Computers & Graphics 48 (2015) 35–47

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

Facing the high-dimensions: Inverse projection with radial basis functions ☆

Elisa Amorim [a,*], Emilio Vital Brazil [a], Jesús Mena-Chalco [b], Luiz Velho [c], Luis Gustavo Nonato [d], Faramarz Samavati [a], Mario Costa Sousa [a]

[a] Department of Computer Science, University of Calgary, Canada
[b] Center for Mathematics, Computation and Cognition, Federal University of ABC, Brazil
[c] National Institute for Pure and Applied Mathematics, Brazil
[d] University of São Paulo, Brazil

ARTICLE INFO

Article history:
Received 3 September 2014
Received in revised form
11 February 2015
Accepted 26 February 2015
Available online 17 March 2015

Keywords:
Multidimensional data
Multidimensional projection
Inverse projection
Exploration of multidimensional parameters
Face synthesis

ABSTRACT

Multidimensional projection has become a standard tool for visual analysis of multidimensional data sets, as the 2D representation of multidimensional instances gives an important and informative panorama of the data. Recently, research in this torojection, a recently proposed resampling mechanism that allows users to generate new multidimensional instances by creating reference 2D points in the projection space. Given an m-dimensional data set and its 2D projection, inverse projection transforms a user-defined 2D point into an m-dimensional point by means of a mapping function. In this work, we propose a novel inverse projection technique based on *Radial Basis Functions* interpolation. Our technique provides a smooth and global mapping from low to high dimensions, in contrast with the former technique (iLAMP) which is local and piecewise continuous. In order to demonstrate the potential of our technique, we use a 3D human-faces data set and a procedure to interactively reconstruct and generate new 3D faces. The results demonstrate the simplicity, robustness and efficiency of our approach to create new face models from a structured data set, a task that would typically require the manipulation of hundreds of parameters.

1. Introduction

Dimensionality reduction for visualization, called *multidimensional projection*, is a traditional approach for data analysis. Principal component analysis and multidimensional scaling, for instance, are mathematical tools that have long been used to find low-dimensional representations (e.g., 2D) of multidimensional data sets. The goal of such representation is to provide an overview of similarities between instances of data in a 2D space or *projection space*. More recently, multidimensional projection techniques have evolved and improved in terms of computational speed [1,2] and user interactivity for data exploration [1,3].

For some applications, however, the visual exploration of multidimensional data may not be the only task at hand and creating new instances of data and extrapolating existing ones may prove to be useful. Taking this into consideration, Amorim et al. [4] have proposed to use the projection space as an interface for the creation and extrapolation of multidimensional data, a procedure called *inverse projection* or *back-projection*.

Given a multidimensional data set $X \subset \mathbb{R}^m$ and its 2D projection $Y \subset \mathbb{R}^2$, inverse projection allows the user to create new multidimensional points ($q \in \mathbb{R}^m$) by creating reference points $p \in \mathbb{R}^2$ in the projection space. The technique presented in the paper of Amorim et al. [4] is the only inverse projection method proposed so far that is used in conjunction with multidimensional projection as an interface. Its mathematical formulation is based on the Local Affine Multidimensional projection (LAMP) technique [3] and is, therefore, named iLAMP (inverse-LAMP). iLAMP operates locally, it does not provide a continuous mapping (e.g., a continuous 2D curve may be mapped in more than one region in the multidimensional space) and uses local affine transformations to map user-defined 2D points into the multidimensional space. It will be demonstrated further that iLAMP can cause distortions in the multidimensional points when the projection layout is modified by the user. In this work, we propose a novel nonlinear and continuous inverse projection technique, based on radial basis function (RBF) interpolation. When we mention continuity in this work, we refer to the standard mathematical definition of continuity for metric spaces.

RBF is a well-established mathematical formulation that has been used in various multivariate approximation problems [5]. The goal of RBF is to construct a function s that interpolates given function values at their corresponding given data points in the

☆This article was recommended for publication by Stefan Bruckner.

*Correspondence to: Department of Computer Science, 602 Information and Communications Technology (ICT), University of Calgary, 2500 University Dr. NW Calgary, Alberta, Canada T2N 1N4. Tel.: +1 (403) 220-6015.

http://dx.doi.org/10.1016/j.cag.2015.02.009
0097-8493/

space. The interpolation function s is formed by a linear combination of radial basis kernels. Depending on the choice of the kernel function, s can be smooth (in this work we say a function or hypersurface is smooth if its second derivative is continuous).

The problem of inverse projection consists of finding a mapping function $s : \mathbb{R}^2 \to \mathbb{R}^m$ and, in this work, we employ RBF interpolation to create this function. In this specific scenario, the RBFs centers are given by the 2-dimensional position of the points in the projection space ($y_i \in Y$), and the outputs are given by the corresponding multidimensional points of the original data set ($x_i \in X$). The mapping function s is, thus, one that exactly maps the given 2D points y_i into their multidimensional counterparts x_i. This function is used to approximate the multidimensional correspondent of a user-generated 2D point. The fact that this function is smooth makes the RBF solution more attractive than iLAMP for some applications. The problem of creating 3D human faces from an existing data set of face models is an example of such applications.

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 [6].

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 data set 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 (see Fig. 5 for an example). 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 inverse projection framework is a very recent topic itself, and this is the first time that a nonlinear smooth mapping from low to high dimension is proposed. Furthermore, we introduce a visual interface with multidimensional projection and inverse mapping is used for the application of face synthesis.

To summarize, the contributions of this work are

- A novel inverse projection technique based on RBF interpolation that is smooth. The smoothness provided by RBF is an interesting characteristic for some applications, like the face-synthesis application presented in this work. Furthermore, the proposed formulation prevents distortions that may be present with the iLAMP method.
- An application on human face models using the proposed inverse projection: given a data set of face models, inverse projection is employed to synthesize new faces.
- A workflow that enables the user to explore the multidimensional space of faces and to create new characters.
- An expression transfer mechanism that can be used to create the various expressions of a new character. Our interface also permits the new characters with different expressions to be incorporated into the multidimensional data set. This allows the construction of a complete character and the further exploration of its expression variations.

2. Related work

Visualization has long been used as a powerful tool to gain insight into multidimensional data sets. Many different visualization techniques have been created to better understand such a complex data configuration. Parallel-coordinates [7] and scatterplot matrices [8], for instance, are popular techniques used to represent multidimensional data in a 2D display. They provide insight about the correlation of parameters on the data set and can also be used to identify possible outliers and clusters. However, such techniques do not provide a mechanism to visualize the neighborhood structure of data under analysis, making harder the process of identifying similar instances.

Multidimensional projection (MP), in turn, represents each instance of a data set as a 2D point in a visual space. The position of each instance is calculated in such a way that distances are preserved as much as possible, i.e., similar instances in the multidimensional space should be positioned close together in the 2D projection space. Thus, this methodology provides insight into the neighborhoods of the data set and is also useful to identify clusters and patterns in the data. Principal component analysis [9] and some multidimensional scaling [10] are traditional linear MP techniques. Linear techniques often present limitations that prevent distances to be well preserved when the data set is not distributed in a 2D manifold structure. On the other hand, nonlinear techniques, like Sammon's mapping [11], prove to be useful in projecting data sets with nonlinear relationships. Techniques such as LLE proposed by Roweis et al. [12] use a locally linear approach but the overall projection result is able to capture nonlinearities in the data set. Isomap [13] is another classic method that is able to perform well in nonlinear data sets as it uses the geodesic distance information in the data set to compute dissimilarities. Recent MP techniques emphasize the importance of fast computation [1,2] and user-interactivity through the use of control points to steer the projection [1,3,14]. Control points have also been recently proposed, for instance, as a means to transform a multidimensional feature space based on user input provided on the projection layout [15].

In order to leverage the neighborhood information provided by MP techniques, Amorim et al. [4] proposed a technique called iLAMP that allows users to create multidimensional points from 2D projections. This process is called inverse projection and can be used to extrapolate data by creating points in the 2D projection space. iLAMP was proposed to be employed to assist users in finding new seed points for optimization problems. More specifically, they consider the common problem in engineering of encountering a combination of parameters that results in a desired output. This is often modeled as a minimization problem where the goal is to minimize a function that measures the error between the desired output and the calculated output. In such cases, the goal is not always to find a global minimum to this function but to ensure that a proper exploration of the parameter space is accomplished. In this context, iLAMP enables us to interactively explore the multidimensional parameter-space from a 2D perspective. Given the 2D projection of a few local minima to the minimization function, iLAMP allows users to interactively create and inspect new seed points in the multidimensional space by sampling in regions of interest in the projection space. The new seed points are then used as starting points for an iterative optimization algorithm and potentially new local minima are found.

Some systems, like the ones presented by Bruckner and Möller [16] and Pretorius et al. [17] have also been designed to assist the user to explore multidimensional parameter spaces. In the paper of Bruckner et al. [16], it was introduced a system that allows the user to explore simulation results for special effects. The focus of their system was not to provide a complete understanding of the parameter space, but to allow users to achieve a desired animation sequence without the hurdle of parameter tuning. Therefore, the system is not designed to support resampling of the parameter space. Likewise, in the paper of Pretorius et al. [17] 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 uniformly sample the parameter space. 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/systems allows for resampling as part of the data exploration process.

Wang et al. [18] propose a system that allows the user to create and edit multidimensional data points from an existing data set or from scratch. They use traditional visualization tools, as parallel-coordinates and scatterplot matrices, to create an interface that permits user to sketch the distribution of new points. In another work, Torsney-Weir et al. [19] present a system that assists the user in resampling a continuous parameter space. Their system is designed to help users to find a good 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 [20] 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 high number of parameters.

In inverse projection, the user is not necessarily aware of the number of parameters in hand, as both analysis and data extrapolation are carried out in the 2D projection space. This concept is very much alike what Endert et al. [21,22] describe as "Observation-level spatialization", that enables users to interact directly with projected data points, instead of manipulating parameters separately. In their work, however, the focus is to provide more insight into a given data set by permitting the user to rearrange the projected points based on a priori knowledge of the data. In our proposed work, even though the user can rearrange the projection results (as will be presented in Section 4.3), the focus is to use the projection of the data to identify regions of interest and generate new samples in these regions.

In the next section we present an overview of the interactive inverse projection framework together with a discussion on data set requirements for its application.

3. 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 data set, that is projected into a 2D projection space. Besides being the interface where data analysis takes place, the projection space also provides the interactive medium over which the user can create new multidimensional samples. With the use of a mouse or pointing device, the user creates 2D points in regions of interest over the projection space. For each user-defined point, inverse projection finds a multidimensional representation for it by means of a mapping function.

One may generalize the main components of the interactive inverse projection framework as (a) a multidimensional data set used as the base for parameter exploration; (b) a projection mapping that projects multidimensional samples into a projection space; (c) an interactive projection space that allows the user to rearrange the projected data and to create points in it; and (d) an inverse mapping that maps 2D user-create points into the multidimensional space. This framework is illustrated in Fig. 1.

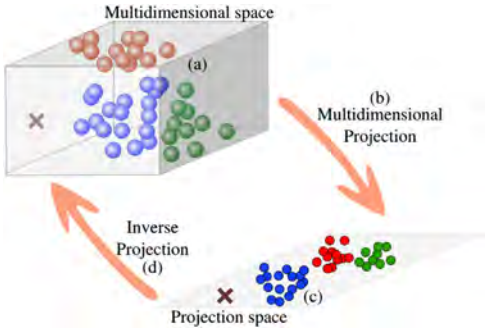

Fig. 1. Given (a) a multidimensional data set (illustrated by the colored circles inside the cube), (b) multidimensional projection maps the data into a (c) projection space. Inverse projection operates in the other direction: via mouse, the user defines a 2D point in the projection space (illustrated by the cross), which is (d) mapped back into the multidimensional space (cross inside cube). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

In order to use the proposed framework, there are two basic premises that need to hold true for the given data set: (1) the objects of the data set can be represented as a numerical feature vector of n-dimensions; and (2) there is a mechanism that transforms any given feature vector of n-dimensions into an object of the same type as the data set.

For the first premise, the numerical representation of the data set is the information used to find the corresponding 2D projection. It is true that some multidimensional projection techniques like MDS only require the dissimilarity information between pairs of instances. However, when the inverse projection takes place it is the direct relationship between 2D points and multidimensional points that permit the creation of a mapping function. The way to find a good feature-vector representation for a data set is very case-dependent. In some cases the object itself is the feature-vector. For example, in the optimization application presented in iLAMP, the instances of data 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 data set of 3D humans faces used in this work requires an appropriate definition of how to extract feature vectors that represent each face. The definition of feature-vectors in this specific scenario is described in Section 5.1. Once each instance in the data set 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 same dimension as the feature-vectors describing each instance of the data. In order to allow the user to fully comprehend a newly created point and to move forward with the exploration, it is essential that a mechanism that will transform this point into an object of the same class as the ones in the data set exists. Such a mechanism transforms, for example, a multidimensional feature-vector into a 3D face model in our face-synthesis application as described in Section 5.3.

A fundamental point in the interactive inverse projection framework is to allow the user to redefine the projection layout in order to achieve a desirable result. For example, when the user has a previous knowledge of the data set, or when each instance of data has a direct visual representation, it can make sense for the user to rearrange the projection layout in order to isolate or augment the influence of some instances in the confection of a new point. In this work, this is possible through the manipulation

of special projected instances, called control points (as will be detailed in Section 4.3). The repositioning of control points by the user causes a modification in the entire projection layout and can be used to redefine regions of interest. When this repositioning occurs, ideally one would still have a coherent mapping from low to high-dimensions. This coherence will be directly impacted by the chosen mapping function.

In fact, the mapping function used to map 2D points into multidimensional samples is an essential part of the inverse projection framework. The mapping function defines important properties that will make each inverse projection method unique. For instance, when the mapping function is smooth, the new samples will lay on a manifold embedded in the multidimensional space. This kind of smooth mapping is important when control points manipulation is carried out because it guarantees that small changes in the control points will result in small changes in the output. Examples of data sets that we believe can be benefited by the possibility of control points manipulation and, consequently, by the smoothness of the mapping function are face models, textures, 3D model animations, or any other data set that contains a direct visual representation of each instance. Thus, smoothness is precisely what we propose in this paper and we achieve it through RBF interpolation, as explained by Monnig et al. [23]. In the next section we present the proposed inverse projection using RBF.

4. Inverse projection through radial basis functions

RBF has become popular to approximate multivariate functions, primarily for its ability to deal with multidimensional data [5]. In RBF interpolation, given data samples $\chi_i \in \mathbb{R}^m$ and respective function values $f_i = f(\chi_i) \in \mathbb{R}, i = 1 \ldots N$, an approximant $s : \mathbb{R}^m \to \mathbb{R}$ is constructed, in such a way that s interpolates the function f over the data samples. The approximant $s(x)$ is formed by a finite linear combination of translations of radially symmetric functions $\phi(\| \cdot \|)$, where $\| \cdot \|$ is the Euclidean norm in $\mathbb{R}^m$. As explained in the book of Buhmann [5], radial symmetry means that the value of the function only depends on the Euclidean distance of the argument from the origin. Since the translations of ϕ are defined by χ_i, the interpolation function s is defined by

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

where λ_i are real-valued coefficients.

RBF has been successfully applied in a recent multidimensional projection technique, named RBF projection [14]. In RBF projection, a function $s : \mathbb{R}^m \to \mathbb{R}^2$ is created to map m-dimensional points to the 2D projection space. In this work, we propose to use RBF interpolation to perform inverse projection, by creating a function s that will map information from the projected space into the original m-dimensional space, i.e., $s : \mathbb{R}^2 \to \mathbb{R}^m$. A detailed description of the technique is presented below.

4.1. Mathematical formulation

Let $X \subset \mathbb{R}^m$ be an m-dimensional data set, 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. To make the formulation clear, we write $s(p) = (s_1(p), \ldots, s_m(p))$, where s_k accounts for the k th output dimension and is written in the form of Eq. (1):

$$s_k(p) = \sum_{i=1}^{N} \lambda_{ki} \phi(\| y_i - p \|), \tag{2}$$

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. 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

$$s_k(y_j) = \sum_{i=1}^{N} \lambda_{ki} \phi(\| y_i - y_j \|) = x_{jk}. \tag{3}$$

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

$$\Phi \lambda_k = b_k, \tag{4}$$

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

$$\begin{bmatrix} \phi_{11} & \cdots & \phi_{1N} \\ \phi_{21} & \cdots & \phi_{2N} \\ \vdots & \vdots & \vdots \\ \phi_{n1} & \cdots & \phi_{NN} \end{bmatrix} \begin{bmatrix} \lambda_{k1} \\ \lambda_{k2} \\ \vdots \\ \lambda_{kN} \end{bmatrix} = \begin{bmatrix} x_{k1} \\ x_{k2} \\ \vdots \\ x_{kN} \end{bmatrix}. \tag{5}$$

Note that the linear system in Eq. (4) 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 actually 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 $p \in \mathbb{R}^2$. Fig. 2 illustrates the process of using RBFs to create a mapping function from low to high dimension. Algorithm 1, in turn, presents a step-by-step procedure to calculate the mapping function s.

Numerical and computational aspects: As it was shown, the RBF interpolation problem comes down to the solution of the linear system given by Eq. (5). 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.

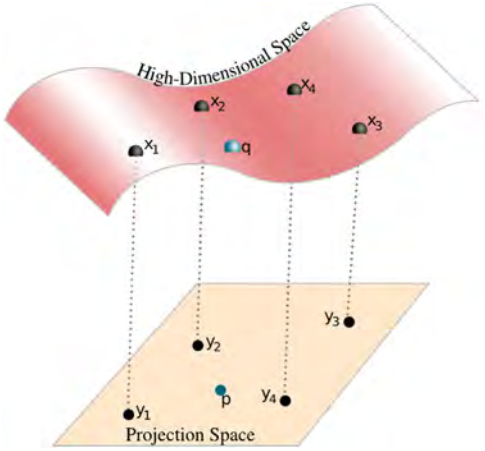

Fig. 2. Inverse mapping using RBF. $\{x_1, x_2, x_3, x_4\} \in \mathbb{R}^m$ represent multidimensional points, while $\{y_1, y_2, y_3, y_4\} \in \mathbb{R}^2$ are their projected counterparts. Our inverse projection method creates a continuous nonlinear mapping function s (illustrated by pink surface) that interpolates between data samples, i.e. $s(y_i) = x_i$. The black points represent samples of the data set, 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). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

The Gaussian ($\phi(r)=e^{-\epsilon r^2}$) and Multiquadrics ($\phi(r)=\sqrt{c^2+\epsilon r^2}$) functions are popular choices in various applications (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 [5] and Wendland [24] 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 choice, 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 [25].

The computation of the mapping function s is generally fast. In fact, we showed that the 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 Fig. 3 were achieved in the same machine configuration described in Section 6. Note that the entire process is extremely fast even for large number of centers and/or dimensions, rendering the technique suitable for real-time applications.

In the proposed framework, the RBF centers come from projected points of a given data set. Depending on the application, and on the size of the data set in hand, all of the points could be used as centers. Or, as an alternative, only a subset of these points could be used. 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 data set. However, in Section 4.2, we briefly discuss possible approaches to filter out and select few samples.

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 in 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. As an alternative, there are some automatic methods designed to indicate a kernel function and parameter definitions that would approximate well a given data. Some of these methods are summarized in the works of Mongillo [26] and Fasshauer et al. [27]. In this work we have found through trial-and-error that the multiquadrics kernel function with $c=0$ performed well in our face-synthesis application, as will be shown in Section 6. We encourage the reader who is keen to use the proposed method to make a thorough investigation of this matter using the aforementioned tools. It is important to keep in mind, however, that these are only guidelines and this may become a challenging task when working with radial basis functions.

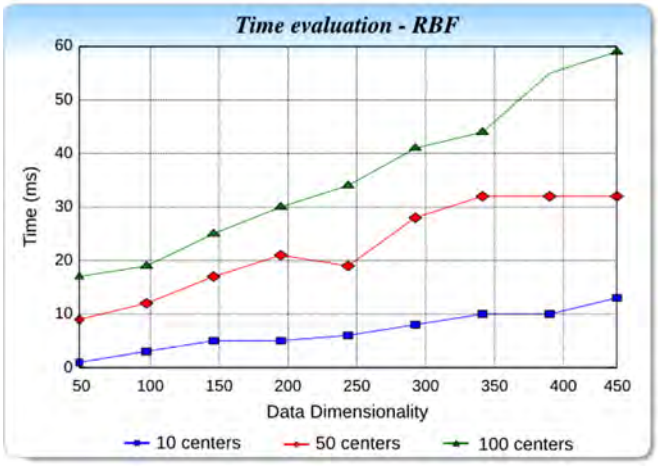

Fig. 3. Time evaluation of RBF inverse projection. For varying number of centers, the time (in milliseconds) spent in the computation of the mapping function and the creation of 100 samples is displayed.

Algorithm 1. Building the RBF for inverse projection.

1: Given $Y=y_1,\ldots y_N\subset\mathbb{R}^2$ (projected data set, RBF centers);
2: Given $X=x_1,\ldots x_N\subset\mathbb{R}^m$ (original data set, RBF function values);
3: Given RBF kernel function $\phi:\mathbb{R}^+\rightarrow\mathbb{R}$
4: // Calculate interpolation matrix Φ
5: **for** $i=1\ldots N$ **do**
6: **for** $j=i\ldots N$ **do**
7: $\Phi[i][j]=\Phi[j][i]=\phi(\|y_i-y_j\|)$
8: **end for**
9: **end for**
10:// Assemble right-hand side of system
11: **for** $i=1\ldots N$ **do**
12: **for** $j=1\ldots m$ **do**
13: $b[i][j]=x_i[j]$
14: **end for**
15: **end for**
16: Solve system $\Phi\lambda=b$, to find λ

4.2. False neighbors and tears

The main goal of multidimensional projection is to reduce the dimensionality of data sets in such a way that distances are preserved as much as possible. In fact, a popular quality metric used to validate the projection results is the *stress* function, which measures how much distances between pairs of instances differ in the high- and low-dimensional spaces. The stress function is written as

$$\text{stress}=\frac{\sum_{i,j}^{n}(d_{ij}-\delta_{ij})^2}{\sum_{i,j}^{n}(d_{ij})^2}, \tag{6}$$

where d_{ij} and δ_{ij} are the distances between instances i and j in the high and low dimensional spaces, respectively. However, some data sets are prone to severe distortions when projected to 2D. Such distortions occur due to information loss during the dimensionality reduction process and are characterized by the appearance of false neighbors and tears in the 2D projection. These artifacts may cause unwanted side-effects in the inverse projection results.

As the names suggest, false neighbors consist of pairs of instances where $\delta_{ij}\ll d_{ij}$, while tears present $\delta_{ij}\gg d_{ij}$ [28]. The presence of such artifacts can result in mapping functions that are not coherent with the given data set. 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 in order to select more appropriate centers for the mapping function. An interesting approach is the method presented in the paper of Amorim et al. [14] for center selection in RBF interpolation. The authors use a technique called *Regularized Orthogonal Least Squares* (ROLS) to select a meaningful set of control points to be used in their multidimensional projection technique. The projection results they achieve indicate that the

control points selection through ROLS presents satisfactory results. Another possibility to deal with false neighbors and tears could be to use the control points of the multidimensional projection as centers of the mapping function.

However, for some applications it may be important to have all instances as RBF centers. This is the case, for example, of the face-synthesis application we present in this work, where we want each face in the data set to be represented in 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 was previously presented and consists of creating a single mapping function using every 2D point y_i as an RBF center. Therefore, the same mapping function is used for all user-created 2D point p, 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 data set, and create one mapping function for each cluster. In doing so, we can reduce the negative impact that false neighbors/tears may cause in the back 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, respectively, be $X_s = x_{c1}, \ldots, x_{ck}$ and $Y_S = y_{c1}, \ldots, y_{ck}$. We propose to have one cluster C_l for each MP control point x_{cl}, where $\{(y_i, x_i) \in C_l | d(x_i, x_{cl}) < d(x_i, x_{cm}) \forall x_{cm} \in X_s\}$. 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 conducted experiments with data sets composed of random samples of hyper-spheres. Hyper-spheres undergo severe information loss when projected to a plane, causing false neighbors and tears. We used hyper-spheres in four different dimensions (3, 5, 10 and 20) and 500 samples in each data set. We generated 200 random points in the projection space, which were projected back to the original hyper-sphere dimensionality. We calculated the distance between the new 200 multidimensional points and the hyper-sphere surface, and the results are presented in Fig. 4. We observe that the multidimensional samples created using this approach are closer to the hyper-sphere surfaces and thus more coherent with the data set.

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 application we propose in this work, the global approach is more suitable.

4.3. Multidimensional projection with control points

So far, we have not made any assumptions over the multidimensional projection technique used to map the original data set 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 special 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 as a black-box. However, recent multidimensional projection techniques provide a mechanism that we believe can be beneficial to the data exploration in the inverse projection framework. This mechanism allows user manipulation of the projection space and is accomplished by the use of *control points*.

Control points have been recently introduced in multidimensional projection as a way to give users control, in some extent, over the projection results [3,14]. They are special 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 can help users to gain more insight about the data set 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. Fig. 5 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 is able to isolate particular instances of faces and generate new faces that are more similar to them.

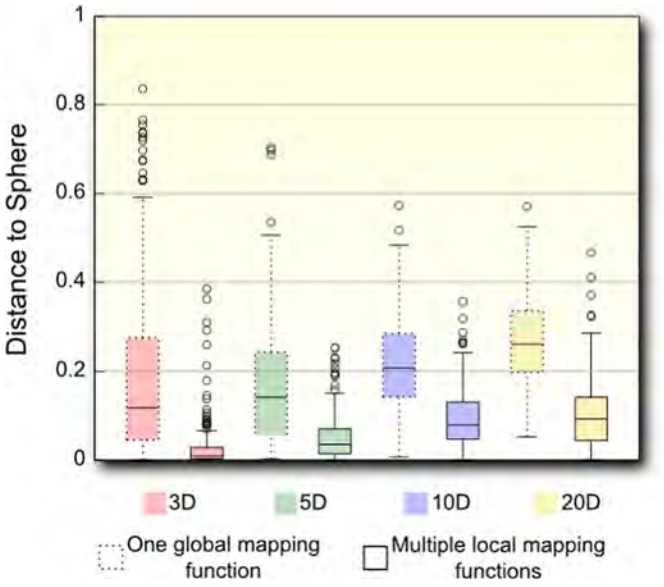

Fig. 4. Experiment to validate the RBF projection when applied to distorted projections using 4 data sets. 500 random samples on the surface of hyper-spheres of 3, 5, 10 and 20 dimensions were used, each dimension forming a separate data set. 200 random sample points are generated in the projection space,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

5. Synthesis of faces and expressions

The synthesis of 3D face models is an important research topic and has been studied for more than three decades [29]. Different approaches have been proposed to achieve 3D face reconstructions [29–32]. Nguyen et al. [33] 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 introduced the first parameterized facial model [34] to shape interpolations and then animate a face. In previous work, Cohen and Massaro [35] used a model of co-articulation and a set of animation parameters to control the face shape. Blanz and Vetter [30] 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 [33].

Recent works were devoted to the study of single view-based 3D face synthesis. Sheng et al. [36], Nguyen et al. [33] and Patel and Zaveri [37] 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.

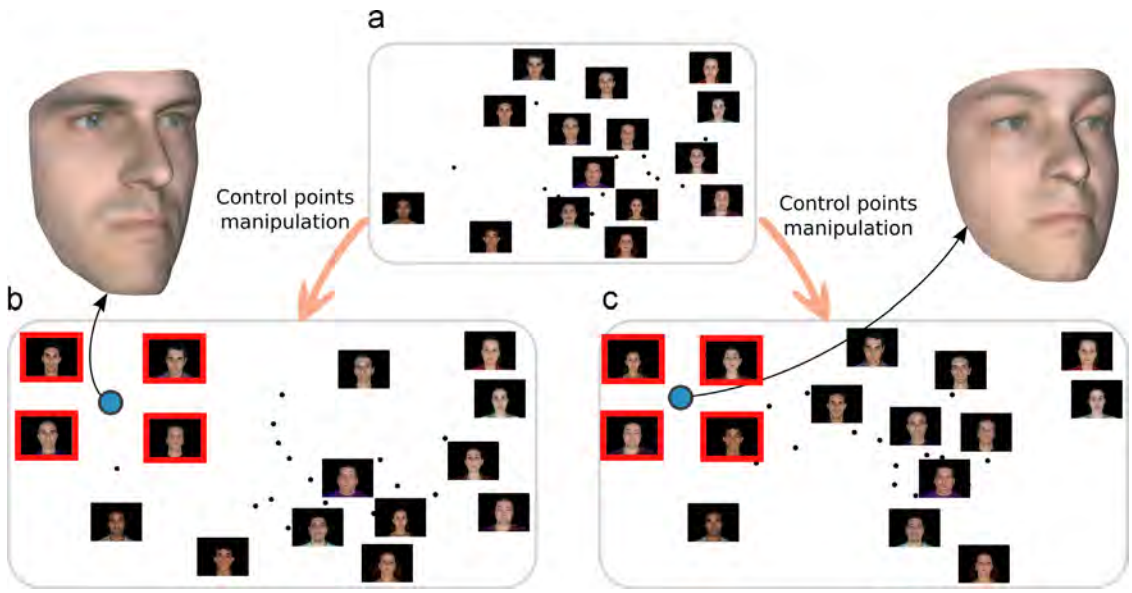

Fig. 5. Example of control points' manipulation in the inverse projection framework, with 3D faces data set; (a) presents the projection of a data set with 30 faces, and 15 control points – the control points are the ones rendered with the face texture, while black points are the remaining instances of the data set; (b) presents the projection after the reorganization of control points positions; in this example, 4 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. (c) is another example of reorganization of control points position, where 4 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. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

The work of Buck et al. [38] uses a frontal face image as an input to generate non-photorealistic faces. Using an initial data set of a hand-drawn character, with 6 mouth and 4 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 certain 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 in order 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. Similar to our work, Lewis et al. [39] use RBFs to create such interpolation. However our approach provides the projection space over which the user can navigate and abstract the various parameters that may be involved in the process.

In this work, we propose to use our inverse projection approach as an interface that allows users to rapidly create new 3D faces, a *by-interaction* process. In this approach the user is not required to provide a 2D image to generate a new face, but an initial data set of images is used to let the user navigate and resample the space of faces. Each face in the data set is represented by a multidimensional vector, that contains geometric and texture information (Section 5.1). The first step of the inverse projection framework is to find a 2D representation of the data set. The 2D projection space becomes the resampling media over which the user is able to create a point $p \in \mathbb{R}^2$ (Section 5.2). The point p is transformed into a multidimensional point q (Section 4). Finally, the point q is passed as an input to a *face-generator* procedure, that returns the geometric and texture information of the new face (Section 5.3). This work flow is presented in Fig. 6.

The application we present in this work is based on the work of Mena-Chalco et al. [6], that uses a single 2D photograph as an input to generate a new 3D face model. In their work, the face synthesis is accomplished through a training set with 210

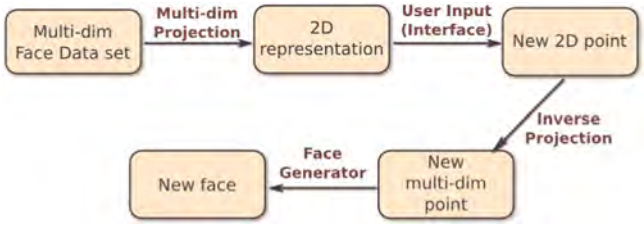

Fig. 6. Synthesis of faces and expression application workflow. Given a data set of faces represented in a multidimensional space, a multidimensional projection technique is applied resulting in a 2D representation of the data. The user is able to create new 2D points that are converted into the original dimensionality of the data set of faces through inverse projection. The new multidimensional point undergoes a series of calculations that give rise to a new facial model.

previously 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 a good application for our inverse projection method.

There are three main reasons that made us choose the face-synthesis application to validate our method and demonstrate the applicability of our technique: (1) the synthesis of faces is an important problem with applications in various areas; (2) there are some important works in the literature that consider the human faces data sets to be embedded in multidimensional nonlinear manifolds [40,12]. This characteristic makes face data sets good candidates for having its dimensionality reduced through multidimensional projection, which is an essential step in our inverse projection framework (Fig. 1); and (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 our technique.

5.1. Input data set

The data set used in this work was created by IMPA [41] and consists of 210 faces acquired from 30 individuals performing

7 basic expressions (happy, sad, anger, disgust, surprise, fear and neutral). Fig. 7 presents the face models of one of the subjects in the data set performing the various expressions.

Each face in the data set contains geometric and texture information measured in $M=9648$ corresponding points of the model. For example, the i th face in the data set 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 data set of faces is given by matrices L^g and L^t, whose i th rows are L_i^g and L_i^t, respectively.

As detailed in the work of Mena-Chalco et al. [6], the representation of the data set of faces is simplified by performing a principal component analysis 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 184 principal components that preserve at least 95% of the original information. 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 data set.

In the aforementioned work, 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 an 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 work, we want to create a new face, both texture and geometry, using a coefficient vector α_x^t as an input, i.e., given a 184-dimensional vector α_x^t, we calculate x^t and α_x^g. (How this calculation is done is presented in details in Section 5.3.) In order to create α_x^t using the inverse projection framework, the input data set needs to be in the same format of α_x^t. Therefore, each face i in the input data set is represented by its coefficient vector α_i^t, giving us a initial data set of 184 dimensions. In the next section we detail the interface for this application.

5.2. Interface

In this application, the interface provided by inverse projection is a screen that depicts the various faces of the input data set, 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. Using control points provided by most recent projection techniques [1,3,14], the user is able to reorganize the layout of the faces. Once a pleasing layout is achieved, the user can create a new face by simply generating a 2D point in 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 3D visualizer window. All the process is done in real time. Fig. 8 depicts the interface of the system.

The process to transform the 2D point into the original multidimensional space is given by the inverse projection methodology

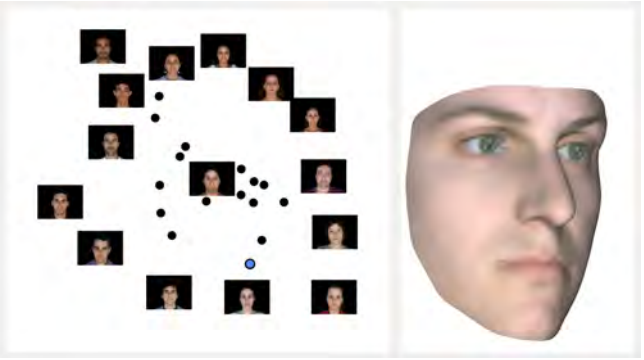

Fig. 8. Interface for facial synthesis application. The left screen contains the projection of the initial data set of faces (in this example 30 faces with neutral expression are used). The control points are rendered with their corresponding facial textures, while the remaining faces are represented as black circles. The blue circle indicates a user-created point, which resulted in the face model depicted in the right screen. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

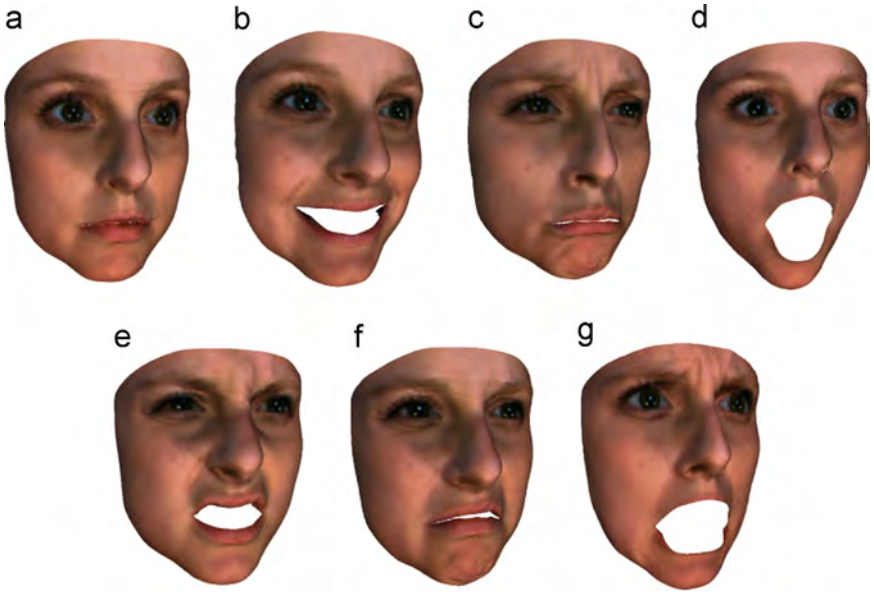

Fig. 7. Geometric model with mapped texture of one of the individuals in the data set performing 7 basic expressions: (a) neutral; (b) happy; (c) sad; (d) surprise; (e) anger; (f) disgust; and (g) fear.

described in Section 4. We present in the next section the process to create the geometry and texture of the new face given the multidimensional vector produced by inverse projection.

5.3. Face generation process

Let t_0 and g_0 be, respectively, the average between the textures and geometries of the training data set. 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 on applying vector α_x^t into basis E^t:

$$x_t = E^t \alpha_x^t + t_0. \tag{7}$$

The reconstruction of the geometry is achieved in 3 main steps: first, the texture coefficients of the faces in the data set are used to calculate coefficients s_x, $\alpha^t s_x = \alpha_x^t$, 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. The second step is to calculate the weight vector α_x^g, as $\alpha_x^g = \alpha^g s_x$. Finally, the geometry x^g is calculated by applying vector α_x^g into basis E^g:

$$x^g = E^g \alpha_x^g + g_0. \tag{8}$$

5.4. Expression transfer

Once a new face model is created, it can be interesting to also create its different expressions. As part of this application, we provide an "expression transfer" mechanism that permits the change of a face model basic expression into any of the other six basic expressions. This is done by means of "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 in terms of expression transfer, its simplicity enables straightforward implementation and yields good results for the given application.

As previously discussed, the input data set 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 calculated for each of the seven expressions $\overline{x}_i^g, i =$ 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

$$\Delta x_{\text{neutral} \to i} = \overline{x}_{\text{neutral}}^g - \overline{x}_i^g. \tag{9}$$

Given a face geometry x_i^g with expression i, we can have expression j transferred into it by doing $x_{i \to j}^g = x_i^g - \Delta x_{Neutral \to i} + \Delta x_{Neutral \to j}$. Notice that this 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, in order to be able to load it in the system for further exploration. We do so by computing the reverse of what is shown is Section 5.3, i.e., we start from the geometric information and recover the texture: $\alpha_x^g = (E^g)^T (x_{i \to j}^g - g_0)$, then $\alpha^g s_x = \alpha_x^g$ and finally $\alpha_x^t = \alpha_t s_x$.

6. Results and discussion

In this section we 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 the inverse projection and in the face-generator process 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 need 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, we used every face in the input data set as an RBF center. This was considered important in this application because it allows the users to recreate every face in the data set and all the faces have equal importance in the inverse mapping. The kernel function used was the multiquadrics with $c = \epsilon = 0$, i.e., $\phi(r) = r$.

As explained in Section 5, the training set of faces used in this work is composed of 210 faces of 30 individuals performing 7 different expressions. Note that this complete 210 faces data set is used to reconstruct the 3D and texture information of the new face, but a subset of it can be used in the inverse projection process. In fact, we experimented with three variations of this data set:

- *Experiment* 1 (*neutral data set*): only the neutral expressions of the 30 individuals are used, i.e., the data set contains 30 faces. In this experiment, the focus is to create a new character with neutral expression.
- *Experiment* 2 (*individual data set*): all the expressions of one single individual are used, i.e., the data set contains 7 faces. In this experiment the focus is to create transitional expressions of a single character.
- *Experiment* 3 (*miscellaneous data set*): all the 210 faces are used. The focus of this experiment is the creation of new characters and different expressions.
- *Experiment* 4 (*expression transfer*): using initially the neutral data set, a new neutral character is created. The remaining six expressions of this character are generated (see Section 5.4) and loaded into the program to create its transitional expressions.

6.1. Results

Experiment 1: Fig. 9 presents some results achieved with the data set of only neutral expressions of 30 individuals. We show 5 examples of user-generated faces. We strategically positioned

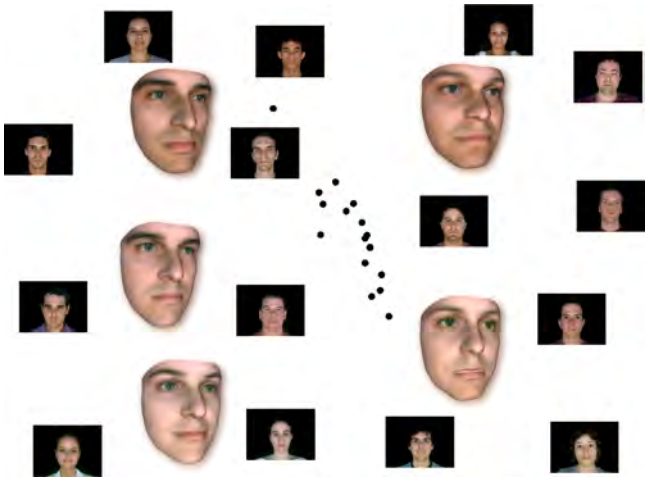

Fig. 9. Example using 30 individuals with neutral expression. The textures with black background represent the control points of the multidimensional projection. Black points represent the remaining faces in the data set. The 3D face models are the user-generated faces through inverse projection.

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 will be more or less similar to specific input faces depending on the position the new 2D point is created.

Experiment 2: Fig. 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 present 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 1–4), fear to neutral (5–7), neutral to angry (8 and 9), angry to happy (10–12) happy to disgust (13 and 14) disgust to sad (15 and 16) and sad to surprise (17–20). The faces generated in points 3 and 4 are good examples of 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.

Experiment 3: In Fig. 11 we present some results achieved using the complete data set with 210 faces. Since we have various individuals performing different expressions as an input, we are able to generate new characters with varying expressions and expressions in between. This is illustrated by 6 new models generated through inverse projection, placed on their corresponding position of the projection space.

Experiment 4: In this example we show that, besides generating a new character, all its different expressions can also be calculated using the displacement vectors explained in Section 5.4. Starting from the neutral data set, the new character with neutral expression is defined (Fig. 12(a) and (b)). The various expressions are then computed for this character (Fig. 12(c)), as described in Section 5.4. The corresponding α_x^t's of each expression are calculated and saved into a separate file, which can later be loaded into the inverse projection system for further exploration of the new character, as illustrated in Fig. 12(d). We create transitional expressions for the new character and we show 7 of them in Fig. 12(e).

The accompanying videos provide a clear idea of how our interface operates integrating the inverse projection method with the face-synthesis application. The first video shows experiments 1, 2 and 3 in action, whereas the second video presents the expression transfer interaction explained in experiment 4.

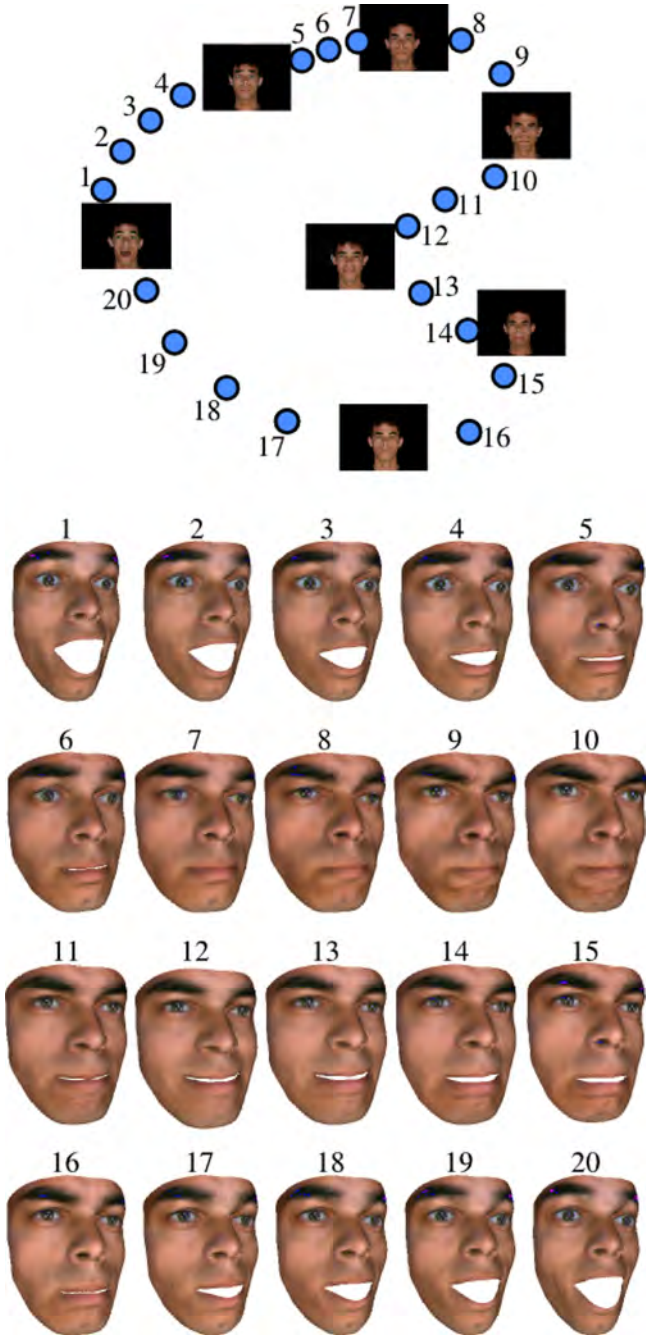

Fig. 10. Example using only one individual performing 7 expressions expressions as data set. All seven face models are used as control points in this example. The user created path through inverse projection, illustrated by the numbered blue points in the projection space, result in the depicted face models. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

6.2. Evaluation

In order to evaluate the *quality* of the face models generated by our system, we have conducted an informal study with 99 people from various age groups, educational levels and knowledge of computer graphics (Fig. 13). The concept of quality here means that the new face model is (1) as realistic as the faces in the data set and (2) unique in comparison with the original faces in the data set.

For the evaluation of (1) we displayed 8 faces (Fig. 14), half of them from the original data set 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 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.

The results of this experiment support the claim that our method is capable of creating faces that are as realistic as the ones from the original data set, since the synthesized faces were not easily distinguished by the participants. For this to hold true, the chosen RBF kernel and its corresponding parameters, when any, need to be carefully chosen 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 data sets.

For the evaluation of (2), we presented two sets with 10 faces each: the first set containing 10 face models from the original data set that were used as an input to generate 10 other faces, presented in the second set (Fig. 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 (35% and 30% respectively), and all of other faces in set 1 received less than 20% of association. Faces C and J also had a large percentage of the participants indicating "NONE" (27% and 28%, respectively), but both were also strongly associated with a Face in set 1: Face C had

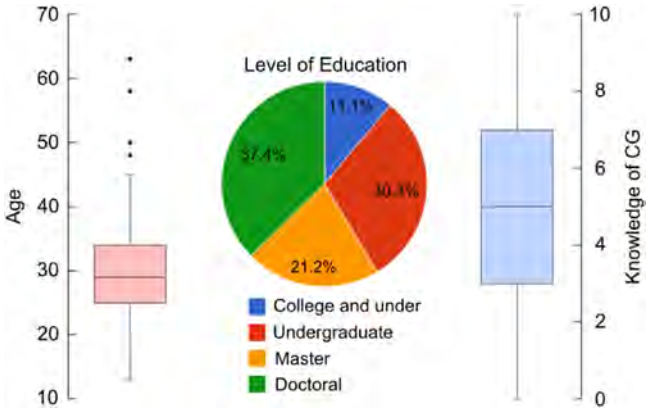

Fig. 13. Summary of the 99 participants that responded to our informal study.

Fig. 11. Example using the complete 210 faces data set, with all individuals and expressions. Using such a data set, we are able to generate new characters with different expressions, as illustrated by the 3D face models depicted in the projection.

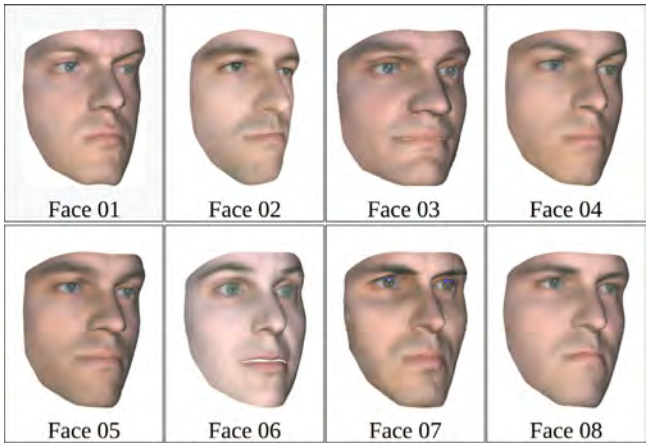

Fig. 14. Set of faces used for the first evaluation. Faces 01, 03, 06 and 07 are from the original data set; the remaining are face models created through our system.

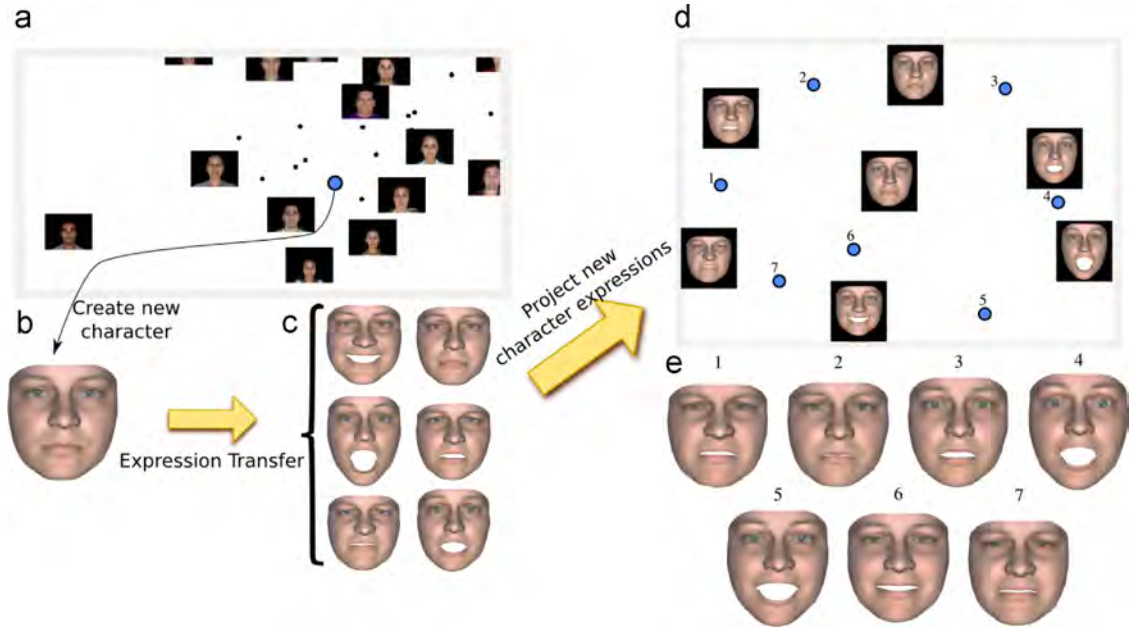

Fig. 12. Expression transfer; (a) projection space and user-defined point (blue); (b) new character with neutral expression; (c) new character with different expressions calculated as demonstrated in Section 5.4. (d) New character with different expressions loaded into inverse projection system; (e) 7 points in the projection space are created to demonstrate transitional expressions for new character. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

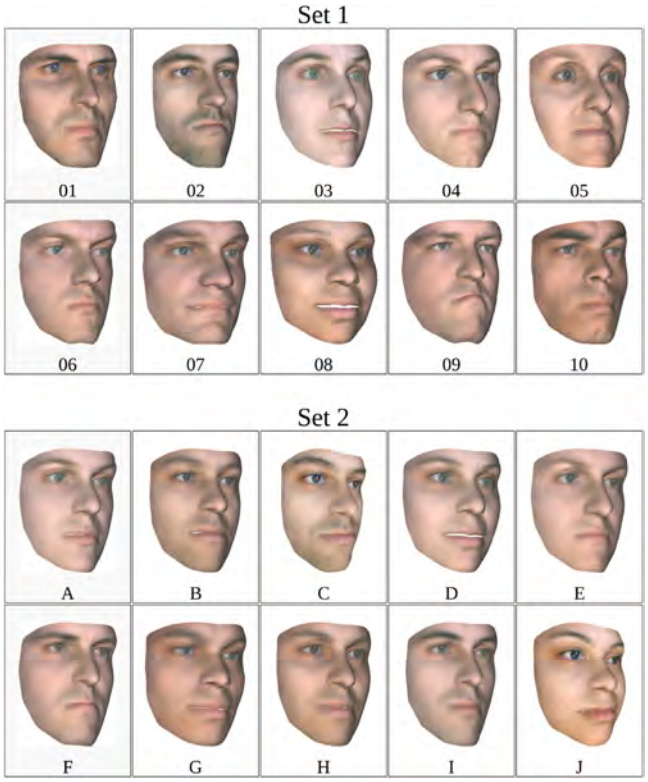

Fig. 15. The two sets of faces used in the second evaluation. Set 1 contains faces of the original data set, that were used as an input to create the faces displayed in Set 2.

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, which indicates 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 considerable 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.

These results are an indicative that our system is able to create new faces that resemble those used in the given data set but are still unique and original, as there was no consensus among 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 4.3), 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 a desired result.

It is important to remark that, as an informal study, these results do not validate our system, but they certainly shed some light into its capability of generating unique yet coherent parameter combinations.

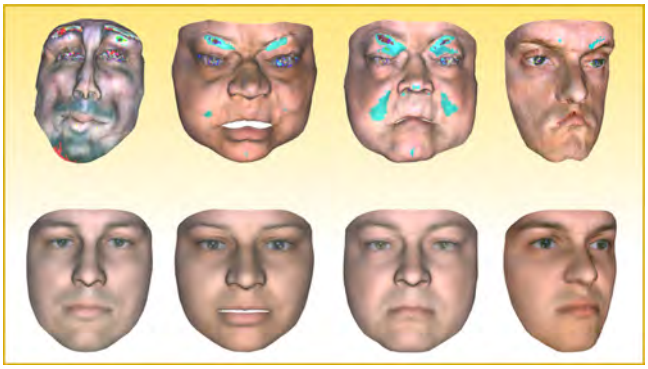

Fig. 16. Top row: faces generate through iLAMP after rearrangement of projection space; bottom row: corresponding faces generated through our RBF solution.

6.3. iLAMP vs RBF

In this section we discuss why the RBF solution presented in this paper is more suitable to 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 data set, 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 4.3).

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

$$\sum_{i=1}^{k} \beta_i \| f_{iLAMP}(y_i) - x_i \|^2, \tag{10}$$

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 generally 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.

In fact, we evaluated the iLAMP error given by Eq. (10) for $k=N$ in the neutral data set. 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 exemplified in Fig. 16.

7. Conclusion and future work

In this work, we have proposed a novel inverse projection technique based on radial basis function interpolation. This method allows the exploration of a multidimensional space in a

2D perspective provided by multidimensional projection. The proposed technique is smooth and global, in contrast to other inverse-projection technique coined iLAMP, and we demonstrate its usability in an application of faces-synthesis. We discuss that the proposed technique is more suitable for some applications, specially when user intervention is expected to be carried out in the projection space.

In the application we present, all the instances of the input data set are used as RBF centers to construct the inverse mapping function. However, data sets with many points might need to have a more careful selection of control points. We presented some alternatives in Section 4.2 but a more careful inspection about the proposed methods still needs to be conducted.

A more thorough investigation regarding the RBF kernel functions ϕ is also an interesting direction in this research. In the examples presented we used $\phi(r) = r$, but it would be interesting if one could determine an appropriate kernel function based on the data set. 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, we plan to evaluate the use of polynomial extensions in our technique [5]. Studies have shown that the use of polynomial terms in RBF kernel functions can improve the function approximation in some cases. The polynomial terms permits, for instance, the creation of a function that samples a sphere embedded in $\mathbb{R}^n$ precisely.

In terms of application, we want to expand the technique so that it can be used with different data sets that have a strong visual appeal, such as texture generation and dance choreography [42].

Acknowledgments

We would like to thank our colleagues for their useful discussions and advice. We also thank the anonymous reviewers for their careful and valuable comments and suggestions. This research was supported in part by the NSERC/Alberta Innovates Technology Futures (AITF)/Foundation CMG Industrial Research Chair program in Scalable Reservoir Visualization. We also acknowledge Brazilian funding agencies Fapesp (#2011/22749-8) and CNPq (#302643/2013-3).

Appendix A. Supplementary material

Supplementary data associated with this paper can be found in the online version at http://dx.doi.org/10.1016/j.cag.2015.02.009.

References

[1] Paulovich F, Silva C, Nonato L. Two-phase mapping for projecting massive data sets. IEEE Trans Vis Comput Graph 2010;16(6):1281–90.
[2] Pekalska E, de Ridder D, Duin RP, Kraaijveld MA. A new method of generalizing Sammon mapping with application to algorithm speed-up. In: 5th annual conference of the advanced school for computing and imaging; 1999.
[3] Joia P, Coimbra D, Cuminato JA, Paulovich FV. Local affine multidimensional projection. IEEE Trans Vis Comput Graph 2011;17(12):2563–71.
[4] Amorim E, Vital Brazil E, Daniels II J, Joia P, Nonato LG, Costa Sousa M. iLAMP: exploring high-dimensional spacing through backward multidimensional projection. In: IEEE VAST. IEEE Computer Society; 2012, p. 53–62.
[5] Buhmann MD. Radial basis functions. New York, NY, USA: Cambridge University Press; 2003.
[6] Mena-Chalco JP, Macêdo I, Velho L, Cesar Jr. RM. 3D face computational photography using PCA spaces. Vis Comput 2009;25(10):899–909.
[7] Inselberg A, Dimsdale B. Parallel coordinates: a tool for visualizing multi-dimensional geometry. In: Proceedings of the 1st conference on visualization, VIS '90; 1990. p 361–78.
[8] Cleveland WS. The elements of graphing data. Belmont, CA, USA: Wadsworth Publ. Co.; 1985.
[9] Jollife I. Principle component analysis; 1986.
[10] Cox T, Cox M. Multidimensional scaling. 2nd ed.Chapman and Hall/CRC; 2000.
[11] Sammon J. A nonlinear mapping for data structure analysis. IEEE Trans Comput 1964;13:401–9.
[12] Roweis S, Saul L. Nonlinear dimensionality reduction by locally linear embedding. Science 2000;290(5500):2323–6.
[13] Tenenbaum J, de Silva V, Langford J. A global geometric framework for nonlinear dimensionality reduction. Science 2000;290(5500):2319–23.
[14] Amorim E, Brazil EV, Nonato LG, Samavati F, Sousa MC. Multidimensional projection with radial basis function and control point selection. In: PacificVis; 2014. p. 209–16.
[15] Mamani GMH, Fatore FM, Nonato LG, Paulovich FV. User-driven feature space transformation. Comput Graph Forum 2013;32(3):291–9.
[16] Bruckner S, Möller T. Result-driven exploration of simulation parameter spaces for visual effects design. IEEE Trans Comput Graph 2010;16(6):1467–75.
[17] Pretorius AJ, Bray MA, Carpenter AE, Ruddle RA. Visualization of parameter space for image analysis. IEEE Trans Comput Graph 2011;17(12):2402–11.
[18] Wang B, Ruchikachorn P, Mueller K. SketchPadN-D: WYDIWYG sculpting and editing in high-dimensional space. IEEE Trans Vis Comput Graph 2013;19(12):2060–9.
[19] Torsney-Weir T, Saad A, Möller T, Hege HC, Weber B, Verbavatz JM. Tuner: principled parameter finding for image segmentation algorithms using visual response surface exploration. IEEE Trans Vis Comput Graph 2011;17(12):1892–901.
[20] van Wijk JJ, van Liere R. Hyperslice: visualization of scalar functions of many variables. In: Proceedings of the 4th conference on visualization '93, VIS '93. Washington, DC, USA: IEEE Computer Society; 1993. p. 119–25. ISBN 0-8186-3940-7.
[21] Endert A, Han C, Maiti D, House L, Leman S, North C. Observation-level interaction with statistical models for visual analytics. In: 2011 IEEE conference on visual analytics science and technology (VAST); 2011. p. 121–30. http://dx.doi.org/10.1109/VAST.2011.6102449.
[22] Endert A, Bradel L, North C. Beyond control panels: direct manipulation for visual analytics. IEEE Comput Graph Appl 2013;33:6–13. http://dx.doi.org/10.1109/MCG.2013.53.
[23] Monnig ND, Fornberg B, Meyer FG. Inverting non-linear dimensionality reduction with scale-free radial basis interpolation. CoRR; 2013. abs/1305.0258.
[24] Wendland H. Scattered data approximation. New York: Cambridge University Press; 2004.
[25] Anderson E, Bai Z, Dongarra J, Greenbaum A, McKenney A, Du Croz J, et al. LAPACK: a portable linear algebra library for high-performance computers. In: Proceedings of the 1990 ACM/IEEE conference on supercomputing, Supercomputing '90; 1990. p. 2–11.
[26] Mongillo M. Choosing basis functions and shape parameters for radial basis function methods. SIAM Undergraduate Research Online; 2011.
[27] Fasshauer G, Zhang J. On choosing "optimal" shape parameters for RBF approximation. Numer Algorithms 2007;45(1):345–68.
[28] Martins RM, Coimbra DB, Minghim R, Telea AC. Visual analysis of dimensionality reduction quality for parameterized projections. Comput Graph 2014;41:26–42. http://dx.doi.org/10.1016/j.cag.2014.01.006.
[29] Levine MD, Yu Y. State-of-the-art of 3D facial reconstruction methods for face recognition based on a single 2D training image per person. Pattern Recognit Lett 2009;30(10):908–13.
[30] Blanz V, Vetter T. A morphable model for the synthesis of 3D faces. In: Proceedings of the 26th annual conference on computer graphics and interactive techniques, SIGGRAPH '99; New York, NY, USA: ACM Press/Addison-Wesley Publishing Co.; 1999. p. 187–94.
[31] Kirtzic JS, Daescu O. Face It: 3D facial reconstruction from a single 2D image for games and simulations. In: 2013 International conference on cyberworlds, vol. 0; 2011. p. 244–8.
[32] Macêdo I, Vital Brazil E, Velho L. Expression transfer between photographs through multilinear AAM's. In: Proceedings of SIBGRAPI 2006—XIX brazilian symposium on computer graphics and image processing. IEEE Computer Society; 2006. p. 239–46.
[33] Nguyen HT, Ong EP, Niswar A, Huang Z, Rahardja S. Automatic and real-time 3D face synthesis. In: VRCAI'09; 2009. p. 103–6.
[34] Parke FI. A parametric model for human faces [PhD thesis]; 1974. AAI7508697.
[35] Cohen MM, Massaro DW. Modeling coarticulation in synthetic visual speech. In: Models and techniques in computer animation. Tokyo: Springer-Verlag; 1993. p. 139–56.
[36] Sheng Y, Sadka AH, Kondoz AM. Automatic single view-based 3-d face synthesis for unsupervised multimedia applications. IEEE Trans Circuits Syst Video Technol 2008;18(7):961–74.
[37] Patel N, Zaveri M. 3d facial model construction and expressions synthesis using a single frontal face image. Int J Graph 2010;1:1–18.
[38] Buck I, Finkelstein A, Jacobs C, Klein A, Salesin DH, Seims J, et al. Performance-driven hand-drawn animation. In: NPAR 2000: first international symposium on non photorealistic animation and rendering; 2000. p. 101–8.
[39] Lewis JP, Cordner M, Fong N. 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. ACM Press/Addison-Wesley Publishing Co.; 2000. p. 165–72.
[40] He X, Yan S, Hu Y, Niyogi P, Zhang HJ. Face recognition using Laplacian faces. IEEE Trans Pattern Anal Mach Intell 2005;27(3):328–40.
[41] Computer Graphics Laboratory, I. VISGRAF faces database. ⟨http://app.visgraf.impa.br/database/faces⟩; 2012. [Online; accessed 31.03.2014].
[42] Schulz A, Velho L. ChoreoGraphics: an authoring environment for dance shows. In: ACM SIGGRAPH 2011 Posters. SIGGRAPH '11; New York, NY, USA: ACM; 2011. p. 1.