ENRIQUECENDO ANIMAÇÕES EM QUADROS-CHAVES ESPACIAIS COM MOVIMENTO CAPTURADO

Bernardo Fortunato Costa

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientador: Claudio Esperança

Rio de Janeiro
Junho de 2018

ENRIQUECENDO ANIMAÇÕES EM QUADROS-CHAVES ESPACIAIS COM MOVIMENTO CAPTURADO

Bernardo Fortunato Costa

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Claudio Esperança, Ph.D.

Prof. Luiz Gustavo Nonato, D.Sc.

Prof. Waldemar Celes Filho, D.Sc.

Prof. Paulo Roma Cavalcanti, D.Sc.

Prof. Ricardo Cordeiro de Farias, Ph.D.

RIO DE JANEIRO, RJ – BRASIL
JUNHO DE 2018

Costa, Bernardo Fortunato

Enriquecendo animações em quadros-chaves espaciais com movimento capturado/Bernardo Fortunato Costa. – Rio de Janeiro: UFRJ/COPPE, 2018.

XIII, 58 p.: il.; 29,7cm.

Orientador: Claudio Esperança

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2018.

Referências Bibliográficas: p. 45 – 49.

1. animação. 2. quadro-chave. 3. motion capture. 4. projeção multidimensional. I. Esperança, Claudio. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

*Este trabalho é dedicado a todos
aqueles que acreditam na
perseverança e na obstinação
enquanto qualidades humanas.*

Agradecimentos

Gostaria de agradecer a todos meus colegas do Laboratório de Computação Gráfica pela ajuda que me foi oferecida, em especial as pessoas de Andrea Lins, Alex Maturano e Pedro Asad, os que mais me ajudaram pessoalmente. Gostaria também de agradecer ao meu orientador Claudio Esperança pela oportunidade de ter realizado este trabalho e também pela grande ajuda em sua realização. Agradeço também aos demais professores do Laboratório de Computação Gráfica Ricardo Marroquim, Paulo Roma e a Antonio Oliveira pela conviência enriquecedora. Agradeço também a Rodrigo Toledo, professor do DCC/UFRJ, por seus conselhos a respeito de um trabalho de doutorado e também a Antonio Adilson pelo esforço pessoal prestado em tentar conciliar este trabalho com meu antigo emprego. Agradeço também à minha família, em particular minha mãe, pela ajuda prestada em momentos de dificuldade. Por fim, gostaria de agradecer a CAPES pela bolsa que possibilitou este trabalho.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

ENRIQUECENDO ANIMAÇÕES EM QUADROS-CHAVES ESPACIAIS COM MOVIMENTO CAPTURADO

Bernardo Fortunato Costa

Junho/2018

Orientador: Claudio Esperança

Programa: Engenharia de Sistemas e Computação

Movimento capturado (*mocap*) produz animações de personagens com grande realismo mas a um custo alto. A utilização de quadros-chave torna mais difícil um resultado com realismo mas torna mais fácil o controle da animação. Neste trabalho, mostramos como combinar o uso de quadros-chaves espaciais – Spatial Keyframing (*SK*) Framework – de IGARASHI *et al.* [1] e técnicas de projeção multidimensional para reutilizar dados de movimento capturado de várias maneiras. Mostramos também como projeções multidimensionais podem ser utilizadas para visualização e análise de movimento. Propomos um método de compactação de dados de mocap utilizando a reconstrução de poses por meio do algoritmo de quadros-chaves espaciais. Também apresentamos uma técnica de otimização para as projeções multidimensionais que melhora a reconstrução do movimento e que pode ser aplicada em outros casos onde um algoritmo de retroprojeção esteja dado.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

ENHANCING SPATIAL KEYFRAME ANIMATIONS WITH MOTION CAPTURE

Bernardo Fortunato Costa

June/2018

Advisor: Claudio Esperança

Department: Systems Engineering and Computer Science

While motion capture (*mocap*) achieves realistic character animation at great cost, keyframing is capable of producing less realistic but more controllable animations. In this work we show how to combine the Spatial Keyframing (*SK*) Framework of IGARASHI *et al.* [1] and multidimensional projection techniques to reuse mocap data in several ways. Additionally, we show that multidimensional projection also can be used for visualization and motion analysis. We also propose a method for mocap compaction with the help of SK's pose reconstruction (backprojection) algorithm. Finally, we present a novel multidimensional projection optimization technique that significantly enhances SK-based reconstruction and can also be applied to other contexts where a backprojection algorithm is available.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Character animation focuses on bringing life to a particular character model. There are several ways of animating a character in a scene. One popular way is rigging a skeleton to a character model (a skin) such that when the skeleton pose is changed, so does the model. This binds the character movements to the degrees of freedom (DOF) of this skeleton.

In standard keyframe-based animation, some key poses are created by artists and interpolated at each frame, sometimes using manually adjusted Bézier curves. However, for complicated or extended animations, this process turns out to be difficult and time-consuming. As an alternative, the approach known as *motion capture* or *mocap* was developed. In it, a human actor wearing a special suit performs movements which are recorded by a set of cameras and sensors. After some processing, it is possible to build a complete description of the actor's movement in the form of a set of skeleton poses and positions sampled with high precision, both in time and space. A typical mocap file has a description of the skeleton's shape and articulations, together with a set of poses, each containing a timestamp, the rotation of each joint and a translation vector of the root joint with respect to a standard rest pose.

Another animation authoring framework called *spatial keyframing* (SK) was proposed by IGARASHI *et al.* [1]. The main idea is to associate keyframes (poses) to carefully placed points on a plane rather than to points in time. Although simple in thesis, spatial keyframing still requires keyframe poses to be authored manually, which is a time-consuming task when many such poses are required or when the skeleton contains many joints. One way to help the process, therefore, is to harvest interesting poses from raw mocap files. The present work investigates algorithms and techniques to accomplish just that. Moreover, we propose using multidimensional projection techniques to automatically suggest an optimal placement for the spatial keyframes on the plane.

Another benefit of finding a good method to project points in pose space to a plane is that it also serves as a tool for the analysis of mocap files. The idea is that the movement contained in a mocap can be visualized as a trajectory in 2D space. Since a good projection method ensures that similar poses are projected onto points close to each other, the plane itself can be viewed as a pose similarity space. Thus, for instance, a cyclic movement is commonly projected onto a closed curve.

Finally, multidimensional projection of mocap data can be used as a tool for motion compression. Since mocap files ordinarily contain in excess of 60 frames per second, a common lossy compression scheme consists of selecting important frames and reconstructing the complete motion by interpolation. Although this interpolation is frequently conducted using time as parameter, we show how spatial keyframing can be adapted for this purpose.

In a nutshell, the following items can be listed as contributions of this work:

1. Repurposes multidimensional visualization techniques to the problem of selecting key poses from mocap data and project them on a plane so that they can be used in the Spatial Keyframe animation Framework (SKF).

2. Empirically evaluates several multidimensional projection schemes in their application to mocap data.

3. Describes the use of SKF in compressing mocap data through decimation and reconstruction and, in particular, introduces a non-linear projection optimization algorithm that yields smaller reconstruction errors. This algorithm is not specific to mocap data, but can also be used with other data, provided a back-projection algorithm and an error metric are available.

Chapter 2

Related work

In this chapter, we review the literature from areas related to the present work.

2.1 Animation authoring

Authoring in the context of animation is the process of creating new content manually or using specialized techniques. Traditional keyframe animation [2] has been widely used in the animation industry as the main way to create content. This technique consists of creating a skeleton for a given character[1] and configuring its pose, i.e., the rotations and positions of its bones on the scene for a set of timestamps. Each pair pose/timestamp is called a *keyframe*. The animation is produced by estimating poses ("in-betweens") for time instants taken in succession using keyframe interpolation.

Later, motion capture (*mocap*) became available to animators as a way to reach realism by reproducing a movement recorded in a real environment by actors. As a general rule, this type of animation achieves greater quality compared to traditional keyframing, where key poses are manually crafted by the animator. At an early stage, mocap data was used as a single shot without concerns to further reuse in the future rather than the simple reproduction of the recorded movement. As the production of mocap-based animation grew, mocap data started to become available for the general public at lower cost. The use of mocap to create new content remained unexplored until KOVAR *et al.*[3] showed a way to reuse mocap data other than by simple reproduction. It uses the definition of motion graphs where short clips taken from mocap files are viewed as vertices and possible transitions between them are modeled as edges. Thus, a longer movement sequence can be defined as a path on the graph where transitions are smoothed

[1]In the context of this work, a character is described by an articulated hierarchy of bones or simply a skeleton. However, keyframing can also be used to animate non-articulated or inanimate objects.

out with the aid of a pose interpolation algorithm. Following their idea, LEE *et al.*[4] enhances motion graphs with a sketching module to describe the movement trajectory.

The work of PULLEN and BREGLER[5] tries to produce new animations using keyframe and motion capture. It it one of the earliest works in the context of animation that tries to mix the strengths of both methods: the quality achieved by mocap with the flexibility provided by keyframes. Their idea is to make mocap data assist keyframe animation lending details not specified by the animator. SAFONOVA *et al.* [6] describes a sketch-based interface so that the animator may roughly specify the character movement which will later be enhanced by the addition of mocap and defined restrictions, such as first and last pose and contacts on the floor or objects. They use a mocap database in compact format to synthesize the sketched movement which later is adapted to satisfy all restrictions imposed by the animator.

Of special interest in the context of the present work is the character animation framework proposed by IGARASHI *et al.*[1] called *Spatial Keyframing* (*SK*). That framework aims at producing casual animations where a small number of poses are manually crafted and associated with special points on a plane that is used as an interaction space[2]. This is combined with a novel pose interpolation algorithm based on Radial Basis Functions (*RBFs* [7]), such that any point on the interaction plane can be mapped to a different pose. A new animation can then be synthesized by moving a mouse or some other pointing device on the plane. This process not only provides the required sequence of poses, but also the timing for the movement. SK does not address the problem of synthesizing positions for the moving character, but only the rotations of the skeleton joints, although an ingenious heuristic is suggested based on tracking the contact point between the skeleton and the floor or other surfaces.

2.2 Pose selection and information extraction

Decimating irrelevant poses from mocap data is a common way to produce a compact representation of a movement. The reconstruction of the original data is done by interpolating the small subset of poses that survive this decimation process. Since these play the same role of keyframes used in keyframe-based animation, the process is also called keyframe extraction.

A popular idea for extracting keyframes is curve simplification. LIM and THALMANN [8] view mocap data as a curve parameterized by time and pro-

[2]The authors even admit using a 3D volume as an interaction space, but interaction in 3D is considerably harder to control.

poses using a technique [9] for approximating such curves with a polygonal line. Later, this algorithm became known as simple curve simplification (SCS). A pseudocode version for it can be seen in Algorithm 1. This idea of using curve simplification heuristics was later enhanced by XIAO *et al.* [10] by employing an optimized selection strategy, which they named layered curve simplification (LCS). They try to optimize the running time of the SCS algorithm by discarding some frames for a first solution. If the reached result is below a certain quality threshold, these discarded frames can be reconsidered to provide a full scan solution. Both SCS and LCS start with a minimal subset of the original mocap data and try to repeatedly select relevant keyframes by measuring the similarity between a local interpolation and the original frame. On the other hand, TOGAWA and OKUDA [11] do the reverse: start with the full set and iteratively discard frames which least contribute to the interpolation, naming their contribution as position-based (PB) strategy. Algorithm 2 shows in pseudocode the details of how PB searches for keyframes.

Other possible approaches include clustering methods, matrix factorization or genetic algorithms. Clustering methods divide frames into clusters and search for a representative frame in each group. The works of BULUT and CAPIN [12] and HALIT and CAPIN [13] fall in this category. A relevant contribution of both is to consider also dynamic information in their clustering metrics. Matrix factorization uses linear algebra methods to reconstruct mocap data represented in matrix format. The work of HUANG *et al.* [14], called *key probe*, and the work of JIN *et al.* [15] are examples of such algorithms based on matrix factorization. Genetic algorithms is a solution search approach inspired in the natural selection of species in biology: possible solutions are mixed to find new ones at each iteration. The ones that show gain produce new "offspring" while the ones that do not are not selected for reproduction in next iterations. The work of ZHANG *et al.* [16] tries to find keyframes using this heuristic. Although all these other approaches present some advantages, curve simplification methods became more popular due to a mix of heuristic simplicity and satisfactory results.

2.3 Multidimensional projection

The key motivation behind the use of dimensionality reduction approaches in the this work is that much of the redundancy found in mocap data can be attributed to DOFs that are hierarchically or functionally related. Note that in the context of this work, the terms "dimension reduction" and "multidimensional projection" are used interchangeably. ARIKAN [17] proposes to use principal component analysis (PCA)[18] on mocap data, together with clustering, to get a good rate of compaction. SAFONOVA *et al.* [6] also use PCA to compact their recorded mocap database and synthesize new movements with the help of a sketch-based interface. PCA discovers the main orthogonal data axes and uses them as a basis to rewrite the original data. By discarding the least relevant axes, a more compact basis can be constructed, which can be used to represent a good linear approximation of the original data. Algorithm 3 details the main steps of how to build a PCA projection given some data. HALIT and CAPIN [13] and JIN *et al.* [15] also use PCA as a way to lower the data dimensionality and save computing time.

Dimension reduction methods can be categorized in many ways. In the context of this work, we divide them with respect to two characteristics: the proposed projection function type and the solution search strategy for building this function. In this way, methods can provide projection functions which might be linear or non-linear. The search for these functions can also favor data locality, in which case we should refer to them as having a local search approach, while a global search approach looks for a solution with no commitment to preserving distance relations in small neighborhoods. PCA, for instance, is a linear projection function with a global search approach.

ZHANG and CAO [19] and JIN *et al.* [15] use locally linear embedding (LLE) [20], a dimension reduction tool, to ease their search for keyframes in the frame set. LLE tries to find a projection where relative distances between each point and their nearest neighbors in lower dimension space is preserved in the least squares sense. The number of nearest neighbors is a parameter of the algorithm. Algorithm 4 describes in more details how LLE finds its projection. LLE produces a non-linear projection using a local search strategy.

ASSA *et al.* [21] also project the motion curve onto a 2D space to find keyframe candidates. They use a variant of multidimensional scaling (MDS) [22, 23] to project the motion curve. MDS tries to find a projection such that relative distances in lower dimension space are as close as possible to the corresponding distances in the original space in the least squares sense. The distance definition is a parameter to be chosen. If the euclidean distance is chosen, MDS produces the same result as PCA. JENKINS and MATARIĆ [24] use another MDS sibling method called Isomap [25] to reduce human motion to a smaller dimension for clustering purposes. Isomap turns out to be MDS with geodesic distance as its distance definition. The geodesic distance matrix can be calculated from the ordinary distance matrix by setting to infinity all distances from element pairs that do not have one of them as their nearest neighbor and then running the Floyd-Warshall algorithm 5 to compute the shortest paths over the matrix until all infinity values are updated. Algorithm 6 describes how a solution can be tailored for both approaches given a distance matrix as input.

It is worth noting that dimension reduction techniques such as PCA, MDS and LLE, besides being used for compaction, can also be employed for other purposes, most notably for visualization of multidimensional data. For instance, TEJADA *et al.* [26] proposed *Force*, a fast iterative approach to project multidimensional data onto a 2D space, based on neighborhood relationships. Initially, a distance matrix in high dimensional space is built, after which a random projection to 2D is constructed and iteratively improved using spring repulsion forces to try to mimic the distance relationships encoded in the distance matrix. Algorithm 7 shows a pseudocode version of Force. This approach is not guaranteed to be optimal but has been shown to produce interesting results for data visualization.

Another dimension reduction technique aimed at visualization of multidimensional data is the t-Distributed Stochastic Neighborhood Embedding (t-SNE) [27, 28]. It has shown promising results with respect to preserving multidimensional data neighborhood on low dimensional projections. Instead of building the low dimensional data to resemble the distance relations of a distance matrix in high dimension, like Force or MDS does, it uses conditional probabilities as its guide the multidimensional projection. The conditional probabilities of both dimensions are calculated using the pairwise distance of points which produces the so called similarity matrix. The high dimension similarity matrix calculus forces it to be symmetric and the low dimension similarity matrix uses a t-Student distribution function to calculate its similarity. The t-SNE tries to find a projection which minimizes the divergence of both similarity matrices by means of gradient descendant optimization, relocating low dimensional data and recalculating its similarity matrix at each iteration, making it a non-linear projection method. Algorithm 8 shows how these calculations are done until a solution is reached. The locality of t-SNE depends on a parameter called perplexity, which determines the number of effective neighbors while calculating the high dimension similarity matrix. The higher the perplexity, the closer to a global solution t-SNE is.

JOIA *et al.* [29] created the so-called local affine multidimensional projection (LAMP), where a small subset of points – called control points – are randomly picked from the original set and projected using some projection algorithm, whereas the remaining points are projected by means of an affine map expressed in terms of their distance to the control points in the original space. The control points can be moved on the projection plane and thus affect the projection of the whole set. This two-step projection approach aims at providing a way to interactively explore the high-dimensional point set by offering smoothly varying projected views. LAMP's locality comes from its weighting scheme where closer control points have greater influence than far ones.

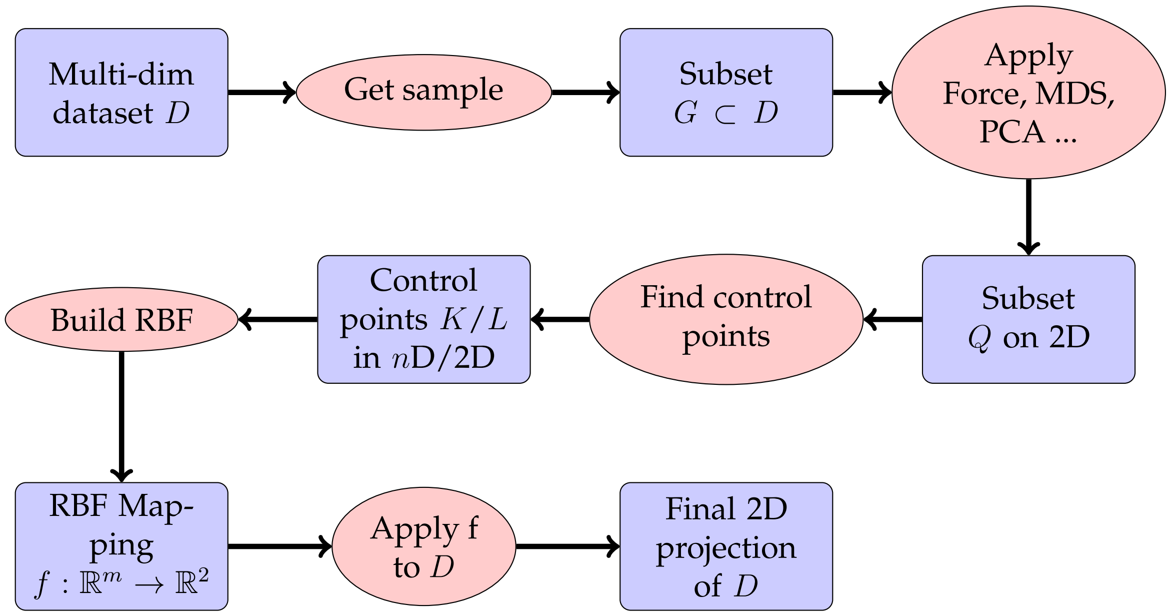

Multi-dim dataset D
Get sample
Subset $G \subset D$
Apply Force, MDS, PCA ...
Subset Q on 2D
Find control points
Control points K/L in nD/2D
Build RBF
RBF Mapping $f : \mathbb{R}^m \rightarrow \mathbb{R}^2$
Apply f to D
Final 2D projection of D

Figure 2.1: Basic RBFP scheme with control points selection using ROLS.

AMORIM *et al.* [30] propose another two-step projection approach which is similar to LAMP, but the affine map is replaced by radial basis functions (RBFs) [7]. In this projection scheme – RBFP for short – control point selection is aided by a technique called regularized orthogonal least squares (ROLS), a more elaborate approach compared to the random selection of LAMP. Notice that RBFs have an advantage over affine maps in the sense that they provide a smoother interpolation. Figure 2.1 shows the stages needed to find a final projection of all points with this scheme.

Algorithm 9 details the *find control points* step where we can consider keyframes as control points in the context of mocap data. Note that this algorithm uses as input a subset of the original data as well as their projections to 2D, computed with some other projection algorithm, e.g., Force, LLE, etc. This is done in order to reduce the computational cost of this first projection, but also of the successive Gram-Schmidt orthogonalizations. ROLS selects each control point by using a standard regression model to determine which instance of G is responsible for the biggest reduction in the amount of squared error. This is done by examining each column of matrix A associated with the RBF interpolation (see Section 2.5). Once the control point is selected, the influence of the corresponding column is eliminated from A using Gram-Schmidt orthogonalization. After the control point set K and their projections L are computed, Algorithm 10 can be invoked to obtain the projection of the remaining (non-control) points.

2.4 Backward projection

"Unprojection" or backward projection is a less developed field compared to dimension reduction. It aims at producing points in the original multidimensional space which do not belong to the input data set. For instance, the iLAMP [31] approach uses the same LAMP[29] heuristic to get a backward projection, swapping high dimension with lower dimension. AMORIM *et al.* [32] use RBF interpolation to transport information from the reduced dimension to the original space. Their work aims at exploring facial expression generation interpolated from a set of control points selected with specialized heuristics.

IGARASHI *et al.*[1] also use RBF interpolation to back-project a 2D point onto a multidimensional skeleton pose, as a way to provide a rapid prototyping environment for animators. In this scheme, each input skeleton pose is modeled and associated with a *marker point* manually placed on a plane. The idea is to build an RBF which maps any point on the plane to a pose smoothly interpolated from the marker point poses. The interpolating RBF uses as a parameter the distances from the point in question to the various marker points, in order to estimate the coefficients of the rotation matrix at each skeleton joint. The input is usually a two dimensional point while the output is a set of 9-dimension vectors corresponding to coefficients of a rotation matrix of each skeleton bone. The resulting matrices are then orthonormalized using a relaxation algorithm that yields pure rotation matrices. The authors convincingly argue that the resulting interpolations are superior to those obtained by interpolating quaternion coefficients.

The SK animation can be built in two stages. In a first stage, a set of poses K chosen as keyframes is associated to a set of markers L. The idea is to build an RBF f such that $f(l_i) = k_i$ for $l_i \in L$ and $k_i \in K$. The markers are usually two-dimensional vectors while the pose is a vector of size nine times the number of rotational degrees of freedom (DOF) of the skeleton. From these values, the RBF coefficients are estimated. These coefficients will be used in a second stage where new poses are created using the position of a controller in the marker space which is handled by the animator in real time. The controller position is the input of the previously built RBF which will produce a matrix of coefficients for every skeleton DOF. These coefficients must be orthonormalized so the resulting matrices are pure rotation matrices. It is also possible to include in the specification a vector describing the position of the root joint for each pose.

Since mocap data is usually thought of as a function that maps time instants onto a pose, unprojection can be regarded as simple time-based interpolation. The most common strategy is to use a simple linear interpolation between the two closest keyframes in time[8], while XIAO *et al.*[10] extended this idea by using hermitian interpolation between successive keyframes as a way to consider dynamic information. In both works, rotation is usually interpolated with Euler angles.

2.5 RBFs and kernels

Radial Basis Functions (RBFs) [7] are used in several places in this work. Firstly, it is the basis of the spatial keyframe back projection algorithm [1]. Secondly, it is the basis for the ROLS control point selection algorithm and the associated RBFP projection approach. An RBF is a real-valued function that approximates output values depending on the current distance between the input point and the points where the function is known. These distance definitions are known as RBF kernels and they smooth the RBF function in different ways, making locality influence greater or smaller.

Equation 2.1 describes the general terms of this function. The RBF kernel is the $\phi(\cdot)$ function and w_j are weights that must be calculated given a pair of input and output values, where the latter is expected to be single dimensional. However, it is still possible to use RBFs to interpolate multidimensional data considering each dimension a different RBF with its proper w_j and $P(x)$ coefficients. There are K values c_j called constraints which are input values where the function output is known. The term $P(x)$ is optional and stands for a linear polynomial component in the form $P(x) = p_0 + p_1x_1 + \cdots + p_nx_n$. ROLS and RBFP do not use this component which is present in the spatial keyframe back-projection algorithm usually as a two dimensional vector.

$$f(x) = \sum_{j=1}^{K} w_j\phi(x - c_j) + P(x) \tag{2.1}$$

Given a known pair of input and output values (c_i, h_i), we can rewrite a set of Equations 2.1 with these values into a linear system problem described in the customary matrix form $AW = B$. Equation 2.2 shows such a system for the case where input points c_i are three-dimensional and the h_i points used in B are one-dimensional. The last four rows are added by imposing restrictions over the sum of weights so a unique solution is guaranteed in the system with a full rank matrix. If output values are k-dimensional, rather than solving the system once for every dimension, it is possible to first invert A and then find the weight matrix W by computing $A^{-1}B$. A pseudo-code for computing the weights for a RBF function that maps an array of points K into a corresponding array of points

L is given in Algorithm 11.

$$AW = \begin{bmatrix} \phi_{11} & \phi_{12} & \dots & \phi_{1k} & 1 & c_1^x & c_1^y & c_1^z \\ \phi_{21} & \phi_{22} & \dots & \phi_{2k} & 1 & c_2^x & c_2^y & c_2^z \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_{k1} & \phi_{k2} & \dots & \phi_{kk} & 1 & c_k^x & c_k^y & c_k^z \\ 1 & 1 & \dots & 1 & 0 & 0 & 0 & 0 \\ c_1^x & c_2^x & \dots & c_k^x & 0 & 0 & 0 & 0 \\ c_1^y & c_2^y & \dots & c_k^y & 0 & 0 & 0 & 0 \\ c_1^z & c_2^z & \dots & c_k^z & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_k \\ p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_k \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = B \tag{2.2}$$

The spatial keyframes scheme uses the modular distance – Equation (2.3) – as its RBF kernel, i.e., a linear kernel in Euclidean space. Such kernel is not guaranteed to preserve locality, but the authors of [1] claim that it produces the best quality animation interpolation. For this reason, we are sticking with traditional spatial keyframe kernel for evaluation purposes.

$$K_m(r) = \|r\| \tag{2.3}$$

RBFP also uses RBFs for selecting control points and projecting the remainder poses, and we follow [30] by using the multiquadric distance – Equation (2.4) – as RBF kernel, where constant ϵ is set to 1.

$$K_{mq}(r) = \sqrt{1 + (\epsilon r)^2} \tag{2.4}$$

Chapter 3

Mocap compaction

In this chapter we discuss the problem of mocap compaction (or compression), i.e., the problem of representing the data for a mocap session with only a few of the original poses. A compression scheme is a coupling between a pose selection strategy and a pose interpolation (or reconstruction) algorithm. We distinguish two broad classes of mocap compression approaches. The first, what we may call the "traditional" approach, consists of selecting a few representative poses from the original set – the keyframes – while the reconstruction uses time as the interpolating parameter. The second approach also selects representative poses, but uses multidimensional projection, so that all poses are mapped to points in 2D, while the reconstruction uses the 2 coordinates of this space as interpolating parameters. This way, we could consider the selected poses for schemes in the first approach as *temporal keyframes* while those for the second approach could be more properly called *spatial keyframes*.

3.0.1 Temporal keyframe schemes

The idea for these schemes is to obtain a compressed representation of the original mocap data by carefully selecting a subset of poses, with their attached timestamps, as shown in Fig. 3.1. Let the original mocap contain n poses $\mathbb{F} = \{p_1, p_2, \ldots, p_n\}$ uniformly spaced in time, and let the compressed representation consisting of a list with $m < n$ *keyframe* poses $\mathbb{K} = \langle p_{i_1}, p_{i_2}, \ldots, p_{i_m}\rangle$ and a list of the corresponding timestamp indices $\mathbb{T} = \langle i_1, i_2, \ldots, i_m\rangle$. Then, in order to reconstruct a pose p_j not in $\mathbb{K}$ we first find the two closest poses p_{i_l} and $p_{i_{l+1}}$ in $\mathbb{K}$ so that $i_l < j < i_{l+1}$. Thus, the reconstructed pose p'_j can be obtained by linear interpolation of p_{i_l} and $p_{i_{l+1}}$. It is also possible to use a higher order function for interpolation, like a cubic hermite curve, if we add the tangent at p_{i_l} and $p_{i_{l+1}}$. In fact, in addition to these two, it is also possible to use infinite support reconstruction functions (RBFs, say), which use all of $\mathbb{K}$.

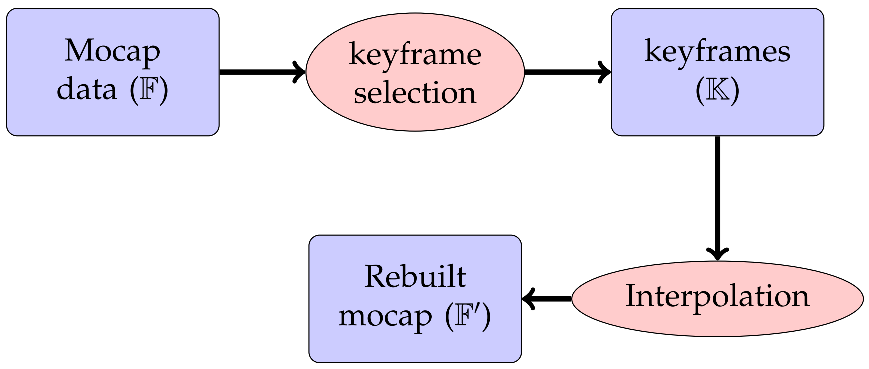


Figure 3.1: Workflow for a temporal mocap compression and reconstruction scheme.

3.1 Spatial keyframe schemes

Time is a natural interpolating domain for mocap data, since successive poses are necessarily similar due to physics constrains. We may, however, imagine an artificial "similarity" domain where two closely related poses are mapped to a pair of close points. We propose constructing such a domain by using multidimensional projection techniques. Thus, we propose adding to the keyframing pipeline a step where poses are first projected onto a plane. A skeleton pose is understood as a vector in high dimensional space, so that a sequence of projected poses with their corresponding timestamps can be regarded as a 2-dimensional trajectory representing the motion. This trajectory can be used to recover the original data with the help of a back-projection algorithm. We adopt the algorithm from IGARASHI *et al.* [1] to recover the multidimensional pose from the projected pose, since this algorithm was specifically devised for this kind of data.

Fig. 3.2 illustrates the workflow for such schemes. Mocap data will feed a keyframe selection algorithm, just as in a temporal scheme, but an additional step is required, where a multidimensional projection algorithm is used to project all poses onto a plane. Thus, in addition to a set $\mathbb{K}$ of keyframes, the compressed representation also includes set $\mathbb{Q} = \{q_1, q_2, \ldots, q_n\}$, where q_i represents a projection of p_i.

We should also distinguish common projection approaches from those employing control points, such as LAMP and RBFP. In the latter schemes, the set of control points could conceivably be conflated with the set of keyframes, in the sense that both consist of representative subsets of the original data. Thus, RBFP-based mocap compression and reconstruction follows the slightly different workflow depicted in Figure 3.3.

For implementing any keyframe selection algorithm in the context of motion capture, we need to define a function for pose distance, i.e., a measure of dissim-

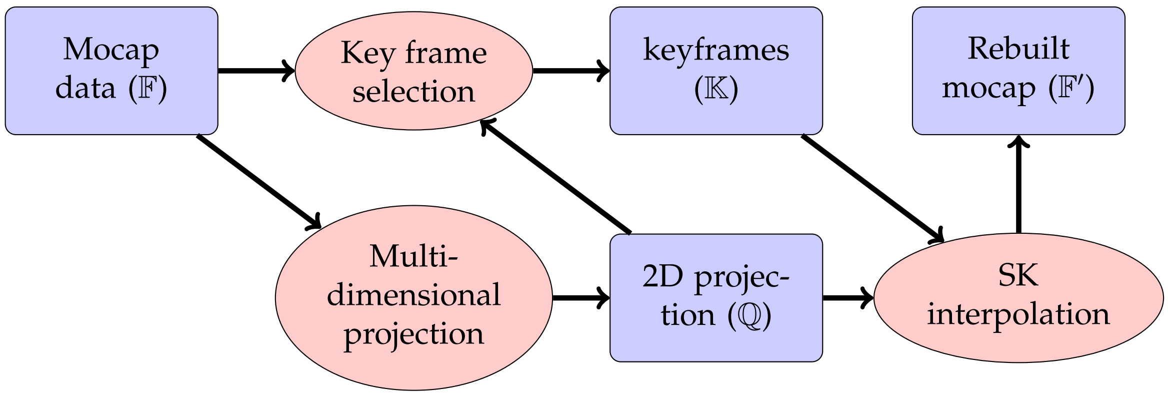

Mocap data ($\mathbb{F}$)
Key frame selection
keyframes ($\mathbb{K}$)
Rebuilt mocap ($\mathbb{F}'$)
Multi-dimensional projection
2D projection ($\mathbb{Q}$)
SK interpolation

Figure 3.2: Workflow for mocap compression and reconstruction with spatial keyframes.

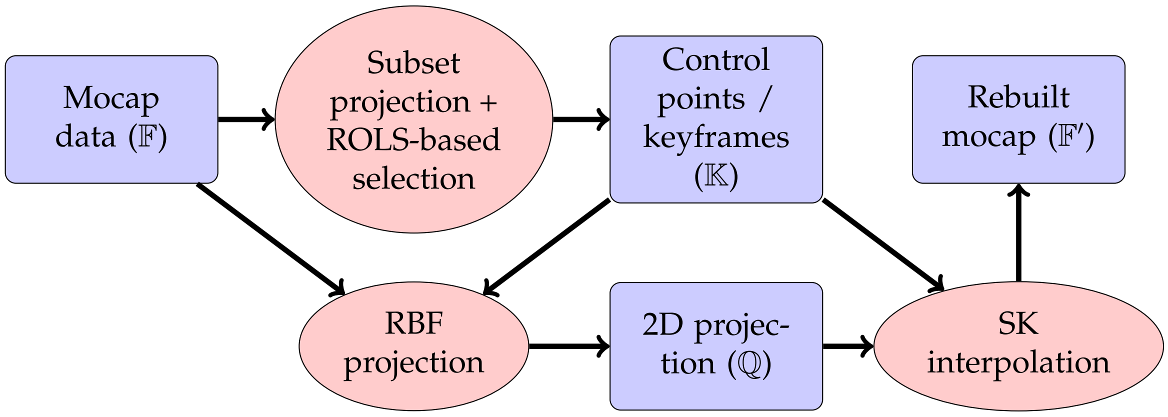

Mocap data ($\mathbb{F}$)
Subset projection + ROLS-based selection
Control points / keyframes ($\mathbb{K}$)
Rebuilt mocap ($\mathbb{F}'$)
RBF projection
2D projection ($\mathbb{Q}$)
SK interpolation

Figure 3.3: Workflow for mocap compression and reconstruction with spatial keyframes and RBF projection.

ilarity. One popular way to measure pose distance is a weighted sum of squared distances between hierarchically-equivalent joint positions. This is shown in Equation (3.1), where A and B are skeleton poses, J is the joint set, and w_j is a weight assigned to the j'th joint according to its importance. If no motivation exists to privilege certain members of the skeleton, it is customary to use uniform weights.

$$D_p(A, B) = \sum_{j \in J} w_j \|P(A_j) - P(B_j)\|^2 \tag{3.1}$$

A problem that arises when we stick to error metrics based on joint positions is that they are not translation-invariant, i.e., a pose can present a huge error if compared to itself repeated at a different position or, conversely, it might present a smaller error with respect to a completely different pose if the skeleton root joint is in the same position. Another way of measuring the pose distance is to use joint data in rotation space such as quaternions, rotation matrices or Euler angles. Quaternions and rotation matrices are preferable to Euler angles since they do not depend on the particular axis order used for the latter. However, this would bring us the difficulty of measuring error contributions in a hierarchical chain of joints, where errors in elements at higher levels of the hierarchy tend to cascade to those at lower levels. To mitigate this effect, we propose to use as joint weights the longest path between the joint and the furthest end effector. Equation (3.2) defines this distance metric in terms of rotation matrices, where A and B are skeleton poses, R is the rotation joint set composed of matrices, $\mathrm{Tr}[\cdot]$ is the trace operator and w_r is a weight assigned to the r'th joint. In our experiments, we use this rotational pose distance definition for pose projection.

$$D_q(A, B) = \sum_{r \in R} w_r \arccos\left(\frac{\mathrm{Tr}[A_r B_r^\top] - 1}{2}\right) \tag{3.2}$$

The following multidimensional projection methods were used in our experiments: among the linear methods and global, we have considered PCA [33], the Force approach [26] and MDS [22]. Among the non-linear and local methods, we have considered RBFP [30], Isomap [25], LLE [20] and t-SNE [27]. Figure 6.2 contains sample results obtained with all of these except RBFP. RBFP adopts a two-step projection approach using control points. In our experiments, the first step of RBFP was evaluated with the use of Force, Isomap, MDS, LLE, PCA and t-SNE. Recall that RBFP prescribes the use of ROLS for keyframe selection. Figure 3.4 illustrates projections for the same dataset using these five RBFP variations.

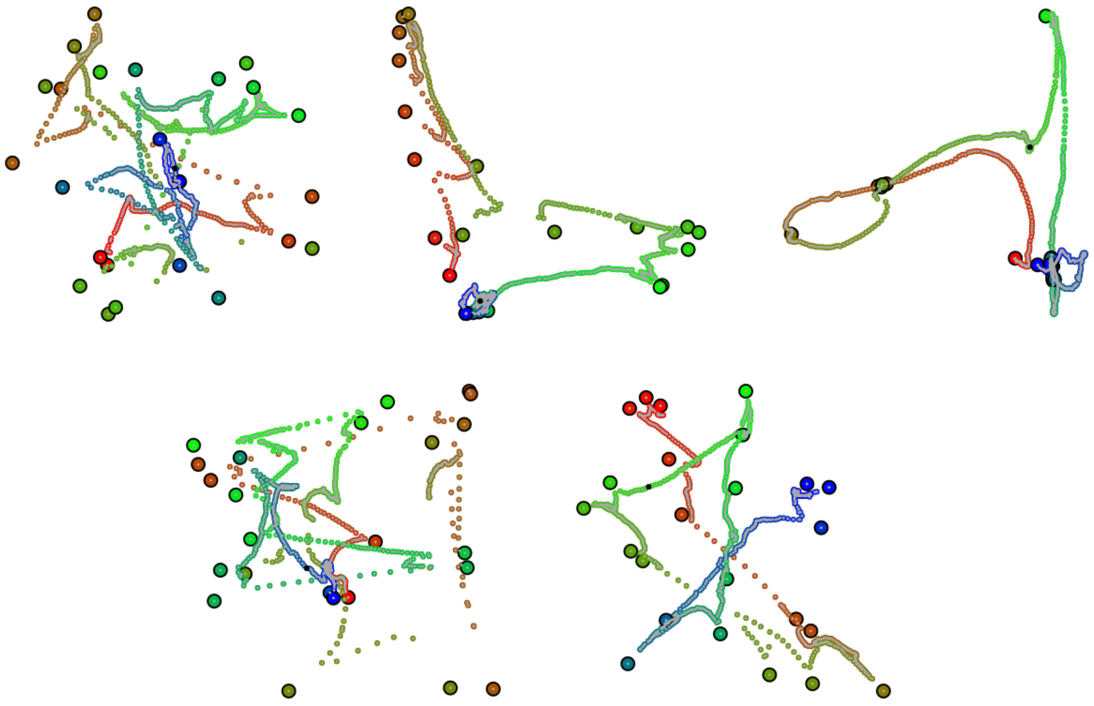


Figure 3.4: Results obtained by using two-step RBFP projection variants on poses of file `05_10.bvh` of the CMU Motion Capture database. The projection for the first step was obtained using the following algorithms (from left to right): Force, LLE, MDS, PCA and t-SNE.

Chapter 4

Projection optimization

In spatial keyframe schemes, the 2D projection of a key pose is reconstructed exactly by the back-projection algorithm, but we expect non-key poses to be reconstructed only approximately. Therefore, an important question is whether the projection of a non-key pose, as prescribed by the projection algorithm, is *optimal*, i.e., whether the back-projection of this point yields the pose with *minimum error* with respect to the original pose.

Having this in mind, we propose an optimization algorithm that will help us to find a better projection of non-key poses. This is shown in Algorithm 12, which takes as input the set of frames from the mocap ($\mathbb{F}$ in Figures 3.2 and 3.3), the subset of keyframes ($\mathbb{K}$), as well as the projection and back-projection algorithms and produces an optimized set of projected non-key poses ($\mathbb{Q}$). This algorithm implements gradient descent[34] optimization where, at each iteration, the gradient of a given function is estimated and steps in the gradient direction are taken to reach a local minimum. The original point computed by the non-optimized projection algorithm is used as a seed for the search.

An important component of the process is the estimation of the gradient at a given point. The numerical process used in our algorithm estimates the gradient around a particular 2D point by back-projecting four neighbor points within a disk of small radius and computing the error with respect to the original pose. If all points yield a bigger error than the original point, then that point is a local minimum. Otherwise, a gradient direction is computed based on the error variation. The radius is calculated taking the average distance between the analyzed point and its two nearest neighbors. Considering mocap data, we can simply choose the previous and next frame in time to save computing time. Parameters α and δ in the algorithm are typically chosen in the interval $[0, 1]$ and were set to 0.5 in our experiments.

Figure 4.1 shows the sample projections of Figure 6.2 optimized with Algorithm 12. We note that the optimization tends to scatter non-key frames with

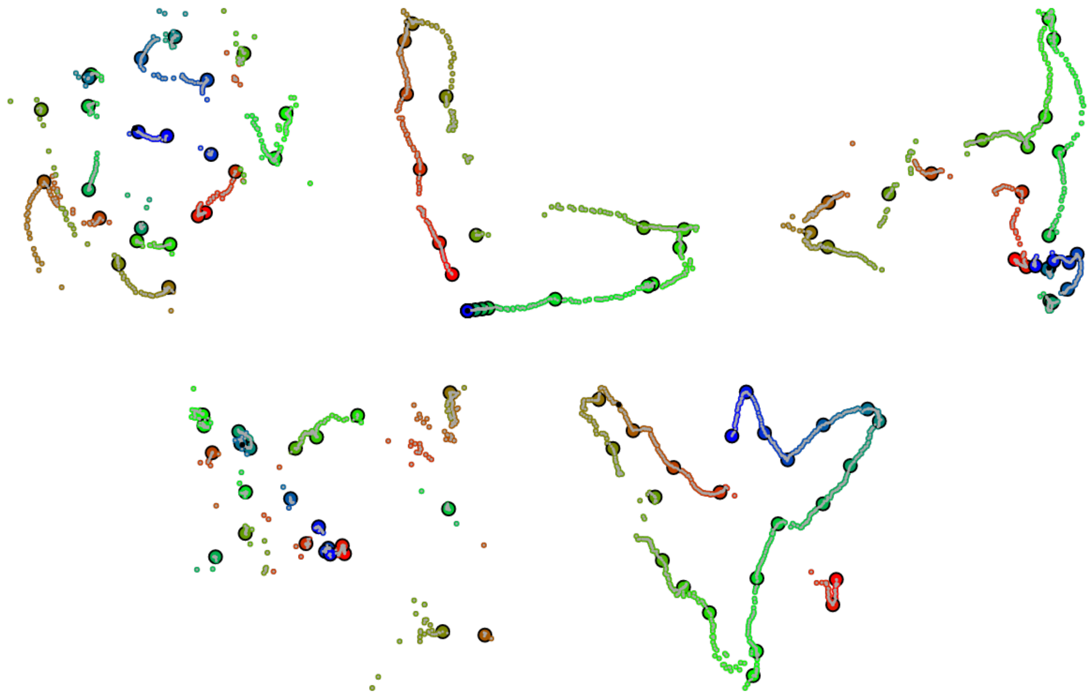


Figure 4.1: Optimized versions of the projections using Uniform Sampling shown in Figure 6.2. From left to right, first row has Force, LLE, MDS while second row has PCA and t-SNE.

respect to the non-optimized projections. Clearly, cases where greater differences exist between the optimized and original projections can be attributed to the original algorithm choosing poorer positions in the first place.

We also note that the optimization algorithm can be adapted to other uses of multidimensional projection for which a back-projection is available and an error metric is defined. Projection algorithms using control points such as LAMP [29] and RBFP [30] already prescribe corresponding back-projection algorithms ([31] and [32], respectively). If no back-projection algorithm is available, it is possible to use RBFs as in [32] or [1]. In this case, the optimization algorithm must be run with a subset of the input data as control points, and will optimize only the projection of the points not included in this set. This means that if, say 50% of the input is used in the back-projection, only the remaining 50% will be subject to displacement by the optimizer. On the other hand, if only a small set of control points is used, the back-projector will probably yield poorer estimations.

The projection optimization algorithm can also be evaluated by means of its capacity to preserve distance and neighborhood relations in the low dimension space, not only by its capacity to reproduce the original high dimension data. A common metric used for this purpose is the stress function (see Equation 4.1) which tries to measure how distance relations in high dimension space given by

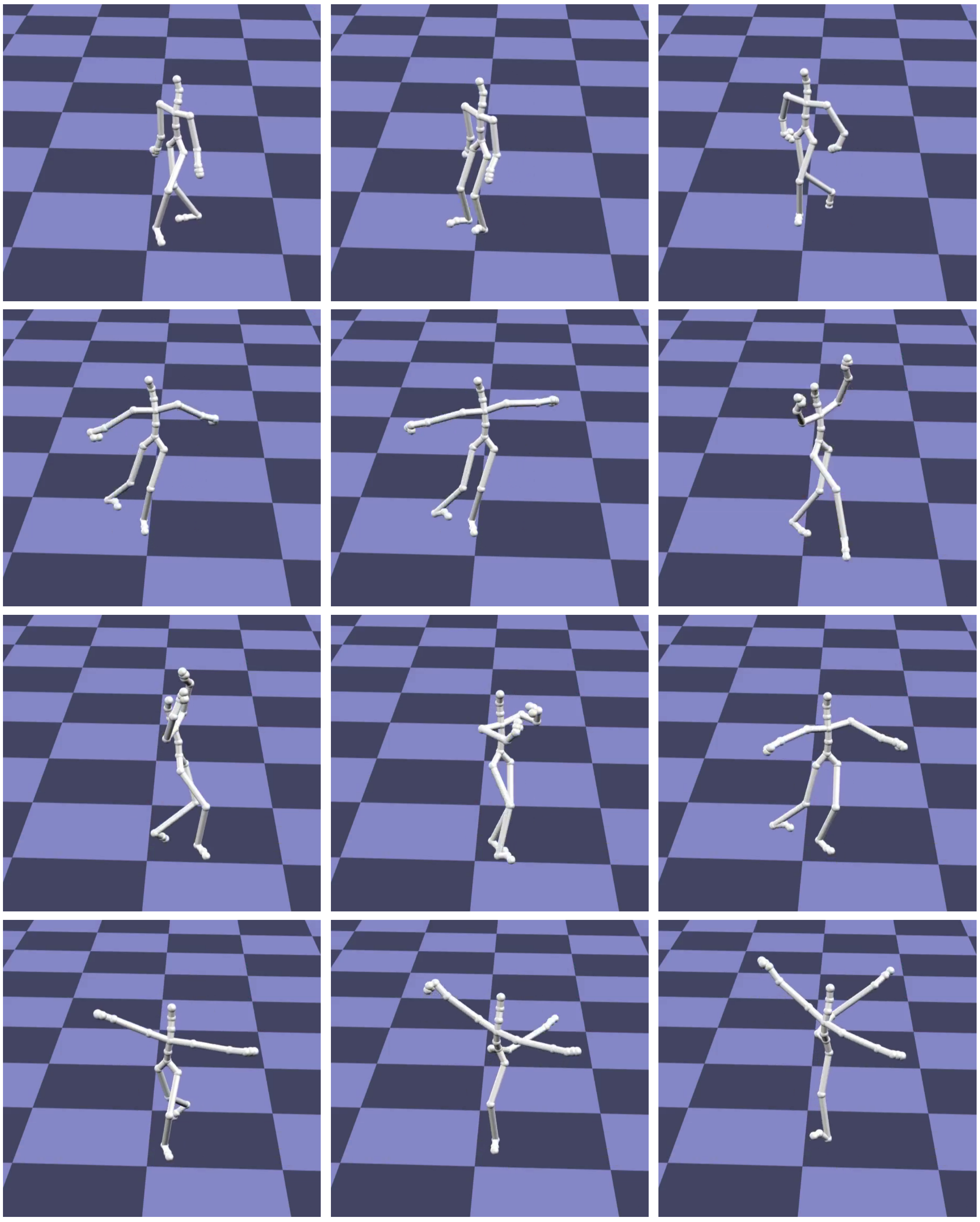


Figure 4.2: A take of 12 poses from file 05_10 which can be seen as projections in Figure 6.2 and Figure 4.1.

$d_{ij} = \|X_i - X_j\|$ are similar to the corresponding distances in low dimension space $\delta_{ij} = \|Y_i - Y_j\|$. In general, as the projection gets better, the stress becomes lower. Stress is also a metric used to measure the quality of multidimensional data visualization in 2D. In Chapter 5 we present several experiments with mocap data and other classic datasets that indicate a significant improvement in the quality of the projection obtained with RBFP.

$$s(X,Y) = \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}(\delta_{ij} - d_{ij})^2}{\sum_{i=1}^{n}\sum_{j=i+1}^{n}\delta_{ij}^2} \tag{4.1}$$

Chapter 5

Experiments

The main question we face is whether there is any gain in terms of quality when using spatial keyframes over traditional temporal schemes. To answer it, each compression scheme is evaluated by measuring the error between the reconstructed and the original animation, for a set of mocap files. The reconstruction quality is estimated using the error measured by Equation (3.2) divided by the number of frames in the mocap file and by the number of joints of the skeleton, to make this measure independent of file size or skeleton topology. We used as input data a subset of the CMU Mocap database[35] with 70 files ranging in size from 129 to 1146 frames. In particular, we used files in BVH format converted for the 3DMax animation software.

All experiments consist of compressing mocap data and measuring the error of the reconstructed mocap with respect to the original. The compaction schemes vary according to three main aspects:

(a) The keyframe selection strategy, chosen from: PB, SCS, ROLS, and Uniform Sampling (US), which selects keyframes at regular time intervals.

(b) The interpolation algorithm. For temporal schemes, these can be Linear or Hermitian interpolation of rotations expressed with Euler angles, or Spherical linear (Slerp) interpolation of rotations expressed as quaternions. All spatial schemes use the back-projection algorithm proposed by Igarashi et al. (see Section 3.1).

(c) For experiments using spatial keyframes we tested 6 main projection algorithms for the markers, namely: Force, MDS, LLE, PCA, Isomap and t-SNE. Our LLE implementation uses the 15 nearest neighbors to reconstruct its surrounding areas. Force uses 50 iterations to reach its final projection. Experiments with Isomap use 14 neighbors for the initial estimation of geodesic distances. Experiments with t-SNE use a perplexity value of 50. The projection algorithms correspond to multidimensional projection of frames using the strategies discussed earlier. In addition to these single-step schemes, we also tested 6 RBFP-based projection schemes (see Figure 3.3) where the first step (keyframe projection) is performed with one of the former projection algorithms and the remaining frames are projected with RBFs.

A preliminary batch of tests was conducted in order to investigate how the compaction schemes fare with respect to the desired compaction ratio. Since this ultimately depends on the ratio between keyframes and total frames (KF ratio), we selected 8 representative animations and four compaction schemes, two temporal (Linear and Slerp) and two spatial (MDS and t-SNE), all run with SCS keyframe selection strategy, and measured the obtained error for ratios between 1 and 10%. Figure 5.1 show the error plot of these files. This experiment reveals that error decreases sharply until reaching a KF ratio of about 4%, after which the error decreases at a slower rate. We used this observation to restrict further comparison tests to a KF ratio of 3%, since a smaller ratio would probably lead to bad reconstructions and a larger ratio would probably not reveal much about advantages of one scheme over another. It could be argued that rather than using a fixed KF ratio, a better comparison would maintain a desired minimum error and gauge what KF ratio would be required to attain it. Such an experimental setup, however, would be considerably more strenuous, since each scheme would have to be run several times, adjusting the KF ratio until reaching the desired error.

Figure 5.2 shows the average error per frame per joint for various combinations of selection and interpolation strategies at 3% KF ratio. The examination of these charts leads us to a few observations:

1. Temporal schemes as Slerp, Linear and Hermitian interpolation work well and are quite similar in terms of average error measure.

2. More sophisticate keyframe selection algorithms do not, in general, produce better results than the naive uniform sampling.

3. Spatial keyframe schemes produce good results but only if the optimized projection of (non-key) frames is used.

In particular, the overall best result was obtained with optimized t-SNE projection and uniform sampling, with an average error of 0.618 per frame per joint. On the other hand, the temporal scheme that yielded the smallest average error – 0.826 per frame per joint – was the Slerp interpolation combined with SCS keyframe selection.

The results shown in Figure 5.2 were produced by averaging the error over all 70 mocap files, which might hide important outliers. For a more detailed comparison, one may examine the box-plot shown in Figure 5.3 (see also the numeric values in Table 5.1)[1]. This chart omits the non-optimized spatial keyframe schemes and the Slerp-based temporal scheme. All experiments were run with keyframes selected by uniform sampling. The figure reveals that all temporal schemes exhibit a larger median than all spatial keyframe schemes. In particular, t-SNE has the smallest median and also the smallest first quartile. Isomap has the lowest minimum and LLE has the lowest third quartile. However, all except PCA show a few outliers. All spatial keyframe schemes with optimized projection show improvements with respect to temporal schemes. The gain related to the use of Spatial keyframe schemes is that the error per frame is reduced by up to 30 %, considering the median of t-SNE optimized projection and Slerp as a temporal scheme.

Scheme	**Boxplot values**					
	W_{bot}	Q_1	Q_2	Q_3	W_{top}	OL
Hermitian	0.252	0.543	0.719	0.992	1.618	5
Linear	0.24	0.545	0.732	1.025	1.721	3
Slerp	0.222	0.52	0.732	0.958	1.583	9
1D-RBF	0.222	0.515	0.729	0.958	1.613	8
Force	0.294	0.463	0.668	0.97	1.608	1
Isomap	0.254	0.429	0.54	0.864	1.293	4
LLE	0.249	0.4	0.525	0.806	1.242	2
MDS	0.291	0.428	0.587	0.815	1.317	2
PCA	0.263	0.437	0.628	0.97	1.698	0
t-SNE	0.252	0.382	0.515	0.809	1.476	1

Table 5.1: Boxplot values. W_{bot} and W_{top} are the values for bottom and top whiskers, Q_1 and Q_3 are first and third quartiles, Q_2 is the median and OL is the number of outliers.

Next, two-step projection schemes were evaluated with respect to their analogous one-step versions. We remind the reader that two-step methods such as LAMP and RBFP differ from their one-step counterparts in that only control points (keyframes in our case) are projected in the first step, while the remainder points use a different projector – affine maps in the case of LAMP and RBFs in the case of RBFP. Since early experiments using LAMP yielded consistently bad reconstructions, we present here only experiments with RBFP coupled with the ROLS selection strategy as suggested in [30]. Figure 5.4 shows a comparison chart between 1- and 2-step schemes, which indicates that RBFP yields poor results unless followed by optimization.

[1]This boxplot chart variation draws the bottom "whisker" at the value of the lowest data point greater than $Q_1 - 1.5IQR$, where Q_1 is the first quartile and IQR is the interquartile range. Similarly, the top "whisker" is the highest data point smaller than $Q_3 + 1.5IQR$. Values lying outside the range between whiskers are considered outliers.

The execution time measurement for all experiments includes (1) time required to perform keyframe selection and (2) time to reconstruct the original file from the selected keyframes. For SK schemes it was also measured the time required for the particular projection algorithm and the time taken by projection optimization if applicable. Of all of these steps, projection is by far the most costly, being three orders of magnitude bigger than both selection and reconstruction times, whereas optimization times range from 1 to 5% of the respective projection time. This indicates that temporal schemes are cheaper and faster than SK schemes. Taking a closer look at Figure 5.5 which shows average projection times for each algorithm, it is possible to notice that the best projectors are also the ones with largest running time. However, non-optimized projections were not as well suited as their optimized counterparts. In this figure, we can see that PCA is the fastest, while t-SNE and LLE have longer projection times. Force and MDS are slower than PCA but four times faster than t-SNE or LLE.

5.1 A note about compression

Our first tests with this framework showed a poor reconstruction for in-betweens if we neglected the 2D projection for each frame and tried to reconstruct mocap by tracing the 2D controller trajectory, using only a simple interpolation between the projected keyframes. This comes from the fact that mocap data is complex and 2D projections, in particular those obtained with optimization, are not smooth but contain cusps and gaps. Thus, the easiest way to encode such trajectories is to store the coordinates of all frame projections. This is not required for temporal compression schemes which only records keyframe timestamps. Once we add more information to encode trajectories, a trade-off arises between better reconstruction quality and a less compact format.

The uncompressed data size can be stated as a function of the number of frames, as described in Equation 5.1, where h is the header size needed for describing the skeleton hierarchy and other constant parameters, f is the number of frames and dof the number of joints. The compressed data size for temporal schemes is described in Equation 5.2, where r_{kf} is the ratio between keyframes and the number of frames. Remember that for each keyframe, we still need to keep all rotations and a timestamp or frame id to locate it in time. Equation 5.3 describes the compressed data size for spatial keyframe schemes, adding two more coordinates to the format.

Considering that for our experiments we have $dof = 31$ and $r_{kf} = 3\%$, we can state that the theoretical lower bound for temporal schemes is near 6%, given that $\lim_{f\to\infty} \frac{c_t(f)}{m(f)} \approx 6.06\%$. Spatial keyframe schemes have a bigger lower bound at around 8%, given that $\lim_{f\to\infty} \frac{c_{skf}(f)}{m(f)} \approx 8.15\%$. The price to be paid for storing the 2 coordinates of each projected frame is around 2% of size of the original data, but is probably less if we consider a header size between 0.5% to 5% of the uncompressed file size. This, however, assumes a keyframe ratio of 3%, i.e., the same ratio used in our error-per-frame experiments. From Equation 5.2 and Equation 5.3, it is easy to see that if both headers h are equal, then the temporal keyframe (r^t_{kf}) ratio has to have values $2/94 \approx 2.13\%$ above the spatial keyframe ratio (r^{sk}_{kf}) so both can have the exact same compression amount, since $r^t_{kf} - r^{sk}_{kf} = 2/(3dof + 1)$. It might be possible to find values of r^t_{kf} and r^{sk}_{kf} where the compression amount is the same for both, but these would result in a quality difference in favor of temporal schemes that the projection optimization would not be able to cover, at least for keyframe ratios from 1% to 4%, where compression schemes would be most useful. This is what the charts in Figure 5.1 suggest, as the error differences shown are mostly above 30 %, considering a difference of 2 % of keyframes between temporal and spatial keyframes schemes.

$$m(f) = h + f(3dof + 3) \tag{5.1}$$

$$c_t(f) = h + f(r_{kf}(3dof + 1) + 3) \tag{5.2}$$

$$c_{skf}(f) = h + f(r_{kf}(3dof + 1) + 5) \tag{5.3}$$

5.2 Projection optimization for visualization

The projection optimization algorithm described in Chapter 4 can also be used for visualization of multidimensional data in general. In order to assess the enhancement provided by the algorithm, we conducted two experiments. The first uses mocap data within the compression framework described in Chapter 3, while the second explores data sets traditionally used in the multidimensional projection literature.

The evaluation of the projection quality can be conducted subjectively by visual inspection, but an objective assessment requires the use of some metric such as the *stress* (Equation 4.1). Table 5.2 shows the average stress ratio for mocap projections made with six different projectors and four keyframe or control point selectors. The average stress ratio is a fraction between the stress of the optimized projection and the stress of the original projection. Gains for the optimization algorithm appear when the ratio is less than one. High-dimension distance is calculated using Equation 3.2, while low-dimension distance is calculated using

Euclidean distance.

In the first experiment, the back-projection algorithm is the same used for all mocap experiments, i.e., IGARASHI *et al.* 's algorithm for character animation, using 3% of the input as control points. We tested 6 single-step projection algorithms with 4 control point selection strategies, as well as the RBFP 2-step projection algorithm where the seeds were projected with the same 6 single-step projection algorithms, while the control points were selected with ROLS. The average results for all 70 data sets used in our compression experiments are shown in Table 5.2. It is possible to observe consistent stress gain for RBFP, except when Isomap is the seed projection heuristic. For all others projectors, there is no real gain in terms of stress. This suggests that the projection optimization algorithm 12 does not, in general, influence the stress measure for these data sets at least when only 3% of the input is used as keyframes except when using RBFP.

Selection Algorithms	**1-step Projection Algorithms**					
	Force	Isomap	LLE	MDS	PCA	t-SNE
PB	1.145	1.07	1.065	1.173	1.245	1.008
ROLS	1.154	1.057	1.033	1.072	0.926	0.969
SCS	1.043	1.085	1.004	1.121	1.046	0.966
US	1.11	1.098	1.049	1.098	1.247	1.215
Selection Algorithms	**2-step RBFP - Seed Projection Algorithms**					
	Force	Isomap	LLE	MDS	PCA	t-SNE
ROLS	0.396	1.013	0.441	0.585	0.518	0.58

Table 5.2: Average stress ratio for 3% keyframe ratio. Each cell shows the ratio between the calculated stress of the optimized projection with respect to that of the original projection. Ratio values below one indicate stress gains only for the RBFP algorithm, except when Isomap is used to project the seeds.

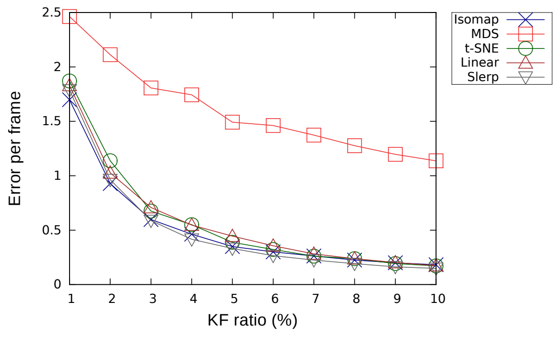
Figures 5.6 through 5.11 display some of the projections of mocap data obtained with RBFP with and without an optimization. Figure 5.6 shows a walk cycle. Figure 5.7 relates to actions such as jump and balance. Figure 5.8 shows the projection of walking over uneven terrain. Figure 5.9 and Figure 5.10 are dance actions. Finally, Figure 5.11 shows a basketball action. In all these instances, the optimized projections' stress is lower than those of the non-optimized projections. The captions of each figure also include the corresponding improvements in the reconstruction error metrics (Eq. 3.2).

The second experiment aims at assessing the utility of the optimizer for other kinds of data sets. For this task, the UCI Machine Learning Repository[36] provides more that 400 datasets which can be used to test the optimization algorithm both with respect to stress gain and visually. Seven datasets were selected for analysis of the RBFP projection algorithm coupled with 2-step ROLS control points selection and six different seed projectors. In all tests, the optimizer uses a 3% sample of the input. Table 5.3 shows the stress ratio for these datasets. Except for the Yeast database using MDS seed projection, there is no other occurrence of a worse projection using the optimization algorithm. Datasets like Ionosphere or Wine show gains for almost all seed projectors while Page blocks, Clouds or Seeds tend to yield modest gains, except for Clouds and Seeds with LLE. The back-projection algorithm used for these experiments is a RBF function described in Equation 2.1 using the modular kernel 2.3 and a polynomial term. For these datasets, we need to discover the two nearest neighbors which must be used in place of the previous and next time "frames" in Algorithm 12, as there is no natural dimension to order the points for these data.

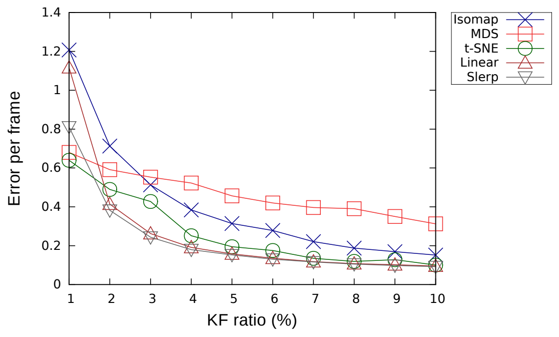
Datasets	**Seed Projection Algorithms**						**Attributes**
	Force	Isomap	LLE	MDS	PCA	t-SNE	
Page blocks	0.977	0.958	0.877	0.96	0.958	0.996	10
Yeast	0.851	0.609	0.119	2.023	0.249	0.299	8
Ionosphere	0.856	0.456	0.405	0.358	0.416	0.715	34
Cloud	1	1	0.073	1	1	0.994	10
Iris	0.96	0.875	0.423	0.551	0.959	0.611	4
Seeds	1	1	0.138	0.845	1	1	7
Wine	0.717	0.71	0.444	0.513	0.849	0.799	13

Table 5.3: Stress ratio for UCL datasets. The ratio is a fraction between stress calculated for 2-step ROLS with and without optimization. Values below one show stress gain for the optimization.

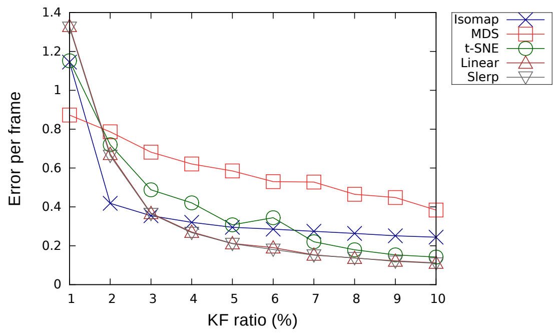
A visual analysis of these datasets can be conducted with the help of Figures 5.12 through 5.17 showing samples of the results, where each figure samples a different seed projection algorithm. Figure 5.12 shows the Ionosphere database using the MDS seed projector. The left projection is always the 2-step ROLS without optimization while the one on the right is the optimized projection. Figure 5.13 is the Iris dataset showing varieties of flowers. Figure 5.14 shows the dataset of wheat varieties. Figure 5.15 shows wine varieties. Figure 5.16 shows classification from blocks of page layout detected by a segmentation process. Figure 5.17 shows classes of protein localization sites in a cell.



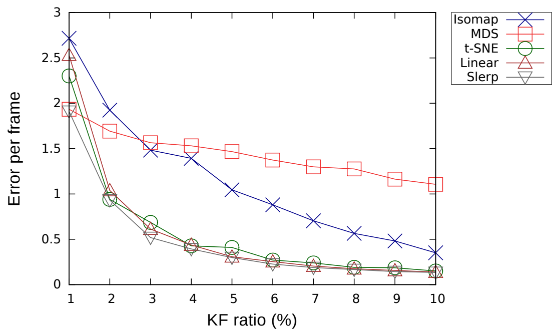
(a) Action: playgound - climb, jump down, dangle, legs push off against



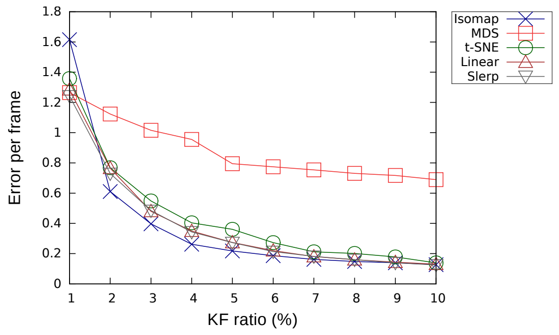
(b) Action: swordplay



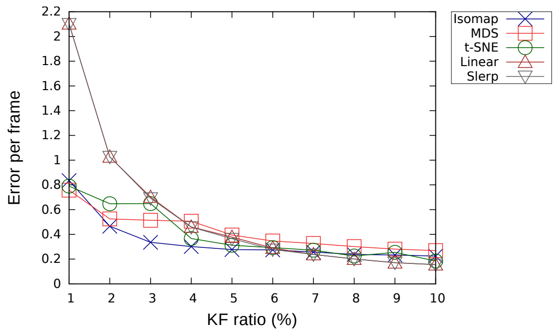
(c) Action: walk on uneven terrain



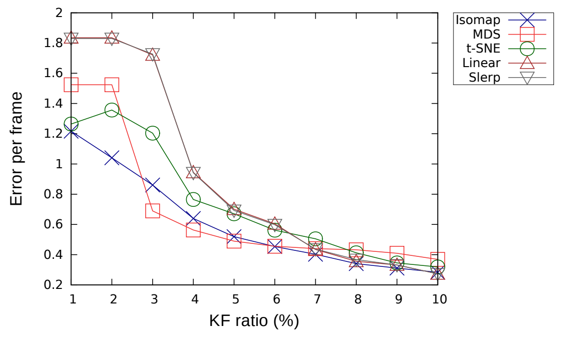
(d) Action: dance - small jetes, pirouette



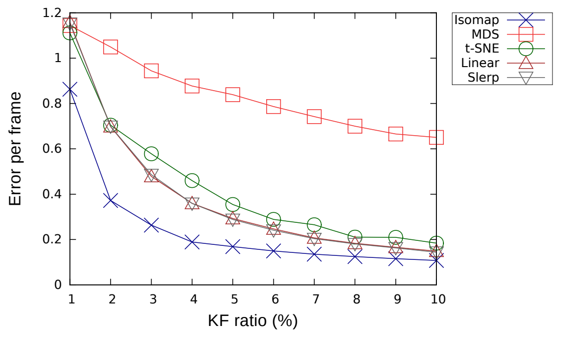
(e) Action: basketball - forward dribble, 90-degree right turns, crossover dribble



(f) Action: walk

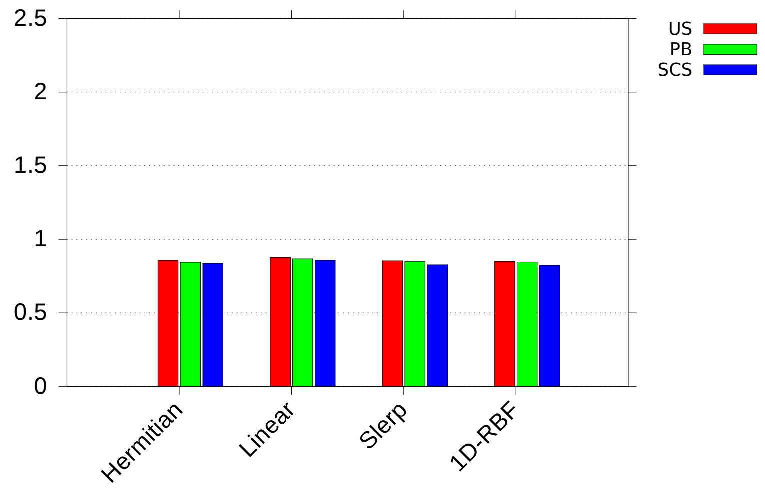


(g) Action: run

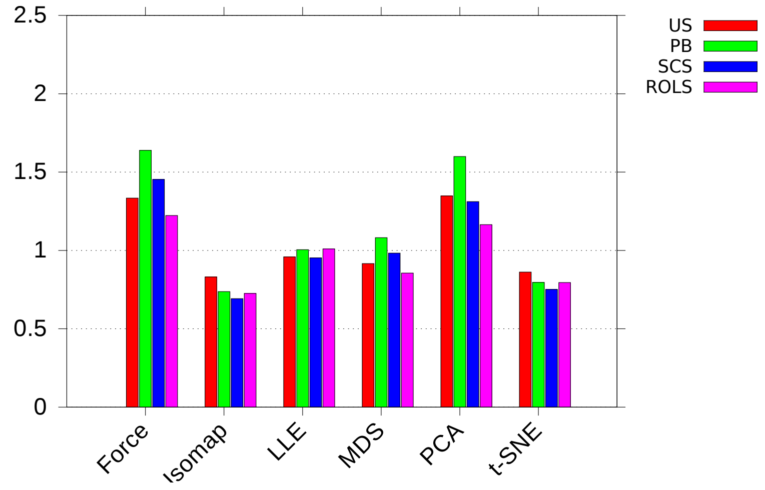


(h) Action: navigate - walk forward, backward, sideways

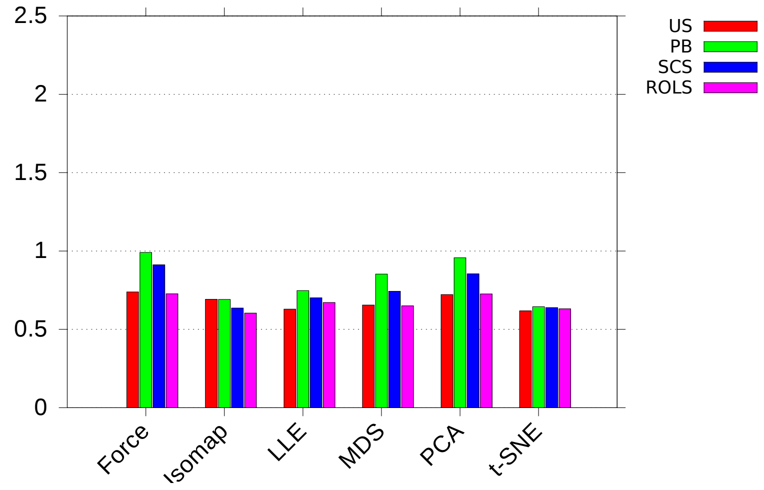
Figure 5.1: Error per frame per joint versus keyframe percentage for five different interpolators with SCS keyframe selector. Each file represents a set of different movements. Error evolution shows fast decreasing until reaching 2 to 4%.



(a) Temporal schemes



(b) SK schemes



(c) SK schemes (2D optimized)

Figure 5.2: Error per frame per joint for temporal and spatial keyframe (single-step) schemes with KF ratio of 3%. Average of 70 files.

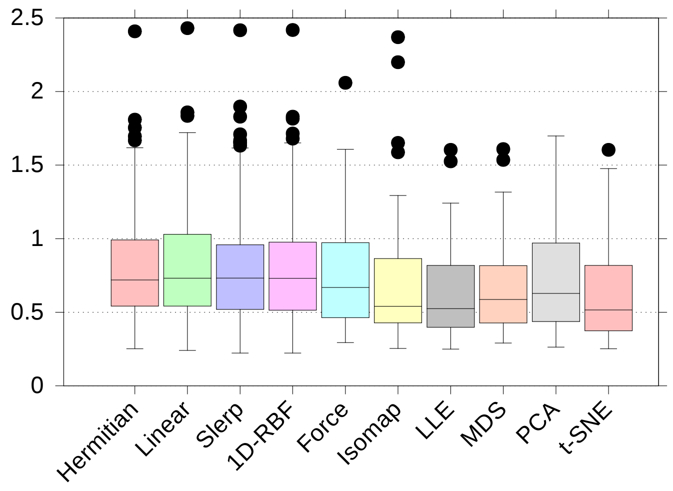
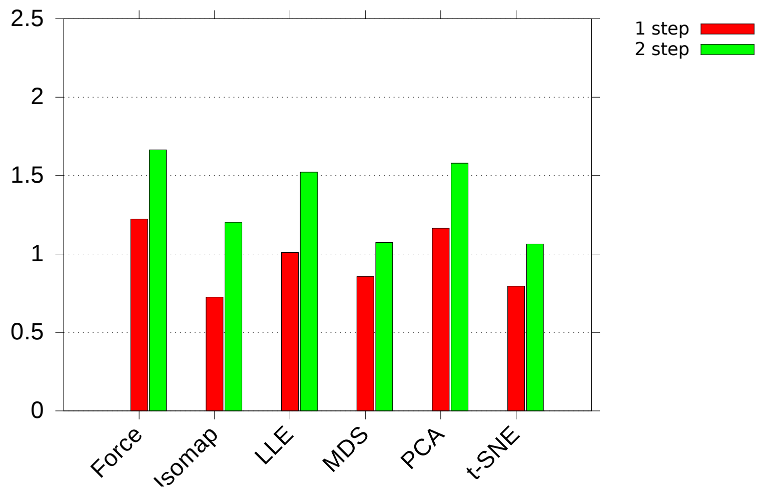
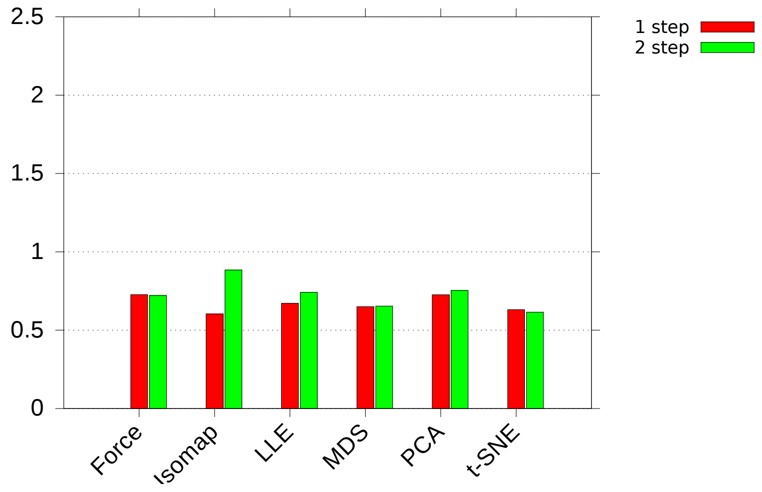


Figure 5.3: Error per frame per joint boxplot for temporal and spatial keyframe schemes with KF ratio of 3%. Values are available at Table 5.1.



(a) SK schemes



(b) SK schemes (2D optimized)

Figure 5.4: Error per frame per joint for ROLS keyframe selection with 1 and 2-step projection approach. KF ratio is 3%. Average of 70 files.

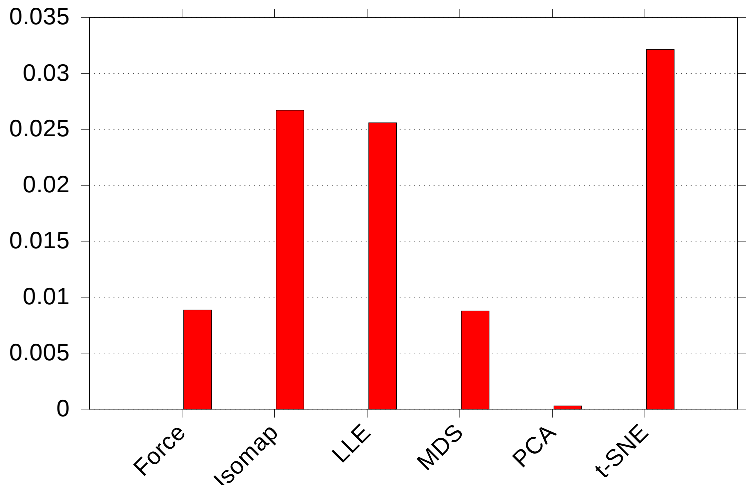


Figure 5.5: Projection running time in seconds per frame for spatial keyframe schemes with KF ratio of 3%.

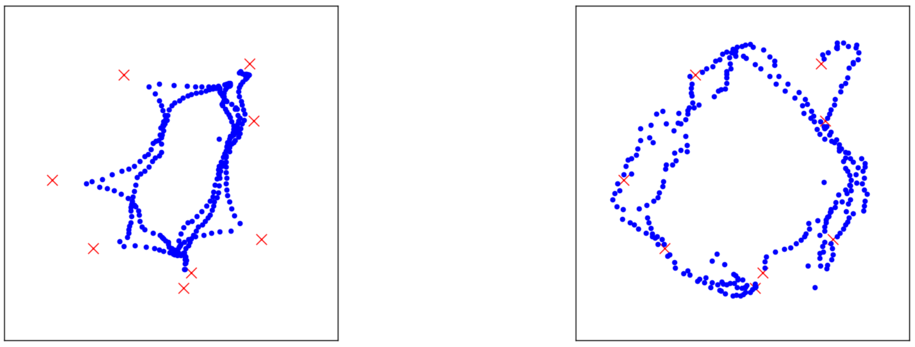


Figure 5.6: Walk cycle. On the left, standard 2-step ROLS projection using LLE. On the right, the same projection using the GD optimization algorithm. Control points are crosses and non-control points are dots. CMU file is 02_02. Stress ratio is 44 %. Error ratio is 35 %.

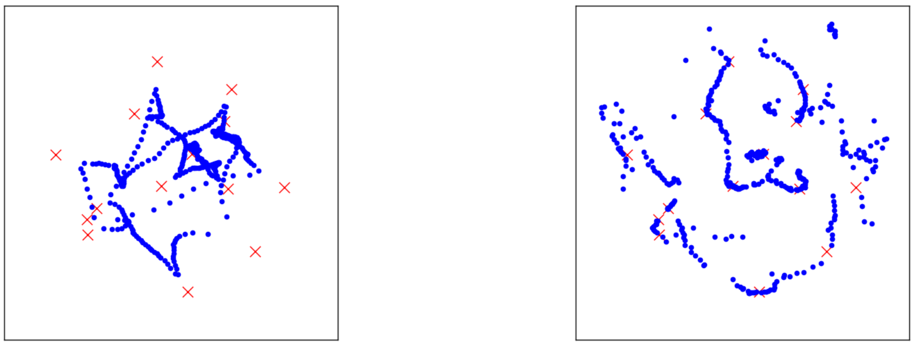


Figure 5.7: Jump and balance. On the left, standard 2-step ROLS projection using Force. On the right, the same projection using the GD optimization algorithm. Control points are crosses and non-control points are dots. CMU file is 02_04. Stress ratio is 40 %. Error ratio is 57 %.

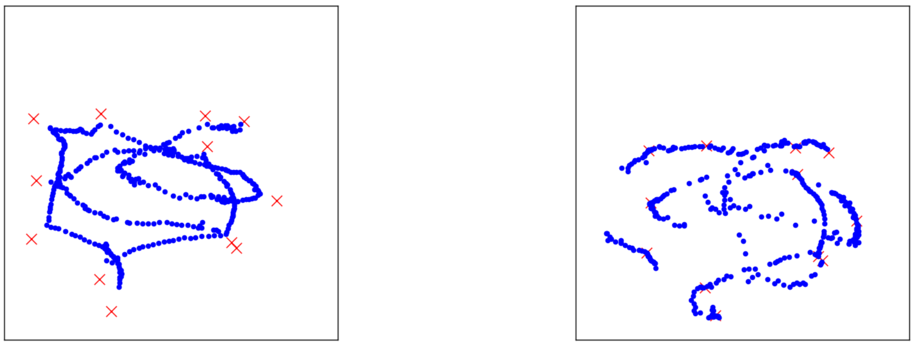


Figure 5.8: Walk on uneven terrain. On the left, standard 2-step ROLS projection using MDS. On the right, the same projection using the GD optimization algorithm. Control points are crosses and non-control points are dots. CMU file is 03_01. Stress ratio is 52 %. Error ratio is 36 %.

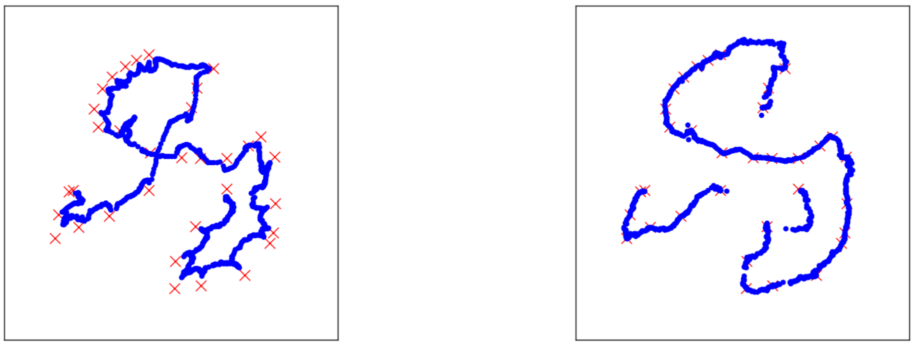


Figure 5.9: Dance: glissade devant, glissade derriere, attitude/arabesque. On the left, standard 2-step ROLS projection using t-SNE. On the right, the same projection using the GD optimization algorithm. Control points are crosses and non-control points are dots. CMU file is 05_09. Stress ratio is 79 %. Error ratio is 51 %.

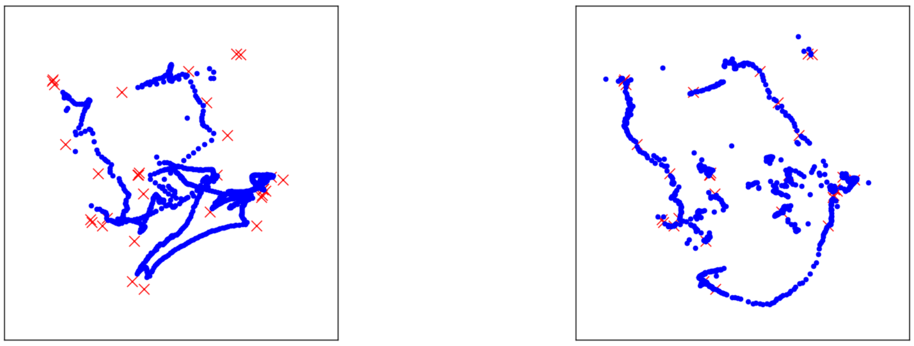


Figure 5.10: Dance: small jetés, pirouette. On the left, standard 2-step ROLS projection using PCA. On the right, the same projection using the GD optimization algorithm. Control points are crosses and non-control points are dots. CMU file is 05_13. Stress ratio is 68 %. Error ratio is 60 %.

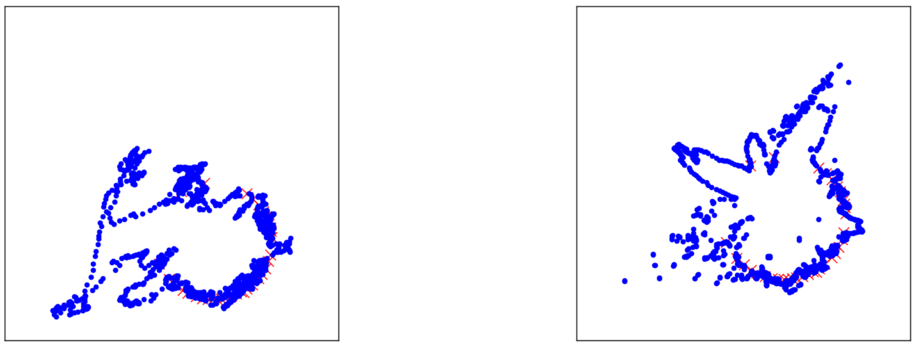


Figure 5.11: Basketball: forward dribble, 90-degree right turns. On the left, standard 2-step ROLS projection using Isomap. On the right, the same projection using the GD optimization algorithm. Control points are crosses and non-control points are dots. CMU file is 06_11. Stress ratio is 88 %. Error ratio is 62 %.

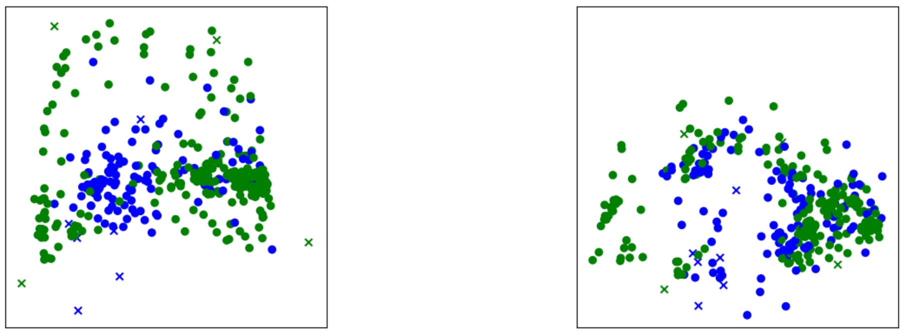


Figure 5.12: Ionosphere database. From left to right: 2-step ROLS and 2 step ROLS optimized both using MDS. Stress ratio is 36 %. Control points are crosses and non-control points are dots. Green means "good" and blue means "bad" where good or bad is a classification corresponding to a perceptron training algorithm.

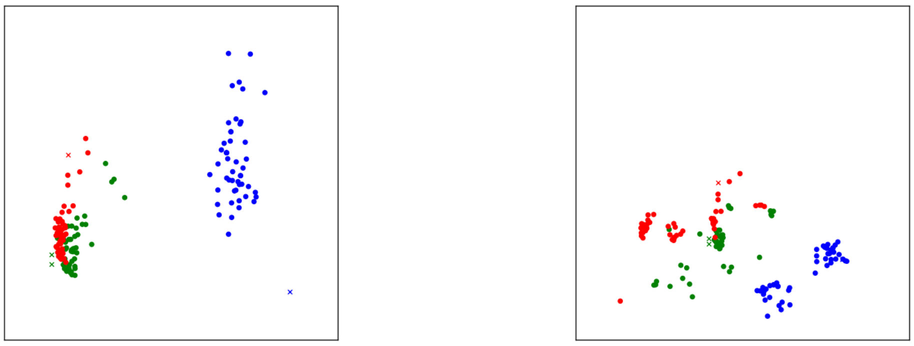


Figure 5.13: Iris database. From left to right: 2-step ROLS and 2-step ROLS optimized both using Isomap. Stress ratio is 55 %. Control points are crosses and non-control points are dots. Blue color means Iris-setosa, green color means Iris-versicolor and red means Iris-virginica.

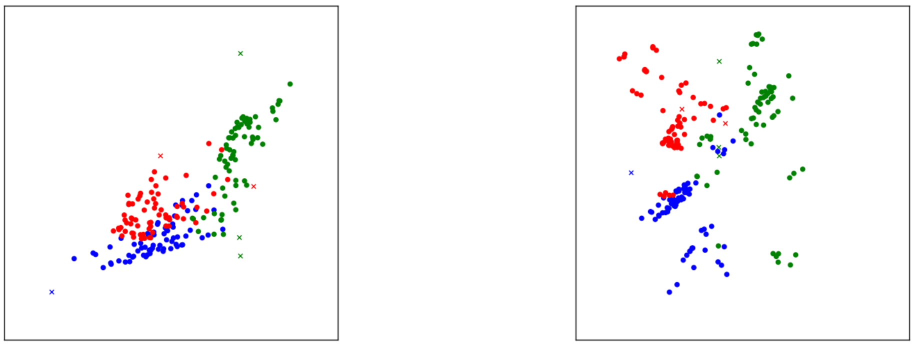


Figure 5.14: Seeds database. From left to right: 2-step ROLS and 2-step ROLS optimized both using LLE. Stress ratio is 14 %. Control points are crosses and non-control points are dots. Classification of this dataset is the kernel of three different varieties of wheat: Kama (blue), Rosa (green) and Canadian (red).

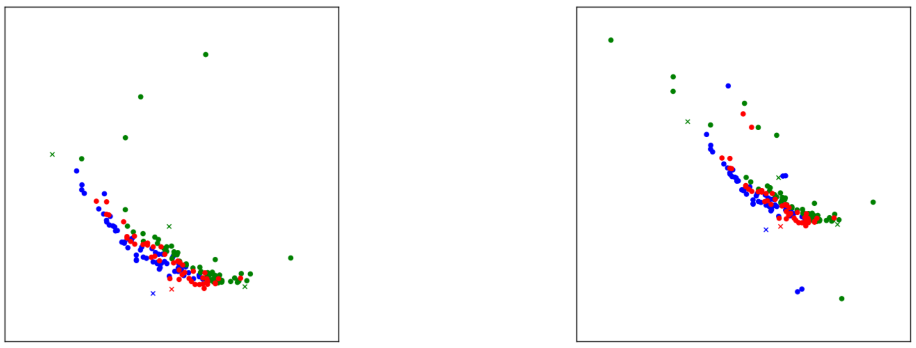


Figure 5.15: Wine database. From left to right: 2-step ROLS and 2-step ROLS optimized both using Force. Stress ratio is 72 %. Control points are crosses and non-control points are dots. Dataset classification divides three different cultivars of wine in Italy: 1 (blue), 2 (green) and 3 (red).

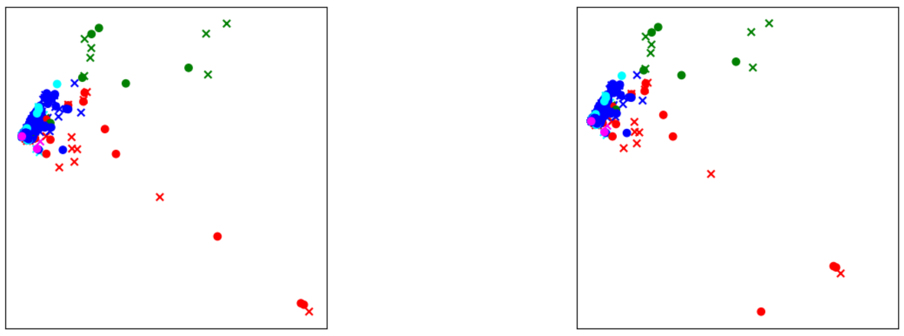


Figure 5.16: Page blocks database. From left to right: 2-step ROLS and 2-step ROLS optimized both using PCA. Stress ratio is 96 %. Control points are crosses and non-control points are dots. Dataset comprises five classes: text (blue), horizontal line (cyan), picture (green), vertical line (magenta) and graphic (red).

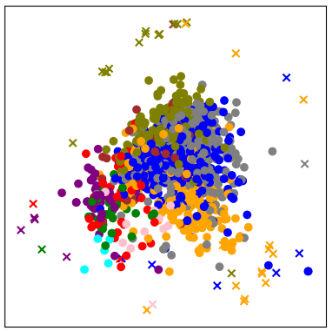
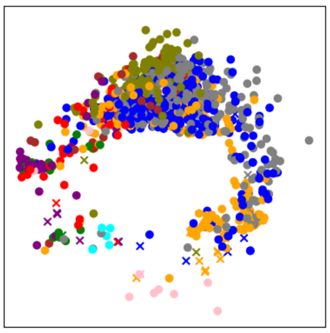
 

Figure 5.17: Yeast database. From left to right: 2-step ROLS and 2-step ROLS optimized both using t-SNE. Stress ratio is 30 %. Control points are crosses and non-control points are dots. Classes of yeast are cytosolic or cytoskeletal (CYT - blue), nuclear (NUC - gray), mitochondrial (MIT - orange), membrane protein without N-terminal signal (ME3 - olive), membrane protein with uncleaved signal (ME2 - red), membrane protein with cleaved signal (ME1 - purple), extracellular (EXC - green), vacuolar (VAC - brown), peroxisomal (POX - pink) and endoplasmic reticulum lumen (ERL - cyan).

Chapter 6

Mocap data reuse in Spatial Keyframing authoring

The original SK paper targets the construction of casual animations using a relatively small number of poses, since these are tedious to construct and their placement on the interaction plane requires a careful planning so that the animator can mentally recall which marker corresponds to which pose. A new animation is created by manually tracing a curve on the plane. In fact, this curve is rather a *trajectory* since the points along the curve have associated timestamps. If the animator makes a mistake, however, the process must be repeated.

In this work we propose techniques to alleviate some of these problems. Firstly, poses are not manually crafted, but rather harvested from a mocap file. Secondly, an algorithm is used for placing the markers on the plane. This can either be a simple algorithm that blindly spreads the markers along a circle, or can consist of a multidimensional projection algorithm that tries to distribute the markers according to the similarity of the poses they represent. Lastly, we propose enriching the authoring pipeline by using sketching techniques such as curve deformation to visually manipulate the trajectories. Some of these are illustrated in Figure 6.1 that shows some snapshots of our SK authoring application.

In our prototype, many of the projection and keyframe extraction algorithms mentioned in Chapter 2 are implemented. Example plots of these are shown in Figure 6.2. When used for authoring, a projection algorithm that clearly distinguishes groups of poses is preferable to those that spread points more uniformly. Similarly, at least one pose in each pose group must be selected by a good keyframe extraction algorithm, even if groups are of different length. In our experiments, t-SNE, LLE and MDS tend to produce nicer results when combined with either SCS or ROLS extraction. Note that the mocap compaction algorithms explored in Section 3 try to obtain good reconstructions with as few poses as possible. On the other hand, when used for authoring, a small set of poses is

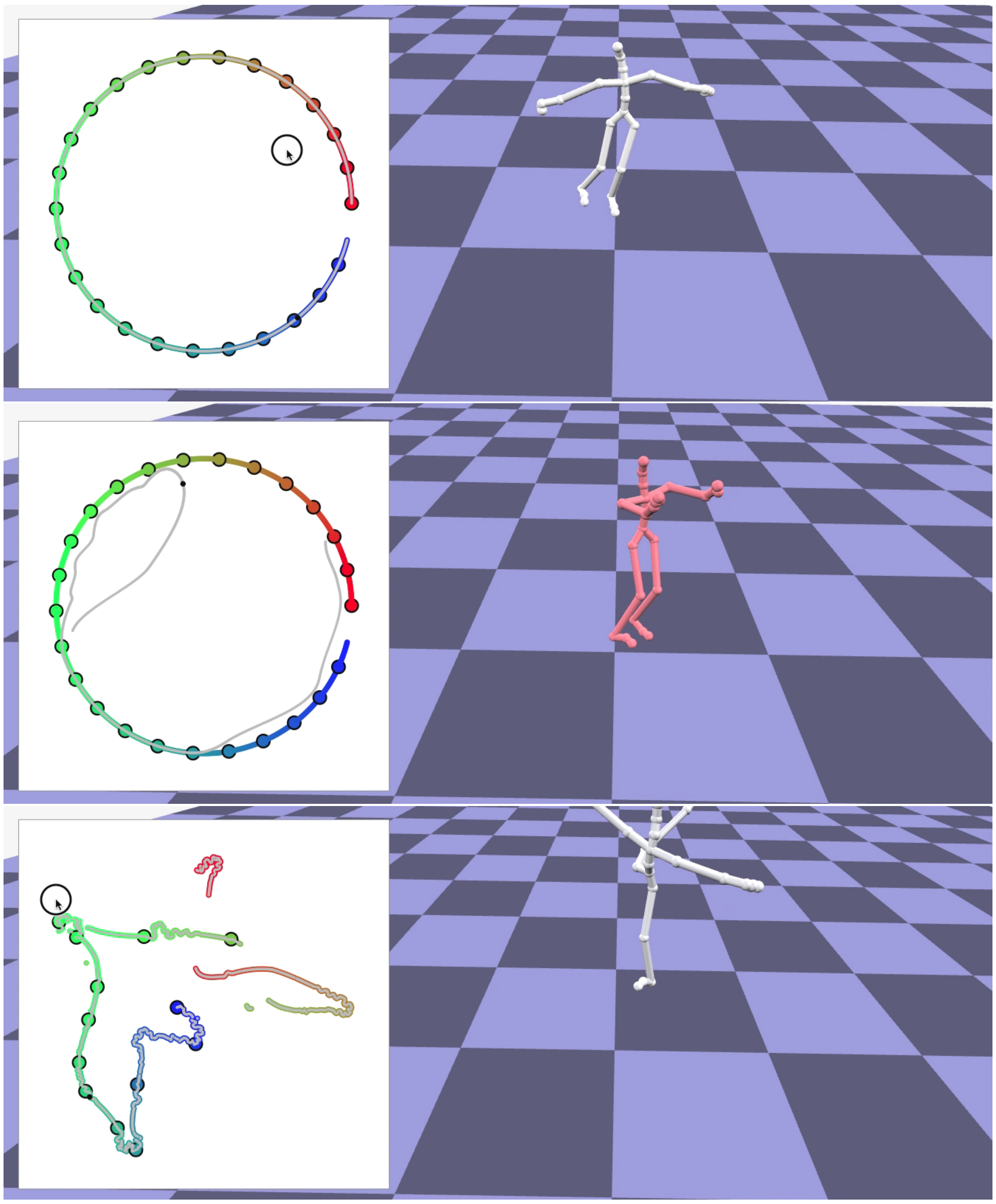


Figure 6.1: SK animation authoring with mocap data. From top to bottom: (1) After a mocap is read and a uniform sample is placed along the curve, poses can be inspected by scrubbing the mouse on the interaction plane. (2) the original trajectory of the mocap is deformed yielding a different animation. (3) A more sophisticated projection and pose extraction algorithm (ROLS/t-SNE in this case) reveals the positions in the interaction plane that correspond to poses of interest, while irrelevant keyframes are removed manually.

preferable to a large set, but keeping the reconstruction errors small is not critical, given that many poses used in the final animation are interpolated. If the animator wants to reproduce verbatim certain segments of the original motion, these should be densely sampled with keyframes, and preferably disposed along a smooth curve on the authoring plane.

Other available enhancements to this authoring pipeline can be used to ensure greater quality in the final animation. For example, to avoid floor sliding or penetration, we can recompute the root joint position for each frame ensuring a fixed position for the foot end effector with lowest Y coordinate at ground level. This can be done by finding the inverse affine transformation that computes the foot position given the root joint, to find the root joint position given the foot placement over the floor. New poses can be created by adjusting some of the available key poses harvested from mocap data. These need a corresponding marker placement on the 2D controller space. If the user does not require a specific location for this marker, our projection optimization algorithm 12 can provide such placement if a seed location is given for it. Another feature made available by this algorithm is the interactive repositioning of keyframes markers on the plane. The user might prefer a different arrangement of the keyframe markers which would yield a completely different projection for all in-between frames. The recomputation of such projections can be done using RBFP in a first step to reach a seed positioning and, with the help of algorithm 12, a final solution can be tailored to present a better mocap reconstruction quality.

Enhancing the synthesized animation by curve editing bring us some concerns on how to deal with timing. In traditional spatial keyframe, there is no other way to improve the animation rather than to redo everything from scratch. However, if we are able to edit the curve, we can improve the animation by relocating its vertices in time and over the 2D plane. We claim that, in a framework enriched by mocap data, it is possible to use the information borrowed from mocap data on the 2D controller interface to bring additional tools to the spatial keyframe framework. A practical application, however, will require that several important issues be addressed. The first key problem is that of temporization. The mocap contains temporization data for only a particular sequence of poses, but not for *any* sequence of poses interpolated from these. Also, the projection algorithms do not use temporal components of the mocap, which might lead to gross temporal distortions. This issue can be alleviated with the help of temporal editing tools similar to those found in traditional keyframe-based animation frameworks. These might be adapted to the sketch-based paradigm used by spatial keyframing. The second key problem is related to the first and has to do with physical plausibility. No care is take by SK in order to ensure that joint rotations agree with physical constraints of a real character. Problems with temporization might lead to hands and feet move with unrealistic speeds and accelerations. In particular, the problem of "sliding feet" was never properly addressed in the

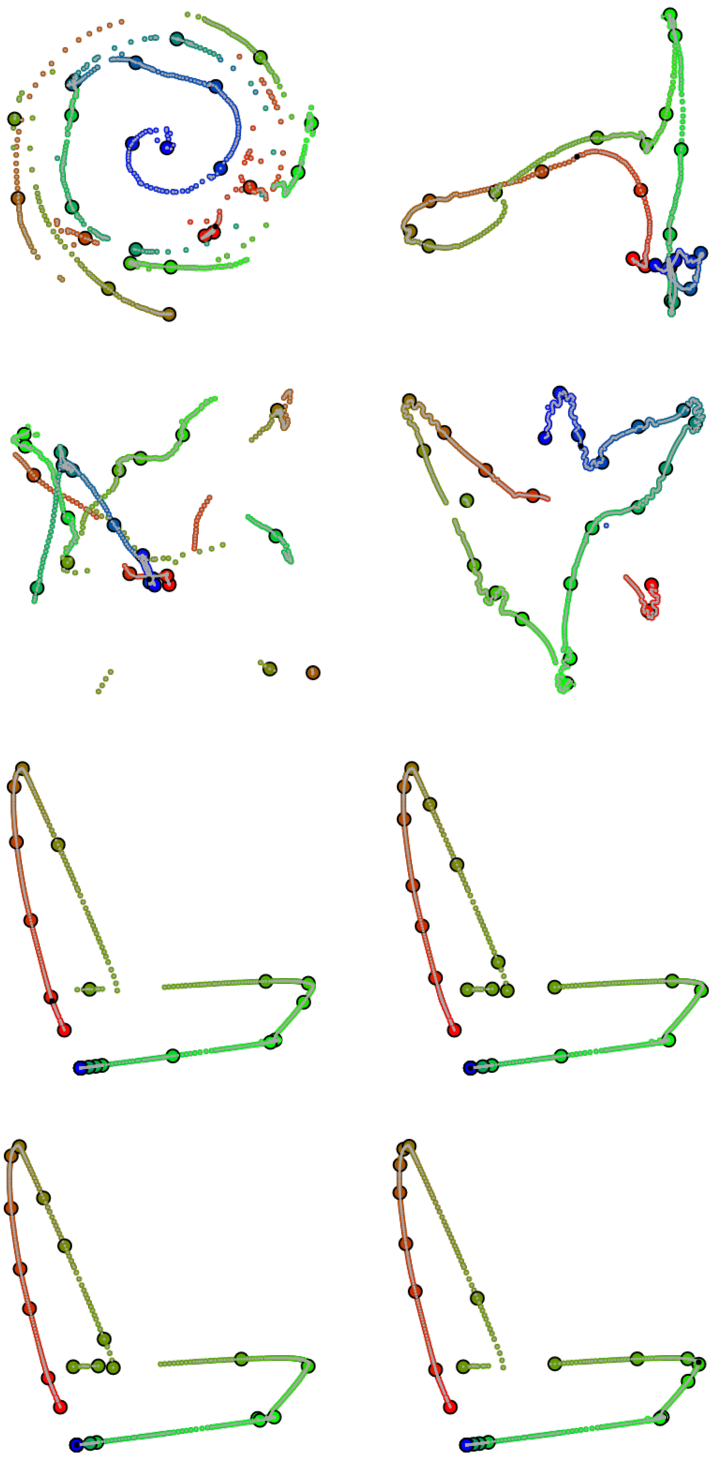


Figure 6.2: Implemented projection and keyframe extraction algorithms. Results obtained by projecting the poses in file `05_10.bvh` from the CMU Motion Capture database. The color gradient from red to green to blue is used to indicate time. Bigger dots with black border are keyframe poses (markers). From left to right, first row has Force and MDS and second row has PCA and t-SNE, where 3% of the poses are selected as keyframes by uniform sampling (US). Third and fourth rows show the results for LLE with keyframes selected with US, SCS, PB and ROLS.

original SK paper, and must be given full attention. These problems can only be solved by bringing in physical simulation tools to the framework.

Chapter 7

Limitations, Conclusions and Future work

In this work, we have evaluated the use of spatial keyframing together with motion capture in three contexts: animation authoring, data visualization and compression. We proposed an algorithm for optimizing projections and evaluated it both in the context of mocap compression and in the context of visualizing other types of multidimensional data. We have built a prototype system where mocap data can be inspected not only by examining poses in time, but also, through multidimensional projection, in a 2D pose similarity space. Our prototype also handles SK-inspired animation authoring where poses can be harvested directly from the mocap and automatically associated with markers on a plane. These features, allied with a sketching interface and the visualization capabilities provided by projection certainly helps the task of creating new animations, but while the system is capable of handling more poses and longer animations than the original SK framework proposed by Igarashi et al., certain limitations of that work are still present, namely those regarding the embedding of the animation in a real scene, such as the authoring of translations. We plan to lift some of these restrictions with future versions of this prototype. Also, a formal evaluation of the tool by professional animators might help us address some of its limitations. With respect to data visualization, we plan to test how would the proposed optimization algorithm respond to different proportions of control points in the data set and test if there is an optimum value for it.

Our investigation of the use of multidimensional projection and SK for mocap compression showed that SK generally attains smaller errors for the same amount of keyframes than temporal compression. Unfortunately, SK compression has not been shown to yield substantial gains with respect to standard temporal compression schemes in practice, mostly because in our experiments trajectories in 2D space had to be stored with no compression. Although our experimentation has been extensive, we still plan on investigating other projection algorithms, as well as using 3D projections which might yield better compression rates.

Bibliography

[1] IGARASHI, T., MOSCOVICH, T., HUGHES, J. F. "Spatial Keyframing for Performance-driven Animation". In: *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '05, pp. 107–115, New York, NY, USA, 2005. ACM. ISBN: 1-59593-198-8. doi: 10.1145/1073368.1073383. Disponível em: <http://doi.acm.org/10.1145/1073368.1073383>.

[2] IZANI, M., AISHAH, ESHAQ, A. R., et al. "Keyframe animation and motion capture for creating animation: a survey and perception from industry people". In: *Proceedings. Student Conference on Research and Development, 2003. SCORED 2003.*, pp. 154–159, Aug 2003. doi: 10.1109/SCORED.2003.1459684.

[3] KOVAR, L., GLEICHER, M., PIGHIN, F. "Motion Graphs", *ACM Trans. Graph.*, v. 21, n. 3, pp. 473–482, jul. 2002. ISSN: 0730-0301. doi: 10.1145/566654.566605. Disponível em: <http://doi.acm.org/10.1145/566654.566605>.

[4] LEE, J., CHAI, J., REITSMA, P. S. A., et al. "Interactive Control of Avatars Animated with Human Motion Data", *ACM Trans. Graph.*, v. 21, n. 3, pp. 491–500, jul. 2002. ISSN: 0730-0301. doi: 10.1145/566654.566607. Disponível em: <http://doi.acm.org/10.1145/566654.566607>.

[5] PULLEN, K., BREGLER, C. "Motion Capture Assisted Animation: Texturing and Synthesis", *ACM Trans. Graph.*, v. 21, n. 3, pp. 501–508, jul. 2002. ISSN: 0730-0301. doi: 10.1145/566654.566608. Disponível em: <http://doi.acm.org/10.1145/566654.566608>.

[6] SAFONOVA, A., HODGINS, J. K., POLLARD, N. S. "Synthesizing Physically Realistic Human Motion in Low-dimensional, Behavior-specific Spaces", *ACM Trans. Graph.*, v. 23, n. 3, pp. 514–521, ago. 2004. ISSN: 0730-0301. doi: 10.1145/1015706.1015754. Disponível em: <http://doi.acm.org/10.1145/1015706.1015754>.

[7] DINH, H. Q., TURK, G., SLABAUGH, G. “Reconstructing surfaces by volumetric regularization using radial basis functions”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 24, n. 10, pp. 1358–1371, Oct 2002. ISSN: 0162-8828. doi: 10.1109/TPAMI.2002.1039207.

[8] LIM, I. S., THALMANN, D. “Key-posture extraction out of human motion data by curve simplification”. Annual Reports of the Research Reactor Institute, Kyoto University. Kyoto University, 2001. Swiss Federal Inst. Technol. (EPFL), CH-1015 Laussane, Switzerland.

[9] LOWE, D. G. “Three-dimensional Object Recognition from Single Two-dimensional Images”, *Artif. Intell.*, v. 31, n. 3, pp. 355–395, mar. 1987. ISSN: 0004-3702. doi: 10.1016/0004-3702(87)90070-1. Disponível em: <http://dx.doi.org/10.1016/0004-3702(87)90070-1>.

[10] XIAO, J., ZHUANG, Y., YANG, T., et al. “An Efficient Keyframe Extraction from Motion Capture Data.” In: Nishita, T., Peng, Q., Seidel, H.-P. (Eds.), *Computer Graphics International*, v. 4035, *Lecture Notes in Computer Science*, pp. 494–501. Springer, 2006. ISBN: 3-540-35638-X. Disponível em: <http://dblp.uni-trier.de/db/conf/cgi/cgi2006.html#XiaoZYW06>.

[11] TOGAWA, H., OKUDA, M. “Position-Based Keyframe Selection for Human Motion Animation”. In: *Parallel and Distributed Systems, 2005. Proceedings. 11th International Conference on*, v. 2, pp. 182–185, July 2005. doi: 10.1109/ICPADS.2005.239.

[12] BULUT, E., CAPIN, T. “Key frame extraction from motion capture data by curve saliency”, *Computer Animation and Social Agents*, p. 119, 2007.

[13] HALIT, C., CAPIN, T. “Multiscale motion saliency for keyframe extraction from motion capture sequences”, *Computer Animation and Virtual Worlds*, v. 22, n. 1, pp. 3–14, 2011. ISSN: 1546-427X. doi: 10.1002/cav.380. Disponível em: <http://dx.doi.org/10.1002/cav.380>.

[14] HUANG, K.-S., CHANG, C.-F., HSU, Y.-Y., et al. “Key Probe: a technique for animation keyframe extraction”, *The Visual Computer*, v. 21, n. 8, pp. 532–541, 2005. ISSN: 1432-2315. doi: 10.1007/s00371-005-0316-0. Disponível em: <http://dx.doi.org/10.1007/s00371-005-0316-0>.

[15] JIN, C., FEVENS, T., MUDUR, S. “Optimized keyframe extraction for 3D character animations”, *Computer Animation and Virtual Worlds*, v. 23,

n. 6, pp. 559–568, 2012. ISSN: 1546-427X. doi: 10.1002/cav.1471. Disponível em: <http://dx.doi.org/10.1002/cav.1471>.

[16] ZHANG, Q., ZHANG, S., ZHOU, D. "Keyframe Extraction from Human Motion Capture Data Based on a Multiple Population Genetic Algorithm", *Symmetry*, v. 6, n. 4, pp. 926, 2014. ISSN: 2073-8994. doi: 10.3390/sym6040926. Disponível em: <http://www.mdpi.com/2073-8994/6/4/926>.

[17] ARIKAN, O. "Compression of Motion Capture Databases", *ACM Trans. Graph.*, v. 25, n. 3, pp. 890–897, jul. 2006. ISSN: 0730-0301. doi: 10.1145/1141911.1141971. Disponível em: <http://doi.acm.org/10.1145/1141911.1141971>.

[18] M.E., T. "Principal Component Analysis (2nd Ed.). I. T. Jolliffe", *Journal of the American Statistical Association*, v. 98, pp. 1082–1083, January 2003. Disponível em: <https://ideas.repec.org/a/bes/jnlasa/v98y2003p1082-1083.html>.

[19] ZHANG, Y., CAO, J. "A Novel Dimension Reduction Method for Motion Capture Data and Application in Motion Segmentation". In: *Parallel and Distributed Systems, 2005. Proceedings. 11th International Conference on*, v. 12, pp. 6751–6760, 2015. doi: 10.12733/jics20150024.

[20] ROWEIS, S. T., SAUL, L. K. "Nonlinear Dimensionality Reduction by Locally Linear Embedding", *Science*, v. 290, n. 5500, pp. 2323–2326, 2000. ISSN: 0036-8075. doi: 10.1126/science.290.5500.2323. Disponível em: <http://science.sciencemag.org/content/290/5500/2323>.

[21] ASSA, J., CASPI, Y., COHEN-OR, D. "Action Synopsis: Pose Selection and Illustration", *ACM Trans. Graph.*, v. 24, n. 3, pp. 667–676, jul. 2005. ISSN: 0730-0301. doi: 10.1145/1073204.1073246. Disponível em: <http://doi.acm.org/10.1145/1073204.1073246>.

[22] COX, M. A. A., COX, T. F. "Multidimensional Scaling". In: *Handbook of Data Visualization*, pp. 315–347, Berlin, Heidelberg, Springer Berlin Heidelberg, 2008. ISBN: 978-3-540-33037-0. doi: 10.1007/978-3-540-33037-0_14. Disponível em: <http://dx.doi.org/10.1007/978-3-540-33037-0_14>.

[23] KRUSKAL, J. B. "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis", *Psychometrika*, v. 29, n. 1, pp. 1–27, 1964.

ISSN: 1860-0980. doi: 10.1007/BF02289565. Disponível em: <http://dx.doi.org/10.1007/BF02289565>.

[24] JENKINS, O. C., MATARIĆ, M. J. "A Spatio-temporal Extension to Isomap Nonlinear Dimension Reduction". In: *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pp. 56–, New York, NY, USA, 2004. ACM. ISBN: 1-58113-838-5. doi: 10.1145/1015330.1015357. Disponível em: <http://doi.acm.org/10.1145/1015330.1015357>.

[25] TENENBAUM, J. B., SILVA, V. D., LANGFORD, J. C. "A Global Geometric Framework for Nonlinear Dimensionality Reduction", *Science*, v. 290, n. 5500, pp. 2319–2323, 2000. ISSN: 0036-8075. doi: 10.1126/science.290.5500.2319. Disponível em: <http://science.sciencemag.org/content/290/5500/2319>.

[26] TEJADA, E., MINGHIM, R., NONATO, L. G. "On Improved Projection Techniques to Support Visual Exploration of Multidimensional Data Sets", *Information Visualization*, v. 2, n. 4, pp. 218–231, dez. 2003. ISSN: 1473-8716. doi: 10.1057/palgrave.ivs.9500054. Disponível em: <http://dx.doi.org/10.1057/palgrave.ivs.9500054>.

[27] VAN DER MAATEN, L., HINTON, G. "Visualizing Data using t-SNE", *Journal of Machine Learning Research*, v. 9, pp. 2579–2605, 2008. Disponível em: <http://www.jmlr.org/papers/v9/vandermaaten08a.html>.

[28] MAATEN, L. "Learning a Parametric Embedding by Preserving Local Structure". In: van Dyk, D., Welling, M. (Eds.), *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, v. 5, *Proceedings of Machine Learning Research*, pp. 384–391, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR. Disponível em: <http://proceedings.mlr.press/v5/maaten09a.html>.

[29] JOIA, P., PAULOVICH, F., COIMBRA, D., et al. "Local Affine Multidimensional Projection", *Visualization and Computer Graphics, IEEE Transactions on*, v. 17, n. 12, pp. 2563–2571, Dec 2011. ISSN: 1077-2626. doi: 10.1109/TVCG.2011.220.

[30] AMORIM, E., VITAL BRAZIL, E., NONATO, L. G., et al. "Multidimensional Projection with Radial Basis Function and Control Points Selection".

In: *Pacific Visualization Symposium (PacificVis), 2014 IEEE*, Pacific Vis'14, pp. 209–216, 2014. doi: 10.1109/PacificVis.2014.59.

[31] DOS SANTOS AMORIM, E. P., BRAZIL, E. V., II, J. D., et al. "iLAMP: Exploring high-dimensional spacing through backward multidimensional projection." In: *IEEE VAST*, pp. 53–62. IEEE Computer Society, 2012. ISBN: 978-1-4673-4752-5. Disponível em: <`http://dblp.uni-trier.de/db/conf/ieeevast/ieeevast2012.html#AmorimBDJNS12`>.

[32] AMORIM, E., BRAZIL, E. V., MENA-CHALCO, J., et al. "Facing the high-dimensions: Inverse projection with radial basis functions", *Computers & Graphics*, v. 48, pp. 35 – 47, 2015. ISSN: 0097-8493. doi: http://dx.doi.org/10.1016/j.cag.2015.02.009. Disponível em: <`http://www.sciencedirect.com/science/article/pii/S0097849315000230`>.

[33] SMITH, L. I. *A tutorial on principal components analysis*. Relatório técnico, Cornell University, USA, February 26 2002. Disponível em: <`http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf`>.

[34] BURGES, C., SHAKED, T., RENSHAW, E., et al. "Learning to Rank Using Gradient Descent". In: *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pp. 89–96, New York, NY, USA, 2005. ACM. ISBN: 1-59593-180-5. doi: 10.1145/1102351.1102363. Disponível em: <`http://doi.acm.org/10.1145/1102351.1102363`>.

[35] "The Daz-friendly BVH release of CMU's motion capture database – cgspeed". `https://sites.google.com/a/cgspeed.com/cgspeed/motion-capture/daz-friendly-release`. Accessed: 2017-05-19. Also available at `http://mocap.cs.cmu.edu/`.

[36] "UCI Machine Learning Repository". `https://archive.ics.uci.edu/ml/datasets.html`. Accessed: 2018-05-30.

Appendix A

Algorithm pseudocodes

Algorithm 1 Simple Curve Simplification (SCS) key frame selection algorithm

```
function GETSCSKEYFRAMES($\mathbb{F}$, $k$)    ▷ Where $\mathbb{F}$ - frame set, $k$ - number of key frames
    $n \leftarrow |\mathbb{F}|$
    Let $f$ be the frame of $\mathbb{F}$ with lowest timestamp
    Let $l$ be the frame of $\mathbb{F}$ with highest timestamp
    $\mathbb{K} \leftarrow \{f\} \cup \{l\}$
    Let $E_{\mathbb{F}}$ be an array of size $n$
    for $i \leftarrow 3,\ k$ do
        Let $\mathbb{I}$ be the reconstructed frame set of all timestamps using $\mathbb{K}$ as keyframes
        for $j \leftarrow 1,\ n$ do
            Let $p$ be the frame of the j'th timestamp in $\mathbb{F}$
            Let $q$ be the frame of the j'th timestamp in $\mathbb{I}$
            Save in $E_{\mathbb{F}}(j)$ the pose distance between $p$ and $q$
        end for
        Let $m$ be the frame in $\mathbb{F}$ corresponding to the maximal value in $E_{\mathbb{F}}$
        $\mathbb{K} \leftarrow \mathbb{K} \cup \{m\}$
    end for
    return $\mathbb{K}$
end function
```

Algorithm 2 Position Based (PB) key frame selection algorithm

```
1: function GETPBKEYFRAMES($\mathbb{F}$, $k$)    ▷ Where $\mathbb{F}$ - frame set, $k$ - number of key frames
2:     $n = |\mathbb{F}|$
3:     Let $E_{\mathbb{F}}$ be an array of size $n$
4:     Let $f$ be the frame of $\mathbb{F}$ with lowest timestamp
5:     Let $l$ be the frame of $\mathbb{F}$ with highest timestamp
6:     $\mathbb{C} \leftarrow \mathbb{F} \setminus \{f, l\}$
7:     $\mathbb{D} \leftarrow \emptyset$
8:     for all $j \in \mathbb{C}$ do
9:         $\mathbb{D} \leftarrow \mathbb{D} \cup \{j\}$
10:        Let $\mathbb{I}$ be the reconstructed frame set of all timestamps using $\mathbb{F} \setminus \mathbb{D}$ as keyframes
11:        Let $p$ be the frame corresponding to timestamp of j in $\mathbb{I}$
12:        Save in $E_{\mathbb{F}}(j)$ the pose distance between $p$ and $j$
13:        $\mathbb{D} \leftarrow \mathbb{D} \setminus \{j\}$
14:    end for
15:    for $i \leftarrow n,\ k$ do
16:        Let $m$ be the frame in $\mathbb{F} \setminus \mathbb{D}$ corresponding to the minimal value in $E_{\mathbb{F}}$, except $f$ and $l$
17:        $\mathbb{D} \leftarrow \mathbb{D} \cup \{m\}$
18:        Let $\mathbb{C}$ have all interpolation neighbor frames of $m$, except $f$ and $l$
19:        for all $j \in \mathbb{C}$ do
20:            $\mathbb{D} \leftarrow \mathbb{D} \cup \{j\}$
21:            Let $\mathbb{I}$ be the reconstructed frame set of all timestamps using $\mathbb{F} \setminus \mathbb{D}$ as keyframes
22:            Let $p$ be the frame corresponding to timestamp of j in $\mathbb{I}$
23:            Save in $E_{\mathbb{F}}(j)$ the pose distance between $p$ and $j$
24:            $\mathbb{D} \leftarrow \mathbb{D} \setminus \{j\}$
25:        end for
26:    end for
27:    return $\mathbb{F} \setminus \mathbb{D}$
28: end function
```

Algorithm 3 PCA projection approach.

```
1: function GETPCAPROJECTION($D$, $p$)    ▷ Where $D$ - data in original dimension, $p$ - projection dimension
2:     Let $n$ be the number of rows in row-wise data matrix $D_{n \times d}$
3:     Let $a = \frac{\sum D_i}{n}$ be the average of all rows in $D$
4:     Let $T$ be the centralized data matrix such that $T_i = D_i - a$ for each row
5:     Compute matrix $C = T^\top T$
6:     Let $\lambda$ be the eigenvalues vector of $C$ in descending order
7:     Let $E$ be the column-wise eigenvectors matrix of $C$ such that eigenvector $E_j$ has eigenvalue $\lambda_j$
8:     Let $M_{d \times p}$ be the matrix equal to the first $p$ columns of $E_{d \times d}$
9:     $P = DM$
10:    return $P$
11: end function
```

Algorithm 4 LLE projection approach. I is the identity matrix.

```
1: function GETLLEPROJECTION($D$, $k$, $p$)    ▷ Where $D$ - data in original dimension, $k$ - number of k-nearest neighbors, $p$ - projection dimension
2:     Let $n$ be the number of rows in row-wise data matrix $D$
3:     $W \leftarrow getWeightMatrix(D, k)$
4:     $M \leftarrow (I - W)^\top (I - W)$
5:     Let $\lambda$ be the eigenvalues vector of $M$ in ascending order
6:     Let $E$ be the column-wise eigenvectors matrix of $M$ such that eigenvector $E_j$ has eigenvalue $\lambda_j$
7:     for $i \leftarrow 1,\ n$ do
8:         for $j \leftarrow 2,\ p+1$ do
9:             $P_{i[p-j+2]} \leftarrow E_{ij}\lambda_j$
10:        end for
11:    end for
12:    return $P$
13: end function

14: function GETWEIGHTMATRIX($D$, $k$)    ▷ Where $D$ - data in original dimension, $k$ - number of k-nearest neighbors
15:    Let $n$ be the number of rows in row-wise data matrix $D$
16:    for $i \leftarrow 1,\ n$ do
17:        Let $Z$ have all $D_j$ which is one of the k-nearest neighbors of $D_i$
18:        Subtract $D_i$ from each row of $Z$
19:        Compute matrix $C = ZZ^\top$
20:        Solve linear system $Cw = \vec{1}$
21:        Set $W_{ij} = \frac{w_j}{\sum_k w}$ for all k-nearest neighbors of $D_i$
22:        Set $W_{ij} = 0$ for all other elements
23:    end for
24:    return $W$
25: end function
```

Algorithm 5 Floyd-Warshall algorithm for building a geodesic distance matrix.

```
1: function GETGEODESICDISTANCEMATRIX($M$, $nn$)    ▷ Where $M$ - distance matrix, $nn$ - number of nearest neighbors
2:     Let $n$ be the order of square matrix $M_{n \times n}$
3:     for $i \leftarrow 1, n$ do
4:         $G_{ii} \leftarrow 0$
5:     end for
6:     for $i \leftarrow 1, n$ do
7:         for $j \leftarrow 1, n$ do
8:             if $i$ and $j$ are one of the $nn$ nearest neighbors then
9:                 $G_{ij} \leftarrow M_{ij}$
10:                $G_{ji} \leftarrow M_{ji}$
11:            else
12:                $G_{ij} \leftarrow \infty$
13:            end if
14:        end for
15:    end for
16:    for $k \leftarrow 1, n$ do
17:        for $i \leftarrow 1, n$ do
18:            for $j \leftarrow 1, n$ do
19:                if $G_{ij} > G_{ik} + G_{kj}$ then
20:                    $G_{ij} \leftarrow G_{ik} + G_{kj}$
21:                end if
22:            end for
23:        end for
24:    end for
25:    return $G$
26: end function
```

Algorithm 6 MDS-like projection approach. Isomap follows the same approach.

```
1: function GETPROJECTION($M, p$)    ▷ Where $M$ - distance matrix, $p$ - projection dimension
2:     Let $n$ be the order of square matrix $M_{n \times n}$
3:     Compute matrix $S$ such that $S_{ij} = M_{ij}^2$
4:     Let $\Delta$ be the Kronecker delta function (Identity of order n)
5:     Compute matrix $H$ such that $H_{ij} = \Delta_{ij} - \frac{1}{n}$
6:     $T \leftarrow \frac{-HSH}{2}$
7:     Let $\lambda$ be the eigenvalues vector of $T$ in descending order
8:     Let $E$ be the column-wise eigenvectors matrix of $T$ such that eigenvector $E_j$ has eigenvalue $\lambda_j$
9:     for $i \leftarrow 1, n$ do
10:        for $j \leftarrow 1, p$ do
11:            $P_{ij} \leftarrow E_{ij}\lambda_j$
12:        end for
13:    end for
14:    return $P$
15: end function
```

Algorithm 7 Force projection approach. Usually, $p = 2$, $r = 50$ and $\alpha = 8$. tol is a small number like 10^{-6}.

```
1: function GETFORCEPROJECTION($D$, $p$, $r$, $\alpha$)    ▷ Where $D$ - data in original dimension, $p$ - projection dimension, $r$ - number of iterations, $\alpha$ - step size
2:     Let $n$ be the number of elements in $D$.
3:     Compute distance matrix $M_{n\times n}$ such that $M_{ij} = M_{ji} = \|D_i - D_j\|$
4:     Initialize matrix $P_{n\times p}$ with random numbers such that $\mu = 0$ and $\sigma = 1$
5:     for $k \leftarrow 1, r$ do
6:         for $i \leftarrow 1, n$ do
7:             for $j \leftarrow 1, n$ do
8:                 $v \leftarrow P_i - P_j$
9:                 $dp \leftarrow \|v\|^2$
10:                if $dp < tol$ then
11:                    $dp \leftarrow tol$
12:                end if
13:                $\delta \leftarrow \frac{M_{ij} - dp}{\alpha}$
14:                $v \leftarrow \frac{v}{dp}$
15:                $P_i \leftarrow P_i + v\delta$
16:            end for
17:        end for
18:    end for
19:    return $P$
20: end function
```

Algorithm 8 t-SNE projection approach. tol is a small number like 10^{-5}. The t-SNE parameter pp (perplexity) usually ranges between 15 and 50. Parameters such as r, η and α are related to gradient descent solution search. Usually, $r = 1000$, $\eta = 500$ and $\alpha = 0.5$.

```
1: function GETTSNEPROJECTION($D$, $p$, $pp$, $r$, $\eta$, $\alpha$)    ▷ Where $D$ - data in original dimension, $p$ - projection dimension, $pp$ - perplexity value, $r$ - number of iterations, $\eta$ - learning parameter, $\alpha$ - momentum parameter
2:     $H \leftarrow getAffinities(D, pp)$
3:     Initialize matrix $P_{n \times p}$ with random numbers such that $\mu = 0$ and $\sigma = 1$
4:     while $r > 0$ do
5:         Let $L$ be the matrix where $L_{ij} = \begin{cases} 0 & \text{if } i = j \\ \frac{(1+\|P_i-P_j\|^2)^{-1}}{\sum_{k \neq l}(1+\|P_k-P_l\|^2)^{-1}} & \text{if } i \neq j \end{cases}$
6:         for $i \leftarrow 1,\ n$ do
7:             $G_i \leftarrow 4\sum_j (H_{ij} - L_{ij})(P_i - P_j)(1 + \|P_i - P_j\|^2)^{-1}$
8:         end for
9:         $P_t \leftarrow P_{t-1} + \eta G + \alpha(P_{t-1} - P_{t-2})$
10:        $r \leftarrow r - 1$
11:    end while
12:    return $P$
13: end function

14: function GETAFFINITIES($D$, $pp$)    ▷ Where $D$ - data in original dimension, $pp$ - perplexity value
15:    Let $n$ be the number of rows in row-wise data matrix $D$
16:    for $i \leftarrow 1,\ n$ do
17:        $\sigma \leftarrow \frac{1}{\sqrt{2}}$
18:        $diff \leftarrow tol$
19:        while $\|diff\| \geqslant tol$ do
20:            Compute $L_{j|i} \leftarrow \frac{\exp(-\|D_i-D_j\|^2/2\sigma^2)}{\sum_{k \neq l}\exp(-\|D_k-D_l\|^2/2\sigma^2)}$
21:            Compute $h = -\sum_j L_{j|i} \log_2 L_{j|i}$
22:            Compute $diff = h - \log_2 pp$
23:            Update $\sigma$ using the bisection method analyzing $diff$ sign: increases if positive or decreases if negative
24:        end while
25:    end for
26:    $L \leftarrow \frac{L+L^\top}{2}$    ▷ Force L to be a symmetric matrix
27:    return $L$
28: end function
```

Algorithm 9 Regularized Othogonal Least-Squares (ROLS) control point / key frame selection algorithm. Regularization parameter r is a small number like 10^{-5}.

```
1: function GETROLSSELECTION($G, Q, k, r$)   ▷ Where $G$ - n-dimensional data, $Q$ - 2-dimensional data, $k$ - number of control points, $r$ - regularization parameter
2:     Let $\phi_{mq}(x) = \sqrt{1 + x^2}$
3:     Build matrix $A$ as shown Equation 2.2 for system $AW = Q$ with kernel $\phi_{mq}$ and points in $G$ as $c_j$
4:     for $i \leftarrow 1,\ k$ do
5:         $m \leftarrow getNewControlPoint(A, G, Q, r)$
6:         Add $m$ to $K$
7:         Eliminate from $A$ the influence of $m$ using Gram-Schmidt orthogonalization
8:     end for
9:     return $K$
10: end function

11: function GETNEWCONTROLPOINT($A, G, Q, r$)   ▷ Where $A$ - RBF matrix, $G$ - n-dimensional data, $Q$ - 2-dimensional data, $r$ - regularization parameter
12:     $emax \leftarrow 0$
13:     for all $g \in G$ do
14:         Let $a$ be the column of $g$ in $A$
15:         Let $e_i = \frac{(a^\top q_i + r)^2}{(q_i^\top q_i)(a^\top a)}$ be the error contribution of $g$ and $q_i$ is the i'th dimension of $Q$
16:         $err = \sum_i e_i$
17:         if $err > emax$ then
18:             $m \leftarrow g$
19:             $emax \leftarrow err$
20:         end if
21:     end for
22:     return $m$
23: end function
```

Algorithm 10 RBF-based projection scheme (RBFP).

```
1: function RBFPPROJECT($D, K, L$)   ▷ Where $D$ - data in high dimension data set, $K$ - control points in high dimension, and $L$ - projected control points
2:     $W \leftarrow buildRBF(K, L)$
3:     $k \leftarrow \|\mathbb{D}\|$
4:     Let $d_i$ be the i'th vector in $\mathbb{D}$
5:     Let $\phi_{mq}(x) = \sqrt{1 + x^2}$
6:     for all $d_i \in \mathbb{D} \setminus \mathbb{H}$ do
7:         Compute $f(d_i)$ as in Equation 2.1 where $w$ and $P$ are given by $W$
8:         Let $l_i = f(d_i)$
9:     end for
10:     return $P$ as all computed $l_i$ and previously known $L$
11: end function
```

Algorithm 11 Compute RBF weights.

```
1: function BUILDRBF($K, L$)    ▷ Where $K$ and $L$ are arrays of points.
2:     Let $\phi$ be the chosen kernel
3:     Build matrix $A$ as in Equation 2.2 using $\phi$ as kernel where constraints $c_j$
       are points in $K$
4:     Build matrix $B$ where elements $h_i$ are points in $L$
5:     return weight matrix $W = A^{-1}B$
6: end function
```

Algorithm 12 Gradient descent optimization for pose projection

```
1: function GETOPTIMIZEDPROJECTION($\mathbb{F}$, $\mathbb{Q}$, $\mathbb{K}$)    ▷ Optimizes projection $\mathbb{Q}$.
   Where $\mathbb{F}$ - frame array, $\mathbb{Q}$ - projected pose array, $\mathbb{K}$ - keyframe array.
2:     $\mathbb{O} \leftarrow \emptyset$
3:     for $i \leftarrow 1,\ maxtries$ do
4:         $done \leftarrow$ true
5:         for all $f_i \in \mathbb{F} \setminus \{\mathbb{K} \cup \mathbb{O}\}$ do
6:             if $gradientDescent(f_i, \mathbb{F}, \mathbb{Q}, \mathbb{K})$ then
7:                 $\mathbb{O} \leftarrow \mathbb{O} \cup \{f_i\}$
8:                 $done \leftarrow$ false
9:             end if
10:        end for
11:        if $done$ then
12:            break
13:        end if
14:    end for
15: end function

16: function GRADIENTDESCENT($f_i$, $\mathbb{F}$, $\mathbb{Q}$, $\mathbb{K}$)    ▷
   Updates $q_i \in \mathbb{Q}$ and returns if $q_i$ has changed. Where $f_i$ - frame, $\mathbb{F}$ - frame set,
   $\mathbb{Q}$ - projected pose set, $\mathbb{K}$ - keyframe set.
17:    $h \leftarrow (0, 0)$
18:    $v_x \leftarrow (1, 0)$
19:    $v_y \leftarrow (0, 1)$
20:    $q_i \leftarrow$ projection of $f_i$ stored in $\mathbb{Q}$
21:    $q_{i+1} \leftarrow$ projection of $f_{i+1}$ stored in $\mathbb{Q}$
22:    $q_{i-1} \leftarrow$ projection of $f_{i-1}$ stored in $\mathbb{Q}$
23:    $r \leftarrow (\|q_i - q_{i-1}\| + \|q_i - q_{i+1}\|)/2$
24:    $e_o \leftarrow Err(f_i, BackProj(q_i, \mathbb{F}, \mathbb{Q}, \mathbb{K}))$
25:    $e_{xp} \leftarrow Err(f_i, BackProj(q_i + \delta r v_x, \mathbb{F}, \mathbb{Q}, \mathbb{K}))$
26:    $e_{xn} \leftarrow Err(f_i, BackProj(q_i - \delta r v_x, \mathbb{F}, \mathbb{Q}, \mathbb{K}))$
27:    $e_{yp} \leftarrow Err(f_i, BackProj(q_i + \delta r v_y, \mathbb{F}, \mathbb{Q}, \mathbb{K}))$
28:    $e_{yn} \leftarrow Err(f_i, BackProj(q_i - \delta r v_y, \mathbb{F}, \mathbb{Q}, \mathbb{K}))$
29:    if $(e_o > e_{xp}) \vee (e_o > e_{xn})$ then
30:        $h \leftarrow h + v_x(e_{xn} - e_{xp})$
31:    end if
32:    if $(e_o > e_{yp}) \vee (e_o > e_{yn})$ then
33:        $h \leftarrow h + v_y(e_{yn} - e_{yp})$
34:    end if
35:    if $\|h\| > 0$ then
36:        $h \leftarrow rh/\|h\|$
37:        $q_i' \leftarrow q_i + \alpha h$
38:        replace $q_i$ by $q_i'$ in $\mathbb{Q}$
39:        return true
40:    else
41:        return false
42:    end if
43: end function
```