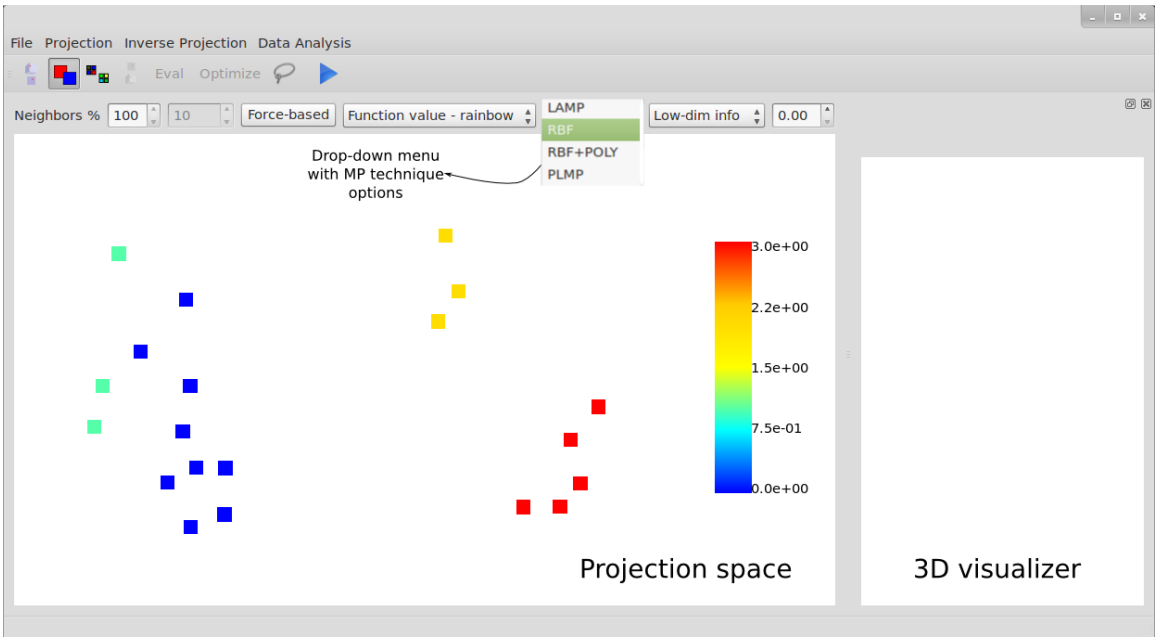


Figure 4.3: Loading the *mammals dataset*, which consists of 20,000 instances and 47 parameters, with four different animal groups, encoded by colors. This figure illustrates 20 control points projected into the projection space through a forced-based technique. The MP method drop-down menu is highlighted; the screen on the right is a 3D visualizer, which is useful in applications such as face-synthesis.

the number of control points to use, and whether random or ROLS-based selection (Section 3.2) should be carried out. Figure 4.3 illustrates 20 control points projected and displayed in the projection space.

In case RBF-projection is selected, one can also choose the basis function to be used in the approximation and the shape parameter of the function (as described in Section 2.6). The projection is then calculated and displayed on the projection screen, which becomes an interactive display (Figure 4.4).

In the prototype system, CPs can be rearranged, and the projection recomputed on-the-fly. New CPs can also be created, or existing ones can be deleted. A free-form region of the projection can be selected and a separate window will display the parallel-coordinates representation [Inselberg and Dimsdale, 1990b] of the selected instances, as illustrated in Figure 4.5.

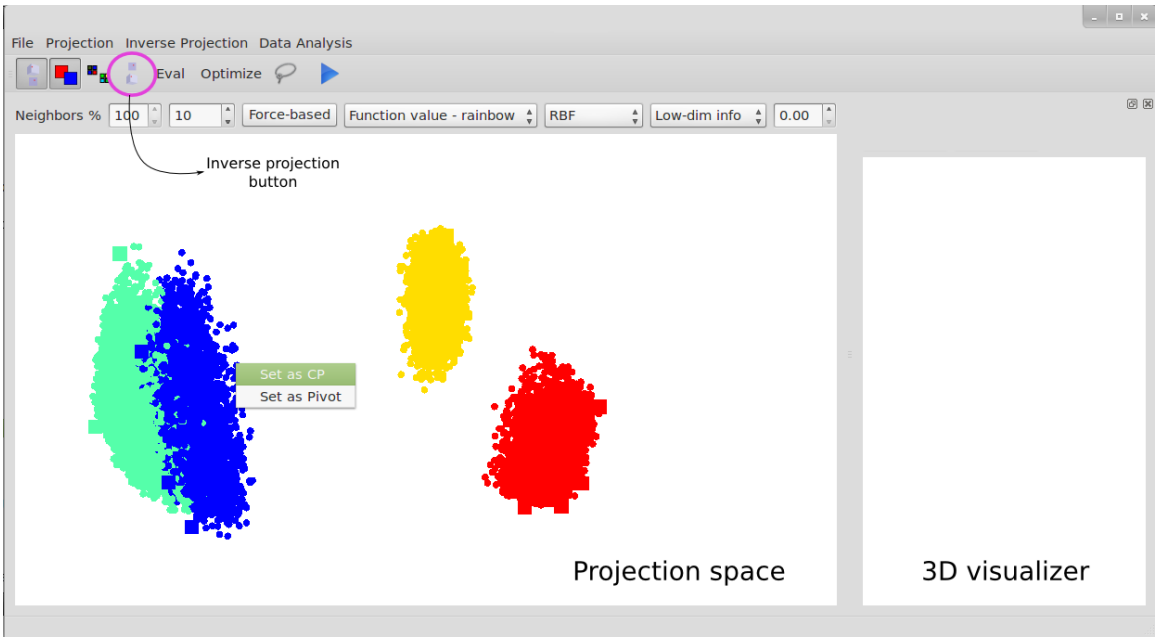


Figure 4.4: The final projection is calculated, using the method selected in the drop-down menu. In this stage, control points can be rearranged or even deleted, or new control points can be added as illustrated.

The inverse-projection mode can be activated by toggling one button in the menu (see Figure 4.4 highlighting this button in the menu). The desired mapping function is chosen (i.e., iLAMP or iRBF) and new 2D points can now be created with the aid of any pointing device. The selected mapping function calculates a multidimensional representation for each point created. At this point, depending on the application in hand, the user needs to plug-in the code that processes the new multidimensional vector accordingly. For example, the optimization application we present in Chapter 5, the new vector is used as a starting point for a gradient-descent calculation. In the face-synthesis application, on the other hand, the new vector is used to compute both a new 3D model and texture information that will become a new face model, which is displayed in the 3D-visualizer window.

This chapter introduced the inverse projection framework, its main components were discussed, as well as the prototype system I developed. The following chapters present the mathematical formulation, as well as the evaluation results and proposed applications for

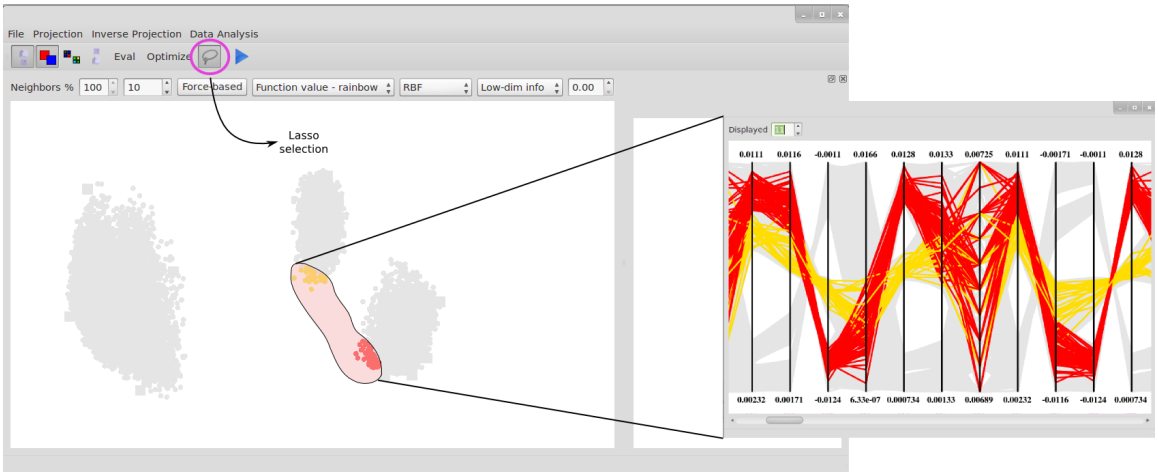


Figure 4.5: By toggling the *lasso selection* button, one can create free-form regions to select instances of interest, which in turn can be rendered as a parallel-coordinates in a pop-up window to further enhance data exploration.

both iLAMP and iRBF.

Chapter 5

Inverse Local Affine Multidimensional Projection

In this chapter, we introduce the mathematical formulation of the iLAMP technique. We also present the evaluation carried out to validate the method and a demonstration application to optimize a function with many local minima.

5.1 Mathematical Formulation

In general terms, iLAMP maps a 2D point in the low-dimensional space, i.e, the visual space, into a multidimensional vector. The proposed method takes as input a multidimensional dataset and its correspondent low-dimensional representation, calculated by a multidimensional projection technique. iLAMP provides an mathematical framework that allows to create new points in the visual space that contains the data projection. iLAMP computes an affine transformation that takes a point $\mathbf{p}$, defined by the user in the visual space, to a point $\mathbf{q}$ in the original multidimensional space.

Let $\mathbf{x}_i$ be an instance in dataset $X \subset \mathbb{R}^m$, and its correspondent instance $\mathbf{y}_i$ in the visual space $Y \subset \mathbb{R}^2$; i.e., $\mathbf{y}_i$ is the correspondent projection of $\mathbf{x}_i$ in the visual space. Both sets, X and Y, are given as input to iLAMP, which uses this information to build local affine transformations $f : \mathbb{R}^2 \to \mathbb{R}^m$ to map a point $\mathbf{p}$ from the visual space to a point $\mathbf{q}$ in $\mathbb{R}^m$.

Given a point $\mathbf{p} \in \mathbb{R}^2$ and $k \in \mathbb{N}$, the first step in the iLAMP algorithm is to find the k closest neighbors to $\mathbf{p}$ among the instances in Y. All the subsequent calculations are done solely based on these k instances and their correspondent multidimensional vectors encountered in dataset X. Let $Y_S = \{\mathbf{y}_1, \mathbf{y}_2 \ldots \mathbf{y}_k\}$ be the subset of Y that contains the k closest points to $\mathbf{p}$ and $X_S = \{\mathbf{x}_1, \mathbf{x}_2 \ldots \mathbf{x}_k\}$ the dataset containing the correspondent multidimensional instances on X.

iLAMP maps $\mathbf{p}$ from the visual space to the original multidimensional space $\mathbb{R}^m$ by finding the isometric affine transformation $f(\mathbf{p}) = \mathbf{p}M + \mathbf{t}$ that minimizes

$$E(M,t) = \sum_{i=1}^{k} \alpha_i \| f(\mathbf{y}_i) - \mathbf{x}_i \|^2, \tag{5.1}$$

where matrix M and vector $\mathbf{t}$ are the unknowns. Note that $f(p)$ is isometric iff $M^T M = I$, where I is the identity matrix. Moreover, the constraint $M^T M = I$ enforces Euclidean distances (norms) to be preserved as much as possible during the mapping, as demonstrated bellow

$$\|M\mathbf{x}\|^2 = (M\mathbf{x})^T M\mathbf{x} = \mathbf{x}^T M^T M\mathbf{x} = \mathbf{x}^T\mathbf{x} = \|\mathbf{x}\|^2,$$

i.e., the Euclidean norm of a vector x remains the same after applying M to x. Since the Euclidean norms are preserved, the orthogonal matrix M acts as an isometric transformation in the mapping. Such methodology results in an accurate mapping scheme, as shown in Section 5.4.

The scalars α_i are weights defined as weights defined as

$$\alpha_i = \frac{1}{\|\mathbf{y}_i - \mathbf{p}\|^2}. \tag{5.2}$$

The main goal is to find the minimizer E (Equation (5.1), and the singular value decomposition (SVD) [Baker, 2005] is a good approach to achive this. To find the minizer E, we set the partial derivative of E with respect to t to zero, thus t can be written in terms of M as

$$t = \tilde{\mathbf{x}} - \tilde{\mathbf{y}}M, \qquad \tilde{\mathbf{x}} = \frac{\sum_{i=1}^{k} \alpha_i \mathbf{x}_i}{\bar{\alpha}}, \qquad \tilde{\mathbf{y}} = \frac{\sum_{i=1}^{k} \alpha_i \mathbf{y}_i}{\bar{\alpha}}, \tag{5.3}$$

where $\bar{\alpha} = \sum_{i=1}^{k} \alpha_i$. The minimization problem described on (5.1) can be written as

$$\min_{M} \sum_{i=1}^{k} \alpha_i \|\hat{\mathbf{y}}_i M - \hat{\mathbf{x}}_i\|, \qquad \text{subject to } M^T M = I, \tag{5.4}$$

where

$$\hat{\mathbf{x}}_i = \mathbf{x}_i - \tilde{\mathbf{x}}, \qquad \hat{\mathbf{y}}_i = \mathbf{y}_i - \tilde{\mathbf{y}}. \tag{5.5}$$

The minimization problem (5.4) can be written in matrix form as

$$\min_{M} \|AM - B\|_F, \qquad \text{subject to} \quad M^T M = I, \tag{5.6}$$

where $\|.\|_F$ denotes the Frobenius norm and A and B are matrices given by

$$A = \begin{bmatrix} \sqrt{\alpha_1}\hat{\mathbf{y}}_1 \\ \sqrt{\alpha_2}\hat{\mathbf{y}}_2 \\ \vdots \\ \sqrt{\alpha_k}\hat{\mathbf{y}}_k \end{bmatrix}_{k\times 2}, \qquad B = \begin{bmatrix} \sqrt{\alpha_1}\hat{\mathbf{x}}_1 \\ \sqrt{\alpha_2}\hat{\mathbf{x}}_2 \\ \vdots \\ \sqrt{\alpha_k}\hat{\mathbf{x}}_k \end{bmatrix}_{k\times m}. \tag{5.7}$$

The rationale behind the minimization problem (5.6) is to build affine mappings that respect the correspondence $\mathbf{y}_i \leftrightarrow \mathbf{x}_i$. Note that when $\mathbf{p}$ is precisely one of the projected points $\mathbf{y}_l, 1 \leq l \leq k$, α_l weight goes to infinity (Equation (5.2)), forcing the minimization to give emphasis to the term $\|f(\mathbf{y}_l) - \mathbf{x}_l\|^2$ (Equation (5.1)). This fact guarantees that the inverse mapping of a projected data $\mathbf{y}_l$ is precisely its counterpart $\mathbf{x}_l$. In practice, to avoid division by zero in Equation (5.2), when $\|\mathbf{y}_k - \mathbf{p}\|^2 < 1.e^{-13}$ iLAMP simply returns the feature vector $\mathbf{x}_k$, i.e., the multidimensional counterpart of $\mathbf{y}_k$.

The solution of (5.6) is given by

$$M = UV, \qquad A^T B = UDV, \tag{5.8}$$

Algorithm 2 iLAMP

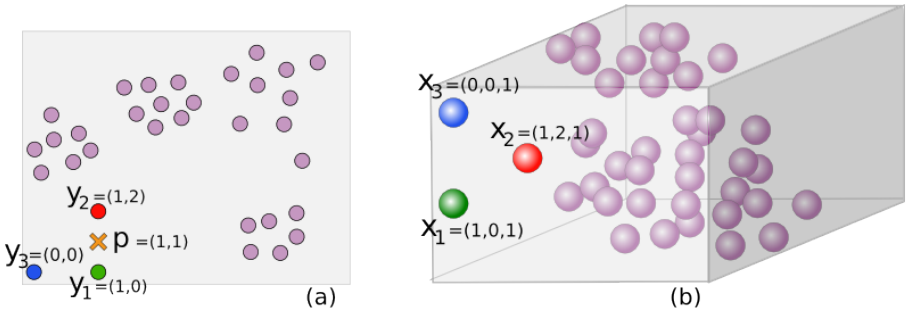
Given $X \subset \mathbb{R}^m$ (multidimensional dataset)
Given $Y \subset \mathbb{R}^2$ (2D projection of X)
Given $p \in \mathbb{R}^2$ (user-generated 2D point)
Given $k \in \mathbb{N}$ (Number of closest neighbors)
Calculate k-dimensional vector α (Equation (5.2))
$\bar{\alpha} = 0$
for $i = 1..k$ **do**
    $\bar{\alpha} = \bar{\alpha} + \alpha[i]$
end for
Calculate m-dimensional vector $\tilde{\mathbf{x}}$ and 2-dimensional vector $\tilde{\mathbf{y}}$ (Equation (5.3)
for $i = 1..k$ **do**
    $\hat{\mathbf{x}}_i = \mathbf{x}_i - \tilde{\mathbf{x}}$
    $\hat{\mathbf{y}}_i = \mathbf{y}_i - \tilde{\mathbf{y}}$
end for
Assemble matrices A and B (Equation (5.7)
$UDV =$ Calculate SVD of A^tB
$M = UV$
$q = (\mathbf{p} - \tilde{\mathbf{y}})M + \tilde{\mathbf{x}}$ (Equation (5.9))
return q

where UDV is the singular value decomposition (SVD) of A^TB. Once M has been computed, the mapping of a point $\mathbf{p}$ to the multidimensional space is accomplished by

$$\mathbf{q} = f(\mathbf{p}) = (\mathbf{p} - \tilde{\mathbf{y}})M + \tilde{\mathbf{x}}. \tag{5.9}$$

The procedure to calculate the inverse projection of a point p through iLAMP is summarized in Algorithm 2.

Figure 5.1 presents a simple numerical example using a 3D dataset and its 2D projection as input and parameter $k = 3$, with the step-by-step calculation of inverse projection through iLAMP.



$$\alpha_1 = 1; \qquad \alpha_2 = 1; \qquad \alpha_3 = \frac{1}{\sqrt{2}} \approx 0.7 \qquad \bar{\alpha} = 2.7$$ (c)

$$\tilde{x} = \frac{(2, 2, 2.7)}{\bar{\alpha}} \approx (0.74, 0.74, 1) \qquad \tilde{y} = \frac{(2, 2)}{\bar{\alpha}} \approx (0.74, 0.74)$$ (d)

$$\hat{x_1} = (0.36, -0.74, 0), \qquad \hat{x_2} = (0.36, 1.36, 0), \qquad \hat{x_3} = (-0.74, -0.74, 0)$$
$$\hat{y_1} = (0.36, -0.74), \qquad \hat{y_2} = (0.36, 1.36), \qquad \hat{y_3} = (-0.74, -0.74)$$ (e)

$$A = \begin{bmatrix} 0.36 & -0.74 \\ 0.36 & 1.36 \\ -0.74 & -0.74 \end{bmatrix}, \qquad B = \begin{bmatrix} 0.36 & -0.74 & 0 \\ 0.36 & 1.36 & 0 \\ -0.74 & -0.74 & 0 \end{bmatrix}$$ (f)

$$U = \begin{bmatrix} -0.3073 & -0.9516 & 0 \\ -0.9516 & 0.3073 & 0 \end{bmatrix}, \qquad V = \begin{bmatrix} -0.3073 & -0.9516 & 0 \\ -0.9516 & 0.3073 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$ (g)

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$ (h)

$$q = (p - \tilde{y})M + \tilde{x} = (0.36, 0.36, 0) + (0.74, 0.74, 1) = (1, 1, 1)$$ (i)

Figure 5.1: A numerical example of iLAMP with a 3D dataset and k=3 closest neighbors. (a) Projection space – user-defined point p (orange cross) and 3-closest neighbors(blue, red and green circles); (b) Multidimensional space – three instances corresponding to the 3-closest neighbors (colored accordingly); (c)-(h) all steps to derive the iLAMP mapping function culminating in (i) applying the function to p to derive $q \in \mathbb{R}^3$.

Before attesting the quality of iLAMP, we discuss some computational and implementation aspects of the technique.

5.2 Computational Aspects and Implementation

In this Section we discuss some implementation and computational aspects of the iLAMP technique. We provide a time analysis of the method, a discussion on the neighborhood of point **p** and interaction scheme of the proposed method.

5.2.1 Compact SVD and Time Analysis

The bottleneck of the iLAMP computation is the calculation of the SVD of matrix $A^T B$ (Equation (5.8)). Matrices A^T and B are $2 \times k$ and $m \times k$, respectively, what makes $A^T B$ a $2 \times m$ matrix. Since $A^T B$ contains only two rows, one may employ *compact* SVD methods to compute the required decomposition, reducing the computational costs considerably when compared to a full SVD scheme. In my implementation I use the LAPACK library [Anderson et al., 1990] compact SVD routine.

The complexity of the iLAMP algorithm depends only on the number of neighbors k and in the dimensionality of the original data, as the calculation of the closest neighbors to point **p** can be accomplished using efficient methods, such as quadtrees [Arya et al., 1998]. Figure 5.2 displays the data dimensionality versus time spent by iLAMP to calculate 100 (one hundred) backward mappings, for three different values of k. The experiments were performed in an Intel Core i7-860 Processor (8M Cache, 2.80 GHz). Fifty runs of the algorithm were performed for each test case, and the illustrated results are the average time acquired. We observe that iLAMP is extremely fast, what allows for interactive real-time applications. For instance, for $k = 100$ and $m = 450$, the average time spent to calculate 100 backward mappings was 263 milliseconds. Notice there is a linear correlation between data dimensionality and time, which indicates that iLAMP is scalable even to datasets with

very high dimensionality.

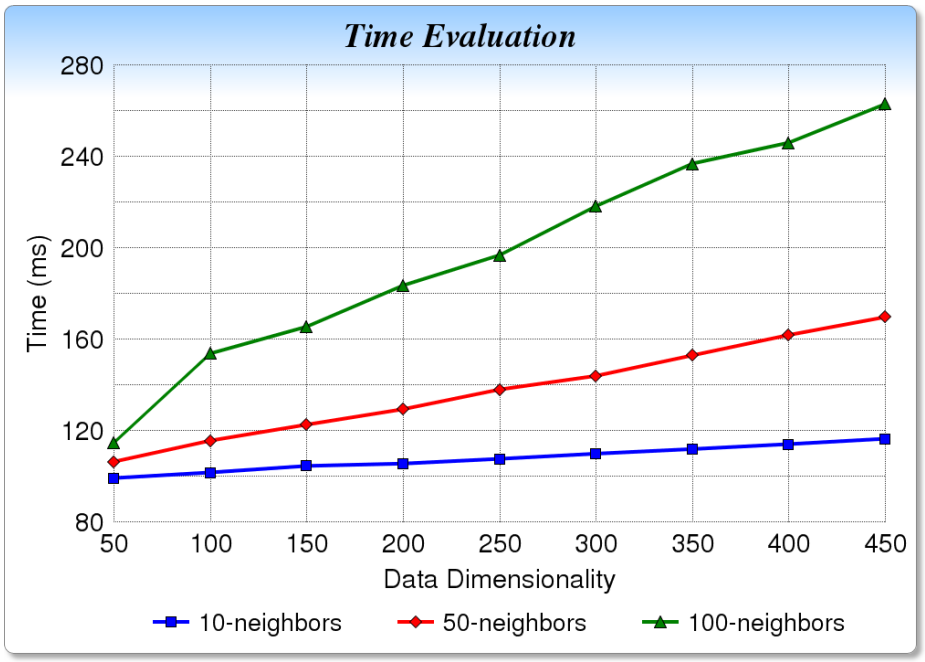


Figure 5.2: Time, in milliseconds, consumed in the generation of 100 samples using iLAMP.

5.2.2 Number of Neighbors

The number of neighbors k is an important parameter for the iLAMP algorithm. It defines how many instances of the original dataset is taken into account on the construction of the affine transformation matrix M that will map point $\mathbf{p} \in \mathbb{R}^2$ to a point $\mathbf{q} \in \mathbb{R}^m$. Thus, the value of k has a significant impact on the quality of the output transformation. In this thesis, we chose k in a heuristic fashion. We experimented with various datasets and we concluded that $3 \leq k \leq 20$ is sufficient to achieve proper inverse mappings.

5.2.3 Interaction Scheme

The main component of our system is a screen that displays the multidimensional projection of a multidimensional dataset. The displayed projection provides the visual feedback that allows the user to analyze the data and identify possible regions of interest to be further

explored. For instance, these can be empty regions in the projection space that are close to instances of particular interest. Using interactive point selection, the user is able to extrapolate the data in such regions, by creating new points in the projection space and mapping them to the original space using the iLAMP technique. This strategy allows the user to experiment what-if scenarios, exploring the parameter space in an interactive and intuitive way.

5.3 Handling False Neighborhoods and Tears

False neighborhoods and tears are artifacts that may appear in multidimensional projections. A false neighborhood occurs when a significant distance in the original space is associated with a small distance in the projection space. This distortion falsely suggests a neighborhood of points that do not accurately reflect the initial data distribution. In contrast, a tear occurs when a small distance in the multidimensional space is associated with a large distance in the projection space. This distortion falsely conveys a large difference between nearby neighboring points.

Some projection techniques attempt to reduce these artifacts; however, in many cases they are unavoidable [Aupetit, 2007]. For instance, consider a dataset composed of samples of a hypersphere (or any other closed surface). The projection of such samples is somehow distorted with the existence of false neighbors, tears, or both [Lespinats and Aupetit, 2009]. More generally, the presence of outliers in a dataset often lead to false neighborhoods in the projection.

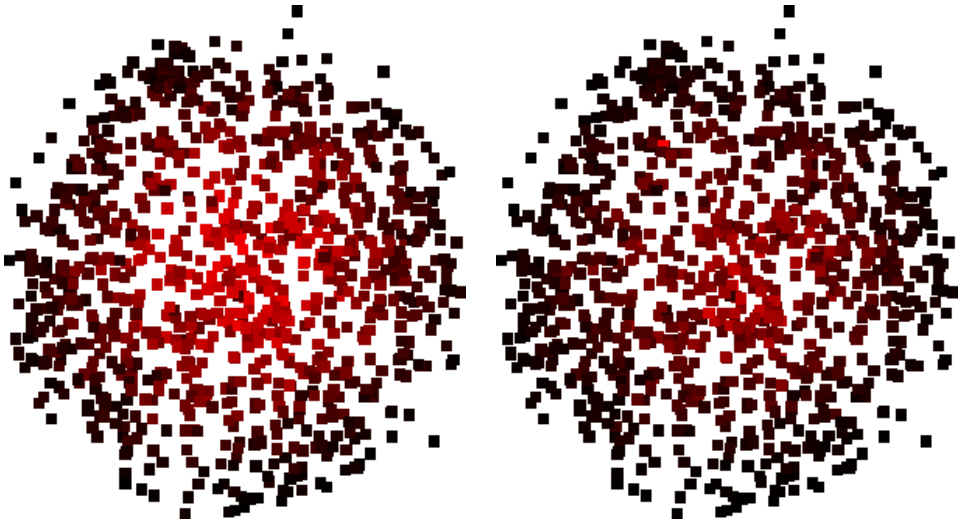
If these artifacts are disregarded in the inverse projection process, iLAMP-generated points may become distorted and end up in unexpected regions of the original space. So far, we have considered that the k closest points to $\mathbf{p}$ are searched among the instances of dataset Y. In fact, this is the most natural procedure, as $\mathbf{p}$ and $\mathbf{y} \in Y$ are defined in the same space $\mathbb{R}^2$. However, if false neighborhoods and/or tears are present in the projection,

using only the low-dimensional information for this task can result in misleading inverse projection mappings.

To accommodate for artifacts in the projection, it is possible to incorporate the multidimensional data in the neighborhood definition of a point **p**. The closest instance $y_i \in Y$ to **p** seeds the neighborhood search in the original space. The $k-1$ nearest neighbors to the multidimensional instance $x_i \in X$ corresponding to y_i, $(x_{i1}, x_{i2}, x_{ik-1})$, define the neighborhood of **p**. This *multidimensional neighborhood search* prevents the usage of a neighborhood set composed of false neighbors.

Note that we do not enforce the usage of the multidimensional neighborhood for the inverse projection mapping. Instead, we allow the user to decide whether to use the low-dimensional information only or to include the multidimensional information. However, the user should have a way to assess the quality of the projection both as a whole or as considering regions of interest of to make an informed decision on how the neighborhood set will be formed.

We provide three visual tools to give indications to the user of artifacts within the projection. These methods are based on proposed visualization techniques of false neighborhoods and tears [Aupetit, 2007]. In particular, [Lespinats and Aupetit, 2009] propose two metrics that should help in the identification of such artifacts and that we use in this thesis. Based on Sammon's [Sammon, 1969] and Curvilinear Component Analysis' (CCA) [Demartines and Herault, 1997] loss functions, each of these metrics account for the identification of tears and false neighbors, respectively. The difference between the calculation of each metric is in the weight assigned to the error between distances in the low and high dimensional spaces. In Sammon's metric, the smaller the distance in the *multidimensional space*, the more the error is penalized. On the other hand, in CCA's metric, the smaller the distance in the *low-dimensional space*, the more the error is penalized.



(a) (b)

Figure 5.3: Color mapping of the (a) Sammon's (tears identification) and (b) CCA's (false neighborhoods identification) error for the projection of a dataset composed of 5D-sphere samples. Colors vary from black (low-error projection) and bright red (high-error projection)

.

The definition of each metric, for a given instance i, is given as follows:

$$P_{Sammon}(i) = \sum_j \left(\|\bar{d}_{ij} - d_{ij}\|^2 \times \frac{1}{\bar{d}_{ij}} \right),$$

$$P_{CCA}(i) = \sum_j \left(\|\bar{d}_{ij} - d_{ij}\|^2 \times \frac{1}{d_{ij}} \right),$$

where $\bar{d}_{ij}$ and d_{ij} are the distances between instances i and j in the high- and low-dimensional space, respectively. Each point has an associated P-value (one for Sammon's and other for CCA's loss function), which can be mapped to colors on the projection screen. In our system, the colors vary from black (low error) to red (high error), as seen in Figure 5.3.

We also propose the visualization of a distance map related to a projected point called a *pivot*. In the projection screen, the user selects a point to be the pivot, and the multidimensional distance to the pivot is used as a color mapping for each instance. The distance-mapping information can be utilized as a guidance for the user in the replacement of LAMP control points, in an attempt to reduce the artifacts aforementioned in this section. It can

also be valuable for the user decision of using the low-dimensional information only or including multidimensional information in the neighborhood used on iLAMP. In this case, the maps vary from black (small distances) to green (high distances). Figure 5.4 presents an example of the usage of such information.

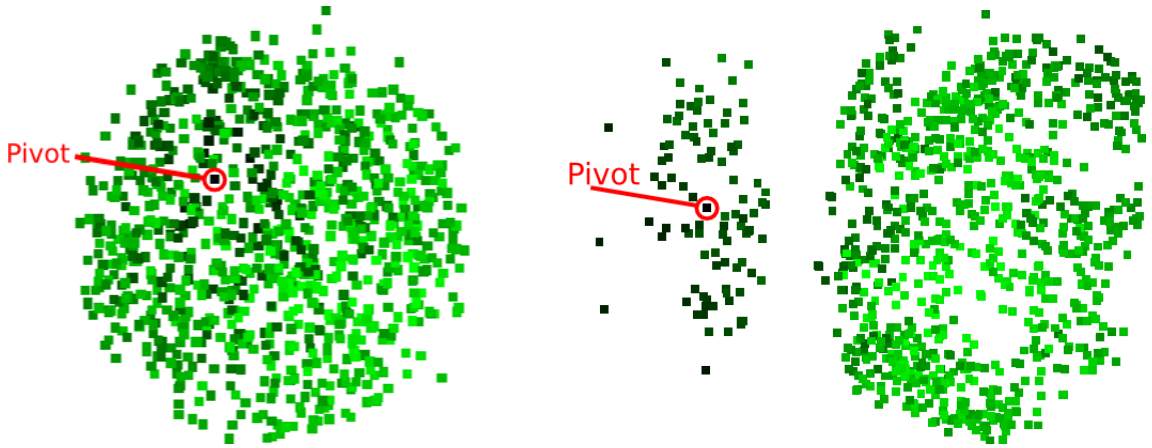


Figure 5.4: Projection of a 5D sphere dataset. (a) Original Projection colored according to the Distance map of the pivot. Dark green and light green points indicate that the corresponding multidimensional instance is close to the pivot or far from it, respectively. It is easy to visualize many false neighbors (light green points close to the pivot). (b) Using distance map information, LAMP control points are rearranged. The left points' cloud contains close neighbors to the pivot. Creating new points close to this cloud minimizes the incident of false neighborhoods and tears.

Thus, we cope with false neighborhoods and tears by providing visual feedbacks that allow the user to identify such artifacts. An informed decision regarding the iLAMP-neighborhood type (low- or multidimensional) can be made to reduce or prevent distortions in the iLAMP-generated points.

5.4 Validation

In this section, we present the experiments and results used to measure the quality of the iLAMP method. As previously stated, iLAMP is used to map an instance $\mathbf{p} \in \mathbb{R}^2$ to an instance $\mathbf{q} \in \mathbb{R}^m$ in a coherent way. While there are an infinite number of vectors that may

be assigned to **q**, the goal of iLAMP is to compute the vector that is consistent with the 2D projection and user-defined point and also with the original dataset. More specifically, we calculate the vector **q** whose multidimensional projection is near to the multidimensional original surface. In this section, we also describe the experiments and measurements that lend credence to our inverse projection technique.

5.5 Curve Back-Projection

We begin with a qualitative analysis by applying iLAMP to a user-designed $2D$ curve back-projecting it into a $3D$ space. Operating in lower dimensions provides visual confirmation of the iLAMP results. In this experiment, we construct a *parallel Swiss roll* dataset by randomly sampling 2000 instances from two adjacent Swiss rolls that are separated by a small gap. The Swiss roll dataset is a 2-dimensional manifold embedded in a 3-dimensional space. It is extensively used as a benchmark to evaluate manifold-learning and dimensionality reduction techniques [Roweis and Saul, 2000b]. In this experiment, we demonstrate that, given the 2D projection of a Swiss roll dataset, iLAMP can be used to sample its 3-dimensional surface as explained below.

Our parallel Swiss roll is projected to the plane using LAMP, in which 80 control points are strategically selected along the edges of each Swiss roll. As demonstrated in Figure 5.5, a free-hand curve is drawn on the $2D$ visual space, between the two projected Swiss rolls. Using 60 iLAMP neighbors, we back-project the user defined curve, mapping it back into the original $3D$ space. Figure 5.5 illustrates the iLAMP reconstructed curve, which appears to be lifted in dimension in a precise and coherent way.

5.6 Hypersphere Reconstruction

For the quantitative analysis of iLAMP, we use hypersphere datasets, with various dimensions and densities. Hyperspheres are very prone to distortions when projected to a lower-

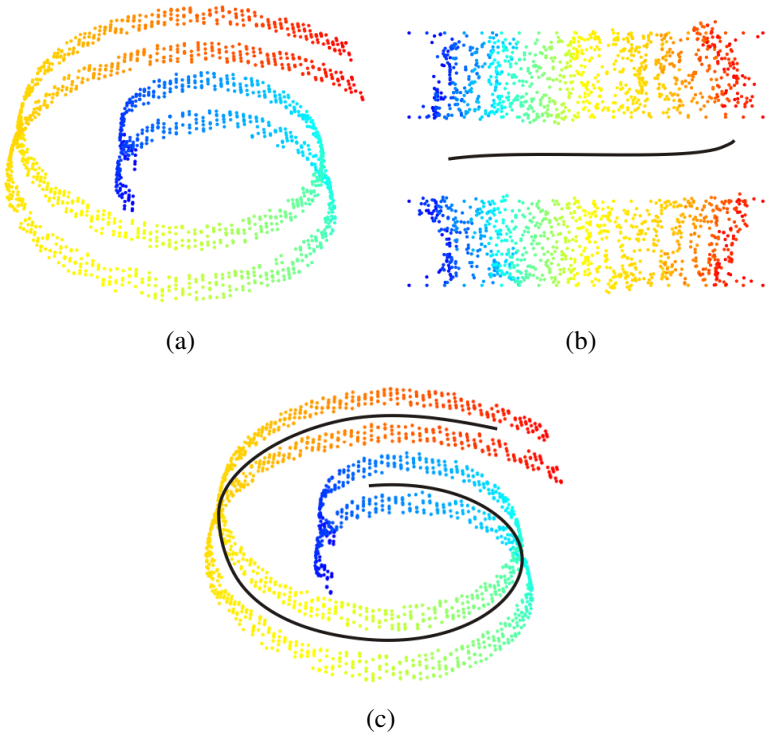


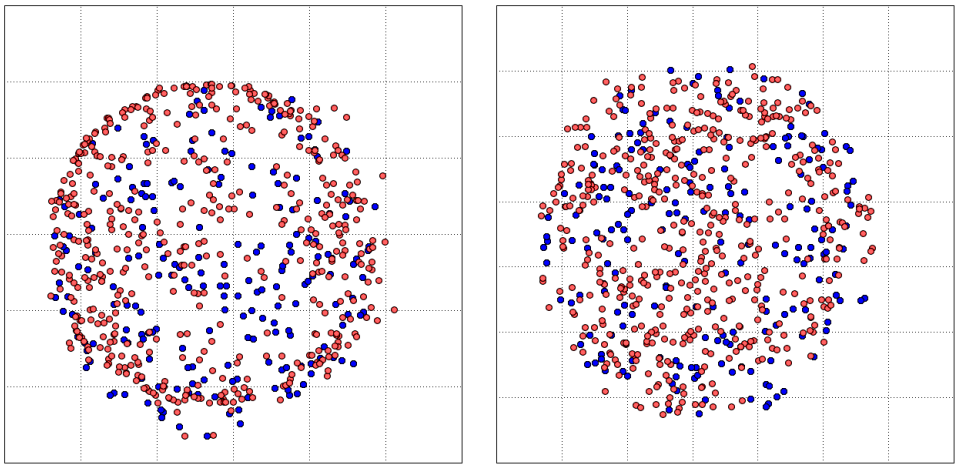
Figure 5.5: An example of inverse projection in a swiss roll dataset. (a) original dataset; (b) projection and 2D curve samples (black); (c) 3D swiss roll and inverse-projected curve.

dimensional space, and a challenge for the inverse projection framework. Thus, the evaluation of iLAMP with such datasets can provide evidence of the robustness and capabilities of iLAMP.

The unit hypersphere embedded in an m-dimensional space, $\sum_{i=1}^{m} x_i^2 - 1 = 0$, provides the basis for our quantitative analysis of iLAMP. We create synthetic hypersphere datasets designed to test the robustness of iLAMP under varying sample density and increasing space dimensionality. The datasets are constructed by randomly sampling hyperspheres with various densities (100, 500 and 1000 instances) and dimensionalities (3-, 5-, 10- and 20-dimensional spaces). For each dimensionality, three datasets were created, with 100, 500 and 1000 random points sampled from the hypersphere's surface. This sampling resulted in 12 unique hypersphere datasets, against which the following tests are run.

For a given hypersphere dataset, we begin by projecting the samples to the planar domain using LAMP. In these experiments, a multidimensional projection is calculated for

each of the data sets. To thoroughly evaluate iLAMP in each hypersphere scenario, 200 points are randomly sampled over the projected domain. iLAMP is used to calculate their corresponding multidimensional position. Figure 5.6 illustrates the projection of the four hypersphere datasets with 500 samples. In this figure, the red points are the projected samples and the blue points are the 200 randomly selected iLAMP input points.



(a) 3D (b) 5D

(c) 10D (d) 20D

Figure 5.6: This figure illustrates the 2D projection (red points) of four hypersphere datasets, with 500 instances each, embedded in (a) 3, (b) 5, (c) 10 and (d) 20 dimensions. The projection is carried out through LAMP [Joia et al., 2011]. In each of these projections, 200 2D points (blue) are created and used as input to iLAMP. The results of inverse projection for these samples are used to validate the iLAMP methodology.

Additionally, note that the iLAMP procedure is applied to each $2D$ point multiple times with different neighborhood sizes. We increment the number of nearest neighbors considered by iLAMP over the interval between 2 and 20, producing 19 different inverse projections.

Three metrics monitor the result accuracy, including (1) the distance between back-projected samples and the analytically defined surface; (2) the stress function; and (3) the *LAMP-validation*.

5.6.1 Distance to Surface

The first measure computes the distance between the back-projected samples, generated via iLAMP, and the nearest point on the unit hypersphere. This value indicates how consistent the iLAMP results are to the originating surface. The distance d_s of the back-projected point $\mathbf{q}$ is defined as,

$$d_s(\mathbf{q}) = |1 - \sum_{i=1}^{m} q_i^2|.$$

Figure 5.7 presents box plots of the distances computed between iLAMP's back-projected multidimensional samples and the hypersphere. Observe that the extrapolated points remain close to the hypersphere surface over which the dataset had been sampled. In particular, the mean distance error is below 0.15 in each experiment. The reported distances rely on the best inverse projection solution found amongst the different iLAMP neighborhood sizes used. The following quality measure analyzes the effects of neighborhood size.

5.6.2 Stress Function

As presented in Section 5.1, the design of iLAMP's transformation matrix is motivated by LAMP. Specifically, the inverse projection closely preserves the relative distances between a $2D$ point $\mathbf{p}$ and its neighbors as the relative distances between the inverse projection of $\mathbf{p}$ with the multidimensional images of its $2D$ neighbors. Projection techniques, such

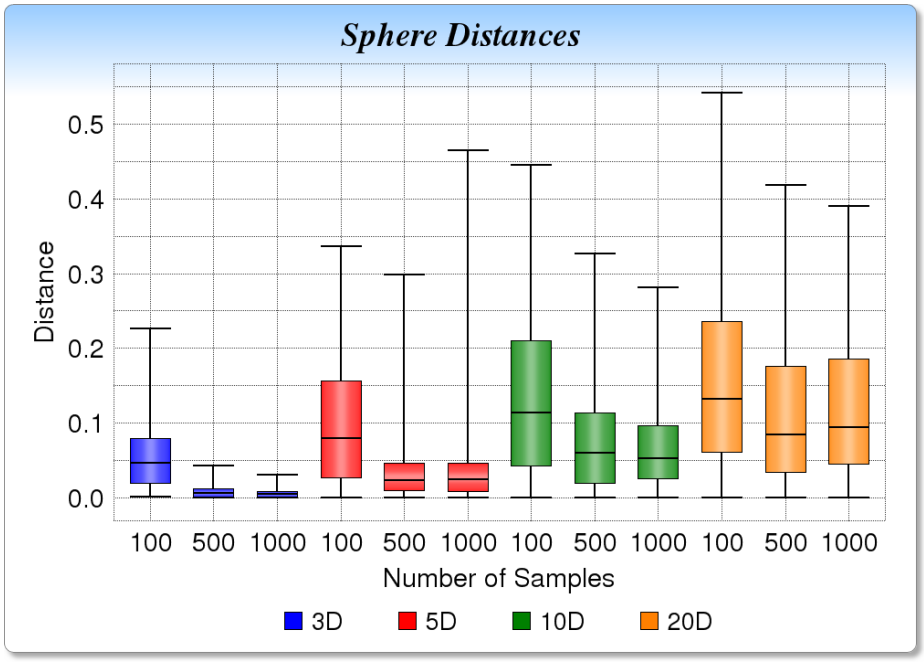


Figure 5.7: Boxplots of distribution of distances between newly created samples and the sphere surface for each hypersphere dataset. Note that, even for a 20-dimensional hypersphere, the distance median is close to 0.1, i.e., most of the points were distant by 0.1 units or less from the hypersphere surface.

as LAMP, rely on the stress function to measure this preservation of relative distances to validate their dimensionality reduction approaches.

To determine the stress function, let l and n be the number of new and original instances, respectively. Let d_{ij} and $\bar{d}_{ij}$ be the distances between $\mathbf{p}_i$ and $\mathbf{y}_j$, and $\mathbf{q}_i$ and $\mathbf{x}_j$, respectively. Recall that $\mathbf{x}_j \in X \in \mathbb{R}^m$ (the original dataset) and $\mathbf{y}_j = \text{LAMP}(\mathbf{x}_j)$. The iLAMP stress function is defined (see Section 5.1 for the notations),

$$s = \frac{\sum_{i=1}^{l}\sum_{j=1}^{n}(d_{ij} - \bar{d}_{ij})^2}{\sum_{i=1}^{l}\sum_{j=1}^{n} d_{ij}^2}.$$

Note that the stress function measures the distance preservation between all pairs of points in the dataset, not just the k nearest neighbors, to measure the global distortion of space.

Figure 5.8 plots the stress function for the iLAMP reconstructions of the multiple hypersphere datasets with respect to the iLAMP neighborhood size (k nearest neighbors).

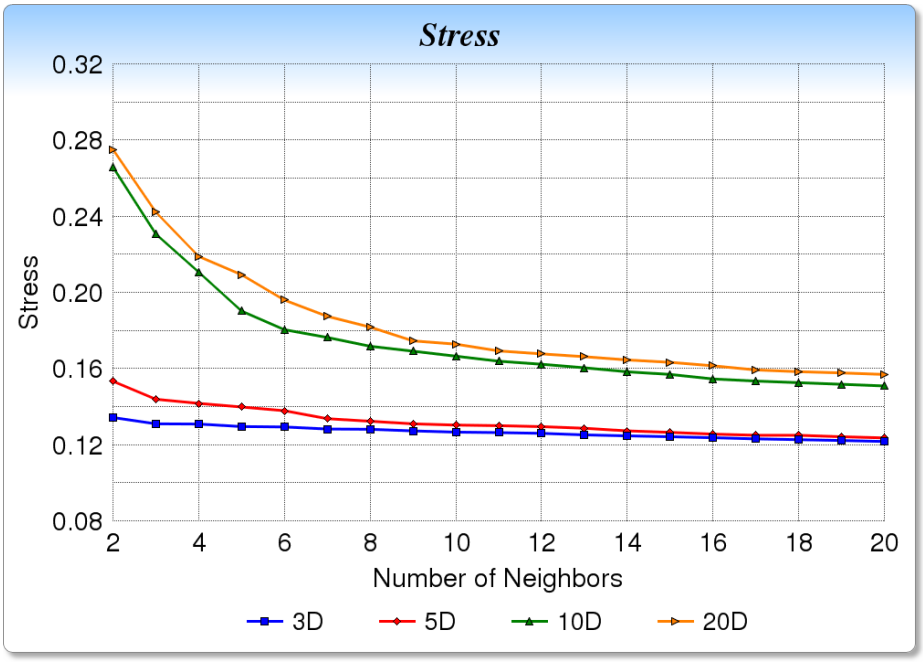


Figure 5.8: Number of neighbors X stress functions obtained with 500 sample's datasets. The lower the dimension of the dataset, the lesser the impact of the number of neighbors in stress. As the dimensionality increases, the results indicate that more neighbor points must be used to maintain stress low. However, this number seems to stabilize after a certain threshold (10-15 in this example).

Observe that the number of neighbors is inversely correlated to the projection's stress function; but, with diminishing returns. Further, larger neighborhoods are necessary as the dimensionality of the original dataset increases to maintain a high quality in the inverse projection.

5.6.3 LAMP-Validation

Lastly, the *LAMP-validation* applies the LAMP projection technique to back-projected points, measuring the distance between the new projection and the original point. The user selected $2D$ point $\mathbf{p}$ is lifted into the original multidimensional space, $\mathbf{q} = \text{iLAMP}(\mathbf{p})$. This point $\mathbf{q}$ is projected back to the $2D$ domain as $\mathbf{p}' = \text{LAMP}(\mathbf{q})$. The *LAMP-validation* measurement becomes,

$$L(\mathbf{p}) = \frac{\|\mathbf{p} - \mathbf{p}'\|}{\|\mathbf{p}'\|}$$

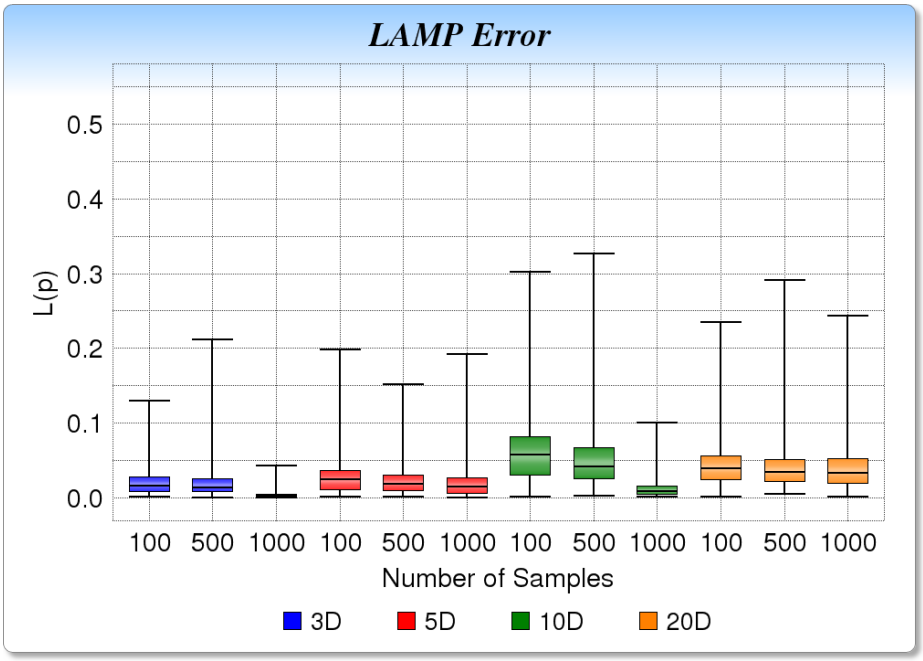


Figure 5.9: Boxplots with error measurement distributions according to the *LAMP-validation* metric. Note that the error distribution is very low (less than 0.05 for 75% of the cases in almost all experiments). This analysis indicates that, if the iLAMP-generated point q existed in the original multidimensional dataset from the beginning, it would have been projected very close to the user-defined 2D point p.

Figure 5.9 presents the *LAMP-validation* error for various hypersphere datasets of various sample density and dimensionality. This test utilizes the LAMP projection method to attest to the coherence of the extrapolated instances. The small distance residuals suggest that the iLAMP extrapolated instances are consistent with the LAMP projection. In particular, the mean error is below 0.1 across each experiment.

The results presented in this section indicate that the iLAMP technique can extrapolate instances of an existing dataset based on local neighborhoods from the layout of a projection. The method creates new instances that are coherent with the original dataset (Figure 5.7) and its projection (Figure 5.9). Further, it closely preserves relative distances between point pairs in the m-dimensional space (Figure 5.8). In the following section, we present applications of iLAMP in more specific scenarios.

5.7 Exploring Parameter Spaces in Optimization Problems

In this section, I present an application in which the user expertise is incorporated into the optimization process through iLAMP. The problem of parameter-space exploration in optimization is often hard to address and, when addressed, it is done in an automated fashion, as multiple parameter combinations can provide reasonable outputs. Inverse projection can be used to provide an interface to include the user in the exploration loop.

In this application, an optimization problem consists of finding $x \in \mathbb{R}^m$ that minimizes a given smooth function $f(x) : \mathbb{R}^m \to \mathbb{R}$ or, in more general terms, finding the best solution among many feasible solutions. It has countless applications in virtually every discipline, like reservoir simulation [Hajizadeh et al., 2012], aerospace engineering [Sobieszczanski-Sobieski and Haftka, 1997] and electrophysiology of the heart [Martins et al., 2006].

In general, optimization problems are solved using automatic methods by either gradient-based or gradient-free techniques. Such techniques start from an *initial guess* and iteratively improve the solution until it cannot be further improved, as it reaches a local minimum. Gradient-based methods are very sensitive to the initial guess, and different minima may be reached depending on the location of the initial guess. However, creating meaningful starting points for such algorithms is challenging because, in the very beginning, the user may have no idea on where good minimizers are located. Moreover, most optimization problems present non-unique solutions, i.e., there are several satisfactory minimization points. Our application integrates LAMP projection and iLAMP inverse projection in a system that allows the user to explore and inspect by a sampling mechanism regions of interest in the multidimensional space of possible solutions.

5.8 System Overview

In this application, I employ iLAMP to allow the user to explore the multidimensional optimization space interactively . The application is composed of three main subprograms: 1)

LAMP projection visualization method; 2) iLAMP inverse projection and 3) an optimization method. A visualization and interaction window integrates the three moduli while allowing the user to interact with the resulting visualization. Figure 5.10 illustrates the system workflow.

5.8.1 System Setup

The proposed application receives as input an initial dataset, composed of precomputed local minima, which is projected to the visual space by LAMP (Figure 5.12-a). LAMP is intrinsically interactive, which allows the user to explore and analyze the initial data by directly manipulating control points (see [Joia et al., 2011] for details). In this exploration phase, the user can identify regions of interest in the multidimensional space and resample those areas by adding new points in the visual space. In fact, the system allows the user to create individual points or a set of random points by clicking or drawing rectangles in the visual space (Figure 5.12-b and c). The new user-defined samples are backward mapped to the multidimensional spaces using our approach. The new points are then used as starting points into optimization procedure (Figure 5.12-d) to reveal new local extrema (Figure 5.12-e). The process can be further refined in specific regions of interest (Figure 5.12-f) to h) until the user is satisfied with the optimization results.

5.8.2 System in Use

To provide details about my methodology, I discuss an example step-by-step, demonstrating that our technique may help in the analysis and inspection of optimization spaces.

To illustrate the system, the *bird function* is used as the function to be minimized and the *Levenberg-Marquadt* method [Gavin, 2011] to perform the optimization. The bird function is one of many multimodal test functions proposed by [Mishra, 2006] to assess the performance of optimization methods. It presents several local minima, making it suitable to help understand how iLAMP can be used to explore the parameter space in search for

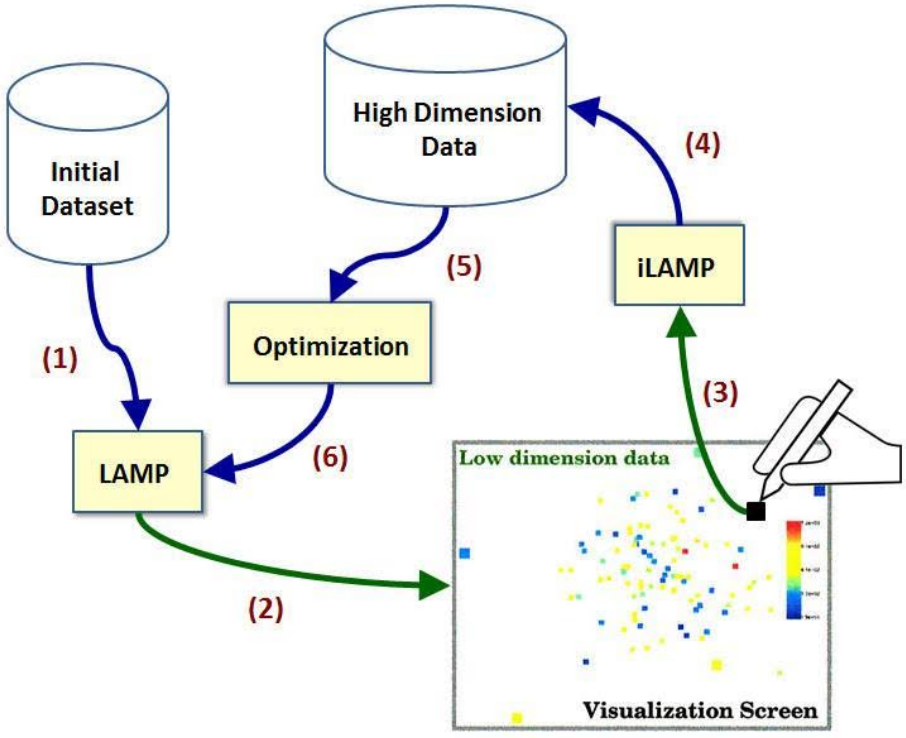


Figure 5.10: Application workflow; green and blue arrows represent the flux of the low and multidimensional data, respectively. Instances colored according to the optimization function value. (1) Initial data given as input to LAMP; (2) Projection (3) User input passed to iLAMP; (4) New multidimensional samples; (5) New data is passed as argument to the optimization method; (6) Optimization result incorporated to the dataset.

some of these local minima. The bird function is defined as follows:

$$b(\mathbf{x}) = \sum_{i=0}^{(m/2)-1} (sin(x_{2i}) * e^{(1-cos(x_{2i+1}))^2} + cos(x_{2i+1}) * e^{(1-sin(x_{2i}))^2} + (x_{2i} - x_{2i+1})^2 + 106.76),$$

where m is the space dimensionality and x_i the coordinates of point $\mathbf{x}$.

Figure 5.11 illustrates the bird function for $m = 2$ in the interval $-2\pi < x_i < 2\pi$, where we can clearly see four local minima. If we make $m = 20$ we end up with thousands of local minima, making difficult to understand where are the best minimizers.

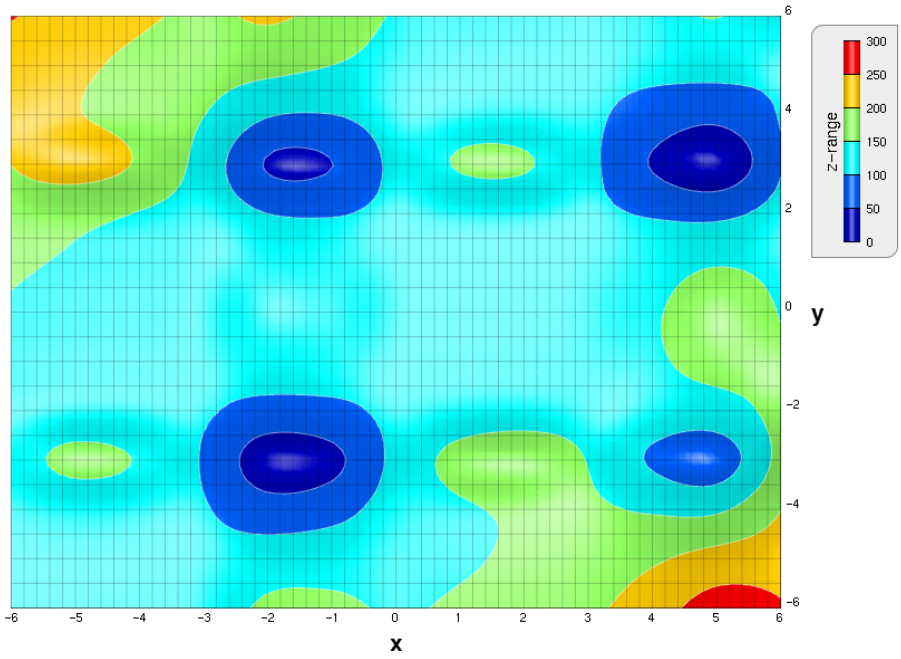


Figure 5.11: Plot of bird-function with 2 parameters $-6 \leq x_1, x_2 \leq 6$. The color maps to the function value, ranging from 0 (dark blue) to 300 (red), and we can immediately spot four local minima (dark blue regions) in this simple 2D example.

We start the 20-dimensional space exploration by running the gradient-based minimization method 100 times with random initial guesses. The smallest minimum contained in the original dataset has a function value $b(\mathbf{x}) = 19$. The original dataset is given as input to LAMP, which projected the instances to the visual space (Figure 5.12-a). Projected samples are colored according to their $b(\mathbf{x})$ function value, which is an essential information to guide the user towards regions containing other minima.

With the projection in hand, the optimization space is explored and interactively resampled. iLAMP allows the user to manipulate control points, which reorganizes the projection accordingly, and potentially brings out regions of interest where the user can resample by clicking points and drawing rectangular boxes in the visual space (Figure 5.12-b). Our approach projects the user-defined samples back to the multidimensional space, which are added to the dataset (Figure 5.12-c).

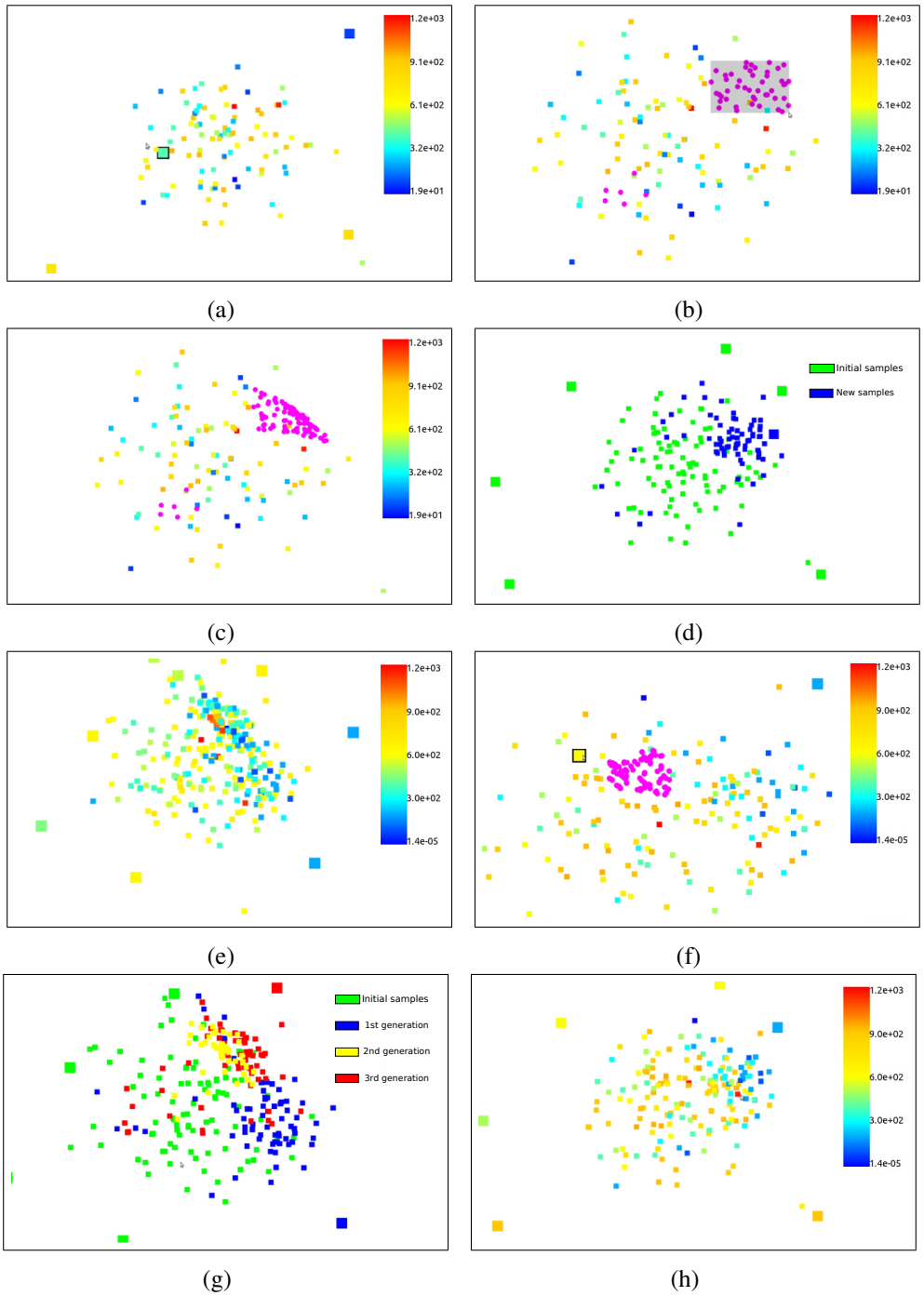


Figure 5.12: Example of the prototype system in action. (a) Initial dataset projected; (b) New samples created (magenta); (c) New samples incorporated to projection; (d) Green: original dataset, Blue: user-generated samples; (e) After using user-generated samples as input to optimization algorithm; (f) Extrapolating data in other regions; (g) Green: initial dataset; Blue, Red, Yellow: First, second and third user-generated sets; (h) Final layout after optimization of new samples. Color scale on figures (a), (b), (c), (e), (f) and (h) represents the function value of each projected point. Big points are the control points used in the LAMP projection technique. The colors in (d) and (g) are used to differentiate the initial samples from the iLAMP-generated samples. Each color refers to one step in the generation of samples.

Figure 5.12-d shows the initial data in green and the new ones created by the user in blue. The new samples are used as input to the gradient-based method which reveals new local minima. Figure 5.12-e shows the new local minima colored according to their function value. Our exploration system includes some of the new samples as control points for LAMP to improve interactivity. A further exploration of the space was performed to bring out more regions of interest (Figure 5.12-f). Zooming in those areas of interest and repeating the process above a couple of times we end up with approximately 300 new samples (Figure 5.12-g) which give rise to many extrema, as illustrated in Figure 5.12-h. In a few seconds we were able to interactively find out minimizer where the bird function is equal to $b(\mathbf{x}) = 1.4e^{-5}$, besides many other local minima that might be of great interest depending on the application.

The example above shows that the proposed system empowers the user with a flexibility not found in other multidimensional data exploration techniques. Indeed, we have tested our approach in optimization functions other than the *bird function* as well as distinct optimization algorithms. With no exception we could mine new minima in a few seconds, always improving the initial results provided by the automated sampling mechanisms built into the optimization software, thus making evident the effectiveness of our approach and the importance of adding the user in the process.

In this chapter, the inverse Local Affine Multidimensional Projection was presented. This iLAMP technique constructs an affine mapping from 2D to multidimensional space to approximate the position of user-generated points. Several experiments were used to demonstrate the effectiveness of the technique, as well as an application example where iLAMP is used to explore the parameter-space of a multidimensional function with multiple local minima.

In the next chapter, a non-linear inverse projection based on radial basis functions will be introduced.

Chapter 6

Inverse Projection through Radial Basis Functions

In the previous chapter, we presented the iLAMP algorithm for inverse projection. iLAMP operates locally and uses local affine transformations to map user-defined 2D points into the multidimensional space. iLAMP does not guarantee a continuous inverse projection mapping, i.e., a continuous 2D curve may be mapped to more than one region in the multidimensional space. iLAMP can also cause distortions in the multidimensional points when the projection layout is modified by the user, as will be presented in Section 6.7. Such characteristics do not pose a challenge for the optimization application described in Section 5.7, but other applications may be better exploited with global and nonlinear mappings. To fill this gap, we propose in this thesis a novel nonlinear and continuous inverse projection technique, based on radial basis function (RBF) interpolation.

As presented in Chapter 3, we employ RBF in Multidimensional Projection (MP). In RBF projection, a function $s : \mathbb{R}^m \rightarrow \mathbb{R}^2$ is created to map m-dimensional points to the 2D projection space. In this section, we propose to use RBF interpolation to perform *inverse projection*. This is accomplished by creating a function $s(p)$ that maps data points from the projected 2D space into the original m-dimensional space, i.e., $s : \mathbb{R}^2 \rightarrow \mathbb{R}^m$.

In the inverse projection scenario, the RBF centers are given by the 2-dimensional position of the points in the projection space ($y_i \in Y$), and the outputs are provided by the corresponding multidimensional points of the original dataset ($x_i \in X$). The mapping function $s(p)$ is, thus, one that exactly maps the given 2D projected points y_i into their multidimensional counterparts x_i. This function is used to approximate the multidimensional correspondent of a user-generated 2D point. Depending on the choice of the kernel function, s can be smooth, making the RBF solution more attractive than iLAMP for some

applications. The problem of creating 3D human faces from an existing dataset of face models is an example of such applications and will be introduced next.

Analysis and synthesis of human faces are important in computer graphics and vision. In general, face models are encoded as multidimensional feature vectors that store information about geometric position and texture values of landmark points on the face. The position of the landmark points characterizes particular expressions and personal features. Synthesis of human faces is usually done "by example", where an input model of a face is given, and a 3D model is generated based on an existing training set of face models [Mena-Chalco et al., 2009].

In this work, we apply the inverse projection as a visual interface to synthesize new face models. The adopted framework is as follows: multidimensional projection is performed on a given dataset with face models represented by their respective feature vectors. The user can then create 2D points on the projection space and have these points mapped back into the original space of feature vectors. In the final stage of the framework the new multidimensional feature vectors are translated into 3D face models (Figure 6.4). The smoothness provided by the proposed RBF technique poses an advantage for this application, as a continuous curve in the 2D projection space creates a fluid transition of face models.

The next section presents a detailed description of the inverse projection technique based on RBF. Later on, an overview of the system is given with more details on the face synthesis demonstration application.

6.1 Mathematical Formulation

Let $X \subset \mathbb{R}^m$ be an m-dimensional dataset, and $Y \subset \mathbb{R}^2$ its projected counterpart, i.e., y_i is the 2D representation of x_i, $i = 1,\ldots,N$. Given any point $p \in \mathbb{R}^2$, we want to find an m-dimensional representation for it, i.e., a point $q \in \mathbb{R}^m$. Thus, a mapping $s : \mathbb{R}^2 \to \mathbb{R}^m$ is

sought. Figure 6.1 illustrates the process of using RBFs to create a mapping function from low to high dimension. To make the formulation clear, we write $s(p) = (s_1(p), \ldots, s_m(p))$, where s_k accounts for the k-th component and is written

$$s_k(p) = \sum_{i=1}^{N} \lambda_{ki} \phi(\|y_i - p\|), \tag{6.1}$$

where $\phi : \mathbb{R}^+ \to \mathbb{R}$ is a given continuous radial basis kernel function, $y_i \in \mathbb{R}^2$ are projected points (RBF centers) and the λ_{ki}'s are the unknown real-valued coefficients. Please refer to Section 2.6 for a general description of RBF theory. Note that we seek a mapping s that interpolates the points for the given data samples, i.e., $s(y_j) = x_j$. Thus, $s_k(y_j) = x_{jk}$ where x_{jk} is the k-th element of vector x_j. The interpolation condition for each function s_k can be written as

$$s_k(y_j) = \sum_{i=1}^{N} \lambda_{ki} \phi(\|y_i - y_j\|) = x_{jk}. \tag{6.2}$$

Therefore, the problem of finding the scalar coefficients λ_k for each function s_k comes down to the solution of a linear system

$$\Phi \lambda_k = b_k, \tag{6.3}$$

where Φ is the interpolation matrix with $\phi_{ij} = \phi_{ji} = \phi(\|y_i - y_j\|)$, $\lambda_k = [\lambda_{k1} \ldots \lambda_{kN}]^T$ and $b_k = [x_{k1} \ldots x_{kN}]^T$. The linear system can be written in matrix form as:

$$\begin{bmatrix} \phi_{11} & \cdots & \phi_{1N} \\ \phi_{21} & \cdots & \phi_{2N} \\ \vdots & \vdots & \vdots \\ \phi_{n1} & \cdots & \phi_{NN} \end{bmatrix} \begin{bmatrix} \lambda_{k1} \\ \lambda_{k2} \\ \vdots \\ \lambda_{kN} \end{bmatrix} = \begin{bmatrix} x_{k1} \\ x_{k2} \\ \vdots \\ x_{kN} \end{bmatrix}. \tag{6.4}$$

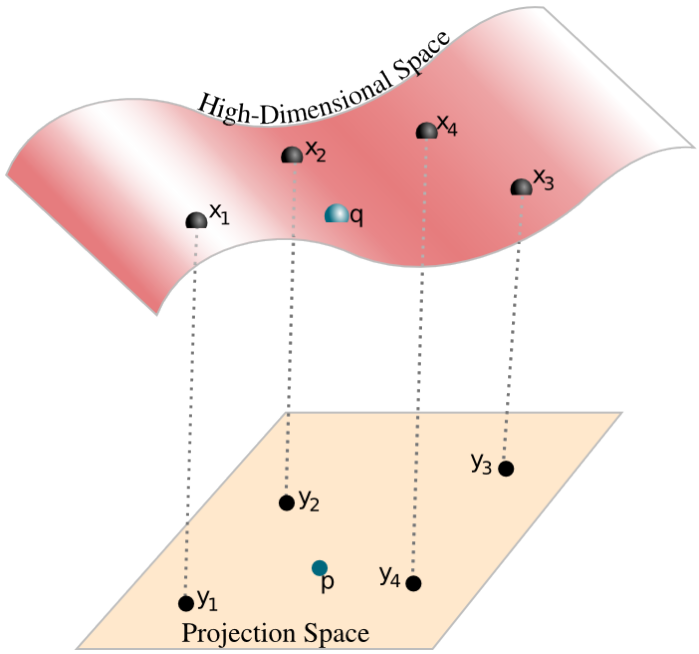


Figure 6.1: Inverse mapping using RBF. $\{x_1, x_2, x_3, x_4\} \in \mathbb{R}^m$ represent multidimensional points, while $\{y_1, y_2, y_3, y_4\} \in \mathbb{R}^2$ are their projected counterparts. Our inverse projection method creates a continuous nonlinear mapping function s (illustrated by pink surface) that interpolates between data samples, i.e. $s(y_i) = x_i$. The black points represent samples of the dataset, while the blue point represents a sample created through inverse mapping (p is the user-generated point; q represents the multidimensional point approximated through the RBF mapping).

Note that the linear system in Equation 6.3 is solved m times to find the parameter λ's for each function s_k. However, the interpolation matrix Φ remains the same and only the right-hand side vector b_k changes for different s_k's. Thus, the linear system can be solved only once in the process by factorizing Φ.

Once the scalars $\lambda_{ki}, k = 1, \ldots, m$ and $i = 1, \ldots, N$ are calculated, the mapping $s = (s_1, \ldots, s_m)$ is complete and can be used to approximate the position $q \in \mathbb{R}^m$ to any given point $p \in \mathbb{R}^2$. Algorithm 3 presents a step-by-step procedure to calculate the mapping function s.

Algorithm 3 Building the RBF for inverse projection

Given $Y = y_1, \ldots y_N \subset \mathbb{R}^2$ (projected dataset, RBF centers);
Given $X = x_1, \ldots x_N \subset \mathbb{R}^m$ (original dataset, RBF function values);
Given RBF kernel function $\phi : \mathbb{R}^+ \rightarrow \mathbb{R}$
// Calculate interpolation matrix Φ
for $i = 1 \ldots N$ **do**
    for $j = i \ldots N$ **do**
        $\Phi[i][j] = \Phi[j][i] = \phi(\|y_i - y_j\|)$
    end for
end for
// Assemble right-hand side of system
for $i = 1 \ldots N$ **do**
    for $j = 1 \ldots m$ **do**
        $b[i][j] = x_i[j]$
    end for
end for
Solve system $\Phi\lambda = b$, to find λ

6.1.1 Numerical and Computational Aspects

As it was shown, the RBF interpolation problem comes down to the solution of the linear system given by Equation 6.4. Note that the invertibility of the interpolation matrix Φ is dictated by the kernel function $\phi(r)$, where r is the Euclidean distance of the argument to the center of the kernel function. Some functions are proven to provide an invertible matrix with the minor assumption that centers y_i are unique. The Gaussian ($\phi(r) = e^{-\varepsilon r^2}$) and Multiquadrics ($\phi(r) = \sqrt{c^2 + \varepsilon r^2}$) functions are common kernel choices (where ε and c are positive parameters). A detailed description of these functions is out of the scope of this work, and we refer the reader to the books of Buhmann [Buhmann, 2003] and Wendland [Wendland, 2004] for more technical details.

There is a vast array of techniques designed to solve linear systems. The interpolation matrix Φ is always symmetric (since $\phi_{ij} = \phi_{ji}$) and, depending on the choice of kernel ϕ, it can be positive-definite. In such cases, the Cholesky factorization is a good option. Otherwise, general factorizations such as LU or QR can be used. The linear solvers used in this work are the ones available in the LAPACK library [Anderson et al., 1990].

The computation of the mapping function s is usually fast. In fact, we showed that the

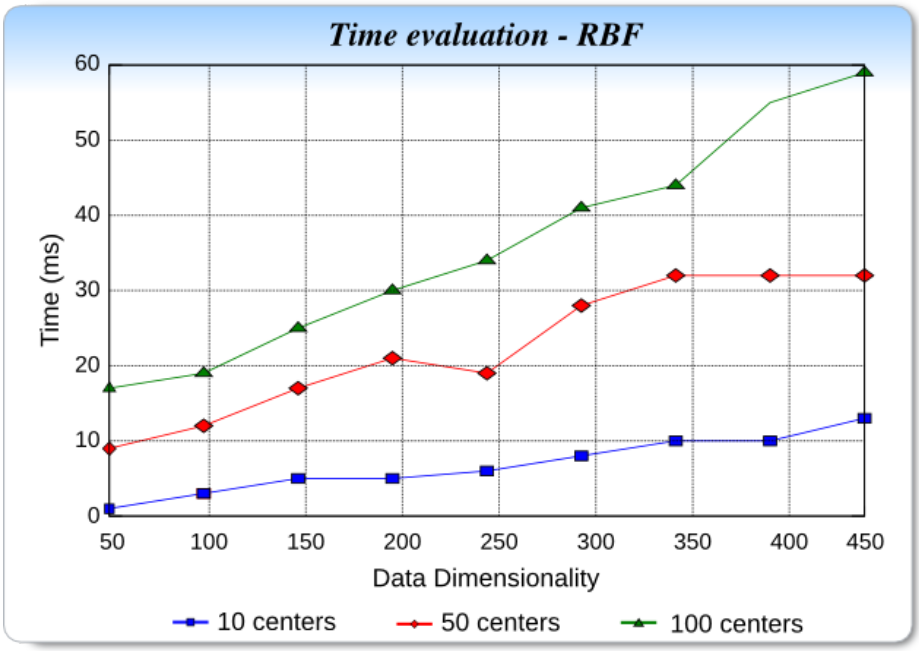


Figure 6.2: Time evaluation of RBF inverse projection. For varying number of centers, the time (in milliseconds) spent in the computation of the mapping function and the creation of 100 samples is displayed.

process comes down to the solution of a system of linear equations, whose size is determined by the number of RBF centers. We assessed the time spent in the computation of the mapping function plus the evaluation of 100 points for a different number of centers (10, 50 and 100) and data dimensionality (varying from 50 to 450). The results presented in Figure 6.2 were achieved in the same machine configuration described in Section 6.4. Note that the entire process is extremely fast even for a large number of centers and dimensions, rendering the technique suitable for real-time applications. Furthermore, note there is almost a linear relationship between time and number of centers (fixing the dimensionality parameter), indicating that many more centers could be used and still make the process fast for real-time applications.

In the proposed framework, the RBF centers come from projected points of a given dataset. Depending on the application, and on the size of the dataset on hand, all of the points could be used as centers. Or, as an alternative, only a subset of these points could be

utilized. In the proposed face-synthesis application, we use all of the points as RBF centers, mainly because we have a manageable number of samples in our faces dataset. However, in Section 6.1.3, we briefly discuss possible approaches to filter out and select few samples.

6.1.2 On the Choice of Radial Basis Kernels

It is clear that an appropriate choice of the kernel function and its parameters (when it is the case) have a significant impact on the quality of the interpolation results. Making this choice is, more often than not, a nontrivial task that requires careful attention. A commonly used approach to find an appropriate kernel is through trial-and-error, by experimenting with various possibilities and analyzing their outputs to make an informed decision. For example, in the proposed application we experimented with various kernels and different shape parameters. Some of the combinations of kernels and shape parameters were prone to overfitting, indicated by the fact that some output face models were very distorted). We found that the multiquadrics kernel function, with $c = 0$ yielded good results for the face-synthesis application, as will be shown in Section 6.4.

As an alternative to the trial-and error approach, there are some techniques designed to automatically select a kernel function and parameter definitions that could potentially approximate well a given data. Some of these methods are summarized in the works of [Mongillo, 2011] and [Fasshauer and Zhang, 2007]. For example, [Mongillo, 2011] proposes methods for predicting which shape parameters will minimize the error of an RBF interpolation, for a given kernel function. These methods take as input a kernel function, an associated shape parameter, and data samples, and return an error metric called *root mean squared* error, or RMS. Essentially, the RMS error indicates how well the specific kernel function and parameter are able to generalize for the given data samples. To select a kernel function, [Mongillo, 2011] proposes to evaluate various different combinations of kernels and shape parameters with respect to their RMS error, and choose the combination that produces the smallest error. There are other approaches to tackle this problem,

and we encourage the reader who is keen to use the proposed method to make a thorough investigation of this matter using the tools above.

6.1.3 False Neighbors and Tears

In Section 5.3, we presented the problem of false neighbors and tears, which are artifacts that may happen in the inverse projection process and may cause unwanted side-effects in the inverse projection results. As the names suggest, false neighbors consist of pairs of instances where $d_{ij} \ll \delta_{ij}$, while tears present $d_{ij} \gg \delta_{ij}$ [Martins et al., 2014], where δ_{ij} and d_{ij} are the original dissimilarities and the 2D distance between instances i and j. The presence of such artifacts can result in mapping functions that are not coherent with the given dataset. Thus, when many false neighbors and tears are present, using all instances as RBF centers in a global mapping function may not be the appropriate choice.

In these cases, one alternative could be to filter out the instances to select more appropriate centers for the mapping function. In Chapter 3, we presented an approach for center selection using RBF interpolation as the basis. Using a technique called *Regularized Orthogonal Least Squares* (ROLS), it is shown how to select a meaningful set of control points to be used in their multidimensional projection technique (Section 1.3.1). The results presented in Section 3.3 indicate that the control points selection through ROLS can improve the quality of multidimensional projection regarding stress. This methodology could be easily coupled into the RBF-based inverse projection to select appropriate centers and alleviate the effects of false neighbors and tears.

However, for some applications, it may be important to have all instances as RBF centers. This scenario is the case, for example, of the face-synthesis application we present in this chapter, where the goal is to have a face in the dataset to be represented by the mapping function. In such cases, we propose to give the user the option of generating one global mapping function, or several local mapping functions. The former consists of creating a single mapping function using every 2D point y_i as an RBF center. Therefore, the same

mapping function is utilized for all user-created 2D points, a good alternative for projections with low numbers of false neighbors/tears. For the latter, we propose to divide the n centers into k clusters, based on the distance information of the multidimensional dataset, and create one mapping function for each cluster. In doing so, we can reduce the negative impact that false neighbors and tears may cause in the inverse projection process.

To create the multidimensional clusters we take advantage of the control points used in multidimensional projection. Let the set of control points and its projected counterpart be, respectively, $X_s = x_{c1}, \ldots, x_{ck}$ and $Y_S = y_{c1}, \ldots, y_{ck}$. We propose to have one cluster C_l for each MP control point x_{cl}, where $\{(y_i, x_i) \in C_l \mid d(x_i, x_{cl}) < d(x_i, x_{cm}) \forall x_{cm} \in X_s\}$, i.e., a pair (y_i, x_i) belong to cluster C_l if and only if x_i is closer to x_{cl} than to any other control point x_{cm}. As previously mentioned, each cluster C_l gives rise to a mapping function s_l, using as RBF centers every $y_i \in C_l$. When a 2D point p is created, we find its closest neighbor y_i and assign the function s_l where $(y_i, x_i) \in C_l$.

In order to validate the clustering approach, we use the same hypersphere datasets presented in Section 5.6. Hyper-spheres are a good dataset to evaluate the impact of false neighbors/tears, as well as the methods used to mitigate their effects, because hyperspheres naturally undergo severe information loss when projected to a plane, i.e., they are prone to many false neighbors and tears. We used hyperspheres in four different dimensions (3, 5, 10 and 20) and 500 samples in each dataset. We generated 200 random points in the projection space, which were projected back to the original hypersphere dimensionality used the proposed RBF solution. The distances between the new 200 multidimensional points and the hypersphere surface are calculated. The rationale is that, the closer the new multidimensional point is to the surface, the better and more coherent the inverse projection is. The results are presented in Figure 6.3. We observe that the multidimensional samples created using the *clustering approach* (solid boxplots) are closer to the hypersphere surfaces than the ones produced through the *global approach* (dotted boxplots), i.e., it produces more

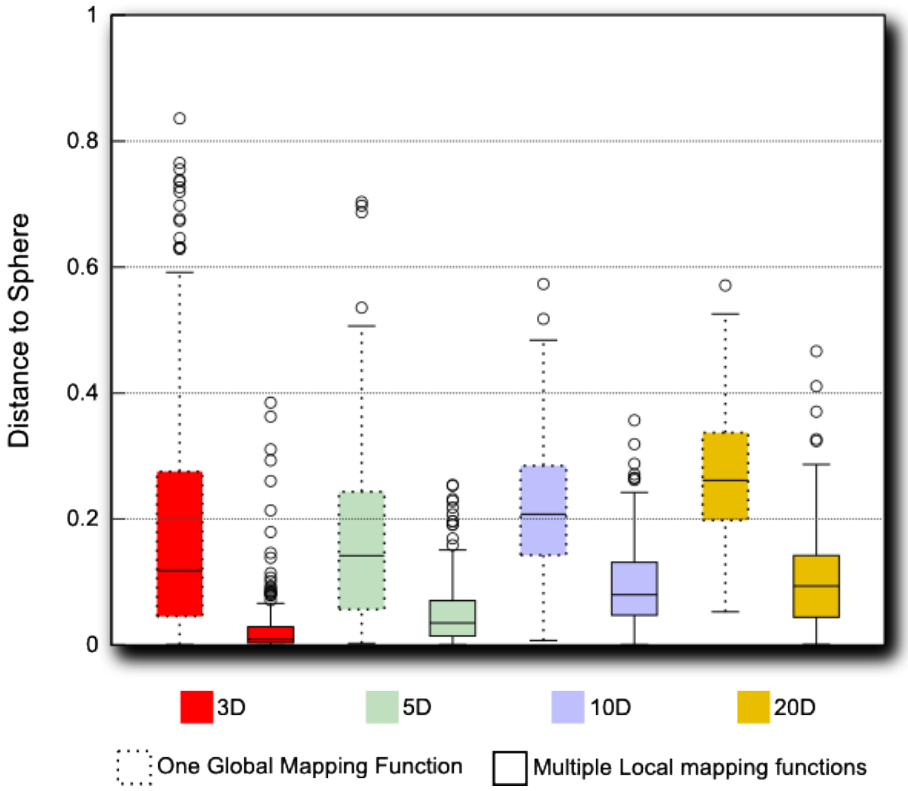


Figure 6.3: Experiment to validate the RBF projection when applied to distorted projections using 4 datasets. 500 random samples on the surface of hyperspheres of 3, 5, 10 and 20 dimensions were used, each dimension forming a separate dataset. 200 random sample points are generated in the projection space, and mapped back using one global and several local mapping functions. The boxplots indicate the distance between the multidimensional points generated and the sphere.

consistent results in this specific dataset.

Note that compact support radial basis kernels could also be used to generate a local effect in the inverse projection mappings. These kernels return zero beyond a given cut off distance, i.e., their influence is limited to a certain radius. Compact support RBFs can be an interesting approach to create local inverse projection mappings, however, as noted by [Buhmann, 2003], they requires careful attention with regards to its support size, since it will depend on how far apart the data samples are located with respect to each other. Also, compact support RBFs generally do not yield good approximations for data with many dimensions.

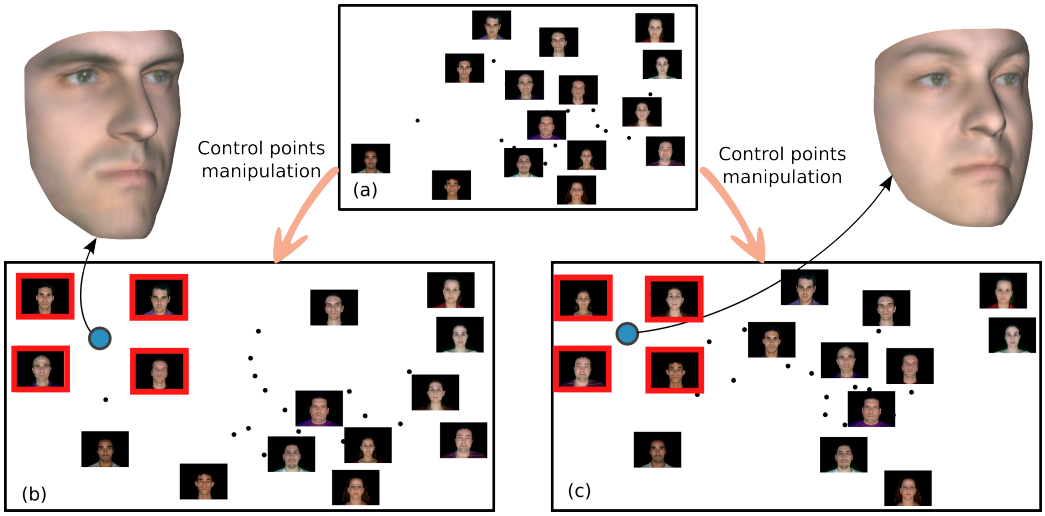


Figure 6.4: Example of control points' manipulation in the inverse projection framework, with 3D faces dataset; Figure (a) presents the projection of a dataset with 30 faces, and 15 control points – the control points are the ones rendered with the face texture, while black points are the remaining instances of the dataset; Figure (b) presents the projection after the reorganization of control points positions; in this example, four control points were isolated in the top left corner of the projection space (highlighted in red); a 2D point created by the user (blue circle) created a new 3D face. Figure (c) is another example of reorganization of control points position, where four different control points were isolated in the same top left corner (again highlighted in red); A new user-defined point, roughly in the same position as in the example (b), generates a different 3D face.

Note that this more local approach is suggested to be used when the projection presents many distortions and all instances of data should be a center in the RBF mapping function. In the case-study we present in this thesis, the global approach is more suitable.

6.2 Multidimensional Projection with Control Points

So far, we have not made any assumptions about the multidimensional projection technique used to map the original dataset into a 2D space. In fact, the mathematical formulation of the inverse projection, as proposed in this work, does not require this mapping to be done in any particular form. It only requires the multidimensional position of the data samples and their corresponding 2D projection, and the process that makes this mapping is not necessarily known and can be considered a black-box. However, the control points

mechanism presented in Chapter 3 can be beneficial to the data exploration in the inverse projection framework.

At this point, the reader should be familiar with the role of control points in multidimensional projection as a way to give users control, to some extent, over the projection results [Joia et al., 2011, Amorim et al., 2014]. They are particular projected data points that can be manipulated and rearranged in the projection space, causing the remaining points in the projection to be rearranged accordingly. This interaction can help users to gain more insight into the dataset and also allows expert knowledge to be incorporated in the projection process. This mechanism is particularly useful in the inverse projection framework, as it can help users to give more focus to target areas, by isolating control points of interest in the projection space. Figure 6.4 presents an example of how the control points can be a valuable tool in the faces-synthesis application. We show that, through the manipulation of control points, one can isolate particular instances of faces and generate new faces that are more similar to them.

6.3 Synthesis of Faces and Expressions

In this work, we propose to use our inverse projection approach as an interface that allows users to create new 3D faces, a *by-interaction* process, rapidly. There are three main reasons for choosing the face synthesis demonstration application to validate my method and demonstrate its applicability. (1) The synthesis of faces is an important problem with applications in various areas e.g., facial expression recognition, visual speech understanding, animating cartoon characters or digital actors using facial motions. (2) There are some important works in the literature that consider the human faces datasets to be embedded in multidimensional nonlinear manifolds [He et al., 2005, Roweis and Saul, 2000a, Chang et al., 2003]. For example, [Chang et al., 2003] argue that, since people change facial expressions continuously over time, it is a reasonable assumption that all images of some-

one's facial expressions form a smooth manifold in the N-dimensional image space, where N is the number of pixels in the images. The intrinsic dimension of the manifold should be much smaller than N. All these examples and characteristics makes face datasets good candidates for having its dimensionality reduced through multidimensional projection, which is an essential step in our inverse projection framework (Chapter 4, Figure 4.1). (3) Humans have the natural ability to evaluate how realistic a face is, making it easy to attest the quality of the faces generated through the proposed technique.

In this approach the user is not required to provide a 2D image to generate a new face, but an initial dataset of images is used to let the user navigate and resample the space of faces. Each face in the dataset is represented by a multidimensional vector, that contains geometric and texture information (Section 6.3.1).

The synthesis of 3D face models is an important research topic and has been studied for more than three decades [Levine and Yu, 2009]. Different approaches have been proposed to achieve 3D face reconstructions [Levine and Yu, 2009, Blanz and Vetter, 1999, Kirtzic and Daescu, 2011, Macêdo et al., 2006]. [Nguyen et al., 2009] classify the problem of 3D face synthesis in three categories based on the input information used: (1) 3D scan, (2) Multi 2D images, and (3) a single 2D image.

Parke introduces the first parameterized facial model [Parke, 1974] to shape interpolations and then animate a face. In previous work, [Cohen and Massaro, 1993] used a model of co-articulation and a set of animation parameters to control the face shape. [Blanz and Vetter, 1999] present a morphable model method based on a 3D face database of registered laser scans. An analysis-by-synthesis process conducts the reconstruction. It is important to note that the output is lifelike, but it requires expensive computation to determine the parameters, besides user interaction and manual work to mark the facial landmarks [Nguyen et al., 2009].

Recent works were devoted to the study of single view-based 3D face synthesis. The

works of [Sheng et al., 2008], [Nguyen et al., 2009] and [Patel and Zaveri, 2010] describe different robust 3D face synthesis systems which use a single frontal face image. None of these approaches represents the face database as a multidimensional space to generate new 3D faces from existing ones.

The work of [Buck et al., 2000] uses a frontal face image as input to create non-photorealistic faces. Using an initial dataset of a hand-drawn character, with six mouth and four eye expressions, the user manually establishes the correspondence between hand-drawn elements and similar expressions given by photographs. Given a new user expression, a tracking system finds particular face features and tries to recreate a hand-drawn character using a combination of the existing ones. Similar to our approach, the authors propose to reduce the dimensionality of the training data to find a Delaunay triangulation which will aid the algorithm in finding the weights for the interpolation. However, this step is automatic and does not involve the user, as opposed to what we propose.

Some works propose to use interpolation to achieve animation between poses or facial expressions. Similarly to this work, [Lewis et al., 2000] uses RBFs to create such interpolation. However, my approach provides the projection space over which the user can navigate and abstract the various parameters that may be involved in the process.

The first step of our inverse projection framework for synthesizing faces and expressions is to find a 2D representation of the dataset. The 2D projection space becomes the resampling media over which the user can create a point $p \in \mathbb{R}^2$ (Section 6.3.2). The point p is transformed into a multidimensional point q (Section 6.1). Finally, the point q is passed as input to a *face generator* procedure, which returns the geometric and texture information of the new face (Section 6.3.3). This workflow is presented in Figure 6.5.

The application we present in this section is based on the work of [Mena-Chalco et al., 2009], that uses a single 2D photograph as input to generate a new 3D face model. In their work, the face synthesis is accomplished through a training set with 210 previously

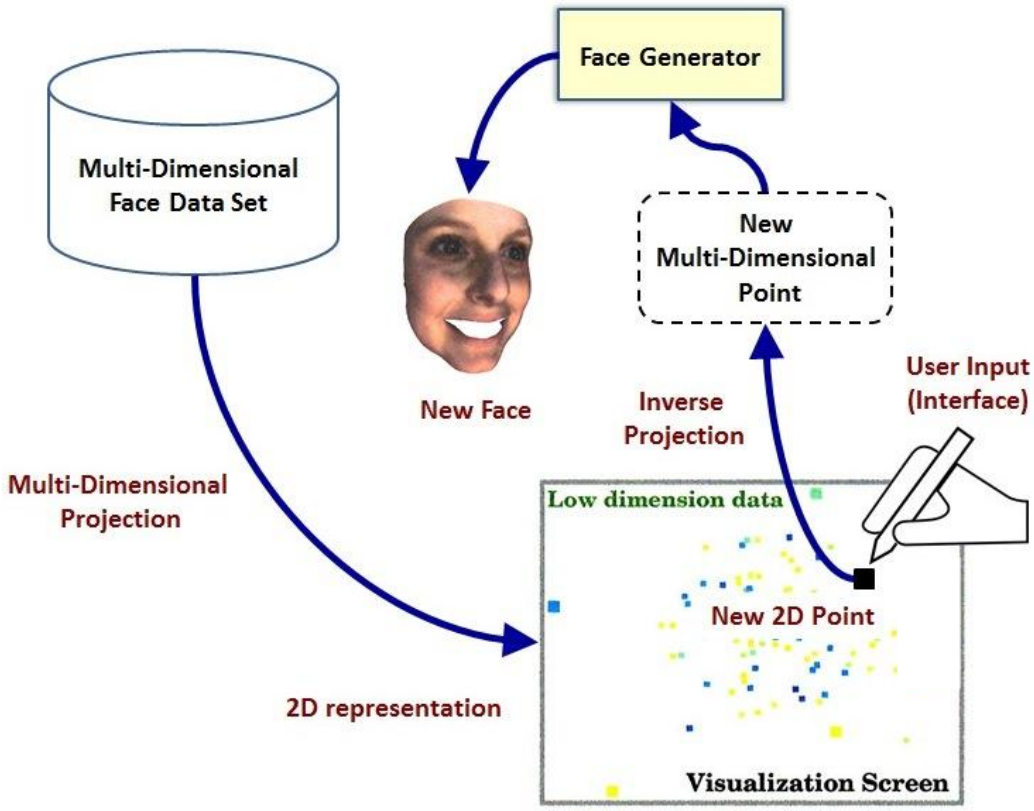


Figure 6.5: Synthesis of faces and expression application workflow. Given a dataset of faces represented in a multidimensional space, a multidimensional projection technique is applied resulting in a 2D representation of the data. The user can create new 2D points that are converted into the original dimensionality of the dataset of faces through inverse projection. The new multidimensional point undergoes a series of calculations that give rise to a new facial model.

acquired faces with different expressions, each of which carrying geometric and texture information. The geometric and texture information of each face is encoded as a multidimensional vector, what makes this an excelent demonstration application for my inverse projection method.

6.3.1 Input Data Set

The dataset used in this work was created at the VISGRAF Lab, National Institute of Pure and Applied Mathematics (IMPA) [Computer Graphics Laboratory, 2012] and consists of $n = 210$ faces acquired from 30 individuals performing seven basic expressions (happy,

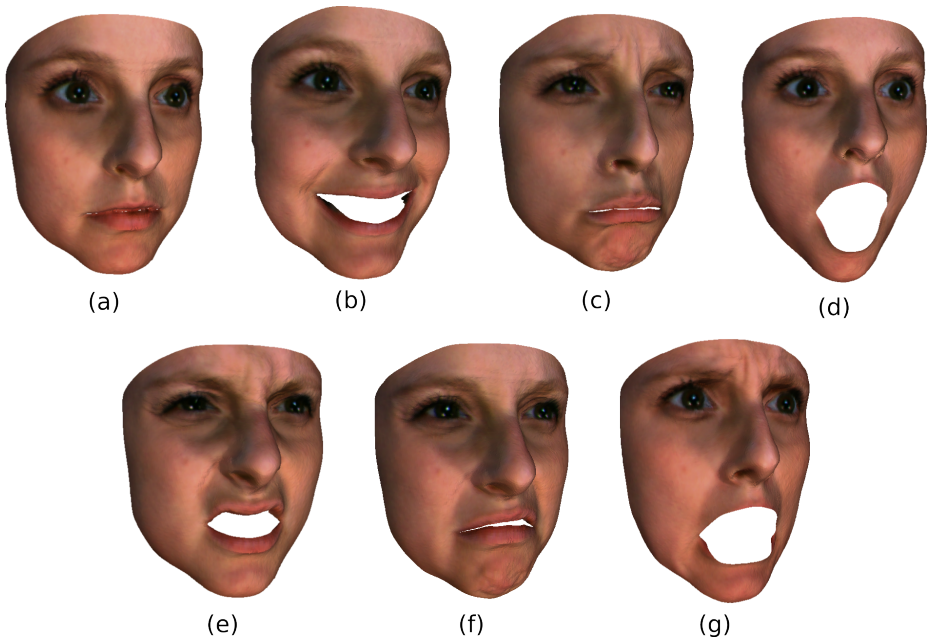


Figure 6.6: A geometric model with mapped texture of one of the individuals in the dataset performing seven basic expressions: (a) neutral; (b) happy; (c) sad; (d) surprise; (e) anger; (f) disgust; and (g) fear

sad, anger, disgust, surprise, fear and neutral). Figure 6.6 presents the face models of one of the subjects in the dataset performing those seven expressions.

Each face in the dataset contains geometric and texture information measured in $M = 9,648$ corresponding points of the model. For example, the i-th face in the dataset can be represented by geometric (L_i^g) and texture (L_i^t) vectors, as seen below:

$$L_i^g = (x_{i1}, \ldots, x_{iM}, y_{i1}, \ldots y_{iM}, z_{i1}, \ldots z_{iM}),$$
$$L_i^t = (r_{i1}, \ldots, r_{iM}, g_{i1}, \ldots g_{iM}, b_{i1}, \ldots b_{iM}),$$

where (x_{ij}, y_{ij}, z_{ij}) and (r_{ij}, g_{ij}, b_{ij}) are, respectively, 3D geometric coordinates and RGB values of the texture of the j-th point. Thus, the representation of the dataset of faces is given by matrices L^g and L^t, whose i-th rows are L_i^g and L_i^t, respectively:

$$L^g = \begin{bmatrix} x_{11} & y_{11} & z_{11} & \dots & x_{1j} & y_{1j} & z_{1j} & \dots & x_{1M} & y_{1M} & z_{1M} \\ & \vdots & & \vdots & & \vdots & & \vdots & & \vdots & \\ x_{i1} & y_{i1} & z_{i1} & \dots & x_{ij} & y_{ij} & z_{ij} & \dots & x_{iM} & y_{iM} & z_{iM} \\ & \vdots & & \vdots & & \vdots & & \vdots & & \vdots & \\ x_{n1} & y_{n1} & z_{n1} & \dots & x_{nj} & y_{nj} & z_{nj} & \dots & x_{nM} & y_{nM} & z_{nM} \end{bmatrix}$$

$$L^t = \begin{bmatrix} r_{11} & g_{11} & b_{11} & \dots & r_{1j} & g_{1j} & b_{1j} & \dots & r_{1M} & g_{1M} & b_{1M} \\ & \vdots & & \vdots & & \vdots & & \vdots & & \vdots & \\ r_{i1} & g_{i1} & b_{i1} & \dots & r_{ij} & g_{ij} & b_{ij} & \dots & r_{iM} & g_{iM} & b_{iM} \\ & \vdots & & \vdots & & \vdots & & \vdots & & \vdots & \\ r_{n1} & g_{n1} & b_{n1} & \dots & r_{nj} & g_{nj} & b_{nj} & \dots & r_{nM} & g_{nM} & b_{nM} \end{bmatrix},$$

where n is the number of faces in the data set.

As detailed in the work of [Mena-Chalco et al., 2009], the representation of the dataset of faces is simplified by performing a principal component analysis (PCA) in both the geometric and texture matrices separately. In this process, vector bases for geometry (E^g) and texture (E^t) are formed, each with the first 184 principal components that preserve at least 95% of the original information present in L^g and L^t. We refer the reader back to Chapter 2, Section 2.3.1, where we present how to derive PCA for a given matrix. E^g and E^t can be written as

$$E^g = \begin{bmatrix} e^g_1 & \dots & e^g_i & \dots & e^g_{184} \end{bmatrix}; \qquad E^t = \begin{bmatrix} e^t_1 & \dots & e^t_i & \dots & e^t_{184} \end{bmatrix}, \tag{6.5}$$

where e^g_i and e^t_i are the i-th principal components the geometric and texture information,

respectively. The vectors L_i^g and L_i^t are then projected into the vector bases E^g and E^t, creating coefficients vectors α_i^g and α_i^t for each face in the dataset:

$$(E^g)^T(L_i^g)^T = \alpha_i^g; \qquad (E^t)^T(L_i^t)^T = \alpha_i^t. \tag{6.6}$$

In the work presented above, the bases E^g, E^t, along with vectors α_i^g and α_i^t, are used as the training data to synthesize 3D face models given 2D textures of a frontal face as input. In their synthesis workflow, the input texture x_t is projected into E^t and the texture coefficients α_x^t are calculated. The last step is to find the geometric coefficients α_x^g based on an equivalence with α_x^t, and the new 3D face model is constructed.

In this thesis, we want to create a new face, both texture and geometry, using a coefficient vector α_x^t as input, i.e., given an 184-dimensional vector α_x^t, we calculate x^t and α_x^g. (more details in Section 6.3.3). To create α_x^t using the inverse projection framework, the input dataset needs to be in the same format of α_x^t. Therefore, each face i in the input dataset is represented by its coefficient vector α_i^t, giving us an initial dataset of 184 dimensions. In the next section, I describe the interface for this application.

6.3.2 Interface

As shown in Figure 6.7, the user interface provided by inverse projection is a screen that depicts the various faces of the input dataset, positioned according to the results of a multidimensional projection technique. With the original dimensionality being reduced from 184 to 2, the parameter-space of the faces can be explored in the 2D layout. By using control points (Chapter 3), one can reorganize the layout of the faces. Once a pleasing layout is achieved, the user can create a new face by directly generating a 2D point on the same screen where the projection is displayed. The complexities of the multidimensional space are completely hidden from the user, who can focus on creating points on regions of interest in the screen. When a 2D point is created, the resulting face is displayed in a separate

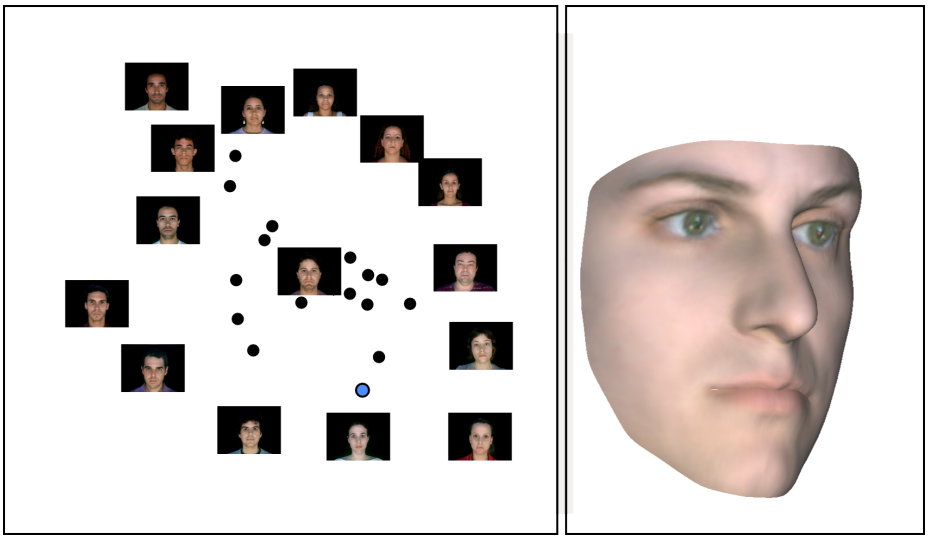


Figure 6.7: Interface for facial synthesis application. The left screen contains the projection of the initial dataset of faces (in this example, 30 faces with neutral expression are used). The control points are rendered with their corresponding facial textures, while the remaining faces are represented as black circles – this is done to prevent cluttering of the projection space, but one can select any instance and see the face image. The blue circle indicates a user-created point, which resulted in the face model depicted in the right screen.

3D visualizer window, as depicted in Figure 6.7. All the process is done in real time.

The process to transform the 2D point into the original multidimensional space is given by the inverse projection methodology described in this chapter. In the next section, we present the process to create the geometry and texture of the new face given the multidimensional vector produced by inverse projection.

6.3.3 Face Generation Process

Let t_0 and g_0 be, respectively, the average of the textures and geometries of the training dataset. Given α_x^t, the process of creating a new face is divided into texture and geometry reconstructions. The reconstruction of texture x_t is straightforward and consists of applying vector α_x^t into basis E^t:

$$x_t = E^t \alpha_x^t + t_0. \tag{6.7}$$

With regards to the geometry reconstruction, Since vector α_x^g is not known at this stage, the geometry is achieved in three main steps:

1. The texture coefficients of the faces in the dataset are used to calculate coefficients s_x,

$$\alpha^t s_x = \alpha_x^t, \tag{6.8}$$

   where α^t is a matrix with i-th row being the texture coefficient α_i^t of the i-th face of the training set; s_x is calculated through a least-squares process, in such a way that $\|\alpha_x^t s_x - \alpha^t\|$ is minimized.

2. Calculate the weight vector α_x^g, as:

$$\alpha_x^g = \alpha^g s_x \tag{6.9}$$

3. Finally, the geometry x^g is calculated by applying vector α_x^g into basis E^g:

$$x^g = E^g \alpha_x^g + g_0. \tag{6.10}$$

Thus, given a vector α_x^t through inverse projection, a new face model can be constructed following Equation (6.7), for texture, and the three-step algorithm (Equations (6.8)– (6.10)), for geometry.

6.3.4 Expression Transfer

Once a new face model is created, it can be interesting also to create its different expressions. As part of this application, we provide an "expression transfer" mechanism that

permits the change of a face model's primary expression into any of the other six basic expressions. This procedure is done using "displacement vectors" that indicate the direction to which each vertex in the geometric model needs to be displaced to achieve the desired expression. Although the technique described in this section is not the most advanced regarding expression transfer, its simplicity enables straightforward implementation and yields good results for the given application.

As previously discussed, the input dataset contains 210 face models of 30 individuals performing the seven basic expressions each. The displacement vectors are calculated using the geometric information of these models. First, a geometric mean face is computed for each of the seven expressions $\bar{x}_i^g, i =$ neutral, happy, sad, anger, disgust, surprise and fear. We calculate displacement vectors $\Delta x_{neutral \to i}$ between each expressions' mean and the neutral mean as

$$\Delta x_{\text{neutral} \to i} = \bar{x}_{\text{neutral}}^g - \bar{x}_i^g. \tag{6.11}$$

Given a face geometry x_i^g with expression i, we can have expression j transferred into it by doing

$$x_{i \to j}^g = x_i^g - \Delta x_{Neutral \to i} + \Delta x_{Neutral \to j}. \tag{6.12}$$

We may understand this equation as a two-step process: first, the original geometry is displaced to achieve the neutral expression; second, the neutral expression is displaced to the desired j expression. Figure 6.8 illustrates this process in a "surprise-to-happy" expression transfer example.

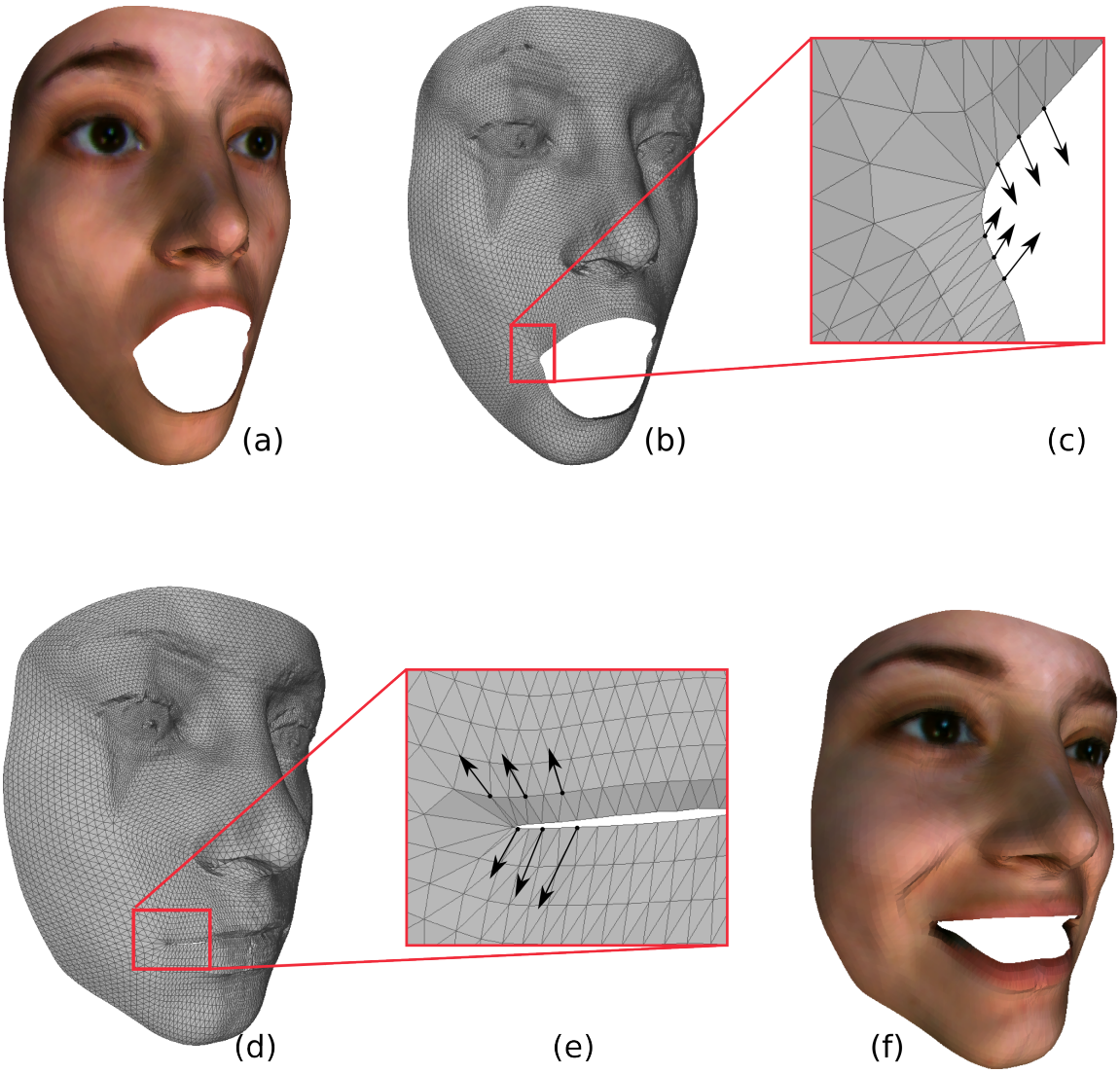


Figure 6.8: Expression Transfer: from "surprise" to "happy" example. (a) A face model performing a surprise expression is given as input, the goal is to get a face model of the same subject showing a happy expression. (b) Only the geometric information is used and (c) every vertex is displaced to achieve the neutral expression, by doing $x_i^g = x_i^g - \Delta x_{Neutral \rightarrow Surprise}$. (d) Once the neutral expression is obtained, (e) vertices are now displaced to make the happy expression, by doing $x_i^g = x_i^g + \Delta x_{Neutral \rightarrow Happy}$. The final face model is illustrated in (f).

Notice that the above procedure only modifies the geometry of the face while the texture remains intact. Once we have the expression mapped into the face model, it is important to recover the α_x^t vector associated with the new face, to be able to load it into the system for further exploration. We do so by computing the reverse of what is shown is Section 6.3.3, i.e., we start with the geometric information and recover α_x^t, following the three steps below:

1. The geometric coefficient vector α_x^g is calculated by applying vector $(x_{i\rightarrow j}^g - g_0)$ into basis E^g:

$$\alpha_x^g = (E^g)^T (x_{i\rightarrow j}^g - g_0), \tag{6.13}$$

2. s_x is calculated as:

$$\alpha^g s_x = \alpha_x^g, \tag{6.14}$$

   where where α^g is a matrix with i-th row being the geometric coefficient α_i^g of the i-th face of the training set; s_x is calculated through a least-squares process, in such a way that $\|\alpha_x^g s_x - \alpha^g\|$ is minimized.

3. α_x^t is finally calculated by applying vector α_t into s_x:

$$\alpha_x^t = \alpha_t s_x. \tag{6.15}$$

6.4 Discussion

In this section, I present some results achieved in the face-synthesis application with inverse projection. The experiments presented in this section were executed in a 2.80 GHz Intel Core i7 CPU 860 with 12 GB of RAM. The computational time spent in both the inverse

projection and the face generator processes is generally in the order of the milliseconds. In fact, the results are achieved in real time, as no noticeable delays are observed from the moment the user creates a new 2D point to the moment when the new face model is displayed. The training phase is the most computationally demanding of the face generator process since a principal component analysis needs to be carried out as well as a least-squares solution. However, this process only takes a few seconds and can be done one time only as the application starts.

In the experiments, I used every face in the input data set as an RBF center. This decision was considered necessary in this application because it allows users to recreate every face in the dataset with all the faces having equal importance in the inverse mapping. The kernel function used was the multiquadrics with $c = \varepsilon = 0$, i.e., $\phi(r) = r$.

As explained in Section 5.7, the training set of faces used in this work is composed of 210 faces of 30 individuals performing seven different expressions. Note that this complete 210 faces dataset is used to reconstruct the 3D and texture information of the new face, but a subset of it can be utilized in the inverse projection process. In fact, we experimented with three variations of this dataset:

- Experiment 1 (neutral dataset): only the neutral expressions of the 30 individuals are used, i.e., the dataset contains 30 faces. In this experiment, the focus is to create a new character with a neutral expression.

- Experiment 2 (individual dataset): all the expressions of one single individual are used, i.e., the dataset contains seven faces. In this experiment, the focus is to create transitional expressions of a single character.

- Experiment 3 (diverse dataset): All the 210 faces are used. The focus of this experiment is the creation of new characters and different expressions.

- Experiment 4 (expression transfer):) by using the neutral dataset initially,

a new neutral character is created. The remaining six expressions of this character are generated (see Section 6.3.4) and loaded into the program to create its transitional expressions.

6.5 Results

In this section, the results of all four experiments are presented. The evaluation of the experiments is presented in the following section, and is done through an informal used study to understand the quality of the face models generated through the proposed inverse projection framework.

Experiment 1: Figure 6.9 presents some results achieved with the dataset of only neutral expressions of 30 individuals. We show five examples of user-generated faces. We strategically positioned the 3D face models on top of the projection space, in the position the 2D point was generated. It is clear that the user-generated faces are more or less similar to specific input faces depending on the position the new 2D point is created.

Experiment 2:

Figure 6.10 presents an example of using only one individual with the seven expressions as input data. We demonstrate how one can create expressions in between existing expressions. Because of the continuity and smoothness provided by RBF projection, the sequence of continuous user-generated faces presents a smooth transition and an animation-like effect.

To illustrate this example, we create a path of 20 user-generated points (blue points in the projection space), and we present the face models generated in those positions. Going from “surprise” to “fear” (points 1 to 4); from “fear” to “neutral” (5 to 7); from “neutral” to “angry” (8 and 9); from “angry” to “happy” (10 to 12); from “happy” to “disgust” (13

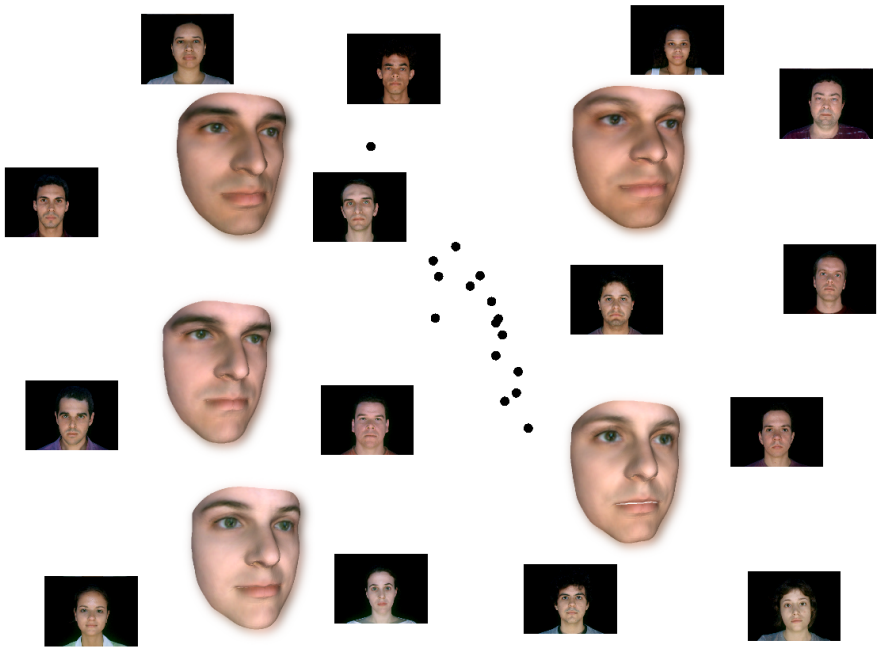


Figure 6.9: Example using 30 individuals with a neutral expression. The textures with black background represent the control points of the multidimensional projection. Black points represent the remaining faces in the dataset. The 3D face models are the user-generated faces through inverse projection.

and 14); from “disgust” to “sad” (15 and 16); and from “sad” to “surprise” (17 to 20). The faces generated in points 3, and 4 are good examples of the blending of two expressions, namely surprise and fear. Note how the half-open mouth indicates surprise, whereas the expression of the eyes indicates fear.

Note that, as the user-generated point gets close to a particular projected point, the face model becomes more and more similar to the original face. The interpolation condition in the RBF formulation also assures that, by creating a point at the same location as a projected point, the resulted face is the exact original model.

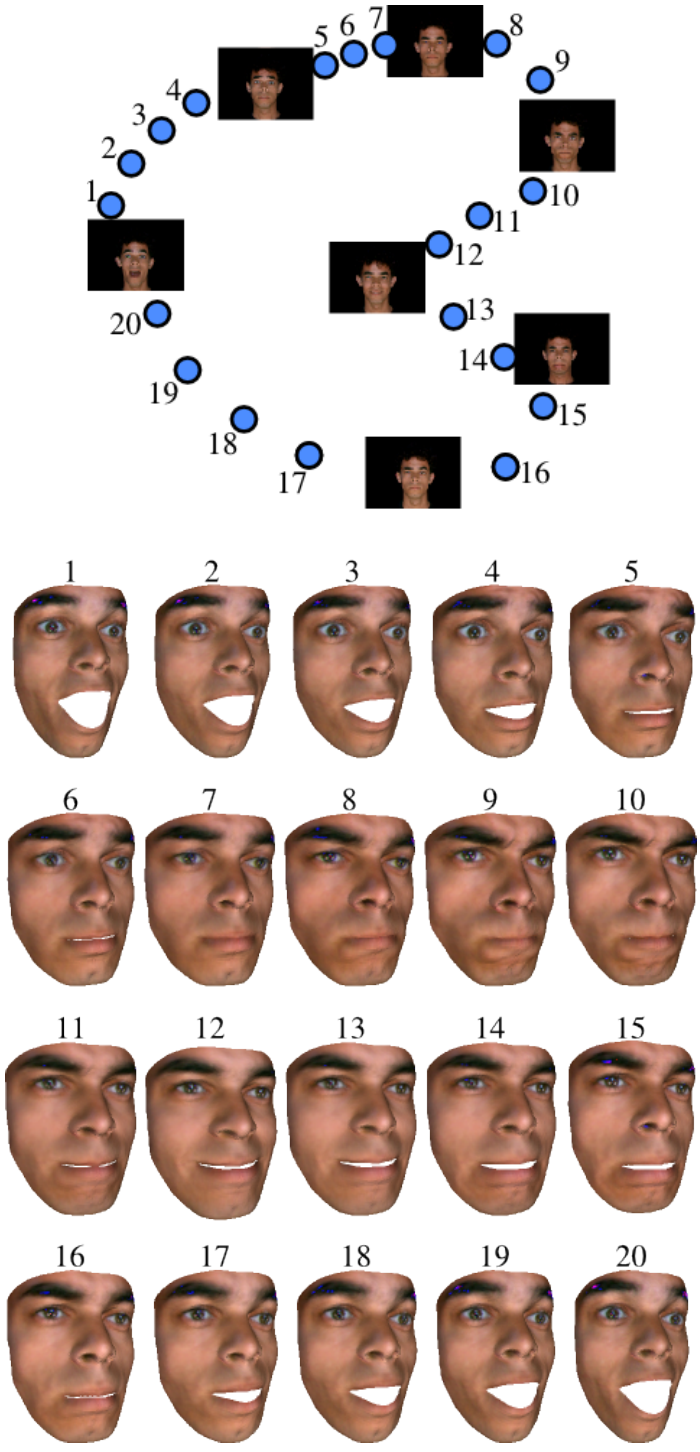


Figure 6.10: Example using only one individual performing seven expressions. All seven face models are used as control points in this case. A path is created through inverse projection, illustrated by the numbered blue points in the projection space, resulting in the depicted face models.

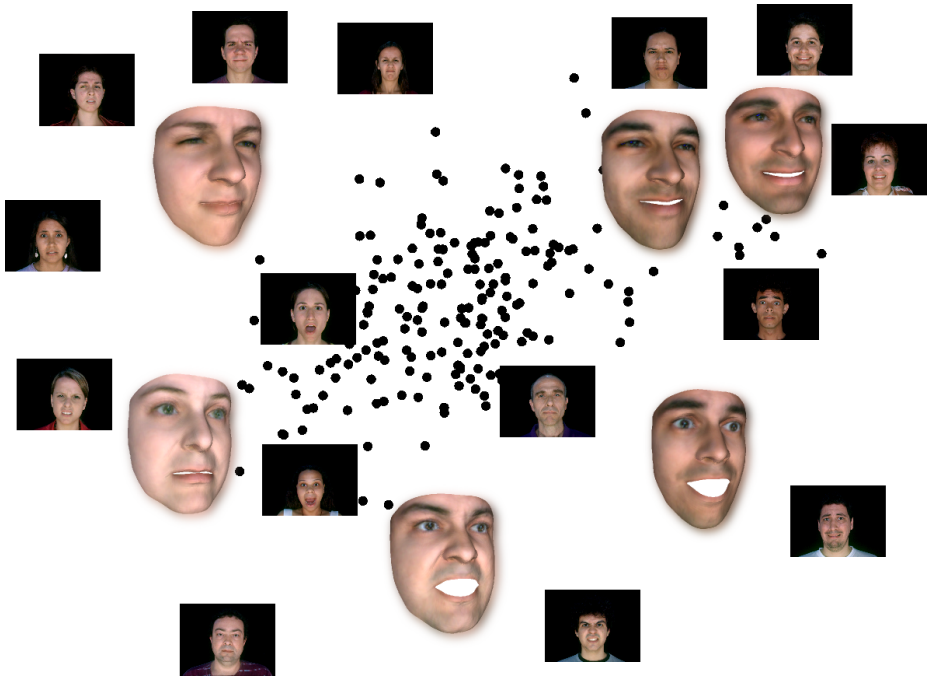


Figure 6.11: Example using the complete 210 faces dataset, with all individuals and expressions. Using such a dataset, we can generate new characters with different expressions, as illustrated by the 3D face models depicted in the projection.

Experiment 3: In Figure 6.11 we present some results achieved using the complete dataset with 210 faces. Since the given dataset has various individuals performing different expressions as input, the prototype system can generate new characters with varying expressions and expressions in between. This functionality is illustrated by six new models created through inverse projection, placed on their corresponding position of the projection space.

Experiment 4: In this example, it is shown that, besides creating a new character, all its different expressions can also be calculated using the displacement vectors explained in Section 6.3.4. Starting with the "neutral" dataset, the new character with neutral expression is defined (Figures 6.12-(a) and 6.12-(b)). The various expressions are then computed for this character (Figure 6.12-(c)), as described in Section 6.3.4. The corresponding α_x^t's of each expression are calculated and saved into a separate file, which can later be loaded into

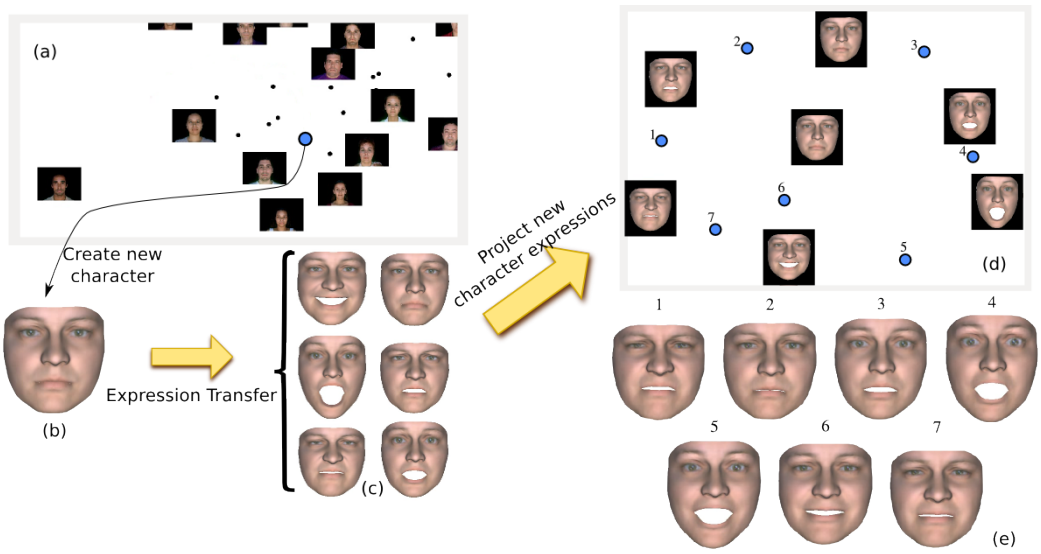


Figure 6.12: Expression transfer; (a) Projection space and user-defined point (blue); (b) New character with neutral expression; (c) New character with different expressions calculated as demonstrated in Section 6.3.4. (d) New character with different expressions loaded into inverse projection system; (e) 7 points in the projection space are created to demonstrate transitional expressions for a new character.

the inverse projection system for further exploration of the new character, as illustrated in Figure 6.12-(d). We create transitional expressions for the new character, and I show seven of them in Figure 6.12-(e).

6.6 Evaluation

An informal study was conducted to evaluate the *quality* of the face models generated by the prototype system. In this study we had 99 people from various age groups, educational levels and knowledge of computer graphics (Figure 6.13). The concept of quality here means that the new face model is (1) as realistic as the faces in the dataset ; and (2) unique in comparison with the original faces in the dataset.

For the evaluation of (1), we displayed eight faces (Figure 6.14), half of them from the original dataset and the other half created using the proposed method. We asked the participants to classify each of the face models as “Scanned” or “Synthesized.” Some face

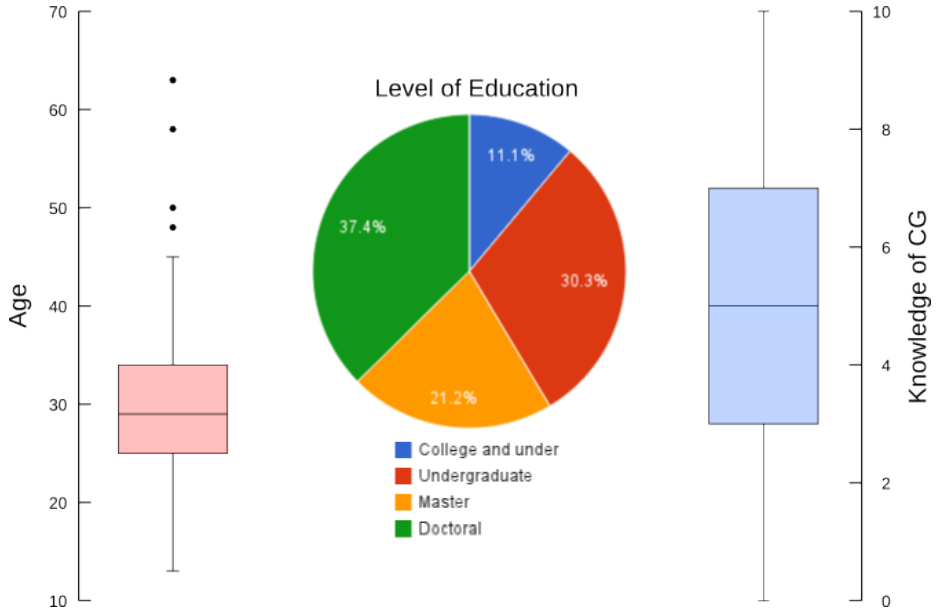


Figure 6.13: Summary of the 99 participants that responded to our informal study.

models were classified incorrectly by the majority of the participants – e.g., Face 02 (74% scanned), Face 03 (66% synthesized), Face 05 (60% scanned) and Face 06 (70% synthesized). The results for Faces 01, 07 and 08 were close to a tie, being 53%, 46% and 44% classified as Scanned, respectively. Face 04 was correctly classified by the majority, having 61% of the participants classifying it as synthesized. This set of results is summarized in Table 6.1.

Face model	**Scanned**	**Synthesized**
Face 01	52.53%	47.47%
Face 02	73.74%	26.26%
Face 03	33.33%	66.67%
Face 04	38.38%	61.62%
Face 05	59.60%	40.40%
Face 06	30.30%	69.70%
Face 07	46.46%	53.54%
Face 08	44.44%	55.56%

Table 6.1: Summary of responses acquired regarding classification (Scanned vs Synthesized) of face models depicted in Figure 6.14. Table entries indicate the percentage of users that classified a given face (row) in each category (column).

The results of this experiment support the claim that our method is capable of creating

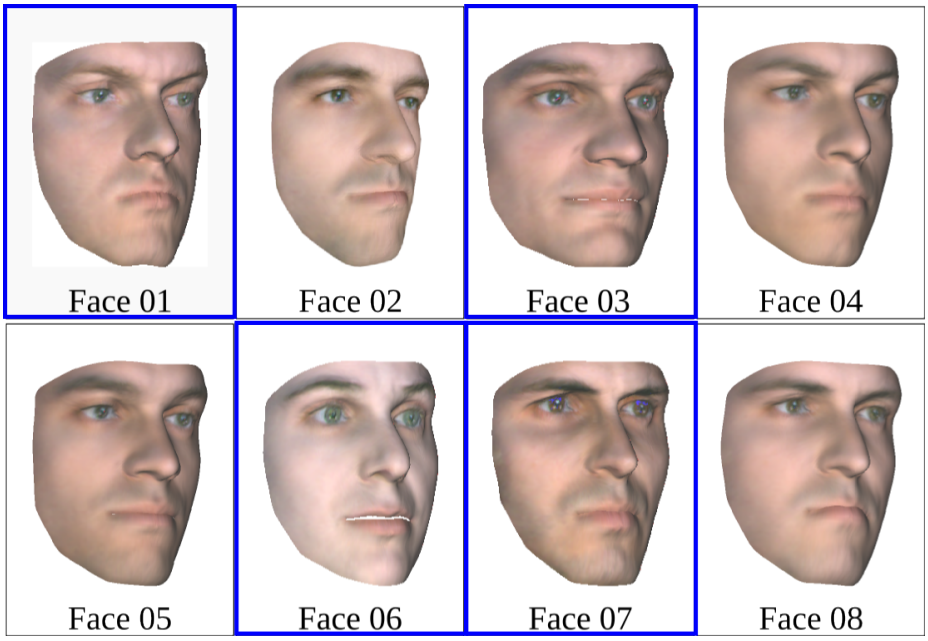


Figure 6.14: Set of faces used for the first evaluation. Faces 01, 03, 06 and 07 are from the original dataset (highlighted with blue borders); The remaining results are face models created through the prototype system.

faces that are as realistic as the ones from the original dataset since the participants did not easily distinguish the synthesized faces. For this to hold true, the chosen RBF kernel and its corresponding parameters, when any, need to be carefully selected and tuned. In this particular application, we have found that the multiquadrics kernel with parameter $c = 0$ is a good choice and gives realistic results, as was stated earlier. We reiterate, however, the importance of carefully evaluating the RBF kernels when applying this method to different datasets.

For the evaluation of (2), we presented two sets with ten faces each: the first set containing ten face models from the original dataset that were used as input to generate ten other faces, presented in the second set (Figure 6.15). We asked the participants to indicate for each face in set 2 to which face in set 1 it was more similar to, or the option to choose "NONE". Faces B and H were mostly found not to be similar to any faces in set 1 (30% and 35% respectively), and all of other faces in set 1 received less than 20.3% of association. Faces C and J also had a significant percentage of the participants indicating

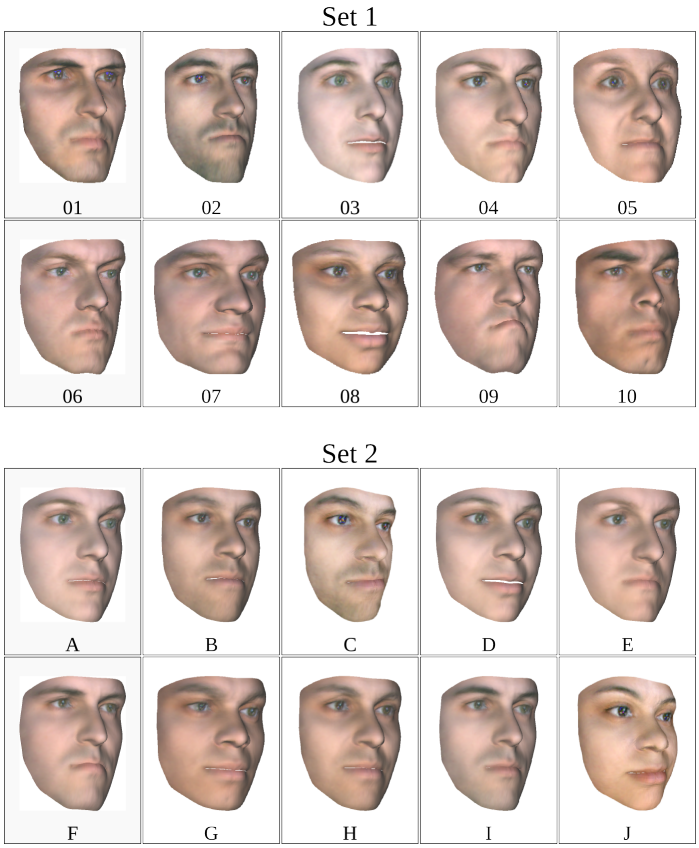


Figure 6.15: The two sets of faces used in the second evaluation. Set 1 contains faces of the original dataset, that were used as input to create the faces displayed in Set 2.

"NONE" (27% and 28%, respectively), but both were also strongly associated with a Face in set 1: Face C had 48% association with Face 02 and Face J 55% with Face 08. For all faces created by our system, "NONE" was the only common answer for all of them (except for Face 04, which had some weak associations – 2% or less - for three of the face models), indicating that our system was able to create unique faces.

Some faces were mostly associated with only one face in set 1. For example, Face A was 67% associated with Face 03, but also presented significant associations with Faces 04 and 06, 14% and 12% respectively; and Face E was associated with Face 04 by 57% of the participants and with Face 06 by 26%. Other faces were strongly associated with two faces in set 1. For example, Face D was considered to be similar to Faces 03 (39%) and Faces 08 (37%); while Face I presented 35% of association with Face 01 and 30%

with Face 02. Finally, some faces were similarly associated with 3 or more faces in set 1. For example, Face G was associated with Faces 07, 08, and 10 - 28%, 15%, and 38% respectively; and Face F was associated with Faces 01, 04, 06, and 09 - 19%, 12%, 27% and 22% respectively. All results of this set of experiments are summarized in Table 6.2.

	01	02	03	04	05	06	07	08	09	10	NONE
A	0.0%	2.0%	66.7%	14.1%	0.0%	12.1%	0.0%	1.0%	1.0%	0.0%	3.0%
B	5.1%	20.2%	2.0%	17.2%	4.0%	11.1%	3.0%	1.0%	3.0%	3.0%	30.3%
C	9.1%	48.5%	2.0%	3.0%	1.0%	4.0%	1.0%	0.0%	0.0%	4.0%	27.3%
D	1.0%	0.0%	39.4%	1.0%	6.1%	1.0%	4.0%	37.4%	0.0%	1.0%	9.1%
E	2.0%	1.0%	2.0%	56.6%	2.0%	27.3%	1.0%	0.0%	2.0%	0.0%	6.1%
F	19.2%	6.1%	1.0%	12.1%	0.0%	27.3%	0.0%	1.0%	22.2%	3.0%	8.1%
G	0.0%	1.0%	0.0%	2.0%	2.0%	0.0%	28.3%	15.2%	1.0%	38.4%	12.1%
H	3.0%	2.0%	10.1%	16.2%	6.1%	6.1%	8.1%	3.0%	1.0%	9.1%	35.4%
I	35.4%	30.3%	1.0%	4.0%	0.0%	13.1%	0.0%	0.0%	5.1%	0.0%	11.1%
J	2.0%	1.0%	0.0%	1.0%	1.0%	2.0%	1.0%	54.5%	5.1%	4.0%	28.3%

Table 6.2: Summary of responses acquired regarding similarity between synthesized faces and original, scanned, face models. Each row corresponds to a synthesized face, letter–coded from A to J; each column corresponds to a scanned face model, number-coded from 01 to 10 (Figure 6.15). Table entries indicate the percentage of users who classified the scanned face to be more similar to the synthesized one.

These results are indicative that our system can create new faces that resemble those used in the given dataset but are still unique and original, as there was no consensus amongst the participants concerning the similarities between faces from both groups. These results can be even more appreciated for the fact that the faces from Set 2 were created entirely based on faces from Set 1. Even though the number of input faces is reduced, the system still demonstrated its ability to generate unique characters. This behavior may be attributed mostly to the control points manipulation (Section 6.2), that permits multiple configurations of faces layouts and, consequently, allows greater output variety. In fact, most of the faces in Set 2 were generated after reorganizing the projection layout to achieve the desired result.

6.7 iLAMP vs. RBF

In this section, I discuss why the RBF solution presented in this thesis is more suitable for the application of face synthesis than iLAMP.

iLAMP is not continuous in its nature as it is designed to use only a few k neighbor points to create the mapping from low to high dimension. Thus, a different set of neighbors is used for each new user-defined point, making the mapping discontinuous. Continuity is an important characteristic for the face synthesis application, as it enforces continuous points in the 2D projection space to create a smooth transition of faces, giving it an animation-like effect. Of course, the discontinuity in iLAMP could be easily prevented by setting the number of neighbors equal to the number of points in the dataset, i.e., $k = N$. However, the iLAMP solution presents yet another disadvantage that is maximized when the number of neighbors k increases: it may become prone to severe distortions if the user modifies the projection layout (as seen in Section 6.2).

For each user-generated point p iLAMP searches for a local affine transformation $s_{iLAMP} : \mathbb{R}^2 \rightarrow \mathbb{R}^m$ to map p into $q \in \mathbb{R}^m$. s_{iLAMP} is the affine transformation that minimizes

$$\sum_{i=1}^{k} \beta_i \| f_{iLAMP}(y_i) - x_i \|^2, \tag{6.16}$$

i.e., it maps, as best as possible, the projected points y_i into their multidimensional counterpart x_i. The β_i weights are used to assign greater importance to the mapping of those y_i closer to the user-defined p. If distances between pairs of instances are well preserved in the projection space, this solution will yield good, non-distorted results. However, when the points in the projection space are deliberately rearranged by the user, the iLAMP solution may produce distorted results, i.e., it may not be able to generate an affine mapping s_{iLAMP} that maps y_i to points close to x_i.

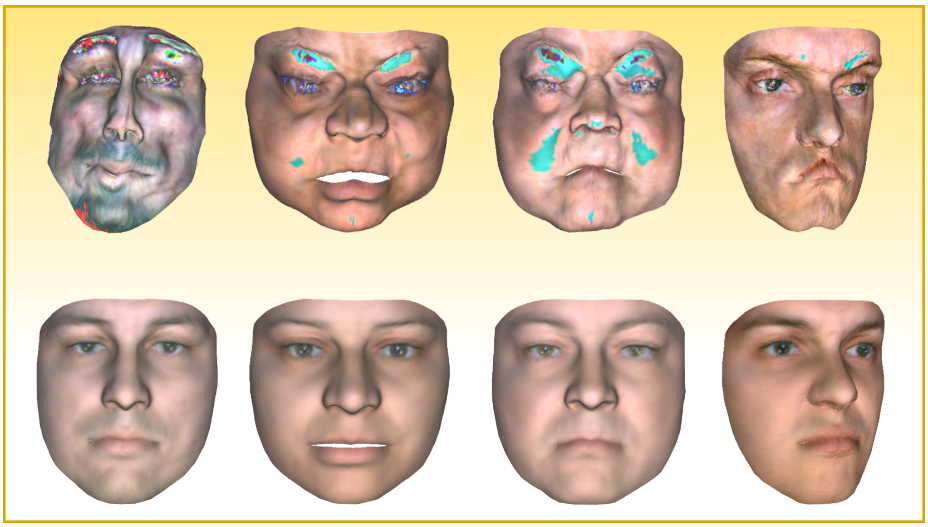


Figure 6.16: Top row: faces generate through iLAMP after rearrangement of projection space; Bottom row: corresponding faces generated through our RBF solution.

In fact, we evaluated the iLAMP error given by Equation (6.16) for $k = N$ in the neutral dataset. When points in the projection space are repositioned from its original projection, this error can become more than two times greater when compared to the error before the repositioning. Such a distorted mapping can result in points q that little have to do with the original x_i vectors, potentially resulting in deformed faces as examplified in Figure 6.16.

To summarize, in this chapter I have introduced an inverse projection technique that uses RBFs to create the mapping function. A face-synthesis application was also introduced, that utilizes the proposed inverse projection framework to create a three-dimensional geometric and texture for digitized face models.

Chapter 7

Conclusions and Future Work

In this thesis, I have presented contributions to the interactive aspects of multidimensional projection methods, in the field of data visualization and exploration.

In Chapter 3, I proposed a mechanism to systematically select control points, special data instances used to approximate the projection results and to provide a steering mechanism ideal to explore the projection space. I demonstrated that a high-quality projection result can be achieved using a very reduced number of control points and that the technique can benefit various CP-based MP methods. In Chapter 4, I introduced the *inverse projection* workflow, which provides a whole new interactive experience by allowing one to extrapolate an existing multidimensional dataset and interactively creating new instances using the 2D projection as a canvas. Chapters 5 and 6 described the two inverse projection techniques we developed, along with two applications that demonstrate the usefulness of the proposed workflow.

I believe the methods and solutions presented in this thesis have the potential to be the precursors of other techniques that could extend even further the interactive capability of MP methods. In the next sections, I will summarize and discuss the foreseen future work for each of the main contributions here presented.

7.1 Control Points Selection for Multidimensional Projection

Chapter 3 introduced a novel multidimensional projection technique, built upon the Radial Basis Functions (RBF) interpolation theory. The main contribution of the presented method is a built-in computational model for selecting control points, with a solid mathematical formulation that results in high-quality projection outputs even with a reduced number of

control points. The results presented in this thesis indicate that the proposed ROLS-based selection procedure yields a good trade-off between computational time and stress while reducing the number of control points necessary to achieve a projection with low-stress.

The fact that the proposed MP technique was built upon the solid formulation of RBF brings many possibilities for future work by itself. There are a significant number of works dedicated to the studies of RBF and to investigating specific aspects of its theory and applications, many of which we believe could be exploited in the technique proposed in this thesis. For example, the choice of the kernel function and its shape parameter(s) [1], when any, are considered a manual step in our workflow. Some works, however, propose automatic ways to select these critical components, such as [Mongillo, 2011] and [Fasshauer and Zhang, 2007]. Our methodology could be further improved if such an automatic mechanisms were incorporated to choose these appropriately according to each dataset.

Another research direction would be to create local projections using the Gaussian kernel and its shape parameter to determine the radial influence of each control point. Local projections, such as LAMP [Paulovich et al., 2010], have the advantages of being more computationally efficient and giving useful results in a broader range of datasets, as discussed by [de Silva and Tenenbaum, 2002]. These aspects would provide more flexibility if incorporated into the method proposed in this thesis.

Regarding control points selection, I have demonstrated how ROLS is capable of selecting a reduced number of meaningful instances while maintaining the projection quality. There are, however, techniques other than ROLS designed to perform the same task, such as the ones described in [Uykan and Guzelis, 1997] and [Zhao et al., 2002]. I believe that applying such techniques to our MP CP-selection methodology would be an important future work to be considered. Another related topic would be to investigate the possibility of creating synthetic control points, i.e., generate instances not necessarily in the original

[1] We refer the reader to Section 2.6 in Chapter 2 for a formal description of kernel functions and shape parameters

dataset to act as control points. I expect this could benefit scenarios where the dataset has many outliers, for instance, and synthetic CPs could better represent the overall dataset.

Finally, regarding implementation, the software prototypes developed as part of this thesis could be integrated with a GPU-based implementation to accelerate the computation of the projection. The work of [Brandstetter and Artusi, 2008] presents a GPU implementation to compute the RBF scalars (given by the solution of the linear system described in Equation (2.9) in Chapter 2), and a computational cost reduction of two orders of magnitude is reported. Furthermore, the application of the RBF to approximate the projection results, given by Equation (2.6), could also be parallelized.

7.2 Inverse Projection

Both inverse projection methodologies described in chapters 5 and 6 of this thesis have a very solid mathematical foundation. The accuracy of the proposed techniques was attested in qualitative and quantitative experiments.Moreover, the applications presented in Sections 5.7 and 6.3, demonstrate that inverse projection can provide a good alternative for making the process of parameter space exploration a more interactive one.

Regarding the overall inverse projection experience, I believe the process could be further enhanced by providing more feedback before starting any data exploration and analytics. In the current design, projected points are the only available guidance used to explore empty regions. This aspect could be addressed by integrating additional interactive visualization and visual analytics techniques. The resulting visual cues would allow users to gain better and new insights into understanding the data and its uncertainties in the areas of interest. Taking the face-synthesis application as an example, a few sample face models could be displayed in areas of the projection where interactive data exploration is likely to happen. These models could be automatically generated, without user intervention, and displayed to guide the interactive data exploration and analytics.

Providing more control over the inverse projection set up is another aspect we believe worth exploring. For example, incorporating the capability to manually configure the weight each instance should have in the inverse projection calculation. So far, this effect can be partially achieved using control points replacement, as described in Section 6.2. A manual setup, however, could expand the diversity of the inverse projection outputs.

Along the same lines, a user-input specification of the radius of influence could also provide more control over the outcome of inverse projection. This radius would specify which instances to be used in the calculation and which ones to be excluded. This functionality could be achieved by allowing the user to draw a closed region in the projection, delimiting the instances to be used. Again, we believe this could potentially result in more diversity of the possible inverse projection outcomes.

Regarding the iLAMP workflow, there are still some points to be improved. One issue is the determination of the number of neighbors k, as the system expects the user to provide this parameter value. Results presented in Section 5.4 indicate that k is an important parameter that should be carefully determined to have the best iLAMP results. So far we do not have an automatic way to determine such parameter, and we depend on heuristic evaluations to decide this value. Furthermore, a next step for iLAMP is to apply the proposed technique into real-world problems. For instance, the history matching problem in reservoir engineering is already benefiting from the use of multidimensional projection techniques [Hajizadeh et al., 2012] and could be a possible venue to be tackled.

Finally, I would like to discuss some specific aspects of the RBF-based inverse projection and direction for future research. In the application we present, all the instances of the input dataset are used as RBF centers to construct the inverse mapping function. However, datasets with many points might need to have a more careful selection of control points. I presented some alternatives in Section 6.1.3, but a more detailed inspection about the proposed methods still needs to be conducted.

A more thorough investigation regarding the RBF kernel functions ϕ is also an interesting related research direction, along with the lines I previously discussed in Section 7.1. In those examples, I used $\phi(r) = r$, but it would be interesting if an appropriate kernel function could be determined based on the dataset. We also intend to investigate the use of the Gaussian kernel and its shape parameter as a way to give the user some control over the radial of influence of each face in the inverse projection. As a last consideration about RBF kernels, in the future, I plan to evaluate the use of polynomial extensions to the techniques I proposed in this thesis. Studies have shown that the use of polynomial terms in RBF kernel functions can improve the function approximation in some cases [Buhmann, 2003]. The polynomial terms permits, for instance, the creation of a function that samples a sphere embedded in $\mathbb{R}^n$ precisely.

Regarding applications, the algorithms and techniques I proposed in this thesis could be expanded and applied in different domains and problems in science, engineering, and content creation for visual effects and animations e.g., [Schulz and Velho, 2011].

Bibliography

[A. Asuncion, 2007] A. Asuncion, D. N. (2007). UCI machine learning repository.

[Amorim et al., 2015] Amorim, E., Brazil, E. V., Mena-Chalco, J. P., Velho, L., Nonato, L. G., Samavati, F. F., and Sousa, M. C. (2015). Facing the high-dimensions: Inverse projection with radial basis functions. *Computers & Graphics*, 48:35–47.

[Amorim et al., 2014] Amorim, E., Brazil, E. V., Nonato, L. G., Samavati, F., and Sousa, M. C. (2014). Multidimensional projection with radial basis function and control point selection. In *IEEE Pacific Visualization*, pages 209–216.

[Amorim et al., 2012] Amorim, E., Vital Brazil, E., Daniels II, J., Joia, P., Nonato, L. G., and Costa Sousa, M. (2012). iLAMP: Exploring high-dimensional spacing through backward multidimensional projection. In *IEEE VAST*, pages 53–62. IEEE Computer Society.

[Anderson et al., 1990] Anderson, E., Bai, Z., Dongarra, J., Greenbaum, A., McKenney, A., Du Croz, J., Hammerling, S., Demmel, J., Bischof, C., and Sorensen, D. (1990). LAPACK: a portable linear algebra library for high-performance computers. In *Proceedings of the 1990 ACM/IEEE conference on Supercomputing*, Supercomputing '90, pages 2–11.

[Arya et al., 1998] Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., and Wu, A. Y. (1998). An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM (JACM)*, 45(6):891–923.

[Aupetit, 2007] Aupetit, M. (2007). Visualizing distortions and recovering topology in continuous projection techniques. *Neurocomputing*, 70(7-9):1304–1330.

[Bache and Lichman, 2013] Bache, K. and Lichman, M. (2013). UCI machine learning repository, http://archive.ics.uci.edu/ml.

[Baker, 2005] Baker, K. (2005). Singular value decomposition tutorial. *The Ohio State University*, 24.

[Bennett and Hays, 1960] Bennett, J. and Hays, W. (1960). Multidimensional unfolding: Determining the dimensionality of ranked preference data. *Psychometrika*, 25(1):27–43.

[Blanz and Vetter, 1999] Blanz, V. and Vetter, T. (1999). A Morphable Model for the Synthesis of 3D Faces. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 187–194, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

[Brandstetter and Artusi, 2008] Brandstetter, A. and Artusi, A. (2008). Radial basis function networks gpu based implementation. *IEEE Transaction on Neural Network*, 19(12):2150–2161.

[Bruckner and Möller, 2010] Bruckner, S. and Möller, T. (2010). Result-Driven Exploration of Simulation Parameter Spaces for Visual Effects Design. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1467–1475.

[Buck et al., 2000] Buck, I., Finkelstein, A., Jacobs, C., Klein, A., Salesin, D. H., Seims, J., Szeliski, R., and Toyama, K. (2000). Performance-Driven Hand-Drawn Animation. In *NPAR 2000 : First International Symposium on Non Photorealistic Animation and Rendering*, pages 101–108.

[Buhmann, 2003] Buhmann, M. D. (2003). *Radial Basis Functions*. Cambridge University Press, New York, NY, USA.

[Chalmers, 1993] Chalmers, M. (1993). Using a landscape metaphor to represent a corpus of documents. pages 377–390. Springer.

[Chang et al., 2003] Chang, Y., Hu, C., and Turk, M. (2003). Manifold of facial expression. In *AMFG*, pages 28–35.

[Chen et al., 1996] Chen, S., Chng, E. S., and Alkadhimi, K. (1996). Regularized orthogonal least squares algorithm for constructing radial basis function networks. *International Journal of Control*, 64(5):829–837.

[Chen et al., 1991] Chen, S., Cowan, C. F. N., and Grant, P. M. (1991). Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks*, 2(2):302–309.

[Chen et al., 2009] Chen, Y., Wang, L., Dong, M., and Hua, J. (2009). Exemplar-based visualization of large document corpus. *IEEE Transactions on Visualization and Computer Graphics*, 15:1161–1168.

[Cleveland and McGill, 1988] Cleveland, W. C. and McGill, M. E. (1988). *Dynamic Graphics for Statistics*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition.

[Cohen and Massaro, 1993] Cohen, M. M. and Massaro, D. W. (1993). Modeling coarticulation in synthetic visual speech. In *Models and Techniques in Computer Animation*, pages 139–156. Springer-Verlag.

[Computer Graphics Laboratory, 2012] Computer Graphics Laboratory, I. (2012). VISGRAF faces database. `http://app.visgraf.impa.br/database/faces`. [Online; accessed 31-March-2014].

[Cox and Cox, 2000] Cox, T. and Cox, M. (2000). *Multidimensional Scaling*. Chapman and Hall/CRC, 2nd edition edition.

[Daniels II et al., 2010] Daniels II, J., Anderson, E., Nonato, L., and Silva, C. (2010). Interactive vector field feature identification. *IEEE Transactions on Visualization and Computer Graphics*, 16:1560–1568.

[de Silva and Tenenbaum, 2002] de Silva, V. and Tenenbaum, J. B. (2002). Global versus local methods in nonlinear dimensionality reduction. *Advances in Neural Information Processing Systems 15.*, pages 705–712.

[de Silva and Tenenbaum, 2004] de Silva, V. and Tenenbaum, J. B. (2004). Sparse multidimensional scaling using landmark points. Technical report, Stanford.

[Deboeck and Kohonen, 2010] Deboeck, G. and Kohonen, T. (2010). *Visual Explorations in Finance: With Self-Organizing Maps*. Springer Finance. Springer.

[Demartines and Herault, 1997] Demartines, P. and Herault, J. (1997). Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets. *IEEE Transactions on Neural Networks*, 8(1):148–154.

[Eades, 1984] Eades, P. (1984). A Heuristic for Graph Drawing. volume 42, pages 149–160.

[Faloutsos and Lin, 1995] Faloutsos, C. and Lin, K.-I. (1995). Fastmap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. *SIGMOD Rec.*, 24(2):163–174.

[Fasshauer and Zhang, 2007] Fasshauer, G. and Zhang, J. (2007). On choosing optimal shape parameters for RBF approximation. *Numerical Algorithms*, 45(1):345–368.

[Few, 2013] Few, S. (2013). *Data Visualization for Human Perception*. The Interaction Design Foundation, Aarhus, Denmark.

[Fisher, 1936] Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(7):179–188.

[Fukumizu et al., 2004] Fukumizu, K., Bach, F. R., and Jordan, M. I. (2004). Dimensionality reduction for supervised learning with reproducing kernel hilbert spaces. *Journal of Machine Learning Research*, 5(Jan):73–99.

[Gavin, 2011] Gavin, H. (2011). The levenberg-marquardt method for nonlinear least squares curve-fitting problems. *Dept. Civil and Environmental Engineering, Duke University*.

[Golub and Van Loan, 2012] Golub, G. H. and Van Loan, C. F. (2012). *Matrix computations*, volume 3. JHU Press.

[Hajizadeh et al., 2012] Hajizadeh, Y., Amorim, E., and Costa Sousa, M. (2012). Building trust in history matching: The role of multidimensional projection. *SPE 152754, 2012 SPE EUROPEC Conference*.

[He et al., 2005] He, X., Yan, S., Hu, Y., Niyogi, P., and Zhang, H.-J. (2005). Face Recognition using Laplacian Faces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(3):328–340.

[Igarashi et al., 2005] Igarashi, T., Moscovich, T., and Hughes, J. F. (2005). Spatial keyframing for performance-driven animation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 107–115. ACM.

[Inselberg and Dimsdale, 1990a] Inselberg, A. and Dimsdale, B. (1990a). Parallel coordinates: a tool for visualizing multi-dimensional geometry. *VIS 90: Proceedings of the 1st Conference on Visualization*, pages 361–378.

[Inselberg and Dimsdale, 1990b] Inselberg, A. and Dimsdale, B. (1990b). Parallel coordinates: a tool for visualizing multi-dimensional geometry. In *Proceedings of the 1st conference on Visualization '90*, VIS '90, pages 361–378, Los Alamitos, CA, USA. IEEE Computer Society Press.

[Janicke et al., 2008] Janicke, H., Bottinger, M., and Scheuermann, G. (2008). Brushing of attribute clouds for the visualization of multivariate data. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1459–1466.

[Joia et al., 2011] Joia, P., Coimbra, D., Cuminato, J. A., Paulovich, F. V., and Nonato, L. G. (2011). Local Affine Multidimensional Projection. *IEEE Trans. on Visualization and Computer Graphic*, 17(12):2563 –2571.

[Jollife, 1986] Jollife, I. (1986). *Principle Component Analysis*.

[Jolliffee, 2002] Jolliffee, I. (2002). *Principal Component Analysis*. Springer-Verlag, 3rd edition edition.

[Kandogan, 2000] Kandogan, E. (2000). Star coordinates: A multi-dimensional visualization technique with uniform treatment of dimensions. In *Proceedings of the IEEE Information Visualization Symposium, Late Breaking Hot Topics*, pages 9–12.

[Kirtzic and Daescu, 2011] Kirtzic, J. S. and Daescu, O. (2011). Face It: 3D Facial Reconstruction from a Single 2D Image for Games and Simulations. *2013 International Conference on Cyberworlds*, 0:244–248.

[Lee and Verleysen, 2007] Lee, J. A. and Verleysen, M. (2007). *Nonlinear Dimensionality Reduction (Information Science and Statistics)*. Springer, 1 edition.

[Lespinats and Aupetit, 2009] Lespinats, S. and Aupetit, M. (2009). False neighbourhoods and tears are the main mapping defaults. How to avoid it? How to exhibit remaining ones? *Quality issues, measures of interestingness and evaluation of data mining models, QIMIE*, pages 55–65.

[Levine and Yu, 2009] Levine, M. D. and Yu, Y. (2009). State-of-the-art of 3D facial reconstruction methods for face recognition based on a single 2D training image per person. *Pattern Recognition Letters*, 30(10):908–913.

[Lewis et al., 2000] Lewis, J. P., Cordner, M., and Fong, N. (2000). Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 165–172. ACM Press/Addison-Wesley Publishing Co.

[Macêdo et al., 2006] Macêdo, I., Vital Brazil, E., and Velho, L. (2006). Expression Transfer between Photographs through Multilinear AAM's. In *Proceedings of SIBGRAPI 2006 - XIX Brazilian Symposium on Computer Graphics and Image Processing*, pages 239–246. IEEE Computer Society.

[Martins et al., 2006] Martins, D. M., Xavier, C. R., Santos, E. P., Vieira, V. F., Oliveira, R. S., and dos Santos, R. W. (2006). Estimating conductivity distribution of transmural wedges of the ventricle using parallel genetic algorithms. pages 49–52.

[Martins et al., 2014] Martins, R. M., Coimbra, D. B., Minghim, R., and Telea, A. C. (2014). Visual analysis of dimensionality reduction quality for parameterized projections. *Computers & Graphics*, 41:26–42.

[Mena-Chalco et al., 2009] Mena-Chalco, J. P., Macêdo, I., Velho, L., and Cesar, Jr., R. M. (2009). 3D Face Computational Photography Using PCA Spaces. *Vis. Comput.*, 25(10):899–909.

[Mishra, 2006] Mishra, S. (2006). Performance of differential evolution and particle swarm methods on some relatively harder multi-modal benchmark functions. `http://mpra.ub.uni-muenchen.de/449/`. Accessed: 2016-06-20.

[Mongillo, 2011] Mongillo, M. (2011). Choosing basis functions and shape parameters for radial basis function methods. In *SIAM Undergraduate Research Online*, volume 4, pages 190–209.

[Nguyen et al., 2009] Nguyen, H. T., Ong, E. P., Niswar, A., Huang, Z., and Rahardja, S. (2009). Automatic and real-time 3D face synthesis. In *VRCAI'09*, pages 103–106.

[Parke, 1974] Parke, F. I. (1974). *A Parametric Model for Human Faces.* PhD thesis. AAI7508697.

[Patel and Zaveri, 2010] Patel, N. and Zaveri, M. (2010). 3d facial model construction and expressions synthesis using a single frontal face image. *International Journal of Graphics*, 1:1–18.

[Paulovich and Minghim, 2006] Paulovich, F. and Minghim, R. (2006). Text map explorer: A tool to create and explore document maps. *Information Visualization*, pages 245–251.

[Paulovich et al., 2008a] Paulovich, F., Nonato, L., Minghim, R., and Levkowitz, H. (2008a). Least square projection: A fast high-precision multidimensional projection technique and its application to document mapping. *IEEE Transactions on Visualization and Computer Graphics*, 14(3):564–575.

[Paulovich et al., 2010] Paulovich, F., Silva, C., and Nonato, L. (2010). Two-Phase Mapping for Projecting Massive Data Sets. *IEEE Trans. on Vis. Comp. Graph.*, 16(6):1281–1290.

[Paulovich et al., 2011] Paulovich, F. V., Eler, D. M., Poco, J., Botha, C. P., Minghim, R., and Nonato, L. G. (2011). Piecewise Laplacian-based Projection for Interactive Data Exploration and Organization. *Computer Graphics Forum*, 30(3):1091–1100.

[Paulovich et al., 2008b] Paulovich, F. V., Pinho, R., Botha, C. P., Heijs, A., and Minghim, R. (2008b). Pex-web: Content-based visualization of web search results. In *Proceedings of the 2008 12th International Conference Information Visualisation*, IV '08, pages 208–214, Washington, DC, USA. IEEE Computer Society.

[Pearson, 1901] Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572.

[Pekalska et al., 1999] Pekalska, E., de Ridder, D., Duin, R. P., and Kraaijveld, M. A. (1999). A new method of generalizing Sammon mapping with application to algorithm speed-up. In *5th Annual Conf. of the Advan. School for Comput. and Imag.*

[Poco et al., 2011] Poco, J., Etemadpour, R., Paulovich, F. V., Long, T. V., Rosenthal, P., Oliveira, M. C. F., Linsen, L., and Minghim, R. (2011). A framework for exploring multidimensional data with 3d projections. In *Proceedings of the 13th Eurographics / IEEE - VGTC Conference on Visualization*, EuroVis'11, pages 1111–1120, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[Pretorius et al., 2011] Pretorius, A. J., Bray, M.-A., Carpenter, A. E., and Ruddle, R. A. (2011). Visualization of Parameter Space for Image Analysis. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2402–2411.

[Rencher and Christensen, 2012] Rencher, A. C. and Christensen, W. F. (2012). *Methods of multivariate analysis*. Wiley, Hoboken, New Jersey.

[Roweis and Saul, 2000a] Roweis, S. and Saul, L. (2000a). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.

[Roweis and Saul, 2000b] Roweis, S. T. and Saul, L. K. (2000b). Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326.

[Saito and Takahashi, 1990] Saito, T. and Takahashi, T. (1990). Comprehensible rendering of 3-d shapes. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '90, pages 197–206, New York, NY, USA. ACM.

[Sammon, 1969] Sammon, J. W. (1969). A nonlinear mapping for data structure analysis. *IEEE Trans. Comput.*, 18(5):401–409.

[Schulz and Velho, 2011] Schulz, A. and Velho, L. (2011). ChoreoGraphics: An Authoring Environment for Dance Shows. In *ACM SIGGRAPH 2011 Posters*, SIGGRAPH '11, pages 14:1–14:1, New York, NY, USA. ACM.

[Sheng et al., 2008] Sheng, Y., Sadka, A. H., and Kondoz, A. M. (2008). Automatic single view-based 3-d face synthesis for unsupervised multimedia applications. *IEEE Trans. Circuits Syst. Video Techn.*, 18(7):961–974.

[Shlens, 2005] Shlens, J. (2005). A tutorial on principal component analysis. In *Systems Neurobiology Laboratory, Salk Institute for Biological Studies*.

[Sobieszczanski-Sobieski and Haftka, 1997] Sobieszczanski-Sobieski, J. and Haftka, R. T. (1997). Multidisciplinary aerospace design optimization: survey of recent developments. *Structural Optimization*, 14:1–23.

[Stolte et al., 2002] Stolte, C., Tang, D., and Hanrahan, P. (2002). Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):52–65.

[Sun et al., 2013] Sun, G., Wu, Y., Liang, R., and Liu, S. (2013). A survey of visual analytics techniques and applications: State-of-the-art research and future challenges. *J. Comput. Sci. Technol.*, 28(5):852–867.

[Swayne et al., 2003] Swayne, D., Lang, D., Buja, A., and Cook, D. (2003). Ggobi: evolving from xgobi into an extensible framework for interactive data visualization. *Computational Statistics & Data Analysis*, 43(4):423–444.

[Tejada et al., 2003] Tejada, E., Minghim, R., and Nonato, L. (2003). On improved projection techniques to support visual exploration of multidimensional data sets. *Information*

Visualization, 2(4):218–231.

[Tenenbaum et al., 2000] Tenenbaum, J., de Silva, V., and Langford, J. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323.

[Torsney-Weir et al., 2011] Torsney-Weir, T., Saad, A., Mller, T., Hege, H.-C., Weber, B., and Verbavatz, J.-M. (2011). Tuner: Principled Parameter Finding for Image Segmentation Algorithms Using Visual Response Surface Exploration. *IEEE Trans. Vis. Comput. Graph.*, 17(12):1892–1901.

[Uykan and Guzelis, 1997] Uykan, Z. and Guzelis, C. (1997). Input-output clustering for determining centers of radial basis function network. In *Proceedings of the 1997 European Conference on Circuit Theory and Design*, pages 435–439, Budapest, Hungary. Technical University of Budapest, European Circuit Society. The 1997 European Conference on Circuit Theory and Design - ECCTD '97, Budapest, Hungary, 30th August - 3rd September 1997.

[Wang et al., 2013] Wang, B., Ruchikachorn, P., and Mueller, K. (2013). SketchPadN-D: WYDIWYG Sculpting and Editing in High-Dimensional Space. *IEEE Trans. Vis. Comput. Graph.*, 19(12):2060–2069.

[Wendland, 2004] Wendland, H. (2004). *Scattered Data Approximation*. Cambridge University Press.

[Wong and Thomas, 2004] Wong, P. C. and Thomas, J. (2004). Visual analytics. *IEEE Comput. Graph. Appl.*, 24(5):20–21.

[Yang et al., 2007] Yang, J., Jiang, Y.-G., Hauptmann, A. G., and Ngo, C.-W. (2007). Evaluating bag-of-visual-words representations in scene classification. In *Proceedings*

of the International Workshop on Workshop on Multimedia Information Retrieval, MIR '07, pages 197–206, New York, NY, USA. ACM.

[Young and Householder, 1938] Young, G. and Householder, A. (1938). Discussion of a set of points in terms of their mutual distances. *Psychometrika*, 3(1):19–22.

[Zhao et al., 2002] Zhao, W., Huang, D.-S., and Yunjian, G. (2002). The structure optimization of radial basis probabilistic neural networks based on genetic algorithms. In *Neural Networks, 2002. IJCNN '02. Proceedings of the 2002 International Joint Conference on*, volume 2, pages 1086–1091.