Data-Driven Constraint-Based Motion Editing

THÈSE N^{O} 4558 (2009)

PRÉSENTÉE LE 10 DÉCEMBRE 2009
À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS
LABORATOIRE DE RÉALITÉ VIRTUELLE
PROGRAMME DOCTORAL EN INFORMATIQUE, COMMUNICATIONS ET INFORMATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Schubert Ribeiro de CARVALHO

acceptée sur proposition du jury:

Prof. R. Hersch, président du jury
Prof. D. Thalmann, Dr R. Boulic, directeurs de thèse
Prof. G. Abou Jaoudé, rapporteur
Prof. T. Vetter, rapporteur
Dr X. Wang, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE
Suisse
2009

To my parents Arneide and Schubert,

and to my wife Milene

"All knowledge is subject to change and revision. Every solution of an existing problem raises new and unsolved questions. These new and as yet unsolved problems require, at times, new methods of inquiry and different conceptual structures; they change the shape and patterns of knowing."

Barbara A. Carper

Résumé

L’augmentation du nombre de systèmes de capture de mouvement contribue directement à la prolifération de bases de données de mouvements humains, principalement car le mouvement humain est important dans de nombreuses applications, allant du divertissement comme les jeux ou les films au sport et à la médecine. Cependant, ces mouvements capturés et variés requièrent usuellement des besoins spécifiques. Ainsi, la modification et la réutilisation de ces mouvements dans de nouvelles situations (par exemple pour les adapter à un nouvel environnement) sont devenues des domaines de recherche en pleine expansion et connues sous le terme d’*édition de mouvement*. Ces dernières années, l’édition de mouvement humain est donc devenue une des activités de recherche les plus actives dans le domaine de l’animation par ordinateur.

Dans cette thèse, nous présentons et examinons une nouvelle méthode d’édition de mouvement humain interactive. Notre principale contribution est le développement d’une technique de cinématique inverse priorisée de basse dimension (LPIK : Low-dimensional Prioritized Inverse Kinematics) qui gère des contraintes de l’utilisateur dans un espace de mouvement à basse dimension (référencé comme espace latent). Sa principale propriété est d’opérer dans l’espace latent au lieu de l’espace articulaire. Par construction, il est suffisant de contraindre un seul instant par LPIK afin d’obtenir un mouvement naturel qui renforce son flot intrinsèque. Le LPIK a l’avantage de réduire la taille de la matrice jacobienne étant donné que la dimension de l’espace du mouvement latent est petite en comparaison à celle de l’espace articulaire d’un mouvement. La méthode offre l’avantage supplémentaire d’être bien adaptée à la gestion de personnage à fort degré de liberté (DoF : degree of freedom), ce qui est une des limitations des méthodes de cinématique inverse basées sur des optimisations dans l’espace articulaire. Grâce à cet avantage, notre méthode fournit toujours des déformations plus rapidement et des résultats de mouvement plus naturels en comparaison de méthodes de la littérature basées sur des contraintes et orientées par des buts. Fondamentalement, notre technique est basée sur les connections mathématiques entre modèles linéaires de mouvement telles que l’analyse en composantes principales (PCA : Principal Component Analysis) et la cinématique inverse priorisée (PIK : Prioritized Inverse Kinematics). Nous utilisons les PCA comme première étape d’un prétraitement pour réduire la dimensionnalité de la base de données afin de la rendre traitable et d’encapsuler un motif sous-jacent. Ensuite, nous relions les solutions de l’IK dans l’espace des mouvements naturels. Nous utilisions le PIK afin de permettre à l’utilisateur de manipuler des contraintes avec différentes priorités tout en éditant interactivement une animation. La stratégie des priorités assure essentiellement qu’une tâche de haute priorité n’est pas affectée par d’autres tâches à priorité plus faible.

De plus, nous proposons deux stratégies basées sur les PCA imposant la continuité du mouvement. Nous montrons des expérimentations qui évaluent et valident (à la fois quantitativement et qualitativement) les bénéfices de notre méthode. Et finalement nous évaluons la qualité des animations éditées par rapport celle issue d’une technique basée sur des contraintes et orientée par des buts. Ceci pour vérifier la robustesse de notre méthode en termes de performance, simplicité et réalisme.

Mots clés: Edition de Mouvement, Analyse en Composante Principale, Cinématique inverse priorisée.

Abstract

The growth of motion capture systems has contributed to the proliferation of human motion database, mainly because human motion is important in many applications, ranging from games entertainment and films to sports and medicine. However, the various captured motions normally require specific needs. Consequently, modifying and reusing these motions in new situations - for example, retargeting it to a new environment - became an increasing area of research known as *motion editing*. In the last few years, human motion editing has become one of the most active research areas in the field of computer animation.

In this thesis, we introduce and discuss a novel method for interactive human motion editing. Our main contribution is the development of a Low-dimensional Prioritized Inverse Kinematics (LPIK) technique that handles user constraints within a low-dimensional *motion space* - also known as the *latent space*. Its major feature is to operate in the latent space instead of the joint space. By construction, it is sufficient to constrain a single frame with LPIK to obtain a natural movement enforcing the intrinsic motion flow. The LPIK has the advantage of reducing the size of the Jacobian matrix as the motion latent space dimension is small for a coordinated movement compared to the joint space. Moreover, the method offers the compelling advantage that it is well suited for characters with large number of degrees of freedom (DoFs). This is one of the limitations of IK methods that perform optimizations in the joint space. In addition, our method still provides faster deformations and more natural-looking motion results compared to goal-directed constraint-based methods found in the literature. Essentially, our technique is based on the mathematical connections between linear motion models such as Principal Component Analysis (PCA) and Prioritized Inverse Kinematics (PIK). We use PCA as a first stage of preprocessing to reduce the dimensionality of the database to make it tractable and to encapsulate an underlying motion pattern. And after, to bound IK solutions within the space of natural-looking motions. We use PIK to allow the user to manipulate constraints with different priorities while interactively editing an animation. Essentially, the priority strategy ensures that a higher priority task is not affected by other tasks of lower priority.

Furthermore, two strategies to impose motion continuity based on PCA are introduced. We show a number of experiments used to evaluate and validate (both qualitatively and quantitatively) the benefits of our method. Finally, we assess the quality of the edited animations against a goal-directed constraint-based technique, to verify the robustness of our method regarding performance, simplicity and realism.

Keywords: Motion Editing, Principal Component Analysis, Prioritized Inverse Kinematics.

Resumo

O aumento dos sistemas de captura de movimento tem contribuído com a proliferação de bases de dados de movimentos humanos, principalmente por que o movimento humano é importante em várias aplicações, indo de jogos e filmes a esportes e medicina. Entretanto, os movimentos capturados normalmente atendem a objetivos específicos. Consequentemente, a modificação e a reutilização destas animações em novas situações - por exemplo, o redirecionamento para um novo cenário - tornou-se uma área de pesquisa crescente conhecida como *edição de movimento*. Nos últimos anos, a edição de movimentos humanos ganhou um foco crescente de pesquisa na área de animação por computador.

Nesta tese, nós introduzimos e discutimos um novo método para edição interativa de movimento de corpos humanos. Nossa principal contribuição é o desenvolvimento de uma técnica de cinemática inversa com prioridades, a qual trata as restrições do utilizador dentro de um espaço de movimentos de baixa dimensão (LPIK : Low-dimensional Prioritized Inverse Kinematics) - também conhecido como *espaço latente*. Sua principal característica é de operar dentro do espaço latente ao invés do espaço de articulações. Por construção, é suficiente restringir uma única pose com LPIK para obter um movimento natural respeitando a continuidade do movimentos. O LPIK tem a vantagem de reduzir a dimensionalidade da matrix Jacobiana por que a dimensão do espaço latente é menor para um movimento coordenado comparado com o espaço de articulações. Além do mais, o LPIK oferece a grande vantagem de ser bem adaptado para gerenciar caracteres que possuam esqueletos com vários graus de liberdade (DoF: degrees of freedom). Esta é uma das limitações dos métodos de cinemática inversa que realizam otimizações no espaço de articulações. Ademais, nosso método ainda realiza deformações mais rápidas e fornece resultados de animações mais realísticos, comparado aos métodos encontrados na literatura baseados em restrições e direcionado por objetivo. Essencialmente, nossa técnica é baseada nas conexões matemáticas entre um modelo linear de movimento como a Análise de Componentes Principais (PCA: Principal Component Analysis) e cinemática inversa com prioridades (PIK: Prioritized Inverse Kinematics). Nós utilizamos o PCA em uma primeira etapa de pré-processamento para reduzir a dimensionalidade da base de dados para torná-la tratável e também para encapsular um padrão de movimento fundamental. E depois, para restringir as soluções de cinemática inversa dentro de um espaço de movimentos naturais. Nós utilizamos PIK para permitir que o usuário manipule restrições com diferentes prioridades, enquanto está editando interativamente o movimento. Basicamente, a estratégia de prioridades assegura que restrições de mais alta prioridade não sejam afetadas por outras restrições com menores prioridades.

Além disso, nós também propomos duas estratégias baseadas no PCA para impor continuidade no movimento. Nós mostramos experimentos que avaliam e validam quantitativamente e qualitativamente os benefícios de nosso método. E finalmente, nós avaliamos a qualidade das animações editadas em relação a uma técnica baseada em restrições e direcionada por objetivo, com a finalidade de verificar a robustez do método proposto em relação ao desempenho, simplicidade e realismo.

Palavras chaves: Edição de Movimento, Análise de Componentes Principais, Cinemática Inversa com Prioridades.

Acknowledgement

It is very hard to list in one small place all the people who directly or indirectly supported me during my PhD. I may be missing some here, but I have to try.

First of all, I want to thank you my thesis director, Professor Daniel Thalmann, who gave me the opportunity to work in his laboratory. Firstly as a research assistant and then as a PhD candidate. The members of my jury Professor Georges Abou Jaoude, Professor Thomas Vetter and Dr. Xuguang Wang for the comments they raised to the examination of this thesis. I want also to thank you my thesis co-director, Dr. Ronan Boulic, who supervised my work during these four years. He was always available and patient to discuss the numerous problems related with my work. I am also very grateful for the times that we spent at *Satellite* drinking and talking about IK :) !

I would also like to thank you several old and new members here at the virtual reality laboratory (VRlab) who helped me with suggestions, directions and support: Anderson Maciel, Ehsan Arbabi, Benoît Le Callennec, Damien Maupu, Renaud Ott, Alejandra Martinez, Pablo Ciechomski, Pascal Glardon, Jan Ciger, Helena Grillon, Stéphane Gobron, Nicolas Pronost and Junghyun Ahn (AJ). Thanks to Josiane to be always kind and caring, and for her good sense of humor. Thanks to Mireille for her work of designer always improving the results of my papers :). She was always attentive, patient and available.

The PhD was a life changing experience, for me the first six months at EPFL were the hardest ones. But, I could meet Anderson Maciel who was finishing his PhD here in the VRlab. He was of great support for me and helped me to handle the most difficult days of my PhD. Thank you very much you are a great! During almost four years, my labmate Ehsan Arbabi offered me not only a wonderful working environment, but he became my friend. Ehsan is the kind of person that never say no for a friend, of course we are not talking about money :) (he knows what does it means!). He was always with good humor, and this makes a lot of difference when you share the same office during so long time.

Thanks to Marcello Kallmann for answered my emails on time and for have put me in contact with Anderson. Thanks to Dora for her attention when I arrived in Geneva.

I am eternally grateful to my aunt Christina for all the support during my education. My aunts and uncles for the good moments in Brazil and my grandmothers Inácia and Elizabeth. Thanks to Socorro Bastos and Ildemar Silva for their attention and care with me and my wife.

In the past four years, all my friends helped me keep my sanity during moments of despair. I hope to not forget nobody: Paulo Dal Fabbro, Leandro (Lele), Tuca, Rafael, Carlos Alexandre, Reni, Alexandre Dal Fabbro, Maria, Paula, Juliano (Catatau), Sami, Marcelo Leite, Montse, Sabrina, Pietro, Simona, Tonho, Janaína. Thanks to Norberto and Francieli for welcoming me and my wife into their home and hearts.

In the past two years, I could ride my bike and even race. I made new friends and could also learn more about cycling. I would like to thank you Xavier Dubal and Jorge for the long rides in the roads of Switzerland. And in special Dubal for the precious advices about training and nutrition that we cannot find in books. Thanks to Alberto, Antunes and the *The Bike* team Boris and Fred for the friendship.

Creto Vidal, from Federal University of Ceara, came in the spring of 2007 for a post-doc at VRlab. We could work together and he helped me to solve some problems of my thesis. Thanks to Vania Vidal for invited me and my wife to eat her delicious *feijoada*!

My parents Arneide and Schubert Carvalho, they have been always present, even with an ocean between us. Thank you for always believe in me and for guided me since childhood in the direction of science and knowledge.

Last, but not least, my wife Milene Carvalho for all the support before and during my PhD. She was of extremely importance during the PhD because she did not let me give up. She remembered me why I was here and she made me believe that I was capable to do it.

Contents

1 Introduction **1**

1.1 Motivation and Overview 1
1.2 Problem Statement and Contributions 3
1.3 Method Evaluation and Validation 5
1.4 Our Approach for Motion Editing 5
1.5 Outline 7

2 State of the Art **9**

2.1 Introduction 9
2.2 Inverse Kinematics 9
  2.2.1 Analytical Methods 9
  2.2.2 Optimization-based Methods 10
  2.2.3 Model-based Methods 12
2.3 Motion Editing and Synthesis 13
  2.3.1 Spacetime Constraints 13
  2.3.2 Frequency-based Methods 14
  2.3.3 Constraint-based Methods 15
  2.3.4 Data-driven Methods 17
2.4 Conclusion 20

3 Motion Pattern Encapsulation **21**

3.1 Introduction 21
3.2 Motion Modeling 21
  3.2.1 The Human Figure 22
  3.2.2 Motion Parameterization 23
  3.2.3 PCA Motion Model 24
3.3 Model Results 27
  3.3.1 Motion Clustering 30
  3.3.2 Motion Encapsulation: The Purpose of Model Generalization 32
3.4 Conclusion 35

4 Data-Driven Constraint-Based Optimization Framework 37
4.1 Introduction 37
4.2 Constraints 37
4.2.1 Key-frame constraints 37
4.2.2 Key-trajectory constraints 39
4.3 Low-Dimensional Inverse Kinematics (LIK) 41
4.4 Low-Dimensional Prioritized Inverse Kinematics (LPIK) 47
4.5 Synergies 49
4.6 Establishing an Appropriate Database Percentage 50
4.7 Conclusion 53

5 Enforcing Temporal Continuity 55
5.1 Introduction 55
5.2 Imposing Continuity from Key-trajectory Constraints 55
5.3 Imposing Continuity from Key-frame Constraints 57
5.3.1 Specific Key 57
5.3.2 Multiple Keys 58
5.4 Duration 60
5.5 Conclusion 62

6 Experiments and Validation 65
6.1 Introduction 65
6.2 Overview of the System 65
6.2.1 Implementation 65
6.2.2 System Interface and Functionalities 66
6.2.3 Editing the Motion 68
6.3 Evaluation 70
6.3.1 Evaluating Duration 70
6.3.2 Evaluating System Performance: Skeletons with Different DoFs 71
6.3.3 Evaluating Motion Quality 73
6.4 Synthesis Comparisons 75
6.4.1 Local vs. Single Models 76
6.4.2 Joint Space vs. Latent Space 77
6.5 Case Study 86
6.6 Conclusion 90

7 Conclusions and Future Work 93

A Background Mathematics 97
A.1 Principal Component Analysis 97
A.1.1 Optimal Reconstruction 97

B HAnim 99
B.1 Level of articulation 99

List of Figures

1.1 System architecture 6

2.1 7-DoFs arm 10
2.2 SGPLVM latent spaces 12
2.3 Spacetime constraints 13
2.4 Frequency-based method example 15
2.5 Constraint-based system 16
2.6 The PCA for geometric animations 18

3.1 Character figure 22
3.2 Walking jump motion models 28
3.3 Golf swings normalization process 29
3.4 Synthesized golf swings 29
3.5 Golf swings motion models 30
3.6 Reaching motion model 31
3.7 Reaching model-based synthesis 33
3.8 Multi-Activity latent space 34
3.9 Synthesized motions from the multi-activity database 36

4.1 Key-frame and key-trajectory constraints 38
4.2 Global synergy 44
4.3 LIK convergence 46
4.4 Walking jumps latent spaces 50
4.5 Motion generation with complete walking jumps database 51
4.6 The influence of the database percentage in convergence performance. 52
4.7 Walking jumps latent space trajectories 53

4.8 Motion generation one subject walking jumps database 54

5.1 Smooth trajectories in Cartesian and latent spaces 56
5.2 Motion editing from key-trajectory constraints 57
5.3 Motion editing from specific key-time constraints 58
5.4 Motion editing from multiple key-time constraints 59
5.5 The behavior of the curve under changing the spline parameters 60
5.6 Recovered duration from key-frame constraints 61

6.1 Data-driven constraint-based motion editing system interface 66
6.2 Optimizer parameters used to assist the editing process 67
6.3 Key-frame mode: spline parameters 67
6.4 Key-frame editing 68
6.5 Key-trajectory editing 69
6.6 Skeletons 72
6.7 Evaluating system performance: lower values mean faster results 73
6.8 Evaluating motion quality: one and two constraints 74
6.9 Evaluating motion quality: three constraints 75
6.10 Motion retargeting 76
6.11 A typical constraint configuration used in our system to edit a golf swing 77
6.12 LPIK convergence 78
6.13 Synthesis comparison joint vs. latent space: golf swing flat ground 79
6.14 Synthesis comparison joint vs. latent space: golf swing down slope ground . . 81
6.15 Synthesis comparison joint vs. latent space: golf swing up slope ground 82
6.16 Synthesis comparison latent space: greater jump 83
6.17 Synthesis comparison joint space: greater jump 83
6.18 Synthesis comparison joint vs. latent space: higher jump 84
6.19 Synthesis comparison joint vs. latent space: reach 85
6.20 Shelf environment 86
6.21 Final reaching postures 87
6.22 Constraints handling 88
6.23 Multiple key-frame editing 89

Chapter 1

Introduction

1.1 Motivation and Overview

For many years engineers and computer scientists have been dealing with different techniques to try to model and generate realistic human motions inside a computer: modeling the human's body movement is a challenging problem due to the fact that human's movement is highly synergistic and dimensional. Nevertheless, once the computer has the internal model of the motion, we can analyze the motion's properties, and we can even modify and reuse recorded movement in new situations (for example, retargeting it either to a new environment or another virtual character [1] with different proportions). In the last few years, many algorithms and techniques from different areas of research such as computer graphics, computer animation, computer vision and robotics have been proposed and developed to represent the human motion by a small set of parameters [Safonova et al., 2004, Urtasun et al., 2004, Calinon and Billard, 2005, Glardon et al., 2006b, Chai and Hodgins, 2007]. Essentially because this representation can be exploited to analyze and synthesize the human motion within a low-dimensional space of physically balanced motions. Although this space can be explored to generate faster and more realistic motions compared to high-dimensional approaches (for example, joint space methods), it can also be used in many applications. These include, but not limited to: motion editing, sports, games entertainment, medicine and motion capture.

Technological advances in motion capture (mocap) - essentially recording the skeleton motion of a human subject - has allowed to provide high quality motions for computer animation. However, the captured animations almost always meet specific needs. Therefore, modifying and reusing these motions in new situations became an increasing area of research known as *motion editing*. A common technique used to address such problem is inverse kinematics (IK). The techniques based on pure IK solutions are also known as *goal-directed* methods. IK gives the possibility to edit a motion by attaching constraints on the character's body to drag its limbs to new positions. Then, an IK solver computes the new character pose satisfying all the user-

---

[1] The virtual character is the virtual representation in our system of a real performer. We also refer to the virtual character as character figure, actor, or simply character.

specified constraints as much as possible. Nevertheless, when highly articulated figures are considered, IK algorithms are iterative. So, they try to find the best solution, for some set of constraints, iteratively minimizing some residual error (for example, the distance between the end-effector [2] and its goal). Furthermore, the majority of the professional animation packages have built-in IK solvers.

IK is the core of constraint-based motion editing techniques [Boulic and Thalmann, 1992, Lee and Shin, 1999, Gleicher, 2001, Liu et al., 2006, Callennec and Boulic, 2006]. One attractive feature of such approaches is that it provides interactive motion editing. Essentially, constraint-based techniques enable the user to specify constraints over the entire motion or at specific times, while editing the animation. An IK solver computes the "best"motion frame by frame, such that each posture achieves the predefined constraints as much as possible. One of the main challenges of per-frame techniques, regardless the construction of the solver, is the enforcement of the temporal continuity, because each frame is adjusted independently. To handle discontinuities, standard techniques take into account more information than just the pose at the current frame. For example, the solver might refer to the previous frame in order to choose a solution for the current one that does not introduce a discontinuity. The main drawback of this type of solution is the computation time needed to force each joint to be attracted toward its original value, which decreases system performance. In addition, filtering techniques are also exploited to enforce the fluidity of the motion [Lee and Shin, 1999]. Besides the great benefits of goal-directed constraint-based approaches providing interactive user manipulation, this type of motion editing technique presents two major drawbacks: (1) the user cannot edit the whole movement by constraining just one key frame, which is important for motions that need to be edited at a specific time (for example, hit time of a golf swing); and (2) the pseudoinverse computation is easily affected by the characters's degrees of freedom (DoFs), thus, the higher the number of character's DoFs, the slower the convergence.

In the past few years, there has been a lot of work in motion editing in the graphics community. The proliferation of motion captured database has allowed the research and development of data-driven or model-based techniques - data-driven approaches usually create animations from motion captured data that contain the movement details from a human actor. The embedded natural flow of the motion is their most attractive feature, because continuity can be encapsulated by construction (for example, when a motion model is created). Basically, these methods focus in constructing a model from mocap data - the data are usually represented by a low-dimensional space known as the *latent space* - and use the model to generate new motions from existing ones. On the other hand, it can become tedious to generate movements to meet user-specified constraints just by manipulating the model's parameters [Shin and Lee, 2006].

To obtain movements enforcing a new set of user-specified constraints within the latent space, data-driven approaches are combined with constrained optimization [Safonova et al., 2004, Grochow et al., 2004, Chai and Hodgins, 2007]. Data-driven techniques are capable to generate more natural-looking motions compared to standard constraint-based techniques, that is, the

---

[2] In our context, an *end-effector* is a device that is attached on the character's joint in order to provide interactive manipulation.

ones that perform optimizations in the joint space [3]. Despite these improvements, most recent model-based approaches present some drawbacks. First, some methods provide a constrained solution where the user is in charge of finding a point in the latent space that meets their constraints, leading to a non-automatic solution [Grochow et al., 2004]. Second, other techniques are so computationally expensive that they slow-down system performance taking several minutes to generate a motion solution [Safonova et al., 2004]. Furthermore, in data-driven motion editing, the accuracy and reliability of a data-driven application is strongly dependent on how well the database fits the motion pattern being edited. It means that, if the solution space does not provide enough information to match user-specified constraints, the generated animation may not be realistic presenting motion artifacts (for example, foot sliding and discontinuities).

In the field of motion editing, model-based approaches are capable to generate more natural-looking motions compared to goal-directed ones, but the solutions are limited to the database. On the contrary, goal-directed approaches, for example, the ones based on pseudoinverse techniques that performs optimizations in the joint space - which includes the great majority of constraint-based methods - can achieve a wide range of user-defined tasks. However, their disadvantage is the frequent lack of naturalness when it comes to reproduce human activities. A natural choice to overcome the limitations of both approaches, should be the construction of a method capable of combining the strengths of data-driven and goal-directed techniques.

1.2 Problem Statement and Contributions

Since motion is normally difficult to create from scratch by using traditional methods (for example, key frame animation), changing existing motions is a faster way to obtain the desired animation. Motion editing is the process used to edit existing motions to achieve the desired effect. It means that, motion editing is used to create new motions from existing ones. In this context, our aim for motion editing is to take an animation that is in need of some adjustments, for example, the motion has the basic shape that we want. This is almost always the case of captured animations, where simple adjustments should be made. For example, consider one adjustment: at a specific time of the motion, there is a new position that should be achieved. Consider Figure 6.10(a) as an example. We have a good golf swing motion, but we want that the golf club head hits the ball onto another position. We can associate a constraint to the golf club head, to change its position, and generate the new motion. Unfortunately, making little adjustments into a good motion may introduce some undesirable artifacts, that may destroy the intrinsic motion flow leading to unrealistic movements. Hence, providing techniques capable to handle motion artifacts preserving the natural motion flow of the animation is the major challenge of motion editing techniques.

In this thesis, we introduce a data-driven constraint-based motion editing technique, which combines the particularities of model-based and the versatility of goal-oriented approaches. Our method is based on the connection between linear motion models such as Principal Com-

---

[3] The joint space expresses the configuration (or posture) of the character figure, which is the configuration of all its joints.

ponent Analysis (PCA), which is used to estimate a set of Principal Components (PC) and Principal Coefficients (PCs) [Jolliffe, 1986], and Prioritized Inverse Kinematics (PIK), which is used to provide interactive motion editing. Essentially, PCA is used as a first step of preprocessing to reduce the dimensionality of the database to make it tractable, to encapsulate the main aspects of a specific motion pattern and to bound IK solutions closer to the space of natural-looking motions. In order to learn the PCA motion model, the training motions need to be time-normalized, such that, each motion must have the same number of samples. Nevertheless, to recover the correct motion size, for each motion, we let the learning algorithm to be sensitive to its corresponding duration. With this strategy, the motion editing system can recover the original motion duration. PIK allows the user to manipulate constraints with different levels of priorities while interactively editing an animation. The major strength of priorities is that constraints belonging to the highest priority layer are enforced first (for example, feet on the ground). Then, those of the next priority-layer are satisfied as much as possible (for example, hit the ball in the case of golf) without disturbing the previous constraints, and so on.

We use the PCA motion model to assist the construction of a framework capable to solve a constraint-based optimization problem within the latent space, which has a much smaller dimension compared to the joint space. In particular, this allow us to build a Low-dimensional Prioritized Inverse Kinematics (LPIK) solver, which reduces the size of the Jacobian matrix before its inversion. This dimensionality reduction speed up the computation of the pseudoinverse. As a result, system performance is improved even when we perform interactive editing of articulated figures with reasonable degrees of freedom, one of the bottlenecks of inverse kinematics techniques that perform optimizations in the joint space. In addition, during the optimization the solver refers just to the frame where a motion editing is carried out. As a consequence, this approach leads to faster deformations and also produces more natural-looking solutions compared to goal-directed ones. Moreover, by using the LPIK solver, a movement enforcing a new set of user-specified constraints can be obtained in just one step. By construction, it is sufficient to constrain a single frame with LPIK (for example, the hit position of the golf club head) to obtain a motion solution satisfying the user-specified constraints and preserving the continuous motion flow.

Furthermore, we build a system to allow users to generate natural-looking motions from key-frame constraints (i.e., the constraint is specified at specific poses) or key-trajectory constraints (i.e., the constraints are specified over a set of frames representing the trajectory of end-effectors). Recalling that, when a motion is adjusted at a specific time or over a set of not well spaced frames, the consistency between frames can be disturbed because temporal motion velocities may not be smooth in space. In order to enforce the spatio-temporal motion flow, we impose continuity through the Principal Components space of the underlying motion pattern, because velocities are intrinsically encapsulated into the model parameters [Glardon et al., 2004a]. In addition, the Principal Component space is smooth and continuous and, therefore, it can handle per-frame adjustments because they can be interpolated [Urtasun and Fua, 2004]. Furthermore, the PC is estimated from a well-defined class of motion behaviors reducing the chances of introducing discontinuities [Safonova and Hodgins, 2005]. As a result, our continuity strategy overcomes the need to consider previous frames and filtering techniques improving system performance. Finally, we discuss two techniques used for handling motion interpolations and extrapolations, which are used to increase the range of motion solutions by keeping user-specified constraints constant.

1.3 Method Evaluation and Validation

We have designed a number of experiments to evaluate the usability and the performance of the proposed technique. We first evaluate performance regarding two skeletons one with 90 and another with 222 degrees of freedom. We measure the computation time needed to compute the pseudoinverse, a pose deformation and a full-body goal-directed motion (i.e., reaching). We investigate with different constraint configurations. The results obtained with the proposed approach were compared against a PIK technique, which performs optimizations in the joint space [Callennec and Boulic, 2006]. We verified that, the LPIK performed faster in all the experiments.

Furthermore, we also compared our method against the PIK approach, regarding performance, robustness, simplicity, continuity and realism. For that, we have first synthesized various styles of golf swings played on three different ground shapes: flat, up and down slopes. This movement is challenging because it is highly synergistic and requires a great precision while moving with high speeds [Farrally et al., 2003]. In addition, to further validate our method, we have also tested with other behaviors such as walking jumps and reaching motions. We have verified that, for motions that need to be edited at a specific key (e.g., the hitting position in the case of golf or a final reach), our method provides a more intuitive editing strategy due to its simplicity for taking care of just one frame, and also produced more natural-looking animations. Despite this advantage, it is still faster: it was 400 times faster to edit a flat ground swing and 34 times faster to retarget a flat ground swing to an up and down slopes. Moreover, our approach have usually required less constraints to produce better quality animations.

We have also evaluated system performance regarding motion models learned from various behaviors and considering different database percentages. In general, system performance improves when optimizations are performed in specific motion model space (e.g., flat ground golf swings), and when the database percentage increases. We therefore evaluated the accuracy of the strategy used to recover the animation duration: the recovered duration presents an error of just one frame, which is acceptable for animations played above 16 frames per second (fps).

1.4 Our Approach for Motion Editing

The proposed framework is motivated by the challenges of providing an efficient and intuitive tool for constraint-based motion editing, where the user should be able to interactively edit the movement without the need of constraining multiple poses on the motion. Moreover, the proposed approach also lets the user specify constrains over a set of poses. In this case, each frame is adjusted individually so that it achieves the predefined constraints as much as possible.

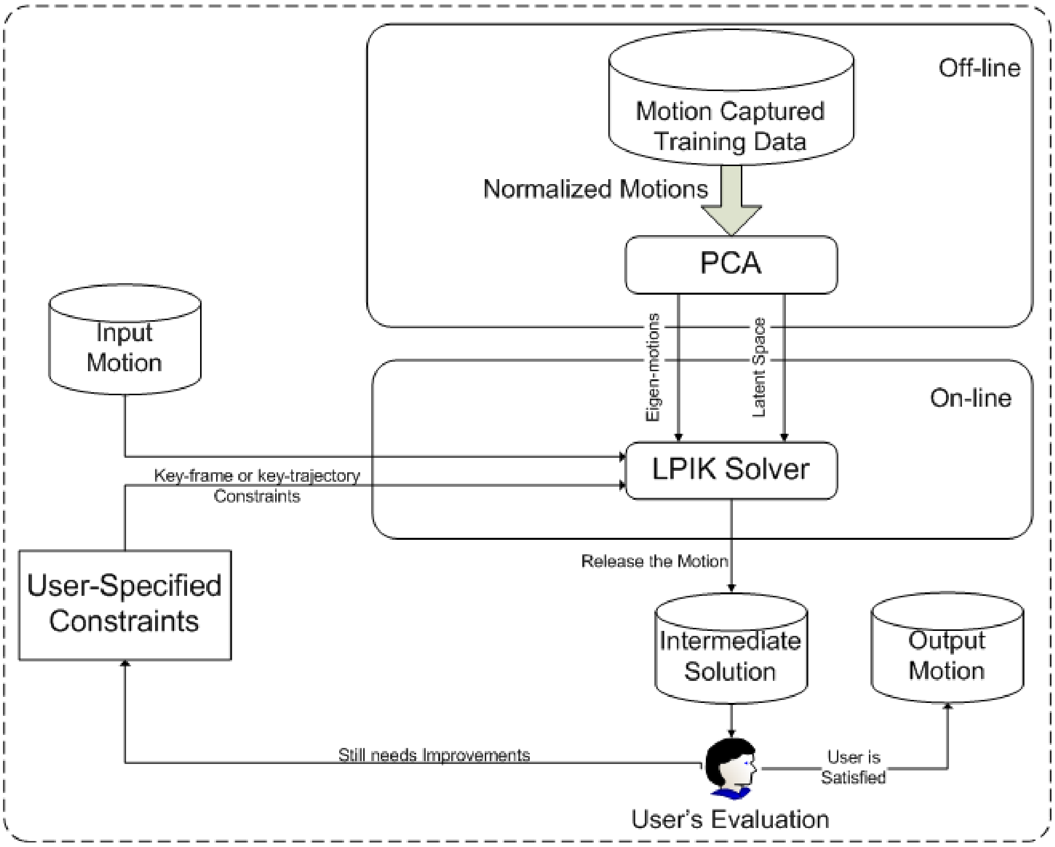


Figure 1.1: System architecture.

The proposed motion deformation technique belongs to the general class of off-line constraint-based methods [Gleicher, 2001]. However, our approach is data-driven.

The IK solver used in our system is the result of a constraint-based formulation, which takes into account the information of a basis constructed from a set of motion captured data. This basis is built by using a linear dimensionality reduction technique such as Principal Components Analysis. In this space, each motion is represented as a set of Principal Components and Principal Coefficients. Therefore, we solve a constraint-based optimization problem within a low-dimensional space of some desired motion pattern. To handle conflict tasks, we have extended our IK solver to handle priorities. The prioritized strategy not only enforces important constraints first (e.g., feet on the ground), but also speed up convergence performance.

A functional description of the system's architecture is shown in Figure 1.1. The system is subdivided in off-line and on-line (i.e., iteratively) computations. As the Principal Components (i.e., the basis) are estimated once and remain constant, they are estimated off-line. The motion capture training data can be composed of motions of the same behavior (e.g., walking jump) or mixed patterns (e.g., golf swings and reaching). As a result, different motion models can be constructed to handle specific tasks. For example, models constructed from mixed motion behaviors are more general and can either handle a large space of solutions or can be used to edit different motion patterns. Nevertheless, mixed models normally performs at a slower convergence rate compared to specific behavior motion models. The PCA can also be learned from small and large dataset. In fact, the performance of our system is not related with the size of the database, but with the percentage considered to approximate the underlying motion pattern. Indeed, a high database percentage showed to both speed up the optimization process and to improve the naturalness of the edited animations, for example, releasing motions with less artifacts such as foot sliding. Once the PCA motion model is constructed, it is loaded into the system's data structure to assist the LPIK solver handling the user-specified constraints within the latent space. The optimization is performed iteratively until the system release the new animation. If artifacts remain in the modified animation, the user can fine tune it by either manipulating the optimizer's parameters and keeping constraints constant or he/she can also restart the editing process by adding/removing constraints. This procedure is repeated interactively until the user is satisfied with the resulted animation.

1.5 Outline

Chapter 2 examines the literature for related work. We look at the connections of our work with the techniques of inverse kinematics and motion editing, regarding model-based and non-model-based approaches.

Chapter 3 describes the strategy used to parameterize a character motion for learning the PCA motion model. We show how the models tend to cluster similar motions and how they can generate new motions, by exploring the mapping from the latent to the motion spaces.

Chapter 4 contains the main contribution of this dissertation: the construction of the low-dimensional PIK solver. We look into the mathematical basis to make it possible to perform optimizations within the latent space. We also describe one algorithm to perform optimizations within the latent space capable to handle conflict tasks.

Chapter 5 describes the mode-based continuity approaches and some important results regarding the estimation of an appropriate database percentage. We also describe two techniques used to handle motion interpolations and extrapolations.

Chapter 6 performs a series of motion editing experiments to validate our method both qualitatively and quantitatively.

Chapter 7 draws conclusions of our techniques, describes strengths and shortcomings, and gives a perspective of on going research and future work.

Appendix A describes the optimal PCA reconstruction and the probabilistic PCA.

Chapter 2

State of the Art

2.1 Introduction

In this chapter, we present previous work related to the problem of posture control and motion editing. Results obtained in these application areas can often be shared. We start with the challenging problem of human body positioning that arises in computer animation applications, when the character body have to met a determined configuration respecting a set of user-specified constraints. Afterwards, we review general motion editing methods due to its importance in character animation. Subsequently, we will draw our attention to more specialized techniques, for example, constraint-based and data-driven motion editing , which are more related with the work developed in this thesis.

2.2 Inverse Kinematics

Due to its application in computer animation and more specifically in the field of constraint-based motion editing. We present a review of inverse kinematics describing the main type of techniques. Note that, because of the vastness of the subject, this review is not intended to be exhaustive.

2.2.1 Analytical Methods

Analytical (or closed-form) solutions IK systems are designed to handle simple kinematic chains, for example, those containing up to six degrees of freedom. This kind of solutions are very fast to compute and can be used for real-time applications. In general, in the field of robotics, robotic manipulators are constructed, so that, solutions can be found by direct resolution of the non-linear equations [Paul, 1981, Craig, 1986]. In the field of computer animation, several researchers have addressed the case of the anthropomorphic arm and leg, to give two important examples, Korein [Korein, 1985] provides an analytic solution for a 4-DoFs

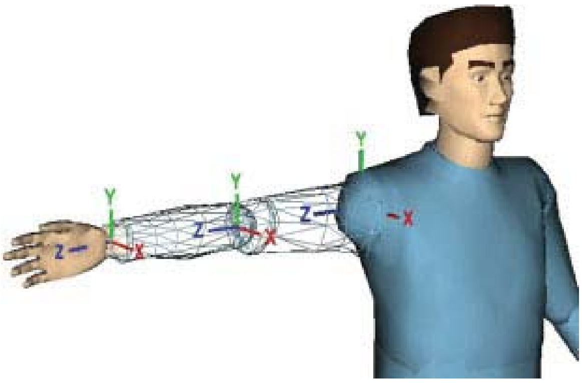


Figure 2.1: Local frames and the zero-posture of the right arm [Kallmann, 2008].

and 7-DoFs kinematic chaing. The first one controls only the position and the second the position/orientation. Tolani et al. [Tolani et al., 2000] also presented a similar technique by using a combination of analytical and numerical methods to solve the IK problem including position, orientation, and aiming constraints. The numerical part is used to handle joint limits. More recently [Unzueta et al., 2008] provided a real-time sequential IK method able to reconstruct full-body poses from a reduced set of end-effectors (six in total). This technique was intended to be used with low-cost human motion capture systems that would track only these six features. The method recovers a pose sequentially using simple analytic-iterative IK algorithms in different parts of the body in a specific order: by readjusting first the spine, then the clavicles, and finally each of the upper and lower limbs. They have designed an analytic-iterative reconstruction method of full spines and analytic methods for the clavicles, arms and legs that achieve very fast and visually acceptable results. They also modeled rotation limits for only a few known anatomical data to constrain joint orientations and developed a simple self-collision algorithm to prevent the elbow of penetrating the torso. Kallmann [Kallmann, 2008] proposed a whole-body analytical IK technique integrating collision avoidance and customizable body control suitable for animating reaching tasks in real-time. The IK method computes the posture of 7-DoFs arms and legs of human-like figures (see Figure 2.1). The whole-body state is achieved by interpolating pre-designed key body postures organized as a function of the direction to the goal to be reached. Moreover, the author also integrated a simple search method for achieving postures avoiding joint limits and collisions. Despite its attractiveness providing fast computations, analytic solutions do not exist for general articulated structures containing many degrees of freedom. In that case, optimization-based methods have to be used to solve the IK problem.

2.2.2 Optimization-based Methods

In this class of methods IK is formulated as a constrained optimization problem. The well-known *resolved motion rate control* introduced by Whitney [Whitney, 1969] is based on a linearization of the non-linear equations, given an initial configuration, characterized by a Jacobian matrix relating differential changes of the joint coordinates to differential changes of task coordinates (e.g., position of an end-effector). Once the linear system of equations is solved, a new joint configuration closer to the goal is computed by adding a joint velocity variations to the initial configuration. By iteratively repeating the process, the system converges to a solution satisfying the constraint, assuming that the initial configuration was close to the desired goal. This method is motivated by the *Newton-Raphson* method for the resolution of non-linear equations [Ortega and Rheinboldt, 1970].

Subsequent work have been focused in the investigation of the redundancy of the IK problem at the differential level. Liégeois [Liegeois, 1977] proposed an extension for the general linearized model of equations expressed in the joint space, by exploiting the null space of the Jacobian matrix. Klein and Hung [Klein and Huang, 1983] provided more insight on this topic investigating the use of the pseudoinverse for controlling redundant manipulators, and on the meaning of the pseudoinverse solution in terms of the Singular Value Decomposition (SVD) representation of the Jacobian matrix. Hanafusa, Nakamura and Yoshikawa [Hanafusa et al., 1981, Nakamura et al., 1987] extended the redundancy exploitation with criteria expressed in Cartesian space. In their method, a secondary Cartesian task that can be satisfied without affecting the primary task can be achieved. This simultaneous resolution of two tasks with different priorities is known as the *task-priority* strategy. In the same sense, Maciejewski and Klein [Maciejewski and Klein, 1985] improved the resolution scheme of Hanafusa et al. by exploiting pseudoinverse properties. In their task-priority formulation, the highest priority task is used to constrain the end-effector to follow a specified trajectory. The lower one is created to handle the obstacle avoidance point. Siciliano and Slotine [Siciliano and Slotine, 1991] generalized the priority strategy to handle an arbitrary number of priority levels. Finally, Baerlocher and Boulic [Baerlocher and Boulic, 2004] improved the generalized priority schema by formulating an efficient and recursive solution speeding up the convergence performance.

When dealing with the Jacobian matrix the management of singularities becomes an important issue, which need to be handle. Nakamura and Hanafusa [Nakamura and Hanafusa, 1986] proposed to use the damped-least squares inverse as an alternative to the classical pseudoinverse to handle singularities. The insight behind their approach is the simultaneous minimization of the residual error and the solution norm. Afterwards, more sophisticated methods have been proposed by [Maciejewski and Klein, 1988] to dynamically determine the damping factor according to the smallest singular value. This approach has the advantage of not disturbing the solution of configurations far from a singularities, while retaining the important characteristics of the damped least squares solution on a singular configuration. To handle algorithmic singularities, which appears in the task-priority strategy when two tasks are in conflict, Chiaverini [Chiaverini, 1997] introduced a new formulation that overcomes the effects of these issues. However, the method proposed by Chiaverini uses the damped least squares inverse to compute the projectors. By doing this, some important properties of the pseudoinverse do not hold anymore. Consequently, this leads to a violation of the priority hierarchy as demonstrated in [Baerlocher and Boulic, 1998]

2.2.3 Model-based Methods

In this class of techniques the human behavior is directly integrated in the process, for example, by learning a model from the data, to obtain more realistic natural-looking solutions. This approach is important for both computer animation applications and ergonomics. Engin and Chen [Engin and Chen, 1986] proposed 3D mathematical model of the shoulder complex motion range based motion database. Their model was capable to predict the range of motion of shoulder joint. Beck and Chaffin [Beck and Chaffin, 1992] proposed a behavioral inverse kinematics algorithm developed from existing regression equations. Verriest et al. [Verriest et al., 1994] used a linear statistical model to predict arm reach postures. Both approaches make use of statistical regression equations developed from a database of measured human behaviors. For the purpose of posing a human arm with a given hand position, Wang and Verriest [Wang and Verriest, 1998] propose an geometric inverse kinematic algorithm specific for the human arm that incorporates realistic shoulder limits. Subsequent work, in the field of computer animation, concentrates in full-body models learned from motion capture (mocap) data. Grochow et al. [Grochow et al., 2004] applied a nonlinear dimensionality reduction technique, called the Scaled Gaussian Process Latent Variable Model (SGPLVM) (a type of non-linear PCA) to human motion data. The SGPLVM was capable of mapping poses from the low-dimensional space directly to the pose space (see Figure 2.2). Furthermore, the learned probabilistic model was combined with kinematic constraints to create new character postures. As a consequence, the system was capable to generate natural-looking motions in simple IK contexts, such as: interactive character posing and trajectory key-framing. As their approach is kernel-based, it cannot handle large data sets due to the limitations of the model.

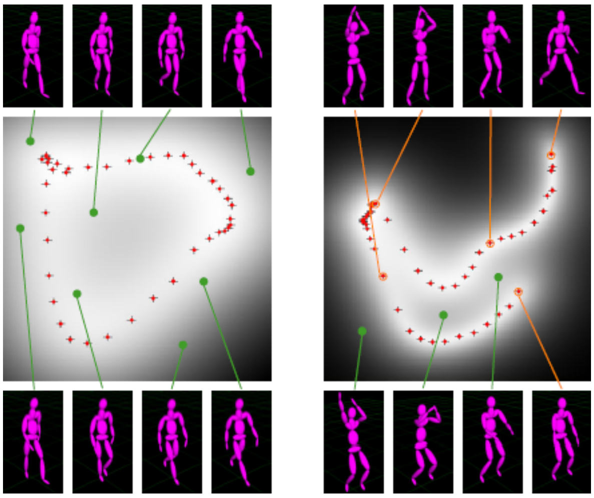


Figure 2.2: SGPLVM low-dimensional spaces learned from a walk cycle and a jump shot [Grochow et al., 2004].

The advantage of all these methods is that they are likely to produce more realistic solutions than those resulting from purely constraint-satisfaction based methods. On the other hand, their application is limited by the underlying model or database. Nevertheless, this type of IK solvers are well-suited for constraint-based motion editing, because the aim of these techniques is to produce small changes in the input motion [Gleicher, 2001].

2.3 Motion Editing and Synthesis

Motion editing techniques aim to slightly modify an existing animation, while preserving some specific geometrical features of the original motion. On the other hand, motion synthesis methods aim to synthesize or generate new motions from existing ones.

2.3.1 Spacetime Constraints

Spacetime constraints treats an entire motion sequence as a single unit, ensuring that the motion solution is physically valid. This method can be used to synthesize new motions (e.g., generating a motion "from scratch") or to edit motion captured animations by using either kinematics or physical-based constraints. For example, to generate an animation the user may specify pose constraints that must be satisfied by the resulting motion sequence (e.g. the character pose at the beginning and the ending of the animation), and also specifies an objective function that is a metric of performance such as the total power consumption of all of the character's muscles. Witkin and Kass [Witkin and Kass, 1988] built a spacetime constraint-based system in which constraints and objectives are defined over *Spacetime*, referring to the set of all forces and positions of all character's degrees of freedom (DoFs) from the beginning to the ending of the animation sequence. The method was applied for animating a luxo with mass and force properties (see Figure 2.3). Following this idea, Cohen [Cohen, 1992] construct a more complete system and introduced the concept Spacetime windows to enable the use of spacetime constraints in an interactive framework, providing tools to the animator to examine and thus guide the optimization process. Subsequent work has addressed solver performance [Liu et al., 1994] and provide alternative solutions [Grzeszczuk and Terzopoulos, 1995, Ngo and Marks, 1993]. Rose et al. [Rose et al., 1996] applied spacetime constraints to obtain realistic transitions between motions. Nevertheless, these works have not been applicable for interactive motion editing. To achieve interactive editing, Gleicher [Gleicher, 1997] provided a different formulation

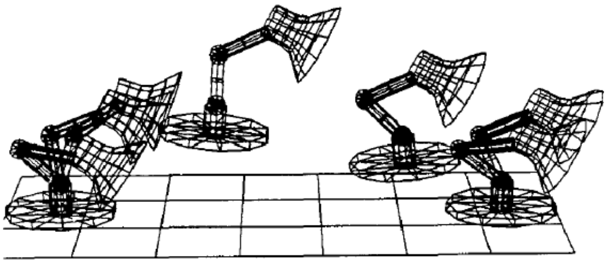


Figure 2.3: Spacetime constraints animating a luxo with mass and force properties [Witkin and Kass, 1988].

to the problem, by combining displacement maps and spacetime constraints. To find an interactive performance for motion editing, he simplified the spacetime problem by removing the physics-related aspects from the objective function. He also applied this technique for motion retargeting [Gleicher, 1998]. Basically, the user defines a set of constraints that the retargeted motion should preserve. Based on this set of constraints, the spacetime solver tries to find a motion that satisfies the previous set of constraints while minimizing an objective function. In his algorithm, Gleicher used an objective function minimizing the distance to the initial posture (in a joint value sense). In [Popović and Witkin, 1999], Newton's laws were applied on a simplified character to minimize computational costs and to preserve the essential physical properties of the motion. They use the spacetime constraints dynamics formulation to keep the realism of the original motion sequence without sacrificing full user control of the editing process. Liu and Popović [Liu and Popović, 2002] introduced a method for rapid prototyping of realistic motions. Starting from a simple animation generated using key framing, physical laws are enforced to produce a more realistic one. Given an input animation, they first detect position and sliding constraints in order to separate the animation into constrained and unconstrained stages. Afterward, they generate transitions between these stages by suggesting the user a set of previously learned transition poses. Finally, they compute the final animation by minimizing the mass displacement, the velocity of the DoFs and by ensuring static balance. This optimization is subject to constraints on the linear and angular momentum that directly depend on whether the character is on the ground or airborne. Generating physically realistic animations using optimization often requires to compute first derivative of joint torques which is of quadratic complexity. This inevitably leads to scale problems. Abe et al. [Abe et al., 2004] used a framework similar to the one presented in [Liu and Popović, 2002] to generate a variety of motions given an input one. They then generated a variety of motions in real-time by using simple interpolation.

The work reported in these papers provides good animation results, but have a number of restrictions. The first one is that they are computationally expensive to solve prohibiting the use of large animation sequences. Moreover, when the number of character's DoFs increases, the computation time becomes rapidly prohibitive. In addition, the character must have full knowledge of the future to optimize their actions. Another limitation is the difficulty in controlling the computations due to the highly non-linear nature of the constraints and objectives, therefore, the available numerical methods often do not converge to acceptable solutions. Another drawback is the failure to identify objectives other than energy consumption. Despite the benefits regarding user interactivity pointed in some of the works cited previously, spacetime constraints requires solving a single mathematical problem for the entire motion. This leads to very large constrained optimization problems that is usually very difficult to solve.

2.3.2 Frequency-based Methods

This class of methods considers the animation curve as a time-varying signal, by applying signal processing techniques the input motion is modified (see Figure 2.4). In the field of character animation, Bruderlin and Williams [Bruderlin and Williams, 1995] adapted multiresolution filtering, multitarget motion interpolation with dynamic timewarping, waveshaping and displacement mapping for editing motion data. Witkin and Popović [Witkin and Popović, 1995] presented the *motion warping* method built from the combination of time warping and displacement mapping techniques. For applying modifications on the input motion, the user conveniently places keyframes in the input animation, which acts as a set of spatial constraints. Subsequently, displacement maps are computed for each animation curve as these latter are warped independently. Finally, the interpolation between keyframes is based on the changes (displacement maps) instead of being based on the absolute motion curves values. Unuma et al. [Unuma et al., 1995] described a simple technique to represent periodic motions using the so-called rescaled Fourier functional model. By using this technique, the motion can be interpolated, extrapolated, or even subtracted with other motions.

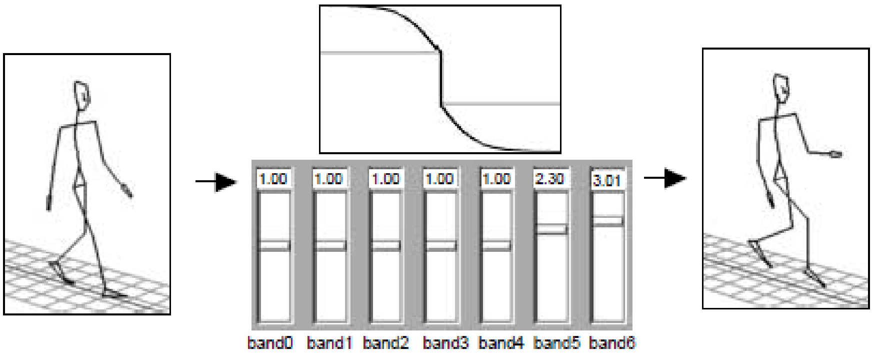
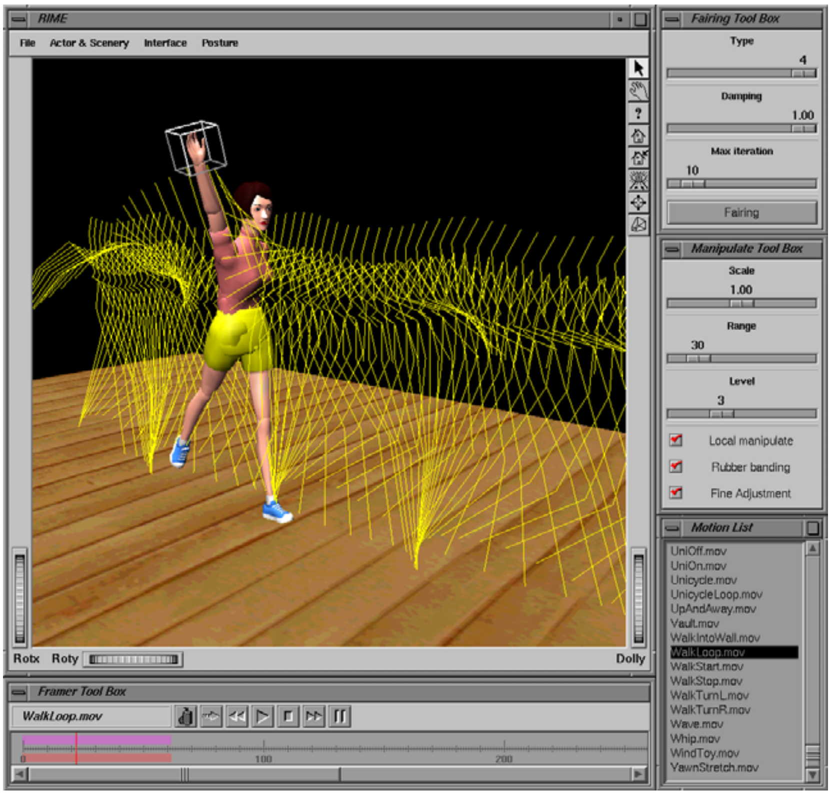


Figure 2.4: Adjusting gains of bands for joint angles [Bruderlin and Williams, 1995].

These methods are in general difficult to use because the manipulation of a time-varying signal is fastidious and non-intuitive, for example, when the animator desires to modify a motion with some precise requests. Furthermore, they do not ensure that original kinematic constraints are preserved. When using motion warping, for example, if the keyframes are not correctly placed, then the final motion may violate important geometric constraints such as the feet penetrating the ground or sliding.

2.3.3 Constraint-based Methods

Gleicher [Gleicher, 2001] proposed a taxonomy of constraint-based techniques. Constraint-based techniques enables the user to specify constraints over the entire motion or at specific times, while editing an animation (see Figure 2.5). An inverse kinematics solver computes the “best”motion, pose by pose, so that each pose achieves the predefined constraints as much as possible. Boulic and Thalmann [Boulic and Thalmann, 1992] presented the first work in motion editing using IK on a per-frame basis to enforce constraints. They used a numerical IK solver in order to simultaneously handle two tasks: the primary and the secondary. The primary task is responsible for kinematics constraints. The latter is in charge of tracking the original motion so that the corrected motion does not deviate too much from the initial one. This paper clearly sets the foundations for future work using this approach. Lee and



Type 4
Damping 1.00
Max iteration 10
Scale 1.00
Range 30
Level 3
☑ Local manipulate
☑ Rubber banding
☑ Fine Adjustment

Figure 2.5: Constraint-based system interface [Lee and Shin, 1999].

Shin [Lee and Shin, 1999] introduced the first example of per-frame plus filtering class of motion editing techniques. Their interactive motion editing system addresses many common tasks such as retargeting, transitions and/or editing. In this work, two concepts were introduced for per-frame IK plus filtering methods: the *intra-frame* and the *inter-frame* consistencies. The intra-frame consistency represents the set of spatial constraints a virtual human must achieve at each frame of the animation. This is usually done by using an IK solver. The inter-frame consistency specifies that neighboring adjusted frames have to be as similar as possible to avoid adding jerkiness in the final animation. This is usually achieved by filtering the changes to the initial motion in order not to add high frequencies. The inevitable consequence is that it potentially destroys intra-frame consistency. As a result, these phases are usually iteratively repeated. Their system makes specific choices for each key aspect of the approach: the IK solver and the filtering process. They implemented a highly specialized IK solver for human-like articulated figures. The final implementation is very fast. The authors have had to make several sacrifices in order to achieve such performances. As a consequence, the set of constraints that it can handle is limited. Inter-frame consistency is enforced by using a hierarchical B-spline-based filter. Each displacement map is adaptively refined by hierarchically fitting B-splines defined by uniform sequences of knots. The greater the knots used, the finer the results. The main problem of using B-spline-based filtering is that it becomes computationally expensive if the number of knots in the B-splines is large. Indeed, the problem of fitting a B-spline to scattered data points is reduced to finding the set of control points which best interpolates the data points. This is done by minimizing an objective function using a least-squares method. Monzani et al. [Monzani et al., 2000] proposed a method to retarget animations to characters having both geometrical and topological differences using an intermediate skeleton. The intermediate skeleton was used as a bridge between all the potential skeletons for which the animation have to be mapped. To achieve smooth transitions between original end-effectors trajectories and constraints trajectories, the authors decided to add an ease-in period (resp. an ease-out period) before (resp. after) the constraints are activated (resp. deactivated). During the ease-in period, the end-effector trajectory is linearly interpolated to smoothly reach the specified goal. During the ease-out period, they preferred to linearly interpolate the postures instead of the trajectories because of potential conflicts between constraints leading to discontinuities when one of them is deactivated. According to the authors, this method only gives good results when the adapted motion remains close to the original one. In particular, linear interpolation may produce noticeable discontinuities because it does not take end-effectors velocity into account. Choi and Ko [Choi and Ko, 2000] presented the first work on online motion retargeting. They built an IK solver based on the motion rate control method. The primary task is to track the motion of the end-effectors. The secondary task is to imitate the motion of the source character as much as possible. As a consequence, the inter-frame consistency is implicitly enforced. Subsequent work [Kulpa et al., 2005] proposed a motion adaptation methodology. They built an efficient implementation of the Cyclic Coordinate Descent algorithm to adapt an animation in real-time. Although this method does not use any kind of filtering, it provides very good results for real-time motion adaptation of virtual characters. To ease the adaptation, the underlying skeleton is divided into groups, each of which being an individual kinematic chain. This has the advantage to ease the computation of a solution. However, as no synergy exists between groups, it may lead to unrealistic results. Liu [Liu et al., 2006] proposed a motion editing technique with collision avoidance to prevent the character's limbs to interpenetrate the body during editing. Displacement maps techniques and a Kalman filter were used to preserve the similarities between the edited and original motions, and to impose continuity between frames. Le Callennec and Boulic [Callennec and Boulic, 2006] proposed an off-line interactive motion editing technique with prioritized constraints. The characteristics of the original motion are preserved by adding the difference between the motion before and after editing as the lowest priority constraint by using an optimization vector. The use of an optimization vector reduces the need to filter the results as each adjusted posture is attracted to its initial configuration. Splines curves were used to impose smoothly varying constraints. Our method differs from all these previous approaches because we constructed a data-driven constraint-based method, which performs optimizations within the latent space of some motion pattern.

2.3.4 Data-driven Methods

Data-driven (or model-based) methods focus in constructing a model from mocap data (e.g., of people performing the same activity many times), and use the model to generate new motions from existing ones. Alexa and Müller [Alexa and Muller, 2000] used PCA to represent animations as a set of principal animation components, with the aim of decoupling the animation from the underlying geometry. In this simple application of PCA, the authors could synthesize the same animation using different geometries (see Figure 2.6). Glardon et al. [Glardon et al., 2004b] generate walking, running and jumping motions by exploring the low-dimensional space constructed from a hierarchical PCA. This further decomposition of the PCA space was used to extract high-level parameters (e.g., walking speed) to provide user manipulation guiding a character through the virtual environment from a small set of well understood parameters. Glardon [Glardon et al., 2006b] also integrated an optimization-based IK solver, as a second preprocessing stage, for preventing foot sliding by exploiting the predictive capability of the PCA motion model. In a subsequent work, still in character navigation, Glardon [Glardon et al., 2006a] developed a motion blending technique in parameter space, such as locomotion speed and jump length, to handle two PCA motion models in order to treat transitions from walking/running to jump motions, while guiding a virtual character with obstacle avoidance through the environment. This strategy avoids the need to perform a preprocessing step involving all the clips of the database allowing the generation of dynamically parameterized motions. IK is also used a second step of pre-processing to clean up foot sliding. Subsequent data-driven approaches use mocap data to restrict the solution within the space of natural-looking motions. Motion graphs and motion interpolation are used to produce new motions from a database [Arikan and Forsyth, 2002, Kovar et al., 2002, Kovar and Gleicher, 2004, Mukai and Kuriyama, 2005]. These techniques have two major restrictions. The first one is that, they are not able to handle end-effector constraints that are not represented in the motion database. The second one, they present limitations in handling constraints over multiple frames. A number of researches have also developed techniques to synthesize human motion in a low-dimensional space, by using both linear and non-linear models. Safanova et. al. [Safonova et al., 2004] proposed a motion synthesis framework able to synthesize new motions by optimizing a set of constraints within a low-dimensional space constructed with PCA. The constraints are expressed in the world frame and then projected onto the low-dimensional

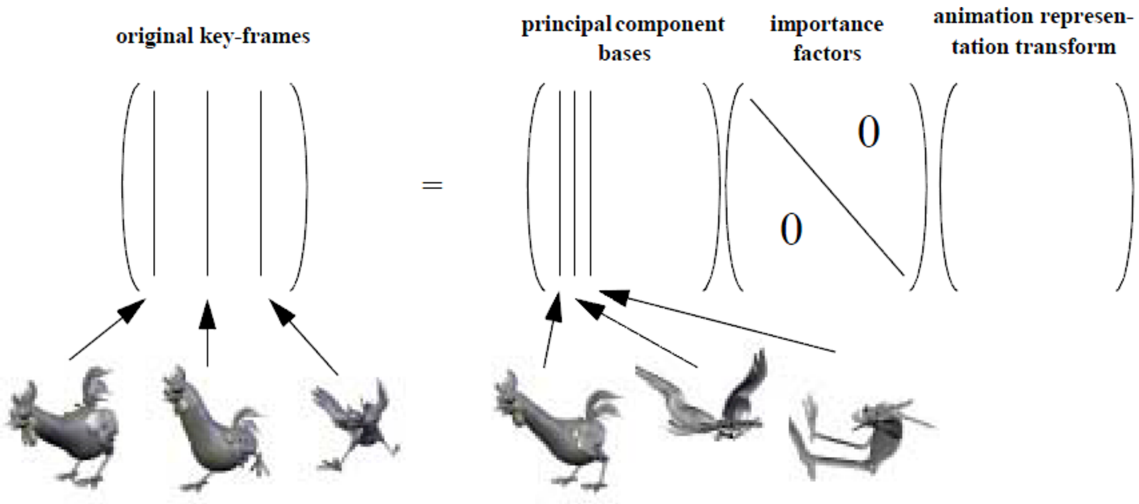


Figure 2.6: The PCA for geometric animations [Alexa and Muller, 2000].

space previously built. Finally, the motion is generated by optimizing its “representation”in the PCA space over time. The optimization uses an objective function which tries to minimize the torques, the jerkiness and the deviation to original motions. The final result is then generated by using B-Splines representing the values of the PCA coefficients over time. Unfortunately, some type of constraints (e.g., feet constraints) cannot be represented with sufficient accuracy by the framework, so they incorporate a simple IK solver, as a second preprocessing stage, to alleviate this limitation and handle foot sliding. Their approach is similar to spacetime constraints, which requires solving a single mathematical problem for the entire motion. This leads to a very large constrained optimization problem that is very time demanding for solving. Urtasun et. al. [Urtasun et al., 2004] proposed a style-based motion synthesis framework able to produce motions with different styles by extrapolating the PCs parameters. In a subsequent work, Urtasun and Fua [Urtasun and Fua, 2004] used the PCA to improve 3D body tracking. Their formulation is similar in spirit to ours because they need to evaluate the PCA coefficients that describes a tracked motion style. However, as their PCA model is not combined with IK the resulted motions generally present motion artifacts such as foot sliding. On the contrary, our approach is capable to handle this issue because it combines PCA with prioritized IK. Shin et. al. [Shin and Lee, 2006] proposed a framework for low-dimensional motion synthesis by investigating three linear models: PCA, multi-dimensional scaling (MDS), and Isomap. They constructed a user interface based on a 2D grid describing a low-dimensional representation of the data, where the user can synthesize a motion by dragging the mouse pointer on this grid. Despite that fact that this approach can generate natural-looking motions, the system has a limitation handling user-specified constraints. So, given a limited set of constraints such as the position of an end-effector or the root segment, they first find a node on the grid of which a pose matches the constraint best and then, the IK solver blends existing data samples to achieve the constraints as much as possible. It means that, if there is no pose matching the constraints the method may fails. The authors do not performed any experiment to verify if their method was capable of handling extrapolations. Chai and Hodgins [Chai and Hodgins, 2007] have proposed the use of a statistical dynamic model to construct a constraint-based motion optimization framework. Their dynamic model is trained over a motion database and then used as a prior for generating natural-looking motions. The system can handle key-frame, key-trajectory constraints and both simultaneously. The first difference with our approach is that the motion is a result of a global optimization process, similar to spacetime constraints techniques, whereas our technique acts in individual poses. The second one, is that, their system can only generate natural motions from a minimum of four key-frame constraints, whereas ours can generate the whole movement by constraining and editing just one key-frame on the motion. Recently, Raunhardt and Boulic [Raunhardt and Boulic, 2009], have combined a data-driven and goal-oriented methods to construct a hybrid postural control approach. The main focus of the work is to treat the issue of controlling a full-body goal-directed motion (i.e. reach) where locomotion is not necessary. The model is learned from *pose* rather than *full motion* data. To generate the final posture, the method requires two steps: (1) set the goal position for an effector that corresponds to the captured motions; (2) set the Cartesian constraints (e.g. feet on the ground). As a result, the system computes a solution respecting the user-specified constraints. This work is based on our work [Carvalho et al., 2007], which combined a linear motion model and a prioritized IK solver to met user-specified constraints within the latent space. The main differences between this work and ours is that, in their work the optimization is performed in the joint space rather than the latent space of the underlying motion pattern and we focus on interactive motion editing. Ikemoto et. al. [Ikemoto et al., 2009] proposed a motion editing technique that uses Gaussian process and a simple IK solver. The system is capable of propagate pose modifications across the motion sequence.

2.4 Conclusion

In this chapter, we reviewed the main works regarding inverse kinematics and motion editing techniques. Nevertheless, it is important to stress the main differences between our method and the ones proposed in [Safonova et al., 2004, Grochow et al., 2004]. The LPIK solver handles kinematic constraints directly in the latent space. Therefore, when the motion is reconstructed by the model it already satisfies all the user-specified constraints. On the contrary, the method presented in [Safonova et al., 2004] applies IK in a second stage of preprocessing, that is, after the motion have been recovered: IK is performed in the joint space. We concentrate our investigation in motion models, instead of pose models [Grochow et al., 2004], because they are capable to encapsulate the natural flow of the motion. We then take advantage from the natural flow of movement provided by a motion database to develop new continuity strategies, which are used in our motion editing system. Furthermore, the proposed data-driven constraint-based motion editing method differs from all the approaches described in section 2.3, due to the nature of our IK solver and the way that the system handles user-specified constraints. The proposed method is shown in more details in the next chapters.

Chapter 3

Motion Pattern Encapsulation

3.1 Introduction

This chapter describes the steps needed to build a motion model. As we are dealing with articulated human body figures, we first concentrate our focus on the issue of joint and motion modeling. The essential feature of a joint is that it permits some degree of relative motion between the two segments it connects. In our approach, the joint model is used to represent a body posture. Nevertheless, since we are dealing with motions the parameterization is extended to handle the character movement. Then, once all the motions are parameterized, they are used as training data for learning the PCA motion model. The complete process is described in the next sections.

3.2 Motion Modeling

Modeling the human body movements is one of the most difficult and challenging problems in computer animation due to its high-dimensional joint space representation. Normally, fifty to sixty dimensions are used to represent a good quality human pose [Safonova et al., 2004, Wang et al., 2008]. Nevertheless, for many types of behaviors (e.g., walking, golf swing, reaching) the movements of the joints are very correlated tending to move in a coordinated pattern [Rose and Gamble, 1994, Safonova et al., 2004, Carvalho et al., 2007]. As a consequence, the dimensionality of human motions can be reduced and modeled by a small set of parameters, for example, by using a dimensionality reduction technique such as PCA.

The PCA is a wide spread statistical technique used to reduce the dimensionality of an input dataset [Jolliffe, 1986, Gonzalez and Woods, 2002]. Despite all the benefits of PCA, the most important one, when compared with other dimensionality reduction techniques such as SG-PLVM [Grochow et al., 2004], is that, it provides a closed form solution: the data are analyzed and reconstructed explicitly from a well-known set of matrices operations. In the next sections, we show and explain more formally, how we parameterize and encapsulate an underlying motion pattern within a low-dimensional space and present important model results.

3.2.1 The Human Figure

Essentially, the human figure is modeled as a hierarchy of rigid segments connected by joints (see Figure 3.1(a)). Segments are usually defined by their length, shape, volume and mass properties, but the bones are not necessarily modeled as 3D objects. The joints are used to modify the posture of the body. To improve the appearance of the character, a mesh can be attached to the skeleton to simulate, for example, the character's skin and clothes (see Figure 3.1(b)).

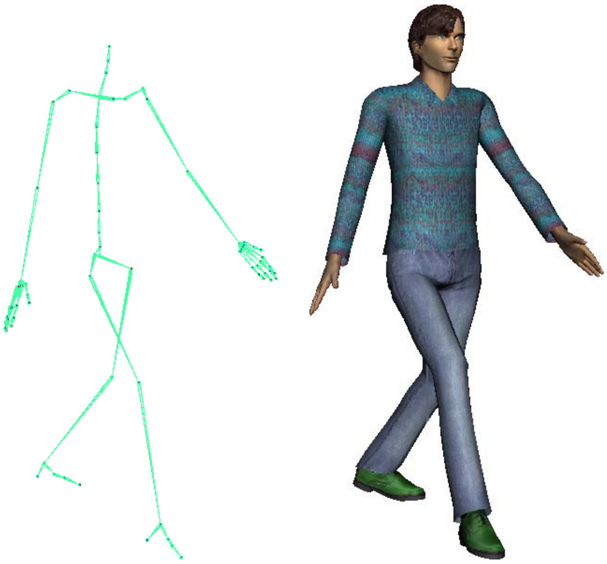

(a) (b)

Figure 3.1: (a) Character skeleton connected by joints. (b) A mesh attached to the skeleton.

To construct the human figure, we use the HAnim standard [H-Anim, 2009]. HAnim provides an abstract representation for modeling three dimensional human figures. One of its most important advantages is the extension of the family of joint types. That is, three dimensional articulations can be represented with a single exponential map. In prior approaches, these articulations were decomposed in a succession of three revolute joints with a common center of rotation. For reasons that will become apparent after, we construct two skeletons with two different levels of articulation (i.e., number of joints). The first with 29 and the second with 73 joints. The root joint is included. The joints used to model both skeletons can be seen in Appendix B.

3.2.2 Motion Parameterization

The choice of a motion representation plays an important role in the estimation of the model parameters. Prior researches [Rose et al., 1998, Park et al., 2002, Kovar and Gleicher, 2003, Glardon et al., 2006b] represent each motion as a set of joint orientations expressed by euler-angles, quaternions or exponential maps [Grassia, 1998]. We follow this approach because it leads to a simple and efficient motion representation, where each frame is represented by a vector. More formally, let us define a character pose [1],

$$\mathbf{\Theta} = \{q\}, \tag{3.1}$$

as a state vector describing the 3D global position, $\mathbf{P}_{root}$ and the 3D global orientation $\mathbf{Q}_{root}$ of the root node, and a set of joint orientations θ, represented by three parameters of the corresponding exponential map representation:

$$\mathbf{\Theta} = \left\{ \begin{array}{c} \mathbf{P}_{root} \\ \mathbf{Q}_{root} \\ \theta \end{array} \right\} = \left\{ \begin{array}{c} q_1 \\ q_2 \\ \vdots \\ q_n \end{array} \right\}. \tag{3.2}$$

Considering that there are ζ orientations (the root plus $(\zeta-1)$ joints) represented by exponential map, the dimension n, corresponds to the number of degrees of freedom of the skeleton model, which is equal to $3(\zeta+1)$. Essentially, no single parameterization of rotation is best and the use of a particular representation depends on its performance in the application [Grassia, 1998]. Euler-angles suffer from *gimbal lock* that occurs when two of the three rotation axes align causing a rotational degree of freedom to be lost. As unit quaternions are embedded in $\Re^4$, there are four directions in which a quaternion can change, but only three rotational degrees of freedom. Moreover, as it has to keep its unit length, quaternion operations can move the quaternion off the unit sphere leading to non-rotations. There are several strategies developed in the literature to handle quaternion problems, but they increase code complexity degrading system performance. We therefore use the exponential map representation because it does not have the problems encountered by euler-angles and the limitations of unit quaternions. However, despite these apparent limitations of unit quaternions, they are well suited for interpolations [Shoemake, 1985].

As we use motion capture (mocap) data from real people, and as each person tends to perform the same activity with some variability in speed, the database contains motions that, in general, have different durations. So, a motion duration normalization is necessary in our approach because the PCA's parameters (see section 3.2.3) are estimated from complete motions. Let us recall the relationship between the frame-rate F_r (e.g., 25 fps), the motion duration T, and the the total number of frames, N:

$$N = T \cdot F_r. \tag{3.3}$$

The normalization is done as follows. As each motion has its corresponding duration, T_i, we preprocess the motion database to compute the mean time, μ_T, of all, d, motions as shown

---

[1] Essentially a skeletal configuration of the virtual character.

by Eq. 3.7. Once the mean time is computed, we use its value directly into Eq. 3.3 to establish the total number of poses that all motions should have to preserve a sufficient sampling frequency. The normalization is carried out by using quaternion spherical linear interpolation (Slerp) [Shoemake, 1985] [2]. Accordingly, every motion has now an identical number of frames for a unit duration. In order to recover the correct motion duration, we use a similar approach as described in [Grochow et al., 2004]: for each motion, we additionally include its original duration (i.e., the value before normalization) in the motion vector. This feature is important to inform the final duration of the synthesized motion, so we let the learning algorithm be sensitive to it. A motion is then represented as a line vector of the form:

$$\boldsymbol{\Upsilon}_i = \{\boldsymbol{\Psi}_i = \{\boldsymbol{\Theta}_1 \ldots \boldsymbol{\Theta}_k \ldots \boldsymbol{\Theta}_N\}, T_i\}. \tag{3.4}$$

We let the pose varies as a function of a phase parameter φ that is defined to be 0 at the beginning and 1 in the end of the motion. The motion is therefore represented as a sequence of N poses indexed by the phase of the motion such that, for frame $k \in [1, \ldots, N]$, the discrete phase $\varphi_k \in [0, \ldots, 1]$ is sampled at regular time intervals. $\boldsymbol{\Theta}_k$ is therefore a pose corresponding to a frame index k. Since a given motion consists of N poses, the motion vector has dimension:

$$D = (n \cdot N) + 1. \tag{3.5}$$

3.2.3 PCA Motion Model

When the mocap data are arranged in a vector form, we use a linear motion model such as Principal Component Analysis (PCA) to find a low-dimensional representation, which efficiently acquires the important nuances of a specific motion pattern. The PCA is based on statistical analysis [Jolliffe, 1986]. Thus, the D elements of the motion vector $\boldsymbol{\Upsilon}$ are considered as statistical variables. In motion capture, if the motion is recorded d times, one can say that the elements of $\boldsymbol{\Upsilon}$ (i.e., the statistical variables) were sampled d times. Therefore, the set of recorded motions are organized in a motion matrix $\mathbf{M}$. More formally, to learn the PCA from motion data, we use following algorithm:

1. Build a motion matrix of dimension $d \times D$, where each motion vector is arranged as a row,

---

[2] The initial representation of the joint angles is exponential maps. Before the time normalization, we convert them to quaternions. Once the normalization is finished, they are converted back to exponential map for estimating the PCA's parameters. Note that, for the root position, we proceed with a simple linear interpolation. We suggest the interested reader to [Herda, 2003] for more information about angle conversions.

$$\mathbf{M} = \begin{pmatrix} \mathbf{\Theta}_{11} & \mathbf{\Theta}_{12} & \dots & \mathbf{\Theta}_{1N} & T_1 \\ \mathbf{\Theta}_{21} & \mathbf{\Theta}_{22} & \dots & \mathbf{\Theta}_{2N} & T_2 \\ \vdots & \vdots & \dots & \vdots & \vdots \\ \mathbf{\Theta}_{d1} & \mathbf{\Theta}_{d2} & \dots & \mathbf{\Theta}_{dN} & T_d \end{pmatrix}. \tag{3.6}$$

2. Compute the D-dimensional mean motion $\zeta_\circ$, that is composed of $\mathbf{\Psi}_\circ$ and μ_T:

$$\mathbf{\Psi}_\circ = \frac{1}{d}\sum_{i=1}^{d} \mathbf{\Psi}_i, \quad \mu_T = \frac{1}{d}\sum_{i=1}^{d} T_i, \tag{3.7}$$

in order to center the whole dataset. Then, construct a new motion matrix mean subtracted:

$$\widetilde{\mathbf{M}} = \begin{pmatrix} \mathbf{\Psi}_1 - \mathbf{\Psi}_\circ & T_1 - \mu_T \\ \mathbf{\Psi}_2 - \mathbf{\Psi}_\circ & T_2 - \mu_T \\ \vdots & \vdots \\ \mathbf{\Psi}_d - \mathbf{\Psi}_\circ & T_d - \mu_T \end{pmatrix}.$$

3. Compute the covariance matrix from the elements of $\widetilde{\mathbf{M}}$, such as:

$$\widetilde{\mathbf{S}} = \widetilde{\mathbf{M}}\widetilde{\mathbf{M}}^T. \tag{3.8}$$

In practice, this stage of the PCA algorithm can be optimized by selecting between $\widetilde{\mathbf{M}}\widetilde{\mathbf{M}}^T$ or $\widetilde{\mathbf{M}}^T\widetilde{\mathbf{M}}$, that is, the matrix having the lowest dimension, because these matrices have the same eigenvalues [Jolliffe, 1986], and the remaining $(D - d)$ eigenvalues of $\widetilde{\mathbf{M}}^T\widetilde{\mathbf{M}}$ are equal to zero. Thereby, its remaining eigenvectors can be discarded because they will not improve the data reconstruction in terms of the mean square error (MSE) minimization [Gonzalez and Woods, 2002] (see appendix A.1.1). Therefore, to compute the PCA it is sufficient to find a transformation matrix that maps the motion vectors into an orthogonal space [3]. This transformation matrix is simply the eigenvectors of $\widetilde{\mathbf{S}}$.

4. Compute the eigenvalue decomposition of $\widetilde{\mathbf{S}}$,

$$\widetilde{\mathbf{S}}\alpha = \mathbf{\Lambda}\alpha. \tag{3.9}$$

The columns of the $d \times d$ matrix α are orthonormal eigenvectors of $\widetilde{\mathbf{S}}$, and Λ is a diagonal $d \times d$ matrix containing the non-negative eigenvalues arranged in descending order, so that $\lambda_i \geq \lambda_{i+1}$. $\{\alpha\}$ are the so called Principal Coefficients (PCs) that characterize the motion (i.e., the *latent space*). Then, the so called $d \times D$ Principal Components (PC) are computed as:

---

[3] This transformation is also known as the *Hotelling* transform or the discrete version of the *Karhunen-Loeve* Transform (KLT) [Gonzalez and Woods, 2002]

$$\mathbf{E} = \alpha^{\mathrm{T}}\widetilde{\mathbf{M}}, \tag{3.10}$$

with,

$$\mathbf{E} = \{\mathbf{E}_{\boldsymbol{\Psi}}, \mathbf{E}_T\}. \tag{3.11}$$

Where the $d \times (D-1)$ matrix $\mathbf{E}_{\boldsymbol{\Psi}}$ is referred as the *eigen-motions* and the $d \times 1$ column vector $\mathbf{E}_T$ as the *eigen-durations*.

One important property of PCA deals with the reconstruction of $\widetilde{\mathbf{M}}$ from $\mathbf{E}$. As the λ_i's decrease monotonically, the reconstruction error can be minimized by selecting the m eigenvectors associated with the largest eigenvalues [Jolliffe, 1986, Gonzalez and Woods, 2002]. Note that, m represents the number of eigen-motions required to approximate a learned motion pattern. One common technique used to determine a value for m is known as the accumulated percentage [Jolliffe, 1986], denoted as:

$$\rho(m) = \frac{\sum_{i=1}^{m} \lambda_i}{\sum_{i=1}^{d} \lambda_i}, \tag{3.12}$$

where, $m \leq d$ controls the fraction of the total variance of the training data that is captured by the subspace denoted by $\rho(m)$. So, by using the PC described by Eq. 3.10, any normalized motion $\boldsymbol{\Psi}_i$ can be approximated as:

$$\boldsymbol{\Psi}_i \approxeq \alpha_i \mathbf{E}_{\boldsymbol{\Psi}} + \boldsymbol{\Psi}_{\circ} \tag{3.13}$$

$\mathbf{E}_{\boldsymbol{\Psi}}$ is now $(m \times (D-1))$-dimensional. And a pose, here also referred as an *eigen-pose*, can be computed as a function of the scalar coefficients, α_i ($i = 1, \ldots, m$) and the frame index k:

$$\psi\left(k, \alpha_1, \ldots, \alpha_m\right) \approxeq \alpha_i \mathbf{E}_{\boldsymbol{\Psi}} + \boldsymbol{\Psi}_{\circ}. \tag{3.14}$$

The final step in the PCA framework is used to compute the duration and therefore generate the output motion. For that, we take advantage of the duration added in the motion matrix (Eq. 3.6). Since, Eq. 3.13 gives the best linear approximation of the normalized motion, we use it to compute the best duration. Towards this end, we compute the time T_i corresponding to the motion $\boldsymbol{\Psi}_i$ as follows,

$$T_i \approxeq \alpha_i \mathbf{E}_T + \mu_T \tag{3.15}$$

$\mathbf{E}_T$ is now $(m \times 1)$-dimensional. Once we have T_i, we use its value directly into Eq. 3.3 to establish the final motion size. Finally, the final motion is computed by using Slerp. To simplify the notation, when the situation allows, we use just $\{\alpha\}$ instead of $\{\alpha_i\}$.

Note that, the main advantage of learning motion models instead of pose models, is that, the eigen-motions can be explored for constructing new continuity strategies, as will be seen in section 5. Nevertheless, eigen-poses, as described by Eq. 3.14, are useful to under-constraining model-based inverse kinematics problems, as will be seen in section 4.3. In the next sections, we show how to encapsulate a motion pattern by constructing motion models using the PCA algorithm described above, and show that these models provide useful properties for motion editing.

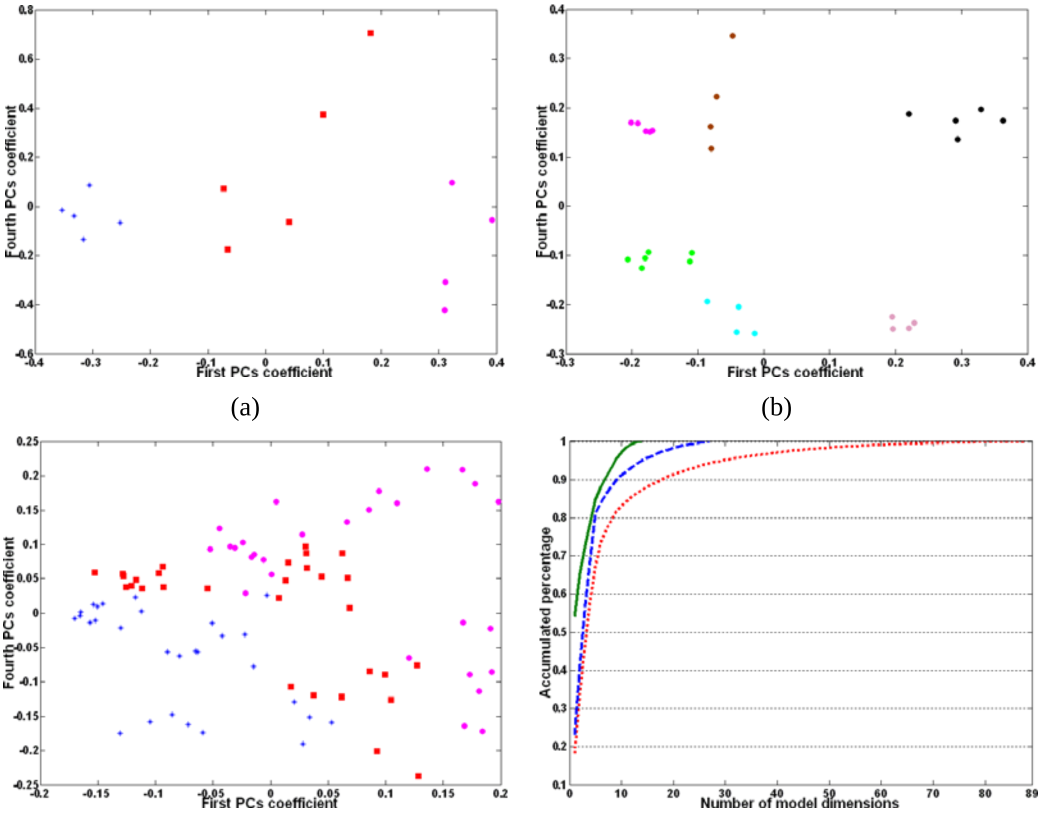
3.3 Model Results

In this section, we show how PCA tends to cluster similar motions and how it generates other motion examples by constructing:

- *Multi-activity* or *heterogeneous* models: i.e., models learned from different behaviors, e.g., both reaching and walking jumps.
- *Single* models: i.e., models learned from a single motion pattern, e.g., only walking jumps. In this case, jumps with different lengths can be considered.
- *Local* models: i.e., specialized single models learned from a specific motion pattern, e.g., golf swings played on a flat ground.

Therefore, to construct a model capable to encapsulate and consequently predict the underlying motion pattern, we first create a mocap database of one or more people performing the same activity many times. In some situations, for example, when we have a reduced dataset, we preprocess the original data creating new samples for increasing data variability. These real or synthetic training data observations are used as input variables for estimating the motion model parameters. For a better understanding of our approach, we propose to describe the process used to learn motion models from walking jumps, reaching and golf swing motions.

1. **Walking Jumps**: we first consider motion models for walking jumps. We used a Vicon$^{\text{TM}}$ optical motion capture system to record the motions of five men and one woman performing walking jumps of 3 lengths ranging from $0.4m$ to $1.2m$, by increments of $0.4m$, for a total of 89 motions. We asked the performers to adapt naturally their speed to be able to execute the requested jump. These data acquire the natural variability of this motion pattern. To virtually animate these motions, we retarget them to a virtual character with $n = 222$ DoFs. Then, we time normalized the data such that each walking jump was represented with $N = 26$ pose samples, which gave motions with $D = 5773$ DoFs. In what follows, we learn two local models, one per subject walking jumps (15 motions) and one per jump at $1.2m$ (28 motions), as well as a single model of walking jumps (89 motions). Figure 3.2 shows the latent space of these database. Figure 3.2(d) depicts $\rho(m)$, in Eq. 3.12, as a function of the eigen-motions for the corresponding models.
2. **Golf Swings**: to learn a motion model for golf swings, we used 16 motions of one subject performing golf swings from the CMU database [CMU, 2009]. Note that, as golf swings are executed in high speeds, editing such movements requires precision during the ball

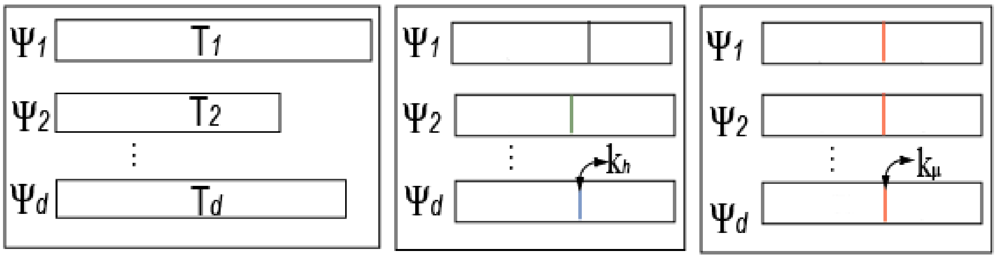


(a) (b)

(c) (d)

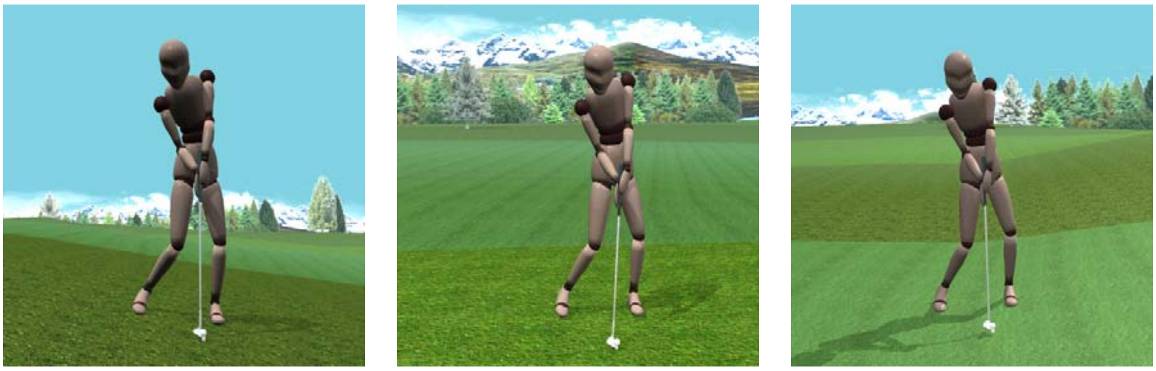
Figure 3.2: Walking jump motion models. Latent space for (a) one male subject at walking jumps ranging from $0.4m$ to $1.2m$; (b) all six subjects at $1.2m$ walking jumps; and (c) all six subjects at walking jumps raging from $0.4m$ to $1.2m$. The data corresponding to different walking jumps (Figure (a) and (c)) are shown in different markers and color, and to different subjects (Figure (b)) are shown in different color. (d) Percentage of the database that can be generated with a given number of eigen-motions for the one subject (solid green), walking jumps at $1.2m$ (dash blue) and all subjects all jumps database(dotted red).

stroke. Consequently, to increase the chances of searching solutions in the space of hits, we linearly time-warp each swing according to the beat frame such that the hit poses are achieved at the same time. A similar approach was used in [Urtasun et al., 2005], but there the authors completely re-sampled four key postures achieved in the beginning, middle and end of the motion. Essentially, the normalization is done in two stages. First, we use the approach described in section 3.2.2 to produce motions with the same duration as illustrated in Figure 3.3(b). In second stage, we preprocess the normalized database searching for the hit frame, K_h and compute the mean hit pose, K_μ. This information is used to subdivide the swing in two parts, such that, the first one from the beginning of the swing to the hit frame (i.e., $[1, ..., K_\mu]$) and second from the hit frame $+1$ to the end of the swing (i.e., $[K_{\mu+1}, ..., N]$). Then, the final time-warped training motions, as illustrated in Figure. 3.3(c), are used to create a swing basis. Note that, what is aligned is the hit frame and not the beat position between the club head and the ball, which can be everywhere in the 3D space.



(a) (b) (c)

Figure 3.3: Golf swings normalization process. (a) Illustrates the swings before the duration normalization. T_i $(i = 1, \ldots, d)$ is the motion duration and the size of the rectangles represents different durations. (b) The motions have the same duration, but the hit frame, K_h, is not time aligned. (c) The motions have the same duration and the hit frame is now time aligned at key time, K_μ.

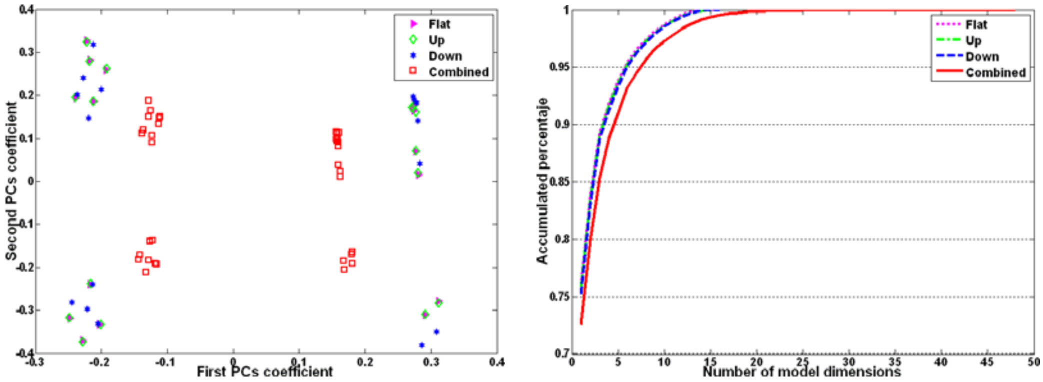


(a) (b) (c)

Figure 3.4: Golf swing motions. (a)(c) Golf swings executed on the down and up slopes synthetically produced from the flat ground motion (b).

Each swing is therefore retarget to a virtual character with $n = 93$ DoFs and each normalized swing is represented by $N = 132$ pose samples, which give motions with $D = 12277$ DoFs. To provide user manipulation of the golf club, we attached an end-effector near the club head to drive the hit to another position. In what follows, we learned motion models from swings played on three different types of ground: 1) one for flat ground, for a total of 16 motion; 2) one for up slopes, for angles ranging from $0.5°$ to $5.0°$ (anti-clockwise), by increments of $0.5°$, for a total of 16 motions; 3) one for down slopes, for angles ranging from $0.5°$ to $5.0°$ (clockwise), by increments of $0.5°$, for a total of 16 motions; and 4) one for the combined down, flat and up grounds.

The golf swings executed on the down and up slopes - see Figures. 3.4(a) and 3.4(c) - were synthetically produced from the flat ground motions by using a motion editing system [Callennec and Boulic, 2006]. Note that, the synthetic motions were generated by adjusting just the position of the feet according to the ground inclination, because by adjusting the position of the golf club head demonstrated to add discontinuities. Figure 3.5 shows the latent space and the accumulated percentage of these models. The process used to recover golf swings is similar to the other motion patterns. The only difference is that, we keep a vector containing the information of the hit frames to recover the hit alignment according to its initial position in the input motion.



(a) (b)

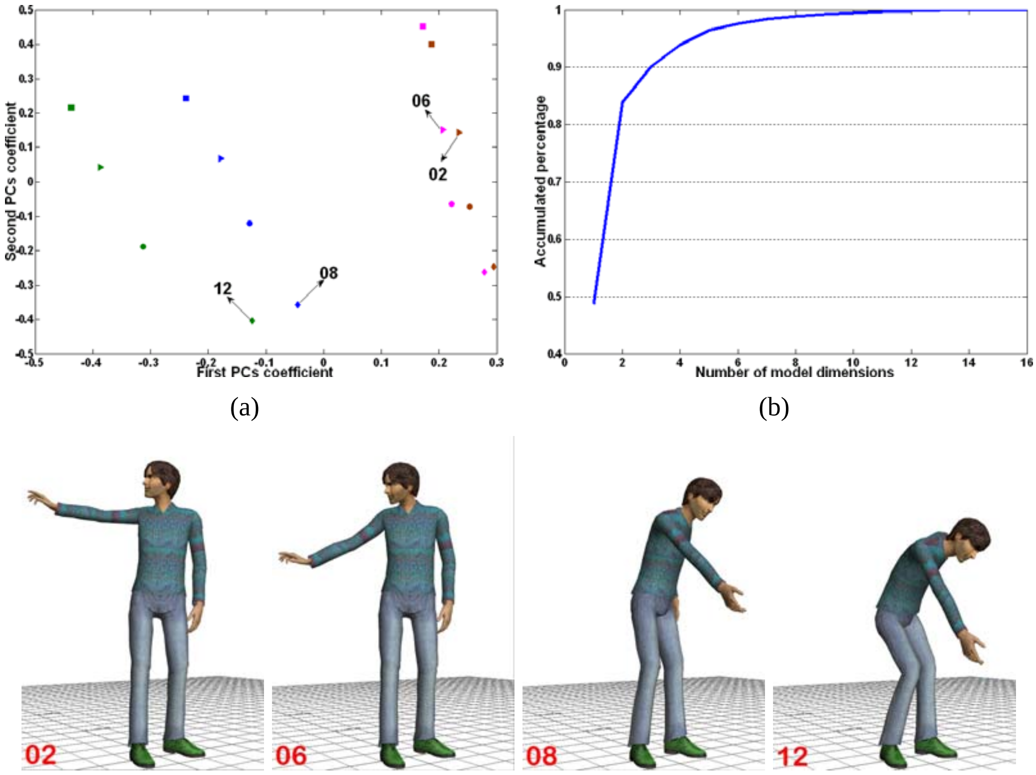
Figure 3.5: Golf Swings motion models. (a) Latent space. The data corresponding to different golf swing models are shown in different color and shapes. (b) Percentage of the database that can be generated with a given number of eigen-motions.

3. **Reaching**: we used the same approach to learn a full-body goal directed motion (i.e. reaching) where locomotion is not necessary. Toward this end, we used $d = 16$ reaching motions performed by one subject provided by [VAL, 2009]. We also used the same character with $n = 222$ DoFs. Then, each reaching was time normalized and represented by $N = 50$ pose samples, which gave motions with $D = 11101$ DoFs. We learned one local model for reaching. Figure 3.6 shows the latent space and the accumulated percentage of this database.

In the sections below, we show how these models cluster similar motions (section 3.3.1), and how the latent space can be explored to generate new motions (section 3.3.2).

3.3.1 Motion Clustering

PCA provides a compact representation of the data describing each sample by a small set of measurements (i.e., the principal coefficients). The latent space retains significant information of the original data samples, which can be used for grouping similar motions into clusters. Cluster analysis or data segmentation aim to grouping or segmenting a collection of



(a) (b)

(c)

Figure 3.6: Reaching motion model. (a) Latent space. (b) Percentage of the database that can be generated with a given number of eigen-motions. (c) Final posture of four reaching motions, in the order they appear in the database, used to illustrate the PCA property clustering similar motions in the latent space.

samples into subsets or clusters, such that those within each cluster are more closely related to one another than samples assigned to different clusters [Jain et al., 1999, Verbeek, 2004, Hastie et al., 2009]. In motion editing, we should preserve some characteristics of the original motion as well as add new features to it [Gleicher, 2001]. In this sense, we aim of statistically preserving the motion's characteristics by performing optimizations in the latent space, because this space encapsulates motion styles and actions of the underlying motion pattern being edited.

In this context, it is interesting to show that the motion models learned previously are capable to grouping similar patterns according to the subjects' styles and actions. Figures. 3.2(a), 3.2(b) and 3.2(c) show the latent space for the walking jumps models projected onto the first and fourth principal coefficients and each point in the clusters represents a full-body motion. We can verify that, the one subject motion model, Figure 3.2(a), describes the inter-action variation for the three different jumps sizes, and the motion model for the jump at $1.2m$, Figure 3.2(b), describes the inter-style variation of each subject. On the contrary, the motion model learned from the complete dataset shows a smaller inter-style and inter-action variation, demonstrating a denser representation. Note that, this model can suffer a further clustering decomposition, for example, by using a hierarchical clustering technique based on a divisive (i.e., top down) strategy [Hastie et al., 2009]. Figure 3.2(d) shows the accumulated percentage. We can see that with a small database the model reaches a high percentage (e.g., 95%) close to the maximum number of dimensions compared to the large dataset model.

The motion models learned from the golf swings also exhibited a good inter-style variation. However, we would not expect a good inter-action variation because the up and down slopes swings were produced synthetically from the flat ground ones. Figure3.5(a) shows the latent space for the golf swing models projected onto the first and second principal coefficients. Note that, the accumulated percentage, shown in Figure 3.5(b), is basically the same for the up and down slopes swing motions. Figure 3.6(a) shows the reaching motions projected onto first and second principal coefficients. Note that, similar reaching motions, here described by the final pose depicted in Figure 3.6(c), are cluster together showing the inter-action variation.

As noticed, the PCA extracts and encapsulates motion similarities into a compact set of coefficients. Such representation, in fewer dimensions, is advantageous because it increases the efficiency and performance of motion editing [Grochow et al., 2004, Shin and Lee, 2006, Carvalho et al., 2007]. Accordingly, similarities have been illustrated by projecting the appropriate set of principal coefficients in two dimensions. Frequently, the first two PCs are projected to illustrate the division of the clusters because they carry more information in terms data variance [Gonzalez and Woods, 2002]. However, the assumption that the first two PCs are the best to represent clusters separation is not completely true, because the feature information that improves the intra-cluster distances may be in the less representative coefficients. Choose therefore the best set of coefficients that improves clustering is a classical problem in pattern recognition known as feature selection [Jain et al., 1999]. Consequently, for the experiments reported in this section, we simply chose a subset of two PCs that best represented the intra-cluster division to show the PCA clustering capabilities.

3.3.2 Motion Encapsulation: The Purpose of Model Generalization

We refer to the process of learning the underlying motion pattern as *motion encapsulation*. In some cases, we need to isolate the core behavior, to reduce or even to eliminate the ambiguity of having multiple solutions satisfying the same set of constraints [Chai and Hodgins, 2005]. In other cases, the aim is not to isolate the core pattern completely, but to gradually increase the behavioral space to either satisfy different constraint configurations or edit distinct motion patterns. The problem may be as simple as learning specialized motion models, or as complex as generalizing the latent space by learning multi-activity models. We believe that, the purpose of model generalization is mainly related with the application. For example, the animator may want to adapt a walking jump motion to reach an object in the middle of the jump. In this case, a model learned from walking jumps and reaching motions needs to be constructed.

Given a database containing motions with different behaviors and a motion to be edited, the animator must then decide which of the behaviors he/she has to select to construct a motion model. This is the first step of the editing process. In order to provide the animators with a first

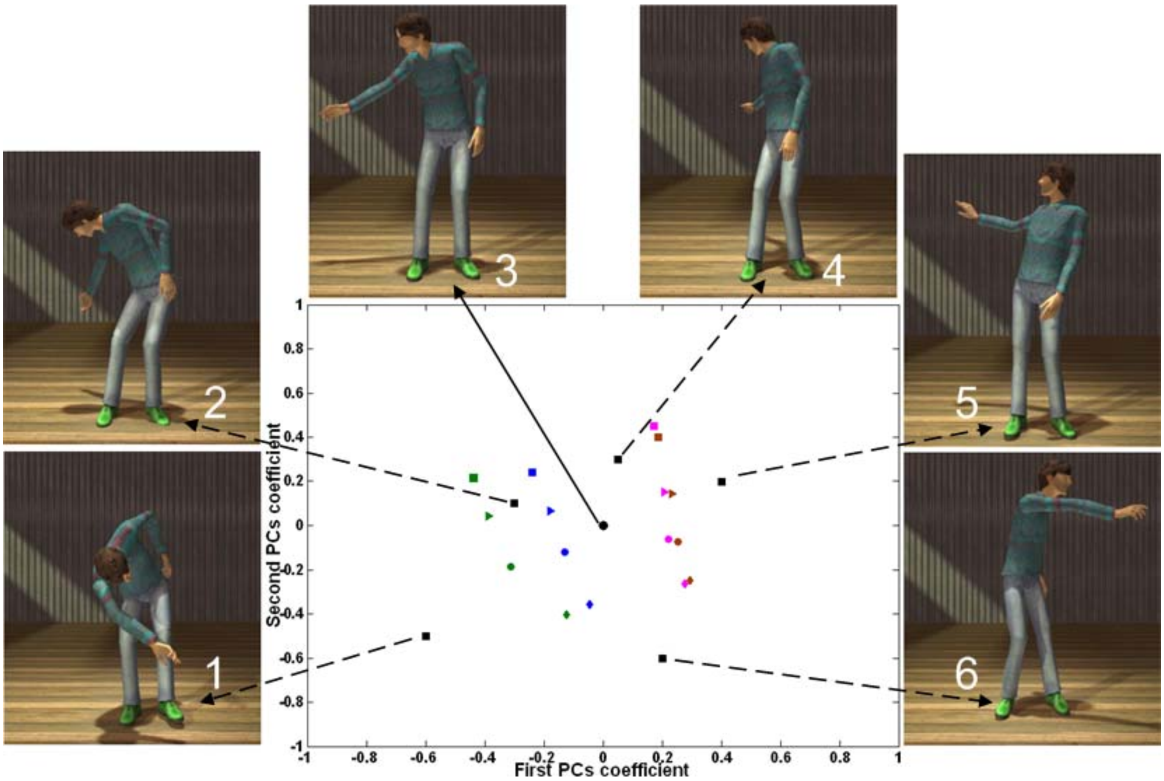
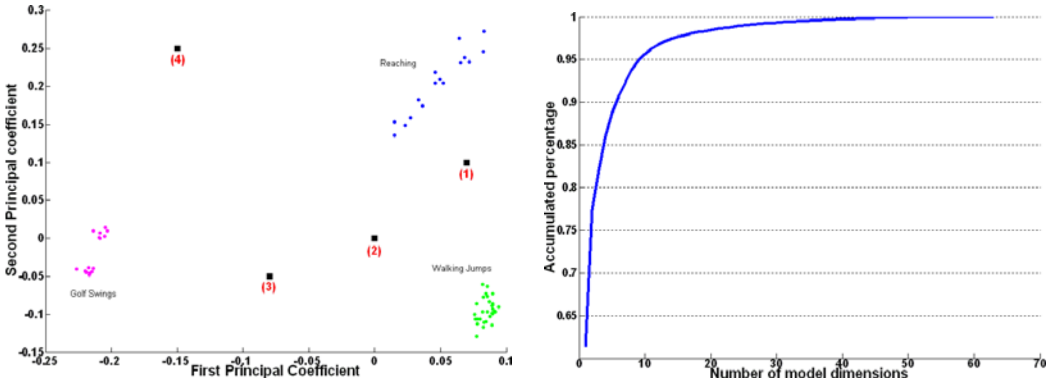


Figure 3.7: Model-based motion synthesis of a database of reaching motions. The latent space is shown in the middle. Random samples are represented by black squares, excepted the mean motion (posture 3 black circle), which represents the origin of this space.

insight of how to select an appropriate motion behavior to learn a specialized model, or how to combine motion patterns to construct a multi-activity model. Accordingly, we explore how the PCA motion model generalizes new motions from random samples in the latent space, this approach is similar to [Urtasun et al., 2006]. Therefore, we experiment with the latent spaces shown in Figure 3.7 created from a database of reaching motions (i.e., specialized model) and in Figure 3.8 created from a database composed of golf swings, walking jumps at $0.8m$ and reaching motions (i.e., multi-activity model).

In the specialized latent space the encapsulation of the core pattern is satisfied in a much greater extent compared to the more generalized space. In other words, we found that random samples outside and between motion clusters all produced genuine reaching motions. Figure 3.7 shows the final reaching posture of random samples computed from the first two eigenmotions for which the remaining coefficients, $\alpha_i = 0$, for $3 \leq i \leq d$. In addition, this model demonstrate to produce plausible motions for samples moving both away from the center and the training data producing pleasing motions, see samples 1 and 6 in Figure 3.7. Nevertheless, we cannot guarantee that all the samples between data points in the latent space will produce natural motions. For example, if the database have both a small number of samples and a high data variability, may we can find regions between data samples that produce motions violating joint limits.

In the multi-activity latent space, we observed that some regions of the sub-space spanned by the first two eigen-motions of the multi-activity model (Figure 3.8) generated physically impossible motions, see Figure 3.9(d). Nevertheless, there are regions of the subspace between motion patterns (e.g., reaching and walking jumps) that do correspond to plausible mixed motions, see Figure 3.9(a). These regions may facilitate the satisfaction of user-specified constraints, for example, in situations where the animator wishes to adapt a walking jumping motion to reach an obstacle in the middle of the jump. These constraints may cannot be easily enforced in a specialized latent space.



(a) (b)

Figure 3.8: Latent space of the multi-activity database composed of golf swings, walking jumps at $0.8m$ and reaching motions. The four random samples that generate the motions of Figure 3.9 are represented by black squares.

From a practical point of view, the latent space can be used to generate new motion samples in an easy and fast way, for example, we could construct a user interface capable of automatically mapping points from the latent space directly into the joint space, by simply using the mouse pointer for providing input data samples [Shin and Lee, 2006]. Despite this being a simple motion generation strategy, the mapping in this way cannot easily handle user-specified constraints (i.e., Cartesian constraints) limiting user needs. One step is this direction could be the combination of motion models with constraint-based optimization. Essentially, when specialized models are combined with optimization techniques the optimizer searches for solutions directly in the core pattern space leading to faster optimizations, as will be seen in section 4.6. Nevertheless, its great advantage is also its major drawback: single models cannot be used to constraint a behavior other than the encapsulated one [Safonova et al., 2004, Carvalho et al., 2007, Chai and Hodgins, 2007]. On the other hand, multi-activity models are more general and can handle a large space of solutions favoring the editing of different motion patterns, but at a slower convergence rate compared to specialized motion models. Therefore, the need of model generalization is related with the application and with system performance. Hence, a tradeoff between system performance and the generalization of the solution space has to be kept when the animatior creates a model.

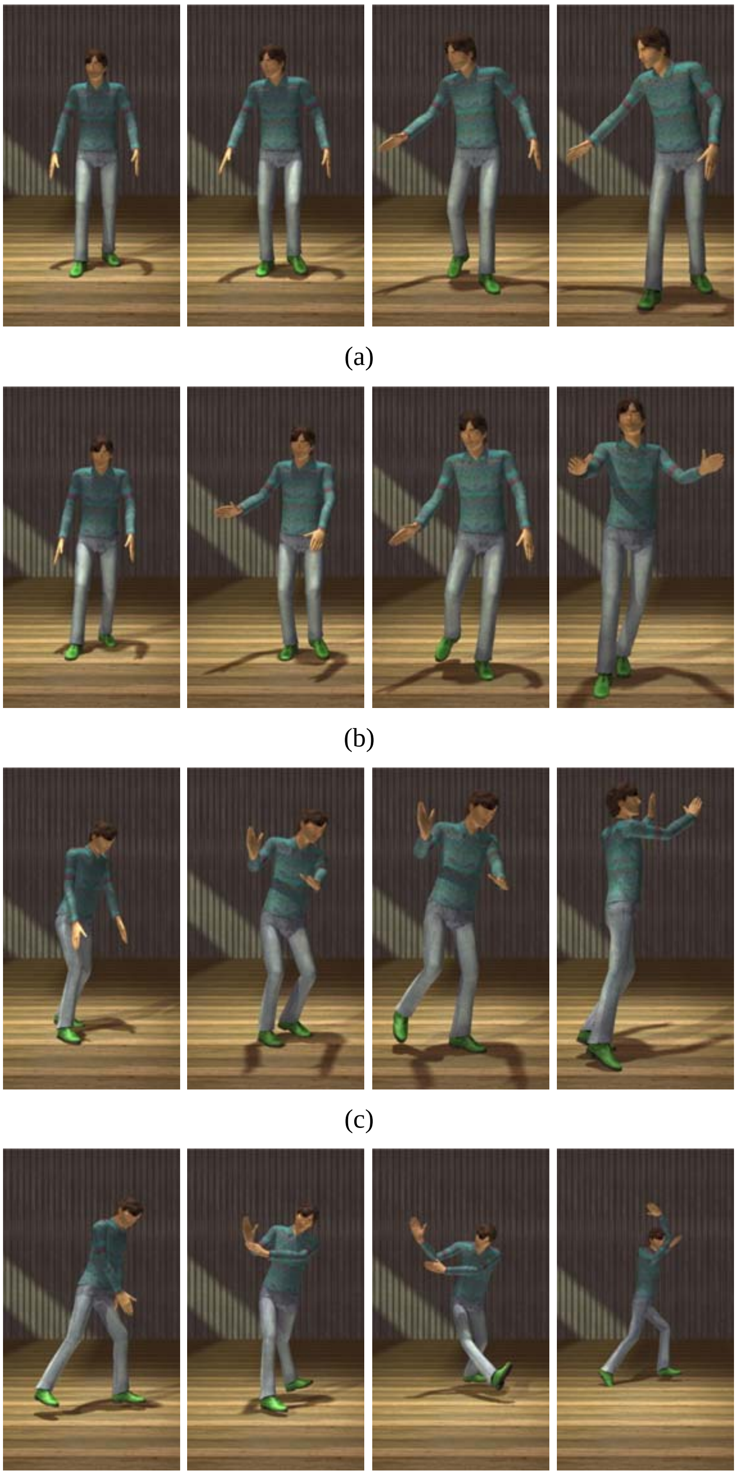
We have observed the encapsulation capabilities of both specialized and more general motion models, by generating new data samples from random points in the latent space. The animator has now the perception to design a variety of motion models according to the task in hands. In practical terms, the animator should create specialized models when he/she wants to edit an animation without the need of generating a mixed motion. More investigation to verify the usefulness of specialized and more general motion models will be carried out in section 6.

3.4 Conclusion

In this chapter, we have described a simple and efficient strategy to parameterize the character motion, for estimating the PCA motion models. As each subject performs the same movement with some variability in duration, the training motions have to be first time-normalized before applying PCA. In some applications, the user may need the motion solution with the correct duration, to provide this feature, we added the duration in the motion vector letting the learning algorithm to be sensitive to it. So, given the latent space the duration can be estimated.

The PCA motion models demonstrated an efficient clustering representation grouping similar patterns according to the subjects' styles and actions. As shown, the PCA motion generation schema maps motions from the latent space directly to the joint space. As a result, new natural-looking motions can be easily generated by manipulating the latent space. However, the major limitation of this synthesis approach is that user-specified constraints cannot be easily satisfied.

In the next chapter, we demonstrate how to explore the motion models presented in this section, by building an optimization framework capable to handle user-specified constraints directly in the latent space. As a result, new principal coefficients are generated handling user needs.



(a)

(b)

(c)

(d)

Figure 3.9: Motions from random samples of the multi-activity database. The PCs of the motions shown in this figure are depicted in Figure 3.8, the points 1 to 4 correspond to the animations (a) to (d). (a) (b) Two motions that resemble a combination of walking jump and reaching. (c) A motion that resembles a combination of walking jump and golf swing. (d) Physically impossible motion.

Chapter 4

Data-Driven Constraint-Based Optimization Framework

4.1 Introduction

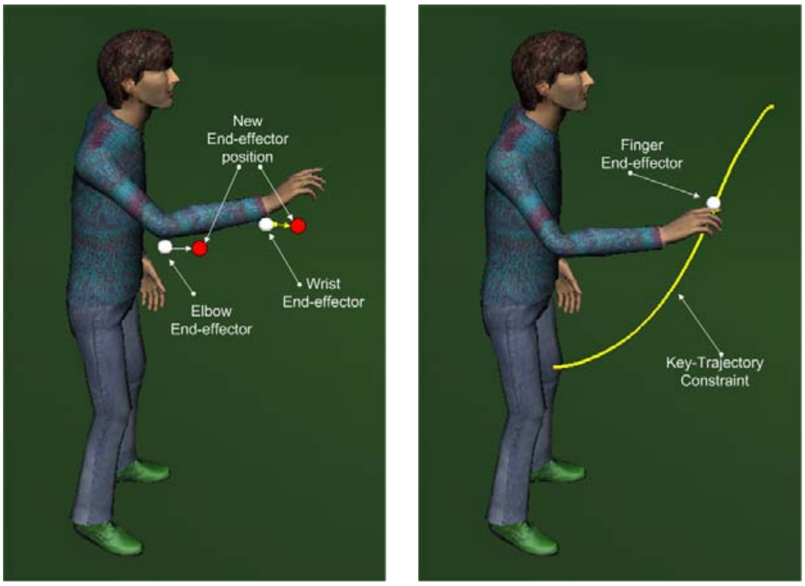
This chapter presents a low-dimensional prioritized inverse kinematics (LPIK) solver based on the combination of the Resolved Motion Rate Control (RMRC) [Whitney, 1969] and Principal Component Analysis (PCA) [Jolliffe, 1986]. The goal is to solve a constraint-based optimization problem within the latent space. The issues related to this framework are discussed in the following sections.

4.2 Constraints

Constraint-based motion editing techniques require the user to specify constraints over an input animation to carry out motion adaptations. Providing interactive ways to manage constraints allow users to easily and rapidly modify preexisting human animations. In particular, geometric constraints, such as the position of an end-effector in the three-dimensional space or the trajectory of an end-effector, are more intuitive for interactive manipulation because the user can specify a goal just by dragging an end-effector to a new position. We equip animators with two types of constraints: key-frame and key-trajectory constraints as illustrated in Figure 4.1, both are firstly created in the joint space and then mapped into the latent space.

4.2.1 Key-frame constraints

In this type of constraint, end-effectors are attached directly on the character's body and dragged to knew positions. Key-frame constraints are pure positional constraints, that is, the effector does not follow any specific trajectory. Figure 4.1(a) depicts this type of constraint. A key-frame constraint is simply represented by a three-dimensional point x expressed in the so-called



(a) Key-frame constraint (b) key-trajectory constraint

Figure 4.1: Example of user-specified constraints. (a) key-frame constraints for modifying a preexisting full-body reaching animation. (b) Key-trajectory constraints represented by a spline curve describing a 3D trajectory of an end-effector across the motion for guiding the reach.

task space. In the case of multiple end-effectors applied at the same key-time, the valency of the task can increase and integrate as many independent scalar or vectorial equations, simply by “piling”them.

The animator can use this type of constraints to edit the whole motion by constraining either one key-frame or multiple key-frames at different key times. Essentially, key-frame constraints are useful to edit motions that need a great precision in time. For example, the user can adapt a walking jump to perform a greater jump by attaching one constraint on the root, at the key frame where the character starts the jump, and move the constraint forward in the jump direction to generate the desire animation. In another example, the user may want to modify a reaching motion to allow that the character reaches two objects in two different instants of the motion. In this case, he/she could edit two character poses at different key times.

Note that, constraints such as the position of the center of mass, which are normally used to control static balance, are not modeled because the solution is achieved within the space of physically balanced motions [Safonova and Hodgins, 2005]. Likewise, it is not necessary to explicitly provide for constraints on velocities because, by construction, they are intrinsically encapsulated in the model parameters [Glardon et al., 2004a]. That is one of the main advantages of learning motions instead of poses.

4.2.2 Key-trajectory constraints

In this section, we summarize how end-effector trajectories are constructed from a set of 3D *constraint points*. This method was first introduced in [Callennec and Boulic, 2006]. However, as will be seen in section 5.2, we have proposed a more efficient strategy for enforcing the motion continuity when key-trajectories are used by the animator.

Key-trajectory constraints allow the animator to edit a motion by specifying end-effectors across the whole motion as continuous trajectories. The aim of this constraint formulation is to reduce the work of the animator of manually specifying complete end-effector trajectories. For example, suppose an animator wants to edit a reaching motion, then instead of specifying a set of points of the trajectory of a hand, which can be cumbersome, he/she can define a high-level constraint specifying important locations this hand should pass through and its whole trajectory is then automatically generated. To provide this feature, we use an efficient interpolation method such that given a set of three-dimensional constraint points the system automatically generates the entire trajectory of an end-effector. Furthermore, the user can also combine multiple trajectories from different end-effectors attached at different key times to carry out motion adaptations. The end-effector's trajectory is represented as a pair of curves $(\mathbf{S}(u), \mathbf{T}(t))$ with u the parameter of curve $\mathbf{S}$ and t the parameter of curve $\mathbf{T}$. $\mathbf{S}(u)$ defines the end-effector's trajectory in space. $\mathbf{T}(t)$ is used to control the timing of $\mathbf{S}(u)$. The parameter t then represents the current time in the animation. These curves are automatically constructed given a set of 3D *constraint points* specified by the animator.

A constraint point is equivalent to a control point of a spline. The difference is that, it also handles timing information to ensure that the end-effector passes through the specified locations at the requested period of time. The time period contains the constraint points. A constraint point $\mathbf{P}$ $(\mathbf{P}_{loc}, \mathbf{P}_{in}, \mathbf{P}_{out})$ is therefore defined as:

$\mathbf{P}_{loc}$: specifies a 3D location that is used to interpolate the final trajectory of the end-effector at each time. It is set by the animator in the world space to provide user manipulation on the control points. This three dimensional location is used as a control point for the trajectory curve $\mathbf{S}(u)$.

$\mathbf{P}_{in}$: specifies the starting time at which the end-effector should reach $\mathbf{P}_{loc}$. This time is used as a control point for the time curve $\mathbf{T}(t)$.

$\mathbf{P}_{out}$: specifies the ending time at which the end-effector should depart from $\mathbf{P}_{loc}$. This time is used as a control point for the time curve $\mathbf{T}(t)$.

The time interval $[\mathbf{P}_{in}, \mathbf{P}_{out}]$ defines a duration for which an end-effector might remain stationary at the specified location $\mathbf{P}_{loc}$. Note that, the animator only needs to define the timing of the end-effector trajectory and the constraint points the end-effector has to pass through. The trajectory and time curves are then automatically computed.

The trajectory of an end-effector is therefore represented as a *Kochanek-Bartels* spline curve [Kochanek and Bartels, 1984]. This class of interpolating splines (i.e., the curve pass through its control points or knots) is based on cubic Hermite basis functions with *tension*, *bias* and *continuity* parameters defined to change the behavior of the tangent. These three parameters are useful to explicitly change (or even break) the continuity of the curve at the control point. Essentially, the tension is used to change the length of the tangent vector, the bias its direction and the continuity the sharpness in between tangents (see [Kochanek and Bartels, 1984] for further details about these parameters). More precisely, given the tension t, the bias b and the continuity c parameters, the source $\mathbf{st}_l$ and destination $\mathbf{dt}_l$ tangents are computed as follows for a control point $\mathbf{P}_l$:

$$\mathbf{st}_l = \frac{(1-t)(1-c)(1+b)}{2}(\mathbf{P}_l - \mathbf{P}_{l-1}) + \frac{(1-t)(1+c)(1-b)}{2}(\mathbf{P}_{l+1} - \mathbf{P}_l) \tag{4.1}$$

$$\mathbf{dt}_l = \frac{(1-t)(1+c)(1+b)}{2}(\mathbf{P}_l - \mathbf{P}_{l-1}) + \frac{(1-t)(1-c)(1-b)}{2}(\mathbf{P}_{l+1} - \mathbf{P}_l), \tag{4.2}$$

The first and the last tangents are set to the velocity of the end-effector on the initial trajectory. In this way, there is no discontinuity when going from the initial trajectory to the deformed one or when going back to the initial one. This formulation enables the addition of sharp corners on the end-effector's trajectory, by setting the tension parameter of a specific control point to 1 when required.

In order to reparameterize $\mathbf{S}(u)$ such that the interval of time $[\mathbf{P}_{in}, \mathbf{P}_{out}]$ corresponds to the same value of parameter u. The time curve has to be handled. Essentially, The time curve $\mathbf{T}(t)$ is defined as an increasing piecewise linear function, which establishes the correspondence between each time of the animation t and the parameter u of $\mathbf{S}(u)$. Given a time t, the corresponding three-dimensional end-effector's position $\mathbf{P}$ in the final trajectory is then defined as:

$$\mathbf{P} = \mathbf{S}(\mathbf{T}(t)) \tag{4.3}$$

The animator has therefore three trajectory modes to carry out motion adaptations:

1. **Absolute**: this mode is useful whenever the animator needs to entirely reshape the trajectory of an end-effector. So, the initial trajectory of the end-effector is not taken into account.

2. **Relative**: in this mode the initial end-effector's trajectory is considered. As a result, the final trajectory that is smoothly deformed while retaining the global shape of the initial one as much as possible. This type of trajectory is useful in cases when the animator only wants to add an offset to the initial trajectory.

3. **Relative with Condensation**: this mode uses basically the same method as for the relative mode. The basic difference is that, as the reference trajectory is moving while the end-effector should stay stationary, the trajectory curve is readjusted such that the final trajectory pauses at required points even if the reference one is moving.

Note that, given these modes, the animator can choose whether he/she wants to take into account the initial end-effector's trajectory or not.

4.3 Low-Dimensional Inverse Kinematics (LIK)

For a better understanding of our method, we start this section recalling a standard inverse kinematics technique based on the Jacobian. In practice, if we consider the pose of a virtual character as a n-dimensional vector (joint space) as shown in Eq. 3.2 and the position of the end-effector, x, as a p-dimensional vector (task space), the IK function can be defined as:

$$\mathbf{\Theta} = f^{-1}(x). \tag{4.4}$$

For a redundant manipulator, i.e., $n > p$, the problem is always ill-posed (i.e., there is no unique solution) [Craig, 1986]. Hence, IK algorithms should determine just one solution to Eq. 4.4 given many possibilities. In the presence of redundant manipulators IK can be formulated as the solution of a nonlinear equation:

$$x(\mathbf{\Theta}) = \mathbf{g}, \tag{4.5}$$

where, $x(\Theta)$ and g are p-dimensional vectors expressed in the task space. The task function $x(\Theta)$ represents the position or orientation of an end-effector frame while g is the goal to be reached. Hence, the problem may be reformulated as an optimization problem that minimizes the residual error:

$$e(\mathbf{\Theta}) = \|x(\mathbf{\Theta}) - \mathbf{g}\| \tag{4.6}$$

The Euclidean norm is used here, and is supposed to be meaningful for $x(\mathbf{\Theta})$.

Since Eq. 4.5 is non-linear in general, it cannot be solved by simple inversion of the function $x(\mathbf{\Theta})$, as shown in Eq. 4.4. One approach to solve IK problems, with optimization criteria, is the well known Resolved Motion Rate Control (RMRC) proposed by [Whitney, 1969]. As most numerical methods, it is an iterative procedure based on a linearization of the constraint equation about the current state $\mathbf{\Theta}$, that results in a Jacobian matrix. By inversion of the Jacobian matrix, the resulting set of linear equations can be solved for an increment $\Delta\mathbf{\Theta}$. Then, a step in this direction is taken to a new configuration that approaches a solution of the task equation $\mathbf{\Theta}$. By iteratively repeating the process, the system converges towards a solution that either satisfies the task equation, or that locally minimizes the residual error when there is no exact solution. Hence, once a direction has been computed, the final pose can be found adding $\Delta\mathbf{\Theta}$ with respect to $\mathbf{\Theta}$:

$$\mathbf{\Theta}^{v} = \mathbf{\Theta} + \Delta\mathbf{\Theta}, \tag{4.7}$$

where, $\mathbf{\Theta}^{v}$ is the new state vector.

To use the RMRC technique, we need to compute the Jacobian matrix of the forward kinematics function, $x = f(\mathbf{\Theta})$, and invert it. In practice, by using a first-order (linear) approximation of the task function, $x(\mathbf{\Theta})$, given by the Taylor series expansion about the current configuration, **Θ**,

$$x(\mathbf{\Theta} + \Delta\mathbf{\Theta}) = x(\mathbf{\Theta}) + J(\mathbf{\Theta})\Delta\mathbf{\Theta} + ... \tag{4.8}$$

and keeping just the linear term of Eq. 4.8 results in:

$$\Delta x \approxeq J(\mathbf{\Theta})\Delta\mathbf{\Theta} \tag{4.9}$$

where, $J(\mathbf{\Theta}) = \partial x / \partial \mathbf{\Theta}$ is the joint space Jacobian matrix, $\Delta x = \mathrm{g} - x(\mathbf{\Theta})$ is a known desired task increment, and $\Delta\mathbf{\Theta}$ is the unknown increment of joint coordinates. The computation of $J(\mathbf{\Theta})$ for exponential maps is summarized in [Baerlocher, 2001]. Once, the Jacobian matrix $J(\mathbf{\Theta})$ and a task increment Δx have been computed, Eq. 4.9 can be solved in order to obtain an increment $\Delta\mathbf{\Theta}$. In the case that, the Jacobian is square $(n = p)$ and non-singular, the solution is unique and can be simple: $J(\mathbf{\Theta})^{-1}\Delta x$. However, in general, the number of constraints p is usually smaller than the number of degrees of freedom n, i.e., $p < n$ leading to a under-constrained configuration, where the classical matrix inverse is not applicable, and the solution is no longer unique. The opposite case, i.e., $p > n$, leads to a over-constrained configuration: in that case, no exact solution exist (i.e., there is no guarantee that all constraints will be satisfied), and the residual $J(\mathbf{\Theta})\Delta\mathbf{\Theta} - \Delta x$ vector cannot be zero. In both cases, additional criteria must be defined in order to select an optimal solution. In the under-constrained case, the best solution must be selected among the valid ones, while in the over-constrained case, the solution that minimizes a reasonable error is selected.

The particular solution is usually based on the least-squares inverse $J(\mathbf{\Theta})^{\dagger}$ of $J(\mathbf{\Theta})$:

$$\Delta\mathbf{\Theta} = J(\mathbf{\Theta})^{\dagger}\Delta x. \tag{4.10}$$

The least-squares inverse, sometimes called Moore-Penrose inverse or pseudoinverse, is a generalized inverse [Boullion and Odell, 1971, Nashed, 1976] that satisfies least-squares criteria in both under- and over-determined situations. However, this solution has one major drawback, in the proximity of singularities the problem becomes ill-conditioned: in an attempt to precisely minimize the residual error, the norm of the resulting least-squares solution may tend to infinity [Maciejewski, 1990]. This is unacceptable in practice since it violates the small increments hypothesis of Eq. 4.10. In such case, the damped pseudo-inverse solution can be used instead [Maciejewski, 1990]:

$$\Delta\mathbf{\Theta} = J(\mathbf{\Theta})^{\dagger^{\xi}}\Delta x \tag{4.11}$$

where, $J(\mathbf{\Theta})^{\dagger^{\xi}}$ is the damped pseudo-inverse of the Jacobian matrix $J(\mathbf{\Theta})$. The damped factor, ξ, is added to prevent Jacobian's singularities and to stabilize IK solutions. Eq. 4.11 is solved by using Singular Value Decomposition (SVD) exploring the null space of the Jacobian matrix [Press et al., 1992]. Additional information to compute the pseudo-inverse using the SVD technique can be found in [Maciejewski, 1990, Baerlocher, 2001].

A difficult problem with the damped least-squares technique is the evaluation of the optimum damping factor. When far from singularities it should be zero to provide the best possible tracking, and high enough to alleviate the unwanted oscillations when close to a singularity. However, a too high value also results in poor tracking accuracy and also degrades convergence performance. A common method is to set a bound b_{max} on the norm of the solution:

$$\|J(\mathbf{\Theta})^{\dagger^{\xi}}\Delta x\| \leq b_{max}$$

We exploit the approach proposed by [Maciejewski and Klein, 1988] to compute an appropriate damping value. In this strategy, the value of the minimum singular value σ_{min} (or an estimate of it), computed with SVD, is sufficient to compute an appropriate, albeit sub-optimal, damping factor. This approach is summarized below:

$$\xi = \begin{cases} a/2 & \text{if } \sigma_{min} \leq a/2 \\ \sqrt{\sigma_{min}(a - \sigma_{min})} & \text{if } a/2 \leq \sigma_{min} \leq a \\ 0 & \text{if } a \leq \sigma_{min} \end{cases}$$

where, $a = \|\Delta x\|/b_{max}$.

The joint space Jacobian has two major drawbacks. The first one is that, its dimension is directly dependent of the character's degrees of freedom, and we know that the higher the DoF is, the slower the pseudoinverse computation is. This problem can be alleviated by decreasing character's DoF or by constructing smaller kinematics chains by dividing the skeleton into groups, where each of which being an individual kinematic chain [Kulpa et al., 2005]. Both approaches have the advantage to ease the computation of a solution. However, the first requires simple skeletons leading to less realistic characters, and the second can break the synergy between groups producing unrealistic movements. The second one is a consequence of the high-dimensionally of the solution space, we have therefore many possible solutions satisfying the same set of constraints. Furthermore, another disadvantage of goal-directed methods is the frequent lack of naturalness when it comes to reproduce human activities.

In order to improve IK solutions and alleviate the issues mentioned previously, the human behavior is directly integrated in the optimization process to obtain more realistic and natural-looking solutions. As a result, we allow IK to compute solutions directly into the low-dimensional motion space, by exploring the latent space of a specific motion pattern. As will be shown, this approach also reduces both the Jacobian's size improving performance of the pseudoinverse computation. As an alternative that requires much less computation time to compute the pose increment, we explore the eigen-motion space described in Eq. 3.10. Nevertheless, instead of taking into account the complete eigen-motion vector, which has a much higher dimension compared to $J(\mathbf{\Theta})$, we consider the portion corresponding to a specific pose, $\mathbf{\Theta}_k$, as shown in Eq. 3.14. Accordingly, optimizing with respect to the eigen-motions means that the pose increment is computed in the latent space rather than the joint space. In practice, to compute the pose increment in the latent space we need to solve the following equation:

$$\Delta\alpha = J(\alpha)^{\dagger^{\xi}}\Delta x \tag{4.12}$$

where, $J(\alpha)^{\dagger\xi}$ is the pseudo-inverse of the Principal Coefficients Jacobian $J(\alpha) = \partial x/\partial\alpha$, which relates a variation Δx in the Cartesian space with a variation $\Delta\alpha$ in the latent space. The computation of $J(\alpha)$ can be easily carried out with the chain rule:

$$\left[\frac{\partial x_l}{\partial \alpha_i}\right] = \left[\frac{\partial x_l}{\partial \theta_j}\right] \cdot \left[\frac{\partial \theta_j}{\partial \alpha_i}\right] \tag{4.13}$$

The derivatives of $\left[\frac{\partial x_l}{\partial \theta_j}\right]$ can be fast and easily computed [Baerlocher, 2001]. Besides, θ_j are linear combinations of the Principal Components $\mathbf{E}$, so $\left[\frac{\partial \theta_j}{\partial \alpha_i}\right]$ is simply extracted from the jth coordinate of $\mathbf{E}$ for the pose at key time k. We denote this Jacobian as $J(\mathbf{E})_k$ its size is $(n \times m)$, to be compared to the much larger size $(D \times m)$ of the complete PC basis. Recalling that the Principal Components are estimated off-line and that they remain constant for a given latent space. As pointed before, the final $(p \times m)$ matrix $J(\alpha)$ is much smaller than the $(p \times n)$ matrix $J(\mathbf{\Theta})$, for a simple illustration consider Figure 4.2. In addition, because of the way $J(\alpha)$ is constructed, the computation of the pseudoinverse is not affected by the skeleton's degrees of freedom, letting the user utilize more complex characters without worrying with a decrease in system performance. So, once we have the new increment $\Delta\alpha$ the new pose can be computed in two steps. First, $\Delta\alpha$ is added with respect to α to obtain the final principal coefficients,α^v:

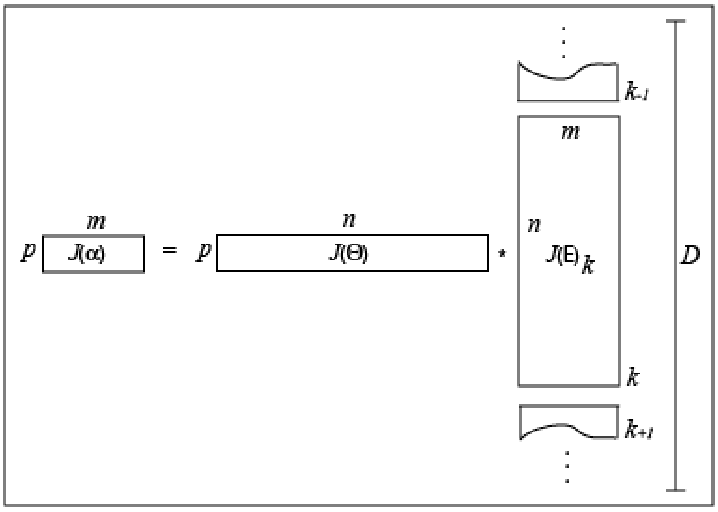


Figure 4.2: Computing $J(\alpha)$ from $J(\Theta)$ and $J(\mathbf{E})_{t_k}$. For a simple illustration, the reaching motions used in this thesis normally have the following configuration: $p = 3$; $m = 9$ and $n = 222$.

$$\alpha^v = \alpha + \Delta\alpha. \tag{4.14}$$

Finally, by using Eq. 3.14, we compute the new state vector as follows:

$$\psi\left(k, \alpha^v\right) = \alpha^v \mathbf{E}_{\mathbf{\Psi}} + \mathbf{\Psi}_{\circ}. \tag{4.15}$$

$\psi(k, \alpha^{\upsilon})$ is equivalent to $\mathbf{\Theta}^{\upsilon}$ computed in Eq. 4.7. The resolution of Eq. 4.12 is achieved in the same way as Eq. 4.11, by iteratively minimizing the norm of the difference between the current state and the desired state: $e(\alpha) = \|x(\alpha) - \mathbf{g}\|$. A termination condition must be provided in order to determine when the iterative process can be stopped, and the current configuration considered as the solution. A convergence tolerance ϵ must be defined, in order to check if the goal has been reached, up to a desired accuracy, i.e. when $e(\alpha) \leq \epsilon$. This condition stops the process when a solution exists and when the convergence occurs. However, when no solution exists, this condition cannot detect the termination of the algorithm, because the distance to the goal is not necessarily small. We handle this issue by terminating the algorithm when the residual error stops to decrease. This stop criterion is acceptable because the solution becomes steady after some iterations, as shown in Figure 4.3(b). Finally, we still use a third criterion in our system determined by the number of iterations, that is, when the iterative process reaches a certain number of iterations previously specified by the user the convergence stops. This condition lets the user follow intermediate solutions. The complete optimization of Eq. 4.12 is shown in Algorithm 1.

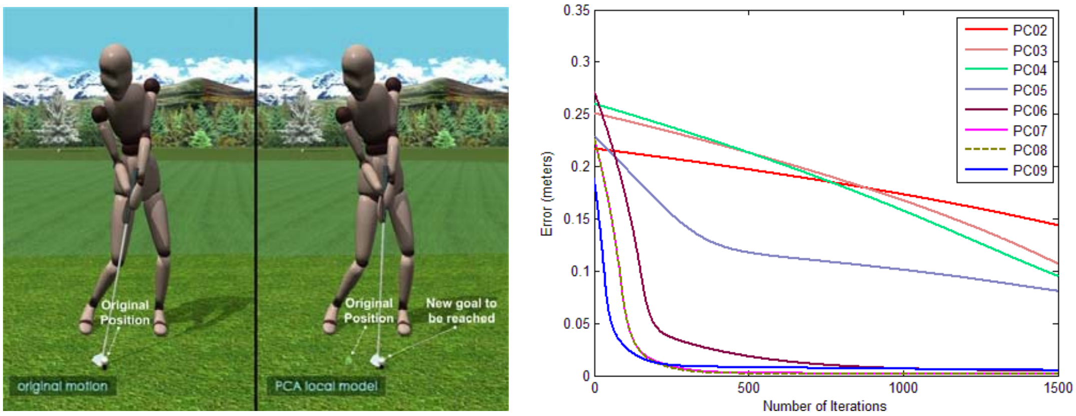
Algorithm 1 : LIK.

```
$\alpha^{\upsilon} \leftarrow \alpha$; $\Delta\alpha \leftarrow 0$; $J(\mathbf{E})_k \leftarrow \mathbf{E}_k$; $\upsilon \leftarrow 0$
while constraints not met do
    Compute $\{J(\mathbf{\Theta}), \Delta x\}$
    $J(\alpha) \leftarrow J(\mathbf{\Theta})J(\mathbf{E})_k$
    $\Delta\alpha \leftarrow J(\alpha)^{\dagger\xi}\Delta x$
    $\alpha^{\upsilon+1} \leftarrow \alpha^{\upsilon} + \Delta\alpha$
    $\psi(k, \alpha^{\upsilon+1}) \leftarrow \alpha^{\upsilon+1}\mathbf{E}_{\mathbf{\Psi}} + \mathbf{\Psi}_{\circ}$
    $\upsilon \leftarrow \upsilon + 1$;
end while
```

Numb. of eigen-motions (PC)	**Time (sec.)**	**Numb. of Iterations**
2	40.000	19106
3	110.000	50214
4	42.000	19929
5	9.000	4194
6	1.625	757
7	0.500	235
8	0.484	221
9	0.531	247

Table 4.1: Computation time of Algorithm 1 needed to converge to a pose solution depicted in Figure 6.10(a), as a function of the number of PC. We used the values $\xi = 10$. The termination condition used to stop the algorithm was: $e(\alpha) \leq 0.01m$. The number of iterations was let free.

We verified the LIK's convergence (Algorithm 1) for a simple example depicted by Fig-



(a) (b)

Figure 4.3: LIK convergence. (a) Left side: original pose. Right side: edited pose. To generate the final pose a latent space with 9 dimensions was used. (b) Convergence of algorithm 1 for different number of model dimensions. The convergence runs faster when more than five PC are used. In this experiment, the parameters of the optimizer were set to: 1500 iterations and $\xi = 10$ for the damping factor.

ure 4.3. In this example, the virtual golfer has to hit the ball onto another position previously specified by the user, as shown in Figure 6.10(a). To generate the new pose, a key-frame constraint was created on the hit frame by attaching one end-effector near the golf club head, which allowed to control the club position. The latent space was constructed from 10 golf swing motions played on a flat ground (i.e., a local model). In Figure 4.3(b) is shown the convergence of Algorithm 1 as a function of the number of dimensions of the latent space.

We observed that the convergence is faster when more than five dimensions are used (see Table 4.1) as the pose solution is more likely to belong to the generated pose space for k. In other words, for this model, less than six dimensions acquired insufficient information for solving this constraint. Thereby, the solver takes more time searching for the correct pose solution. We also noticed that when less than six dimensions were used, the reconstructed poses presented some artifacts (e.g., loss of contact between the feet and the ground). Note that, this issue is a consequence of the low percentage, $\rho(m)$, corresponding to such a small number of eigen-motions, used during reconstruction [Safonova et al., 2004] and not of the quality of the new α^v coefficients. Further investigation by considering different models will be seen in section 4.6.

In this section, we have recalled the main aspects of a classical technique for the numerical resolution of the inverse kinematics problem in the joint space. Then, due to the main drawbacks of the joint space, we proposed a new formulation of the IK problem in the latent space of a specific motion pattern. In addition, we verified that the convergence performance of Algorithm 1 is influenced by the dimension of the latent space. In the next section, we show how to extend the current approach to deal with multiple tasks and to resolve their possible conflicts in the latent space.

4.4 Low-Dimensional Prioritized Inverse Kinematics (LPIK)

Let us define a set of p tasks, each one satisfying its goal $\mathbf{g}_j$:

$$x(\alpha)_j = \mathbf{g}_j,$$

$j \in [1, \ldots, p]$, and having its corresponding increment:

$$\Delta x = (\Delta x_1, \ldots, \Delta x_p)\,.$$

In the case of multiple tasks occurring at the same key time k, the optimal solution α^* should satisfy all of them, i.e., $x(\alpha^*)_j = \mathbf{g}_j$, $\forall_j$. However, this may be difficult because some of the tasks may be in conflict. Tasks are said to be in conflict when they are not achievable simultaneously. The extreme case happens when all the tasks are not achievable (e.g., the goals are not reachable).

The first possibility to solve a conflict task is to assign a weight to each task. The purpose of each weight is to represent the relative importance of each task with respect to the others. The aim of the weighting strategy is to control the distribution of the residual error, by solving a weighted least-squares problem [Ben-Israel and Greville, 1974]. However, this strategy cannot guaranty the satisfaction of the tasks because merely weighting the tasks does not allow for a clear prioritization. When tasks get into conflict, the total residual error is distributed among all of them according to their weight. This implies that no task is exactly satisfied.

Because of the limitations of the weighting strategy, the so called priority strategy was proposed to handle conflicting constraints [Maciejewski and Klein, 1985]. The major strength of this technique is that prioritized constraints are sorted into priority-layers. As a result, constraints belonging to the highest priority layer are enforced first. Then, those of the next priority-layer are satisfied as much as possible without disturbing the previous constraints, and so on. Siciliano and Slotine [Siciliano and Slotine, 1991] proposed a scheme for solving multiple tasks with priorities. They generalized the priority strategy to handle an arbitrary number of priority level. However, its major drawback is the computation of the so called augmented Jacobian. This matrix increases from a level to the other and it has to be inverted each time degrading system performance. Baerlocher [Baerlocher and Boulic, 2004] improved the generalized priority schema by formulating an efficient and recursive solution speeding up the convergence performance.

Due to the benefits of the prioritized technique compared to the weighting strategy, we explore a formulation for solving conflict tasks similar to [Baerlocher and Boulic, 2004]. However, the main difference between both strategies is that ours solve conflicting tasks within the latent space instead of the joint space. This has still resulted in a more efficient algorithm for solving skeletons with many DoFs. In order to construct a consistent framework linking priorities to our LIK solver, we reformulate the prioritized strategy as follows:

$$\Delta\alpha \;=\; J(\alpha)^{\dagger^{\xi}}\Delta x + \mathrm{P}_{N(J(\alpha))}\mathrm{z} \qquad (4.16)$$

$$\mathrm{P}_{N(J(\alpha))} = \mathrm{I}_m - J^{\dagger}(\alpha)J(\alpha).$$

with the notation defined as below:

$\Delta\alpha$	m-dimensional PCs variation vector
Δx	p-dimensional high priority constraints
$J(\alpha)$	$p \times m$ Jacobian matrix
$J(\alpha)^{\dagger}$	$m \times p$ pseudo-inverse of J_α
$J(\alpha)^{\dagger\xi}$	$m \times p$ damped pseudo-inverse of J_α
$\mathrm{P}_{N(J(\alpha))}$	$m \times m$ projector operator onto the null-space of J_α
I_m	$m \times m$ identity matrix
z	m-dimensional PCs variation vector

The complete solution that extends Algorithm 1 and solves the Low-dimensional Prioritized Inverse Kinematics (LPIK) problem is described by Algorithm 2.

Algorithm 2 : LPIK: Handling Conflicts

```
1:  $\alpha^v \leftarrow \alpha$; $J(\mathbf{E})_k \leftarrow \mathbf{E}_k$; $v \leftarrow 0$;
2:  while not converged do
3:      $\Delta\alpha_\circ \leftarrow \mathbf{0}$; $\mathrm{P}_{N(J_{\alpha^v})} \leftarrow \mathrm{I}_q$;
4:      Compute $\{J(\mathbf{\Theta}), J(\alpha^v), \Delta x\}$
5:      for $j = 1$ to $p$ do
6:          $\Delta\widehat{x}_j \leftarrow \Delta x_j - J(\alpha^v)\Delta\alpha_{j-1}$
7:          $\widetilde{J}(\alpha^v)_j \leftarrow J(\alpha^v)_j \mathrm{P}_{N(J^{\mathrm{A}}_{\alpha^v})_{j-1}}$
8:          $\Delta\alpha_j \leftarrow \Delta\alpha_{j-1} + \widetilde{J}(\alpha^v)_j^{\dagger\xi}\Delta\widehat{x}_j$
9:          $\mathrm{P}_{N(J^{\mathrm{A}}_{\alpha^v})_j} \leftarrow \mathrm{P}_{N(J^{\mathrm{A}}_{\alpha^v})_{j-1}} - \widetilde{J}^{\dagger}(\alpha^v)_j \widetilde{J}(\alpha^v)_j$
10:     end for
11:     $\Delta\alpha \leftarrow \Delta\alpha_p + \mathrm{P}_{N(J^{\mathrm{A}}_{\alpha^v})_p}\mathrm{z}$
12:     $\alpha^{v+1} \leftarrow \alpha^v + \Delta\alpha$
13:     $\psi\left(k, \alpha^{v+1}\right) \leftarrow \alpha^{v+1}\mathbf{E}_{\mathbf{\Psi}} + \mathbf{\Psi}_\circ$
14:     $v \leftarrow v + 1$;
15: end while
```

Essentially, from lines 5 to 13 the following operations are performed:

5 : compensation of motion due to higher priority tasks;

6 : restriction by projection on $N(J^{\mathrm{A}}_{\alpha^v})_{j-1}$;

7 : accumulation of the partial solutions;

8 : incremental update of the projector;

10 : add a criterion minimization term z;

11 : compute the new principal coefficients;

$12:$ compute the new state vector;

$13:$ compute the next configuration if not converged.

The core of the priority strategy is performed in the **for** loop and in line 10, where the term with lowest priority is added in the solution. $J^{\mathrm{A}}(\alpha)_i$ is the so-called *augmented Jacobian,* here defined to act in the latent space as:

$$J^{\mathrm{A}}_{\alpha_i} = \begin{bmatrix} J_{\alpha_1} \\ J_{\alpha_2} \\ \vdots \\ J_{\alpha_i} \end{bmatrix} \tag{4.17}$$

each line represents a priority level. The main advantage of using a prioritized strategy in constraint-based motion editing is the guarantee to converge to a motion that at least enforces the most important constraints in terms of realism (e.g., feet on the ground).

In practice, the regularization factor, ξ, is used in our system to smooth out artifacts. Note that, by giving a high value for the damping stabilizes singular context but at the cost of a slower convergence. Such singular context happens when the LPIK tries to achieve unreachable poses, i.e, poses that are not in the database. The termination condition used to stop the convergence of the Algorithm 2 is achieved in the same way as described in section 4.3.

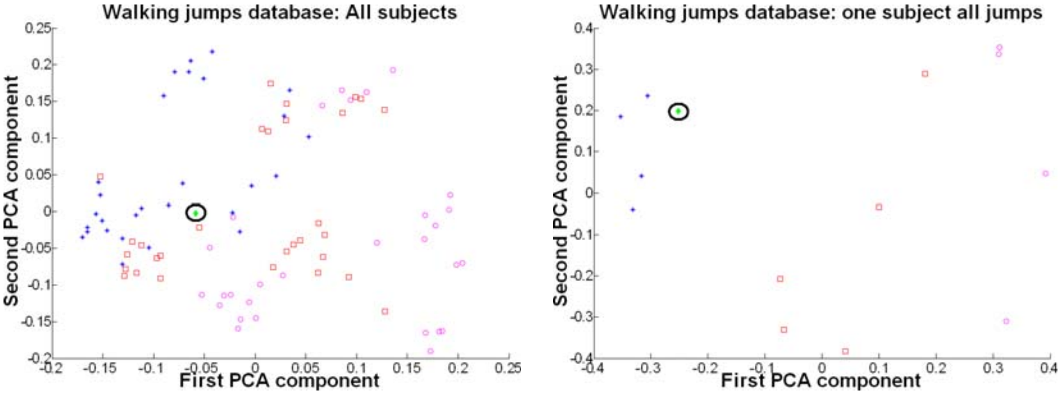
In this section, we presented a method for the resolution of task conflicts in the latent space, which introduces an order of priority to arbitrate the conflicts leading to a new set of solutions. In the next section, we discuss an important property addressed by the LPIK solver called *synergy,* which is important to improve the naturalness of the edited motion.

4.5 Synergies

One common approach used to improve the performance of a constraint-based system consists in partitioning the articulated human figure into independent substructures, this strategy offers closed form solutions for user-defined constraints [Kulpa et al., 2005]. An advantage of this approach is that one does not have to use any kind of filtering, which can provide good results for real-time applications. However, its main drawback is the lack of global synergy between groups, which may lead to unrealistic animations. In addition, there is also a lack of cooperation in solving a set of conflicting constraints: some solutions might exist but the partitioning prevents their emergence. Another technique described in [Callennec and Boulic, 2006] uses a strategy called joint recruiting, which allows a constraint to recruit all or part of the joints from its parent, up to the root joint. This approach provides good results in solving conflicting tasks due to its priority strategy, which is similar to our approach, the major difference is that it acts in the joint space. However, using just one constraint recruiting all the joints can produce motion artifacts, e.g., foot contact can be broken, making the character move through the air (this will be illustrated in Chapter 6). On the contrary, the present approach allows a more global synergy between groups, because all the character's joints are considered during optimization. This synergy is obtained from the matrix product between the eigen-motions and the joint space Jacobian, as shown in Figure 4.2. Accordingly, our solver can produce more natural-looking motions compared to solvers that provide less synergies between joints (see section 6.4).

4.6 Establishing an Appropriate Database Percentage

When caring out motion adaptations, the animator is mostly interested in the visual quality of the final animation and the time taken by the system to release the desired motion, because animators usually need to deliver good quality animations in a short period of time. In our system, the satisfaction of both demands is related with the dimensionality of the latent space. We have seen in section 4.3 that, for a small database (e.g., 10 motions) both system performance and motion quality improve as the number of model dimensions increases, that is, as the data reconstruction becomes more accurate. However, it is not interesting to always use all the model components, if we have a large dataset, because optimization performance decreases with the increase of the system's degrees of freedom. Nevertheless, as the accuracy of the motion reconstruction is controlled by the database percentage $\rho(m)$, the animator's task consists in determining an appropriate $\rho(m)$ value to establish the final number of model dimensions, m. In this context, it is important to provide the animators with a first insight of how to define an appropriate $\rho(m)$ from small and large motion database, because the greater the style variability present in the data, the higher the number of components required to better reconstruct a data [Safonova et al., 2004].



(a) (b)

Figure 4.4: Walking jumps latent spaces. (a)(b): latent space constructed from the complete and reduced walking jumps database. The green point surrounded by a black circle represents the position of the edited motion in both latent spaces.

We, therefore, investigated how the database percentage improves both system performance and the generation of natural-looking motions. For comparison purpose, we learn two motion models from the same motion pattern (i.e., walking jumps of $0.4m$, $0.8m$ and $1.2m$, each

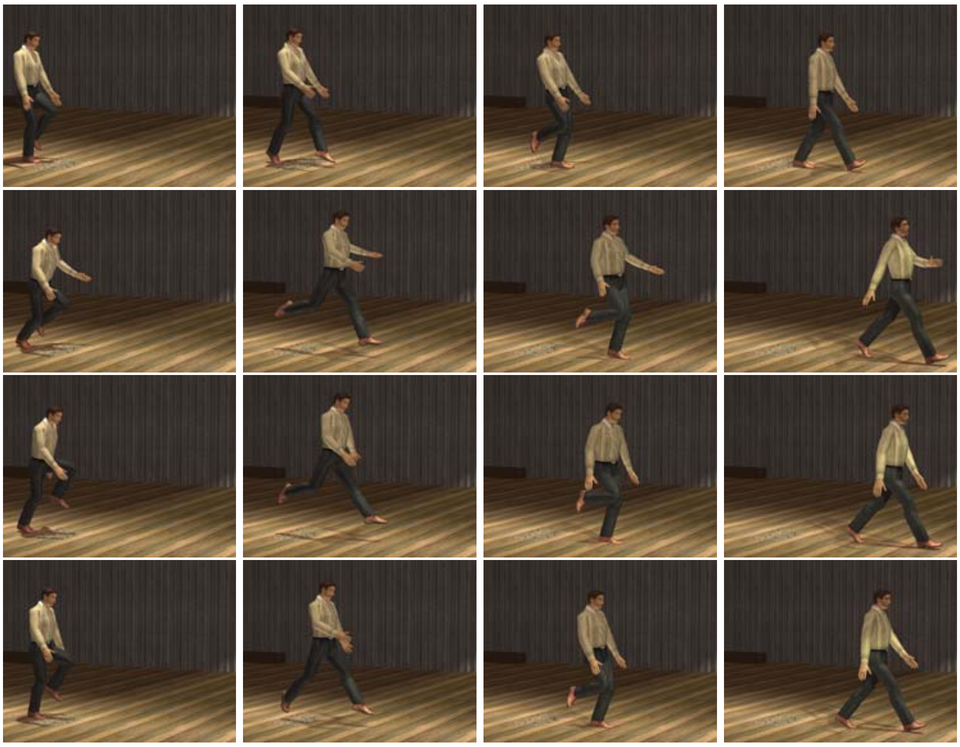
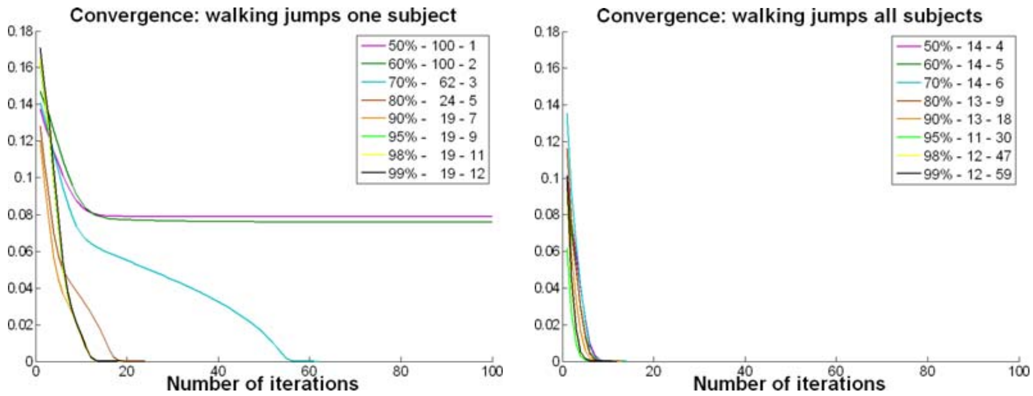


Figure 4.5: Motion generation with complete walking jumps database. Top row shows the input motion: walking jump at $40cm$. The second, third and fourth rows show the motion results obtained by optimizing the user-specified constraints in the embedded motion model space constructed from the complete dataset, by considering a database percentage of 60%, 80% and 98%. From left to right frames 7, 13, 18 and 25.

motion has 25 frames) differentiating in the number of data samples and subjects' style: the largest dataset contains 89 training motions from six subjects, and the smallest one 15 motions from one subject. The latent space is depicted in Figures. 4.4(a) and 4.4(b). The green point surrounded by a black circle represents the position of the edited motion in both sub-spaces. Therefore, to verify how the database percentage improves a motion solution, we experimented eight percentage values ranging from 50% to 99%, giving a total of 16 editing operations. In the experiments, we set the parameters of the optimizer constant, that is, 10 for the damping factor (ξ), 100 for the maximum number of iterations and $\mathrm{e} = 10^{-3}$ (i.e., $1mm$) for the convergence threshold. In what follows, we synthesized a walking jump of $40cm$ - first line of Figure 4.5 - to generate a greater jump, by attaching a key-frame constraint on the root node at frame 7, that is when the character starts the jump, and moved it $20cm$ forward (i.e., in the jump direction). This displacement led to jumps greater than $80cm$.

We observe in Figure 4.6(a) that for $\rho(m) \leq 60\%$ the convergence becomes steady before the end-effector reaches the goal, and for a higher value the end-effector reached the goal before the optimizer has achieved the maximum number of iterations. In contrast, when the motion variability is increased the task can be satisfied with sufficient accuracy independently of the



(a) (b)

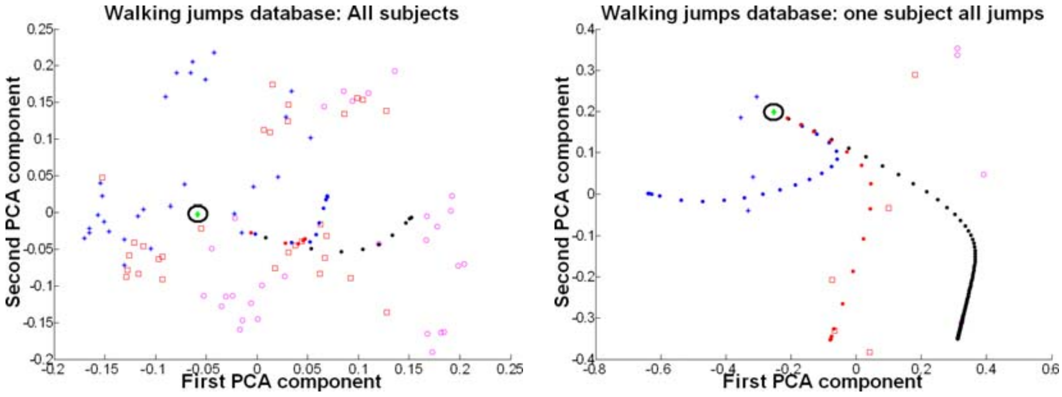
Figure 4.6: (a) convergence performance regarding number of iterations obtained by optimizing the user-specified constraints in the latent space of the complete walking jumps motion model, percentage values ranging from 50% to 99%. (b) the same, but for the reduced motion model. In both figures is shown the information of the percentage value, number if iterations and model dimensions (up right rectangle from left to right).

value of $\rho(m)$, as shown in Figures 4.6(b). We can clearly see that the model that encapsulates a higher motion variability has a much lower degradation in optimization performance (i.e., number of iterations needed to reach the goal) with the decrease of the database percentage than the model that presents less data variability.

In Figures 4.7(a) and 4.7(b) we compare the effect of the motion pattern encapsulation acquired by each model for three database percentages $\rho(m)$: 60%, 80% and 98%. The red, blue and black balls depicted in Figures 4.7(a) and 4.7(b) show the solution path obtained during optimization for these three percentage values, respectively. We can observe that when the motion variability is increased the solution locality is improved in the sense that, independently of the database percentage, solutions are found in regions near of the training data increasing the chances of releasing more realistic animations, as shown in Figure 4.5. Moreover, we can also observe in Figure 4.5 that different values of $\rho(m)$ generate different motions. On the other hand, optimizations performed in the reduced latent space, without an appropriate percentage value, can optimize solutions out the space of feasible motions. In other words, unrealistic movements can be generated as shown in the third line of Figure 4.8.

We have observed that an appropriate value of $\rho(m)$ improves both system performance and animation's realism, independently of the motion variability acquired by the model. Nevertheless, this value gets higher as the motion variability decreases. In practical terms, for a motion database containing a great motion variability, the animator can consider $\rho(m) \geq 90\%$ as good start value, but if it is not the case, $\rho(m) \geq 95\%$ is the appropriate one.

A similar study to our was conducted in [Safonova et al., 2004], but the authors investigated the quality of motion reconstruction from the number of model dimensions. As we have noticed, the number of model dimensions is not appropriate to keep a tradeoff between system performance and motion quality, if the data do not have enough generality to generate other



(a) (b)

Figure 4.7: The green point surrounded by a black circle represents the position of the edited motion in both latent spaces. (a) the black, blue and red balls represent the PCs solutions path obtained by optimizing the user-specified constraints in the latent space of the complete motion model, by considering percentages of 60%, 80% and 98%. (b) the same, but for the reduced motion model.

motions. For example, with only five dimensions (i.e., $\rho(m) = 60\%$) the model with higher motion variability could produce a realistic walking jump motion, but the other model could not produce the same quality result for the same number of dimensions (i.e., $\rho(m) = 80\%$).

4.7 Conclusion

In this chapter, we first described the two types of user-specified constraints that our system can handle: key-frame and key-trajectory constraints. Therefore, we have recalled the main aspects of a classical technique for the numerical resolution of the inverse kinematics problem in the Cartesian space, which served as a basis for the construction of the data-driven constraint-based inverse kinematics method. In a subsequent stage, the low-dimensional IK (LPIK) solver was extended to take into account conflicting tasks by using a prioritized strategy. The proposed LPIK solver optimizes user-specified constraints within the latent space of the underlying motion pattern. By solving the IK problem in the latent space rather than the joint space has two compelling advantages: (1) it reduces the Jacobian size before its inversion, making the system independent of the character DoFs reducing computation time and (2) it restricts the IK solutions in the space of feasible motions because optimizations are carried out within the pattern space of the edited motion.

The next chapter discusses the strategies used in the proposed framework to impose continuity between poses. The first one is in charge to handle key-frame adjustments and the second per-frame deformations. Both strategies are based on the motion model parameters.

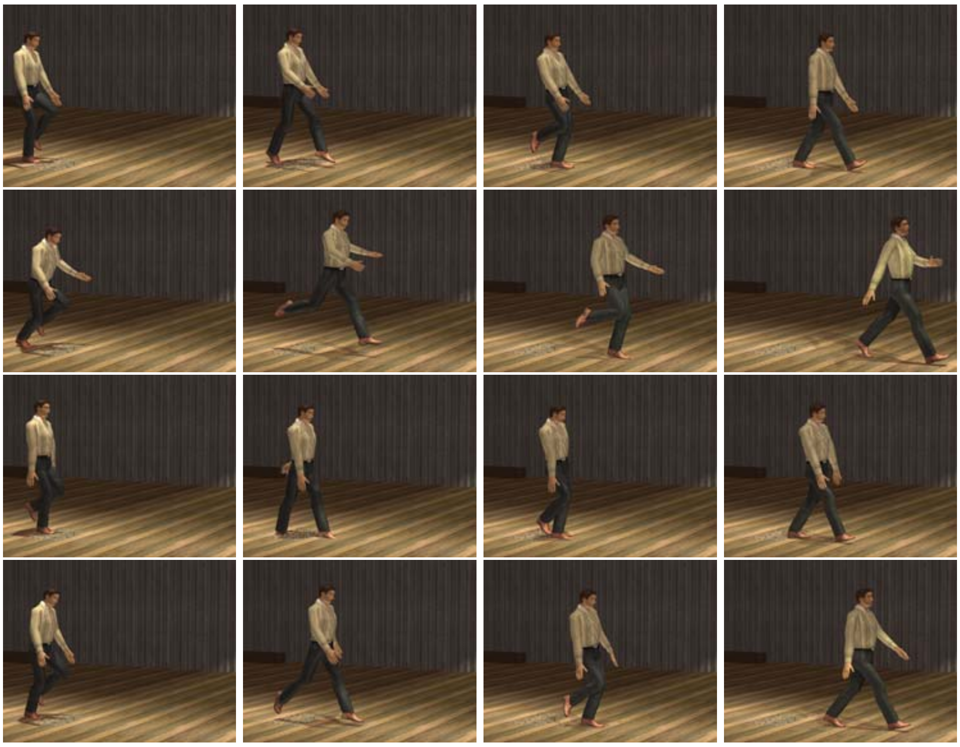


Figure 4.8: Motion generation one subject walking jump database. Top row shows the input motion: walking jump at $40cm$. The second, third and fourth rows show the motion results obtained by optimizing the user-specified constraints in the embedded motion model space constructed from the reduced dataset, by considering a database percentage of 60%, 80% and 98%. From left to right frames 7, 13, 18 and 25.

Chapter 5

Enforcing Temporal Continuity

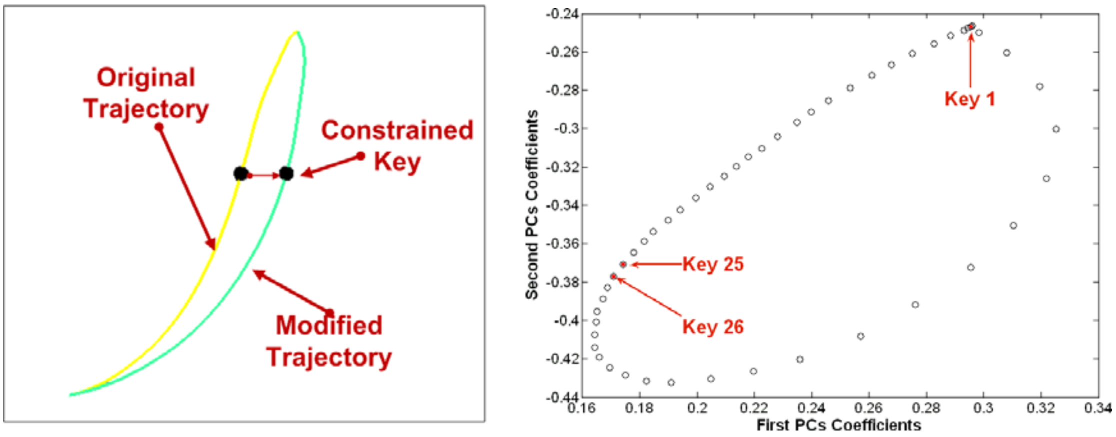
5.1 Introduction

In this chapter, we describe our approach to enforce temporal continuity between frames. Adjusting just one frame individually or a set individual frames at different key-times can destroy the inter-frame consistency. To handle these issues and both to ease the work of the animator and improve system performance, we introduce new techniques for imposing motion continuity based on PCA. The proposed methods offer the compelling advantage that they require neither filtering nor previous deformations to handle discontinuities. The next sections describe these strategies in more details.

5.2 Imposing Continuity from Key-trajectory Constraints

The strategy elaborated for imposing continuity from key-trajectory constraints uses the eigen-motion space. This space is smooth and continuous and, therefore, can handle per-frame adjustments because they can be easily interpolated [Urtasun and Fua, 2004]. As in standard per-frame constraint-based motion editing techniques, the user needs to relax the motion by using ease-in/ease-out time intervals to prevent discontinuities. However, two important differences arise between our approach and traditional ones. First, during the IK stage, it is not necessary to force each joint to be attracted toward its original value. Second, filtering is not required. As a consequence, the editing process runs faster. Figure 5.1 shows the latent trajectory obtained from a continuous Cartesian trajectory.

In our framework position constraints are not considered as locations in space but as continuous trajectories. Firstly, to enforce a position constraint on a specific key, k, during a time interval smaller than the motion duration, the animator has to define a trajectory between keys $k - \Delta k_{OG}$ and $k + \Delta k_{GO}$, where Δk_{OG} is the duration to go from the original trajectory to the goal location and Δk_{GO} the duration to go from the goal location back to the original trajectory. In practice, these time intervals control the activation/deactivation of constraints and they



(a) (b)

Figure 5.1: Smooth trajectories in Cartesian and latent spaces. A Continuous end-effector trajectory in Cartesian space (a) results in a smooth latent trajectory (b).

must be sufficiently large to provide smooth transitions from the goal location to the original trajectory. Otherwise, discontinuities may be introduced. Secondly, to enforce a position constraint on different keys the ending of the trajectory of the first key should join smoothly with the beginning of the trajectory of the second key, and so on. Otherwise, discontinuities may be introduced.

Once constraints are created and the trajectory determined, the system can release the motion. Note that, the $\{\alpha\}$ vector describes a complete motion belonging to the latent space. Therefore, solving the problem in the latent space for many poses for determining a unique $\{\alpha^v\}$ satisfying all the user-specified constraints across the motion, can easily over-constraints the problem due to the small size of the latent space. Moreover, treating the entire motion sequence as a single unit means to solve a spacetime constraints problem, but - as discussed in section 2.3.1 - this approach has many drawbacks. In practice, we ease the constraints by using a frame by frame approach, such that, from to beginning to the ending of the motion sequence the LPIK optimizes the initial $\{\alpha\}$ for each constrained frame obtaining a new set of principal coefficient vectors, such as: $\{\alpha^v_1, ..., \alpha^v_k, ..., \alpha^v_N\}$. Proceeding in this way, key-trajectory constraints are optimized without over-constraining the solution. Figure 5.2 illustrates the optimization process.

Due to the way in which PCA is constructed each vector α^v_k represents a full-body motion. In fact, the set $\{\alpha^v_1, ..., \alpha^v_k, ..., \alpha^v_N\}$ provides a $N \times N$ motion matrix of the form:

$$\begin{pmatrix} \boldsymbol{\Theta}^v_{11} & \boldsymbol{\Theta}^v_{12} & \cdots & \boldsymbol{\Theta}^v_{1N} \\ \boldsymbol{\Theta}^v_{21} & \boldsymbol{\Theta}^v_{22} & \cdots & \boldsymbol{\Theta}^v_{2N} \\ \vdots & \vdots & \cdots & \vdots \\ \boldsymbol{\Theta}^v_{N1} & \boldsymbol{\Theta}^v_{N2} & \cdots & \boldsymbol{\Theta}^v_{NN} \end{pmatrix}.$$

In order to release deformed motion $\{\boldsymbol{\Psi}^v\}$, we proceed with a frame by frame approach

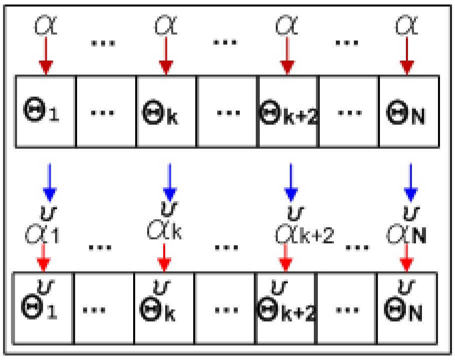


Figure 5.2: Motion editing from key-trajectory constraints. Since a full trajectory is considered all poses are constrained. Once, the LPIK converges to a $\{\alpha_k^v\}$ satisfying all the constraints of a pose k, the system uses Eq. 3.14 to recover the deformed pose: $\mathbf{\Theta}_k^v = \alpha_k^v \mathbf{E_\Psi} + \mathbf{\Psi}_\circ$. The process is repeated in a frame by frame basis until the system solves the last constrained pose.

such that, the first pose $\{\mathbf{\Theta}_1^v\}$ of the motion is computed as: $\psi\left(1, \alpha_1^v\right) = \alpha_1^v \mathbf{E_\Psi} + \mathbf{\Psi}_\circ$. In other words, the pose $\{\mathbf{\Theta}_1^v\}$ is the pose $\{\mathbf{\Theta}_{11}^v\}$ of the above matrix. Thereby, the other poses of the deformed motion are given by the diagonal of the above matrix.

5.3 Imposing Continuity from Key-frame Constraints

When a motion is deformed on a specific time or at different key-times, the motion consistency between frames can be disturbed. In our framework, the animator can modify the whole motion by either constraining and editing a pose on a *specific time* or a set of individual poses on *different key-times*, without considering the end-effector trajectory. As in key-trajectory constraints, the system releases the motion using neither filtering nor previous deformations.

5.3.1 Specific Key

In our framework, imposing continuity from specific key-time is a straightforward process using the Eq. 3.13. As the principal coefficients are learned from complete motions, each vector $\{\alpha\}$ characterizes a complete animation. Therefore, if the parameter vector is optimized for a specific key, k, resulting from a constraint imposed on a pose $\mathbf{\Theta}_k$, the updated set $\{\alpha^v\}$ defines one full motion belonging to the latent space. Once the solution coefficients are computed, they are also used to define the other poses on the motion by using Eq. 3.13. Figure 5.3 shows an schematic illustration of the process. Furthermore, this technique also provides a great benefit to animators, which is called *graphic strength*. This concept is related to the production of frame sequences that look good kinematically and dynamically [Ikemoto et al., 2009]. Producing therefore graphically strong frame sequences is a hard task due to the high kinematic detail involved. By construction, our approach can automatically propagate strong modifications across the motion enforcing an efficient and straightforward graphic strength.

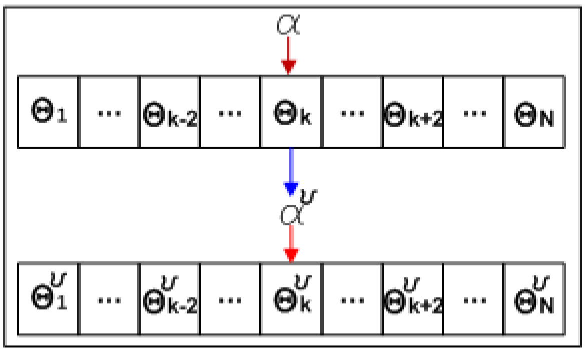


Figure 5.3: Motion editing from specific key-time constraints. The complete animation $\{\mathbf{\Psi}^{v}\}$ is computed as: $\mathbf{\Psi}^{v} = \alpha^{v}\mathbf{E}_{\mathbf{\Psi}} + \mathbf{\Psi}_{\circ}$. In this process, each pose $\{\mathbf{\Theta}_{k}^{v}\}$ is recovered by using the same $\{\alpha^{v}\}$ vector.

5.3.2 Multiple Keys

Impose continuity from multiple key-times requires a different strategy compared to the one used in the previous section, because a set of constraints attached on a pose at time k will probably destroy the constraints attached on previous poses. We observed in Figure 5.1 that a continuous end-effector trajectory in Cartesian space leads to a smooth latent trajectory without introducing discontinuities. According to this observation, we assume continuity as the fact that two successive locations in the Cartesian space are also successive in the latent space. Note that, the opposite is also true: points close in the latent space are close in the joint space, since the PCA models a mapping either from α to $\mathbf{\Psi}$ or from α to $\mathbf{\Theta}_{k}$. Accordingly, when considering a temporal sequence this fact produces smooth latent trajectories.

In order to impose continuity between multiple constrained keys at different key times, we need to handle smooth latent transitions because it will impose continuity in the joint space. For clarity purposes, we focus our discussion on two keys. Nevertheless, as the method is straightforward, it is easily extended to cases with several keys. The construction of smooth latent trajectories that holds over the interval $[\alpha_{l}^{v}, \alpha_{l+1}^{v}]$ is represented as a piecewise *Kochanek-Bartels* spline [Kochanek and Bartels, 1984]. More formally, given the knots α_{l}^{v} and $\alpha_{l+1}^{v} \in \Re^{m}$ parameterized by $h \in [0,1]$. Each segment of the spline is computed as:

$$splineKB(\alpha_{l}^{v}, \alpha_{l+1}^{v}, 0) = \alpha_{l}^{v} \tag{5.1}$$

$$splineKB(\alpha_{l}^{v}, \alpha_{l+1}^{v}, 1) = \alpha_{l+1}^{v} \tag{5.2}$$

$$splineKB(\alpha_{l}^{v}, \alpha_{l+1}^{v}, h) = \begin{bmatrix} h^{3} & h^{2} & h & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \alpha_{l}^{v} \\ \alpha_{l+1}^{v} \\ \mathbf{st}_{l} \\ \mathbf{dt}_{l} \end{bmatrix}. \tag{5.3}$$

Note that, at the beginning and ending of the motion sequence a choice for the source and destination chords of the tangent vectors shown in Eqs. 4.1 and 4.2 must be done. One choice could be to use $\alpha^v_{l-1} = \alpha$. However, this option is not so attractive because in the first and last intervals $[1, \ldots, l]$ and $[l+1, \ldots, N]$, features of the original movement, such as foot contact, will also be interpolated. This may introduce foot sliding. For example, if on a specific key value of the input motion the foot is on the ground at position x_{inp}, in the deformed motion the foot can be onto another position x_{def}. So, the interpolation in the latent space will blend these two contact positions resulting in foot sliding. Unless the animator wants to introduce foot sliding to simulate, for example, a walking on ice, this vector can be used as the $l-1$ knot. Keep in mind that, when the interpolation is done in the latent space, we are in fact interpolating either complete motion sequences or full-body poses. Accordingly, to reduce the risks of adding foot sliding and other motion artifacts introduced from contact parts, we performed the interpolation only between constrained keys. The piecewise interpolation is constructed as follows:

1. At the beginning of the motion sequence, if the constraint is not in the first key, each pose in the interval $[1, \ldots, l]$ is recovered by using the same α^v_l vector.
2. Each segment corresponding to a time interval $[l, \ldots, l+1]$ between two constrained keys is interpolated from Eqs. 5.1 to 5.3, for estimating the principal coefficient vectors.
3. At the ending of the motion sequence, if the constraint is not in the last key, each pose in the interval $[l+1, \ldots, N]$ is recovered by using the same α^v_{l+1} vector.
4. Compute the final motion using Eq. 3.14.

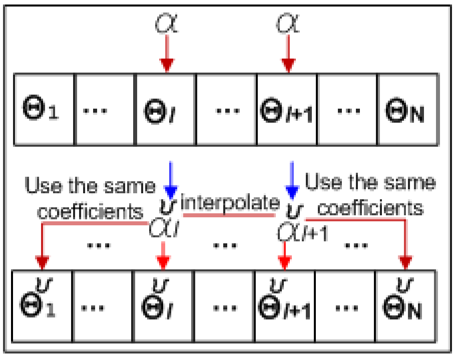


Figure 5.4: Motion editing from specific key-time constraints.

Figure 5.4 shows an illustration of the piecewise algorithm described above. This figure depicts the evolution of the principal coefficients across the motion for two key-frame constraints.

We show in Figure 5.5 the behavior of the curve under changing the spline parameters for different values of: tension, continuity and bias. As described in [Kochanek and Bartels, 1984], we assume the default value as: $t = 0, c = 0, b = 0$. We can see that the continuity parameter introduces a discontinuity at a knot. This is observed by a pronounced corner in the curve. The animator can use this parameter to chance, for example, the direction of the motion.

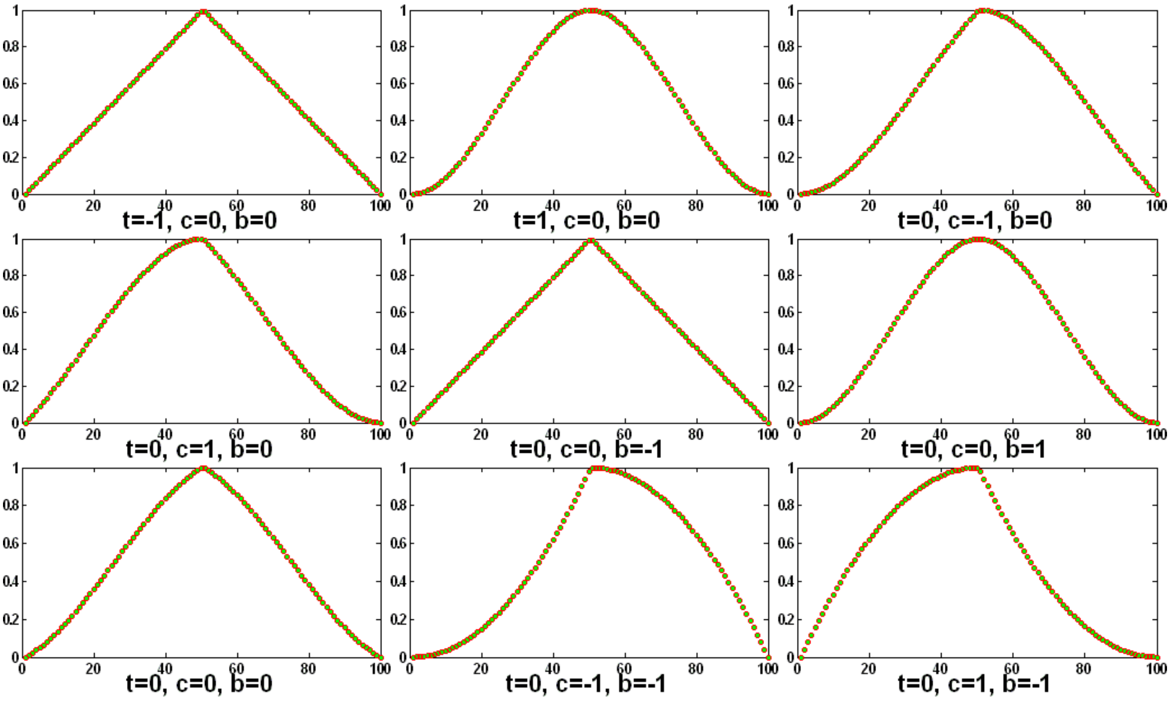
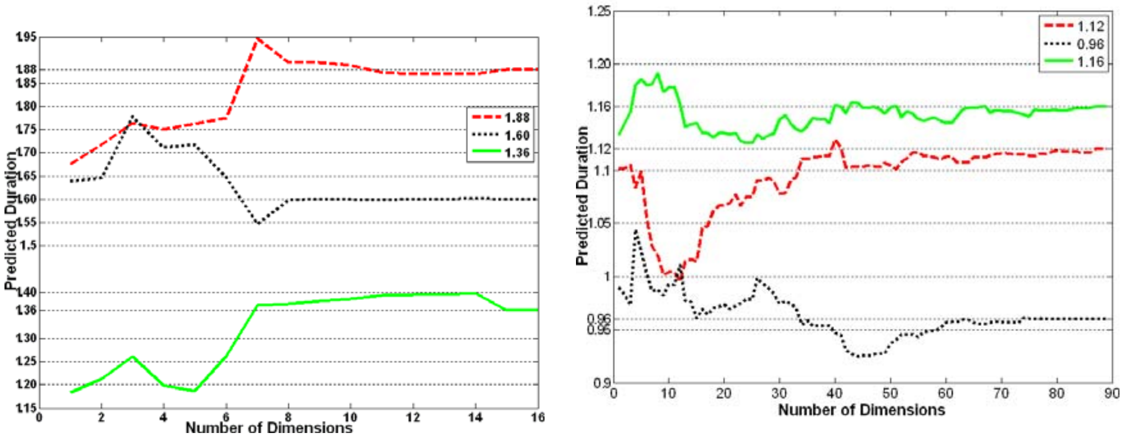


Figure 5.5: The behavior of the curve under changing the spline parameters.

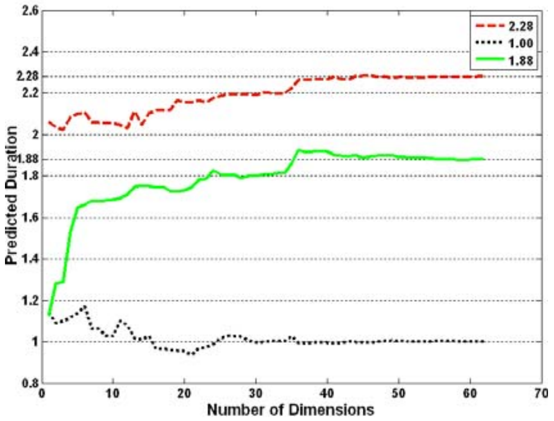
5.4 Duration

The system provides a normalized motion. If the user is satisfied with the quality of this animation regarding its duration then the process stops. Otherwise, if the user need to warp back to the original time domain further step is necessary. In fact, this is a two-step process: first, the system recovers the correct duration using the process described in the end of section 3.2.3; second, the final motion is warp back using Slerp [Shoemake, 1985].

In order to estimate the best value for the duration, we need to determine an appropriate size for the latent space in terms of the database percentage, $\rho(m)$. Therefore, to achieve this goal, we need to investigate how the database percentage influences the quality of the reconstructed duration, T^v. Of course, if all dimensions are considered the reconstruction is perfect, but this approach is prohibitive for a large dataset because the number of dimensions can be very high. In what follows, we learn three motion models: a single model learned from a reaching database, as shown in Figure 3.6; a model for walking jumps, as shown in Figure 3.2(c); and a multi-activity model learned from a database of golf swings, walking jumps and reaching motions, as shown in Figure 3.8. Therefore, we consider, for each model, the time from three different animations. In more details, for the reaching model was considered the time corresponding of a reaching in three different locations; for the walking jumps, the jumps at $0.8m$ from three different subjects; and for the multi-activity, the time of one animation of each behavior. In Figure 5.6 is shown the predicted duration as a function of model dimensions: for the reaching (Figure 5.6(a)), walking jumps (Figure 5.6(b)) and multi-activity (Figure 5.6(c))



(a) Reaching (b) Walking jumps



(c) Multi-activity

Figure 5.6: Recovered duration from key-frame constraints.

motion models. As the number of model dimensions increases T^v approaches the true value, as expected. The related database percentages corresponding to the model dimensions can be seem in Figures3.6(b), 3.2(d), and 3.8(b), respectively.

For the purpose of determining a suitable database percentage, we assume a reconstruction error of one frame, which means a tolerance of $\pm 0.04s$ for animations played at 25 frames per second. This error is sufficiently acceptable as it does not change the perception of the scene for animations played at this frame rate [Fleishman and Endler, 2000]. The latent space learned from a small database containing motions of the same behavior (e.g., reaching motions) requires a higher database percentage to approximate the duration, because the database do not have enough generality. When the data variability is increased (e.g., walking jumps database) the duration can be approximated by using a smaller database percentage. On the other hand, when motions with different behaviors are incorporated, the required dimensionality of the space increases leading to a higher database percentage compared to model specific behaviors [Safonova et al., 2004, Carvalho et al., 2007]. Note that, the higher the percentage database, the faster the IK convergence performance, as seen in section 4.6. Table 5.1 shows the mean ± standard deviation (S.D.) error for the three motion models and the corresponding database percentage necessary to approximate T_i^v with sufficient accuracy. The values shown in Table 5.1 were computed from the minimal number of dimensions. For example, considering the reaching model, T_i^v can be well approximated with at least 7 dimensions. Therefore, 8 or 9 dimensions can be considered as well. Hence, the mean ± S.D. were computed from dimensions 7 to 16 for this model. We proceed in the same way for the other models.

Models	T_1^v	T_2^v	T_3^v	*d*	*m*	$\rho(m)$
Reach	1.88 ± 0.02	1.59 ± 0.01	1.38 ± 0.02	16	8	97%
Walking jumps	1.11 ± 0.01	0.95 ± 0.01	1.15 ± 0.01	89	18	90%
Multi-activity	2.27 ± 0.02	1.01 ± 0.01	1.89 ± 0.02	63	26	99%

Table 5.1: Summary of the mean ± S.D. for the T_i^v durations. These results are for the 3 motion models investigated in this experiment. The true duration values T_1, T_2 and T_3 are shown in Figure 5.6. d is the number of motions used in the database. m is the dimensionality of the latent space necessary to approximate T_i^v with sufficient accuracy.

The process for estimating the duration described previously may not be appropriate if more than one pose is constrained. As each vector $\{\alpha_k^v\}$ is used to recover a full-body pose and not the whole animation, the use of Eq. 3.15 provides many possible duration solutions and each of them satisfying a constrained pose. In order to estimate the best T^v, we proceed with a similar strategy based on a previous work of [Boulic et al., 1990, Boulic et al., 2004, Glardon et al., 2004a]. Our solution is therefore limited to the construction of a function returning the corresponding total time given the intermediate times of the animation. More specifically, at each time step k, the total time is incremented according to the parameter vector $\{\alpha_k^v\}$ and the phase increment $\Delta\varphi = 1/N$, as follows:

$$T^v \approxeq \Delta\varphi \cdot \sum_{k=1}^{N} \alpha_k^v \mathbf{E}_T + \mu_T. \tag{5.4}$$

5.5 Conclusion

In this chapter, two methods for imposing temporal continuity between frames have been presented. The first one for enforcing key-trajectory constraints and the other key-frame constraints. Both techniques are performed in the PCA parameters space. These continuity strategies provide a range of solutions for the animator. For example, the animator may want to modify a reaching motion for describing a cubic path may by constraining two key frames and creating two trajectories connected to each other. The animator can also use the spline parameters for creating multiple solutions by keeping constraints constant. For example, by given a larger value for the tension the animator can create more tightly movements.

In the next chapter, we validate our framework in a number of experiments regarding performance, robustness, expressivity and simplicity. We also compare the results of our approach against a per-frame prioritized IK technique, where solutions are computed in the Cartesian space.

Chapter 6

Experiments and Validation

6.1 Introduction

An important application of constraint-based motion editing is in the modification of existing animations. For example, by adapting an existing motion to a new environment or another character with different proportions. As motions are hard to create from scratch and most of them are not reusable, we concentrate the experiments and validation of the proposed framework in motion adaptation to new environments and situations.

In this chapter, we present the data-driven constraint-based motion editing system, which was built based on the techniques described in the previous chapters. It allows us to illustrate the usefulness of linear motion models combined with prioritized inverse kinematics, allowing us to illustrate the usefulness of task priorities in the latent space.

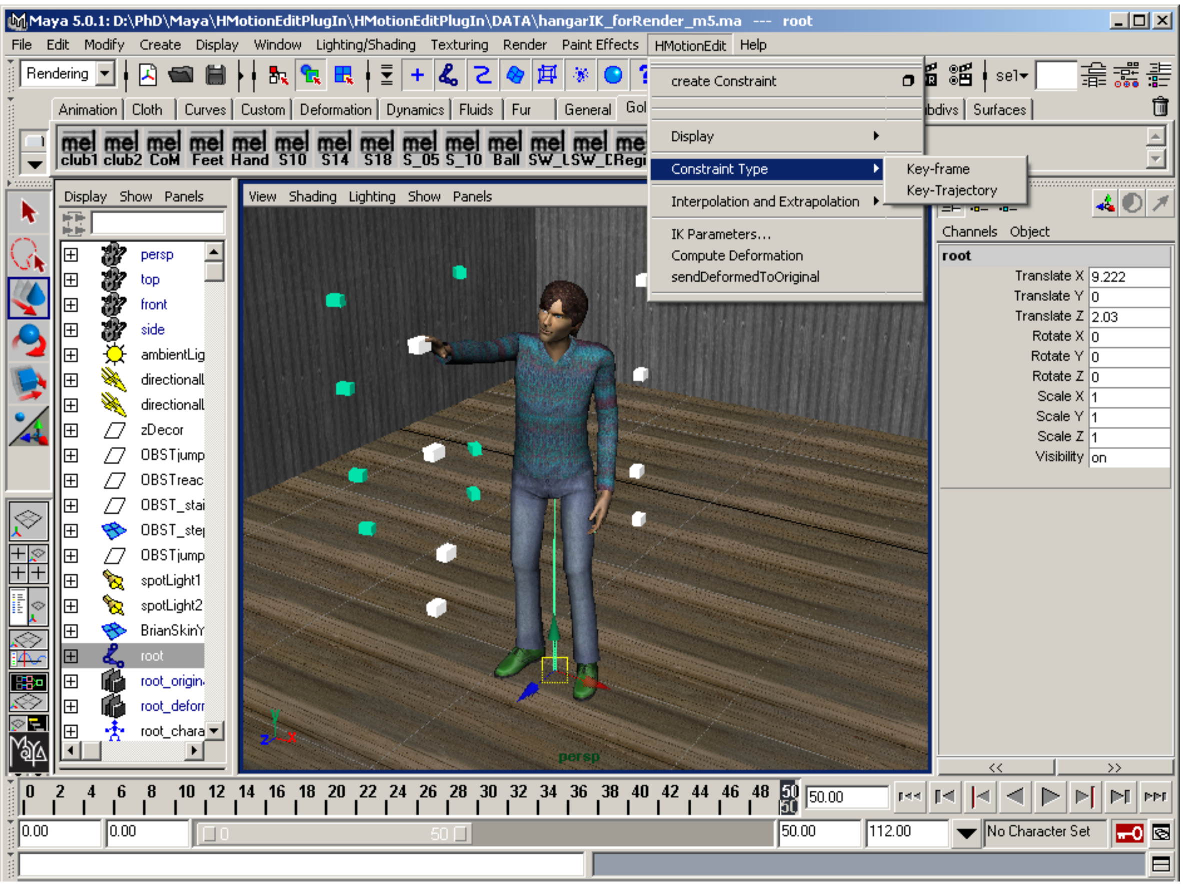
6.2 Overview of the System

6.2.1 Implementation

The motion editing system that we have proposed and used as a basis for all experiments has been implemented in three languages: C++, C and Matlab. The system is subdivided in off-line and on-line computations, as shown in Figure 1.1. We used Matlab (i.e., MATrix LABoratory) to construct the PCA motion model by following the steps of the algorithm described in section 3.2.3. We chose Matlab because it allows easy matrix manipulation, it is stable and it is well adopted in the scientific community. The PCA parameters are stored as text files, which are loaded in the system's data structure when it is started. The computationally expensive calculations, such as the optimization and the motion generation strategy, and the PCA synthesis are all done in C and C++. In all the experiments reported in this chapter are run on a 3.2GHz Pentium Xeon(TM) with 1GB memory.

6.2.2 System Interface and Functionalities

The data-driven constraint-based motion editing system proposed in this dissertation have been integrated into the Autodesk/Maya software as plug-in and MEL scripts, as shown in Figure 6.1. The character scene is created by using the Autodesk/Maya software.

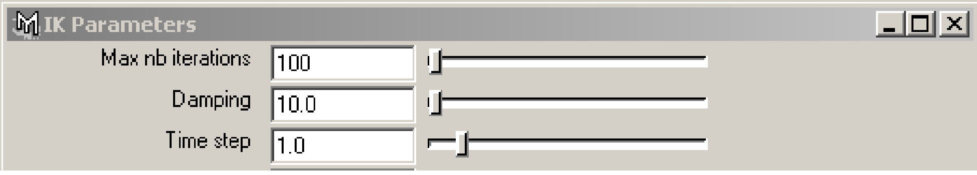


Translate X 9.222
Translate Y 0
Translate Z 2.03
Rotate X 0
Rotate Y 0
Rotate Z 0
Scale X 1
Scale Y 1
Scale Z 1
Visibility on

50.00
0.00 0.00 50.00 112.00

Figure 6.1: Data-driven constraint-based motion editing system interface.

The character model can be loaded from description files, and interactively edited in the application. The deformed animation can be saved for future use, either as a full-body motion or as a set of latent variables (i.e., the principal coefficients). The window devoted to the management of end-effectors and other objects in the scene is shown in left side of Figure 6.1. There are two ways to create an end-effector, which is represented by a sphere attached to a joint on the hierarchy. First, the end-effector can be created by selecting among a list of predefined joints available in the window (left side of Figure 6.1). This is a simple and accurate way to select standard joints such as the root, the shoulders, or the hands in order to control the reach of the character, and attach end-effectors to them. Another alternative and more direct solution is to pick a point on the surface of a character body with the mouse pointer, in the application window. This allows to control any visible part of the character body just by clicking on it.



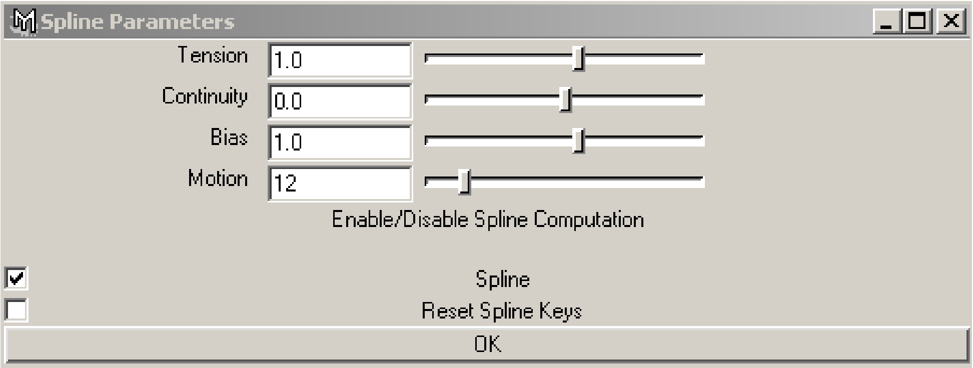
Max nb iterations 100

Damping 10.0

Time step 1.0

Figure 6.2: Optimizer parameters used to assist the editing process.

A set of criteria, which can be useful to improve the editing process, is available in a separate panel. In Figure 6.2 we show the optimizer's parameters. In this window, the animator can manage the optimizer's parameters controlling its major number of iterations, the value of the damping factor and the length of the task increment (*Time step* slide bar). The animator should be aware that due to the non-linearity of the task function, if the step is too large, the path of the end-effector over several iterations may be erratic, rather than straight. For this reason, this parameter is normally set to the unit value.



Tension 1.0

Continuity 0.0

Bias 1.0

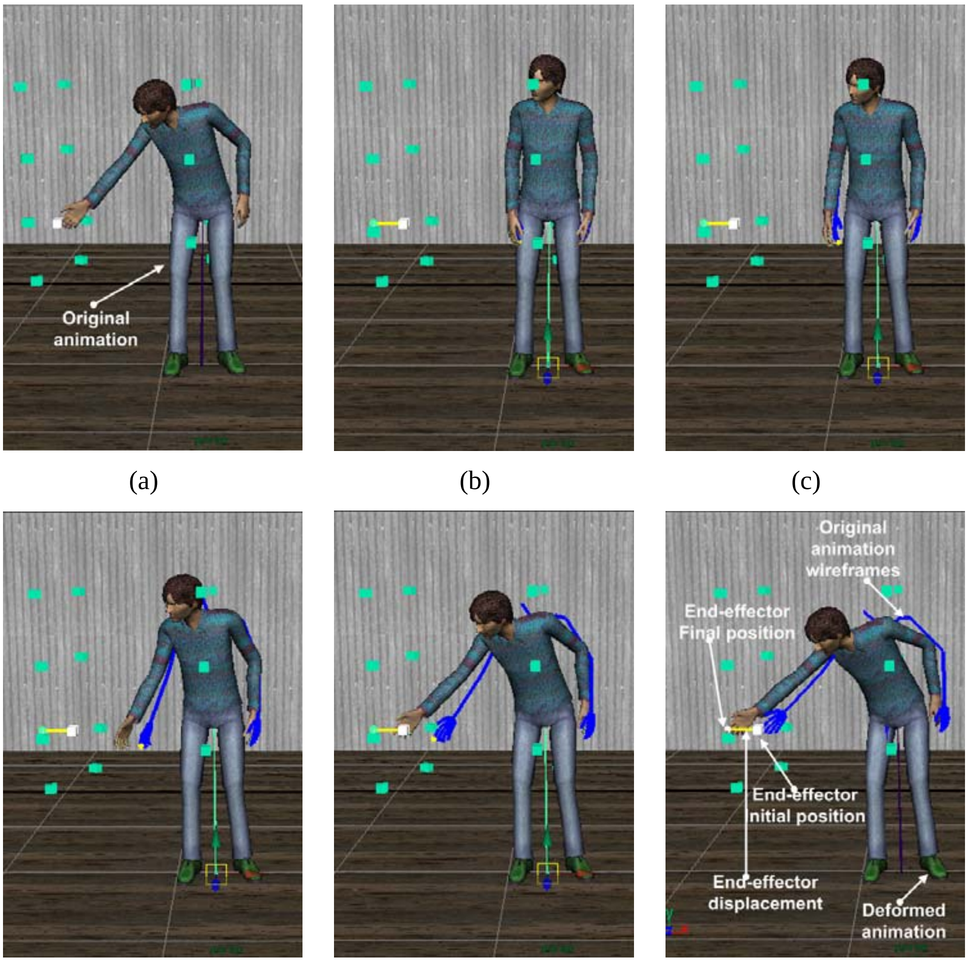
Motion 12

☑ Spline

☐ Reset Spline Keys

Figure 6.3: Spline parameters.

Another system functionality is associated with the key-frame mode. Once in this mode, animators have the possibility to edit multiple poses and generate the final movement based on the spline model described in section 5.3. In Figure 6.3 we show the spline interface. The animator can produce different motion solutions by tuning the spline parameters: tension, continuity and bias. Note that, if just one pose is edited there is no need to use the spline interpolation. In this case, the animator has to disable the spline mode by unchecking the *Spline* checkbox. In some situations the animator may want to reedit an animation be constraining less poses than before. In this case is recommended to reset the spline knot values by checking the *Reset spline keys* checkbox, to avoid undesirable computations. Once the PCA files are loaded, the animator can edit any motion in the database by tuning the *Motion* slide bar. Furthermore, the system provides the user the option of visualizing how much the deformed animation differs from the original one, by displaying the original wireframed motion, as shown in Figure 6.4.


Original animation
Original animation wireframes
End-effector Final position
End-effector initial position
End-effector displacement
Deformed animation

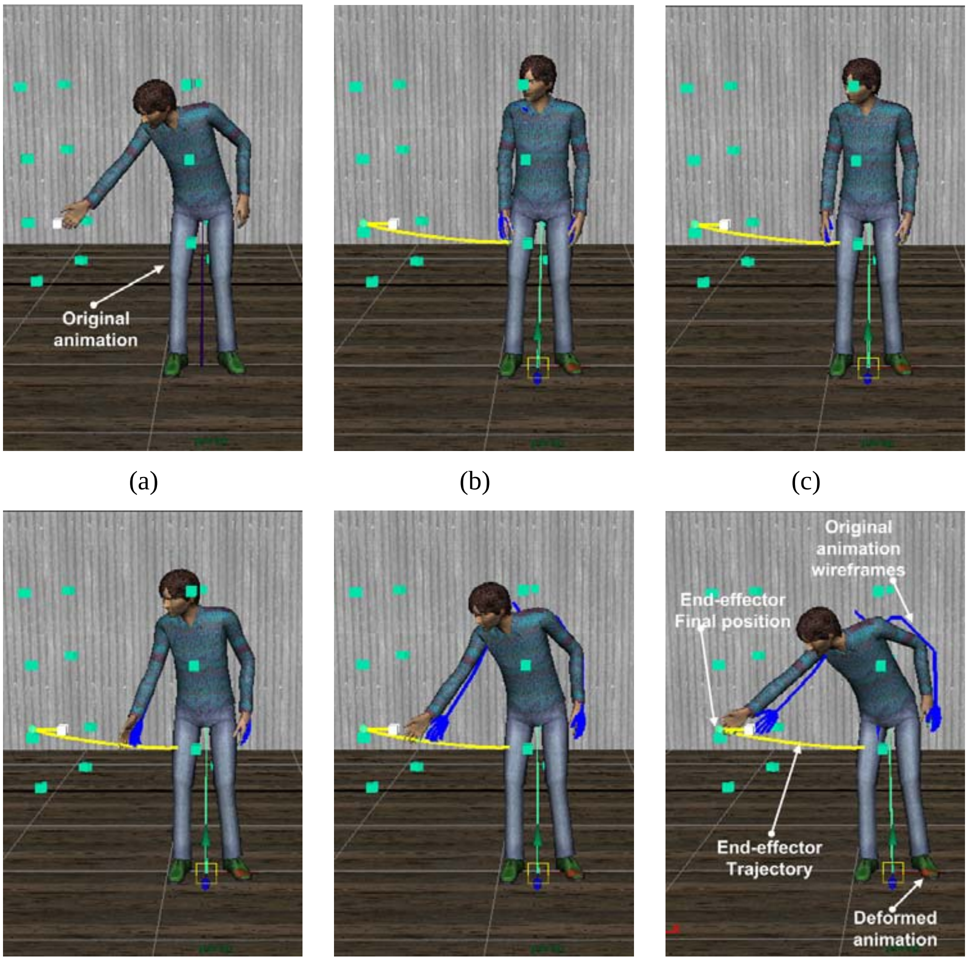
(a) (b) (c)

(d) (e) (f)

Figure 6.4: Key-frame editing. (a) Original animation, the last pose is depicted. (b)-(f) Deformed animation, frames 1, 10, 20, 30 and 50. The original wireframed animation is shown by the blue skeleton.

6.2.3 Editing the Motion

The editing process is quite simple. Basically, the user has to create a set of constraints C_s $(s = 1, ..., p)$ described by end-effectors and specify its type: key-frame or key-trajectory. Once constraints are created, the user has to specify new goals by dragging end-effectors to new positions and associate priorities to them. For simplicity, we suppose that each constraint is assigned a distinct priority level. This allows us to use the index s as the priority level, with $s = 1$ being the highest level. In the case an animator wants to use key-frame constraints, the system can provide two solutions according to the number of constrained frames. This stage is performed automatically, each time the animator creates a constraint the system automatically adds the key pose value k in a vector of constrained poses. Therefore, if just one frame is constrained the system releases a solution based on a single set of principal coefficients vector describing the whole motion. Otherwise, the system releases a solution based on a piecewise spline interpolation performed in the latent space, both solutions are described in section 5.3. In the case of key-trajectory constraints are chosen, the system automatically adds the key time values k according to the duration of the trajectories, the system releases the solution using a frame by frame strategy as described in section 5.2. Once these stages are completed the new motion can be generated by selecting “Compute Deformation”in the motion editing plug-in interface, as shown in Figure 6.1.



(a) (b) (c)

(d) (e) (f)

Figure 6.5: Key-trajectory editing. (a) Original animation, the last pose is depicted. (b)-(f) Deformed animation, frames 1, 10, 20, 30 and 50. The original wireframed animation is shown by the blue skeleton.

To illustrate the key-frame approach consider Figure 6.4. We focus in one key frame, the case of multiple keys will be shown later in this chapter. In this example, a new reaching motion is generated by changing the reach position in the last pose (Figure 6.5(a)). The white cube represents the initial reach and the green ones the reaches of the motions in the database, but the cube that has to be reached is not in the dataset. Once a pose solution is found, the optimized latent variables are automatically propagated across the motion. In what follows, from Figure 6.4(b) to 6.4(f) is shown the deformed motion represented by the character body mesh and the original animation is represented by the blue skeleton. On the other hand, when key-trajectory constraints are selected, the user has to change the end-effector's trajectory and consider ease-in/ease-out time intervals for which constraints are activated/deactivated. We illustrate this type of constraints by using the same reaching motion as described in the key-frame constraints experiment. In the example shown in Figure 6.5, the character's hand has to follow the end-effector trajectory (yellow line), to reach the same green cube as before. The motion is generated on a frame by frame basis. In what follows, from Figure 6.5(b) to 6.5(f) is shown the deformed motion represented by the character body mesh and the original animation is represented by the blue skeleton.

Note that, the two deformed reaching motions depicted in Figures 6.4(e) and 6.5(e) are slightly different regarding its dynamics, but both animations reach the final goal, as seen in Figures 6.4(f) and 6.5(f). The original motion duration is $T = 1.16$ and the resulting durations are $T^v = 1.24$ and $T^v = 1.19$ for the key and trajectory constraints, respectively. The latent space used in both examples is described in section 3.3, we used a database percentage $\rho(m) = 98\%$, which gave $m = 15$ dimensions, and the character has $n = 222$ DoFs.

6.3 Evaluation

In this section, we have designed a number of experiments to evaluate the usability and the performance of the proposed data-driven constraint-based motion editing system.

6.3.1 Evaluating Duration

In this section, we evaluate the accuracy of Eq. 5.4 used to recover the duration. We have used reaching and walking jumps sequences and have edited them by increasing both the number of constraints and constrained keys. To perform a quantitative validation we have taken two reaching sequences with durations of $1.12s$ and $1.16s$, and two walking jump sequences with durations of $1.08s$ and $1.24s$. We then have used a reaching database with 16 normalized motions, such that each sequence is $2.0s$ long, and a walking jump database with 89 normalized motions, such that each sequence is $1.04s$ long. For both models, we have considered a database percentage $\rho(m) = 98\%$, which gave $m = 15$ and $m = 70$ dimensions, respectively.

We then have verified how the Eq. 5.4 behaves by editing these motions using both key-trajectory and key-frame constraints. We run the system in ten cases:

1. **Reaching motions** We edited them using key-trajectory constraints. We first constrained two keys on the motion and used two end-effectors to change the reaching trajectory. Then, we have augmented the editing by attaching two foot constraints with higher priorities.

2. **Walking jump motions** We edited them using key-frame constraints. We constrained two keys and attached one end-effector on the character's root at frame 6 to increase the jump length, and another at frame 13 to increase the jump height. We have augmented the editing by attaching one hand constraint at key 13 with higher priority, to allow the character to reach an object in the middle of the jump. Then, we augmented the editing by attaching another constraint on the character's hand at frame 17 to allow the character to reach another object in the end of the jump.

Duration										
Original	1.12	1.12	1.16	1.16	1.08	1.08	1.08	1.24	1.24	1.24
Recovered	1.12	1.11	1.13	1.13	1.09	1.06	1.06	1.26	1.23	1.26

Table 6.1: Original and recovered durations.

We observed in table 6.1 that by increasing both the number of constrained keys and end-effectors, Eq. 5.4 provides a predicted duration with an error less than one frame. Furthermore, the proposed duration model is sufficiently robust to handle deformations from key-frame and key-trajectory constraints.

For the next experiments, we illustrate with normalized durations. This is necessary in order to better visualize the editing differences, such that, the animations can be played slowly.

6.3.2 Evaluating System Performance: Skeletons with Different DoFs

One important factor that decreases system performance is the number of DoFs used to represent the character structure. This issue is mainly related with the IK solver used in the motion editing system. To evaluate how the LPIK solver improves system performance, we have measured the computation time needed to (1) perform the pseudo-inverse, (2) perform a pose deformation and (3) perform a full-body motion deformation. The pseudo-inverse is computed in line eight of Algorithm 2, the pose deformation is basically the computation of Algorithm 2 and the computation of a motion deformation is the result of applying Algorithm 2 in each constrained pose. We compared the results obtained with the LPIK solver, which performs optimizations in the latent space, against the PIK solver, which performs optimizations in the joint space. In all the experiments considering the PIK techniques, the character's joints have been recruited from its parent to the root.

In what follows, we edited a reaching animation performed by two characters with different skeletons DoFs: $n = 222$ and $n = 90$, both skeletons are based on the H-Anim standard [H-Anim, 2009]. The first skeleton (Figure 6.7(b)) has 73 joints. The simplified one (Figure 6.7(c)) has 29 joints. In both skeletons, we also added the two more joints and for controlling foot position: right and left heel. Recalling that the root joint has six degrees of freedom. The joints used to model both skeletons can be seen in Appendix B. For this evaluation, we used a motion model, as described in section 3.3, learned from 16 time-normalized

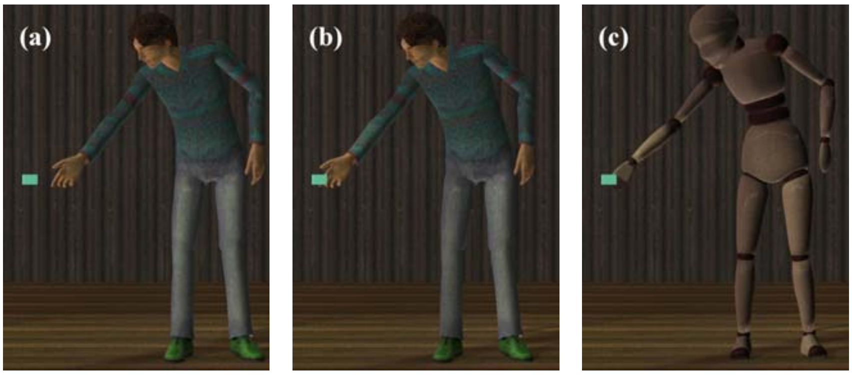


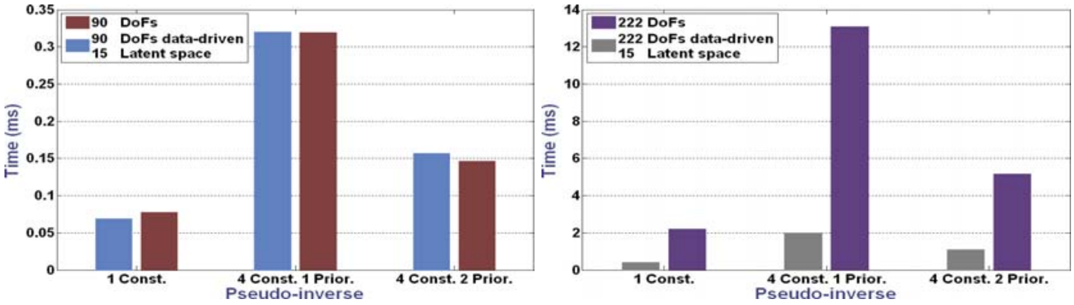
Figure 6.6: Skeletons. (a) final goal indicated by a blue cube. (b)(c) Skeletons with 222DoFs and 90DoFs, respectively.

training motions, such that each motion has $N = 50$ poses. We considered a database percentage $\rho(m) = 98\%$, which gave $m = 15$ dimensions. In the task to be executed the character had to reach a $3D$ position in cartesian space (green cube) in the end of the reach, as shown in Figure. 6.7. Note that, this animation is illustrated in Figures 6.4 and 6.5. We therefore considered the following setups:

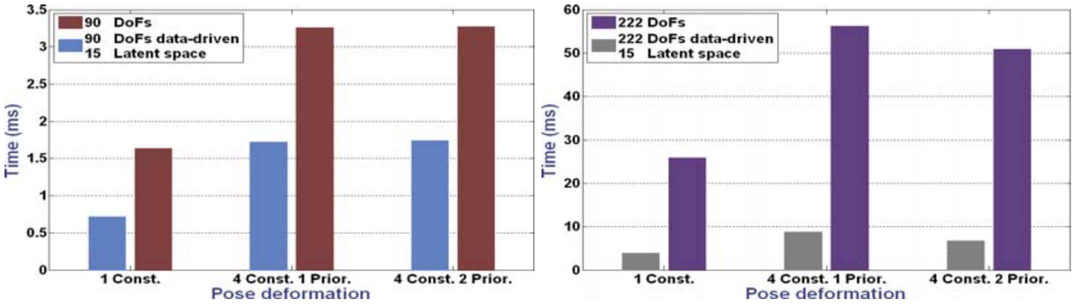
- *Pseudo-inverse and pose deformation*: one key-frame constraint on the character's hand on the last frame;
- *Full-body motion deformation*: one key-trajectory constraint on the character's hand, activated from frames 1 to 50.

In all cases, we considered three configurations regarding the number of end-effectors and its priorities: (1) one end-effector attached on the character's hand; (2) one end-effector attached on the character's hand and three on the character's feet (i.e., left toe, left heel, right toe) with the same priorities; and (3) one end-effector on the character's hand with lower priority and three on the character's feet with highest priority.

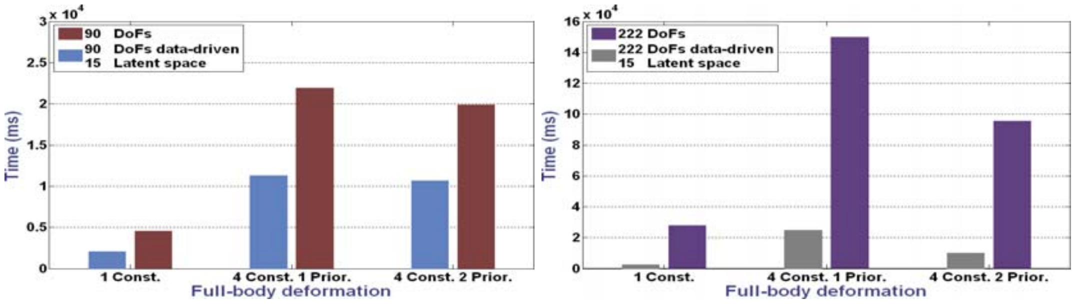
The LPIK-based system performed faster in the majority of the experiments compared to the PIK. We observe in Figure 6.7 that as the character's DoFs and the number of constraints increase the PIK performance decreases faster than the low-dimensional solver introduced in this thesis. We can draw important quantitative conclusions based on these experiments. For example, consider the case of four constraints with the same priorities and the three experiments (pseudo-inverse, pose and full-body motion deformation), the optimizations performed in the joint space were 40.90, 17.18 and 4.8 times slower when the skeleton's DoFs increased. On the other hand, the same optimizations performed in the latent space were 6.21, 5.02 and 2.21 times slower. This means that the LPIK solver is 6.60, 6.43 and 5.98 times faster than the PIK when the skeleton increases from 90 to 222 DoFs. We clearly see that the system performance is strongly related with the dimensionality of the space. The LPIK performs optimizations in a space of dimensions 15×3 or 15×12, regardless the skeleton. On the contrary, the



(a) Pseudo-inverse



(b) Pose deformation



(c) Motion deformation

Figure 6.7: Evaluating system performance: lower values mean faster results.

PIK has to solve a problem in spaces with dimensions 90×3 or 90×12 and 222×3 or 222×12. Although the optimizations within the latent space are carried out in a space of fixed dimension - for the same number of constraints and model dimensions - we observed a slight difference in performance results when the skeleton's DoFs increased (blue and purple bars). This is a consequence of the time required to compute the joint space Jacobian and the PCA synthesis, because both stages depend of the character's DoFs. Nevertheless, the decreasing in performance gets more accentuated in the joint space (green and red bars).

6.3.3 Evaluating Motion Quality

To evaluate motion quality and perform a qualitative validation of our framework, we have taken one motion sequence from the reaching database and use it as a testing sequence. In order to verify how the motion quality can be improved, we have increased the numbers of constraints and assigned different levels of priorities to them. We have used a reaching motion model learned from 16 time-normalized training motions, such that each motion has $N = 50$ poses. We considered a database percentage $\rho(m) = 98\%$, which gave $m = 15$ dimensions. In the task to be performed the character has to reach a 3D position in cartesian space, represented by a green bar, in the end of the reach as shown in Figure 6.8. This task cannot be satisfied directly by any motion in the database. We then evaluate the quality of the generated animations by using five key-frame constraints configurations according to table 6.2. The optimizer parameters were kept constant in all the experiments.

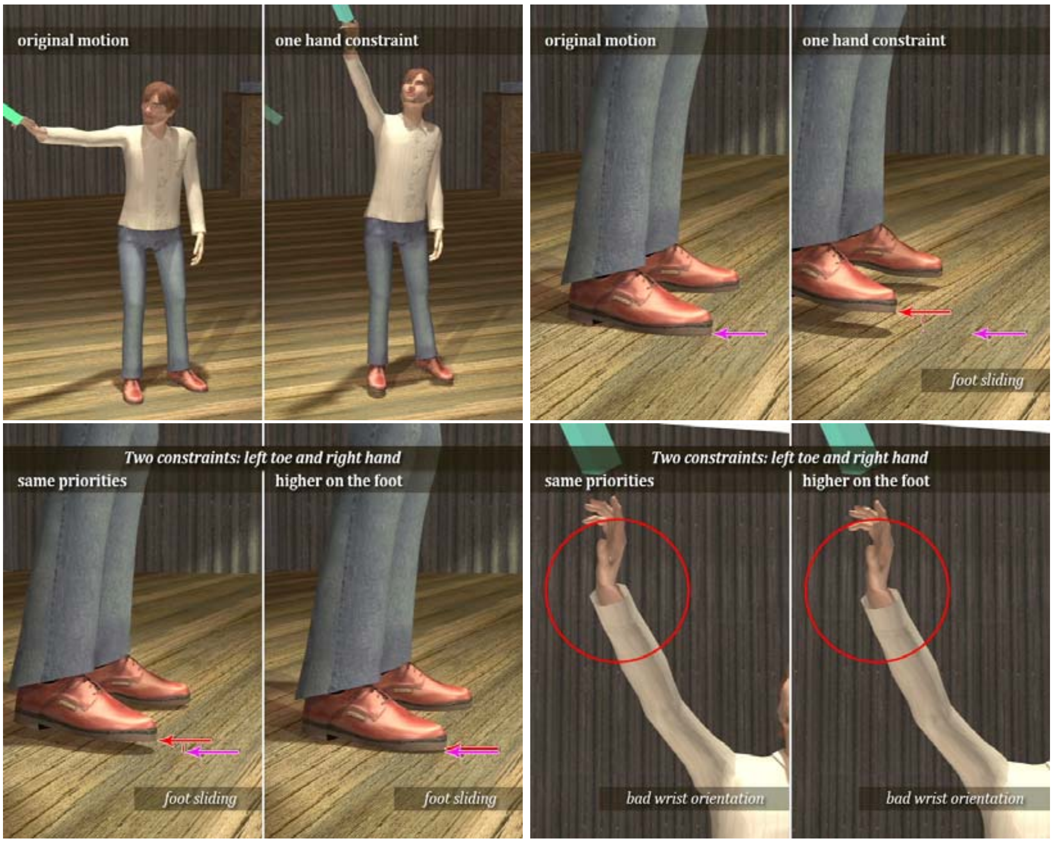


Figure 6.8: The pink arrow in represents the original foot position and the red one the sliding gap. First row: the character reaches the object, but foot sliding is introduced. Second row: the character does not reaches the object and the right wrist presented an unnatural orientation, but foot sliding is considerably reduced when the priority is higher on the foot.

When one hand constraint was used the character reached the object, but foot sliding was introduced as shown in Figure 6.8. The results improve when the animator slightly increases the number of constraints and giving them different priorities: foot sliding is reduced and the right wrist orientation becomes more natural, as observed in the second and first rows of Figures 6.8 and 6.9. Furthermore, slightly increasing the number of constraints and give them different priorities also demonstrated to improve the naturalness of the movement: in the beginning of

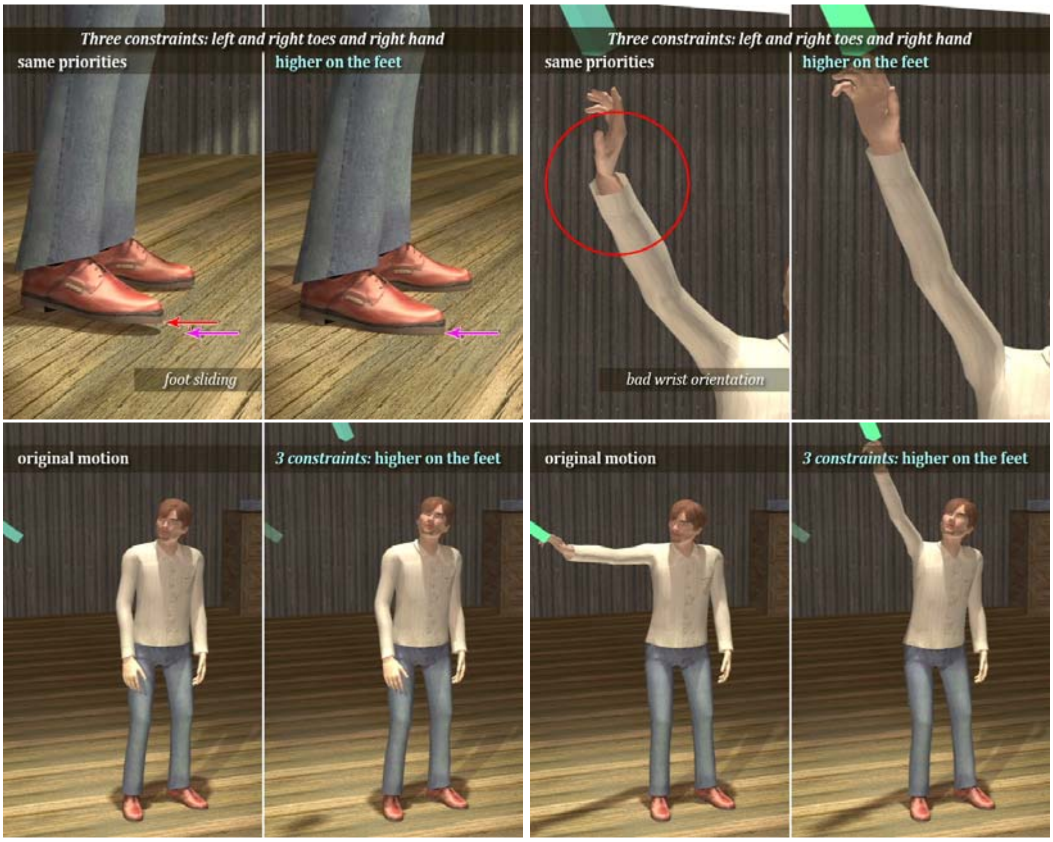

Three constraints: left and right toes and right hand
same priorities
higher on the feet
foot sliding
Three constraints: left and right toes and right hand
same priorities
higher on the feet
bad wrist orientation
original motion
3 constraints: higher on the feet
original motion
3 constraints: higher on the feet

Figure 6.9: The pink arrow represents the original foot position and the red one the sliding gap. First row: motion quality is improved when foot constraints with higher priorities are used. The character reaches the object and there is no foot sliding. Second row: the result is natural-looking. In the beginning of the motion the character looks to the object before reaching it.

the motion the character looks to the object before reaching it, as observed in Figure 6.9.

6.4 Synthesis Comparisons

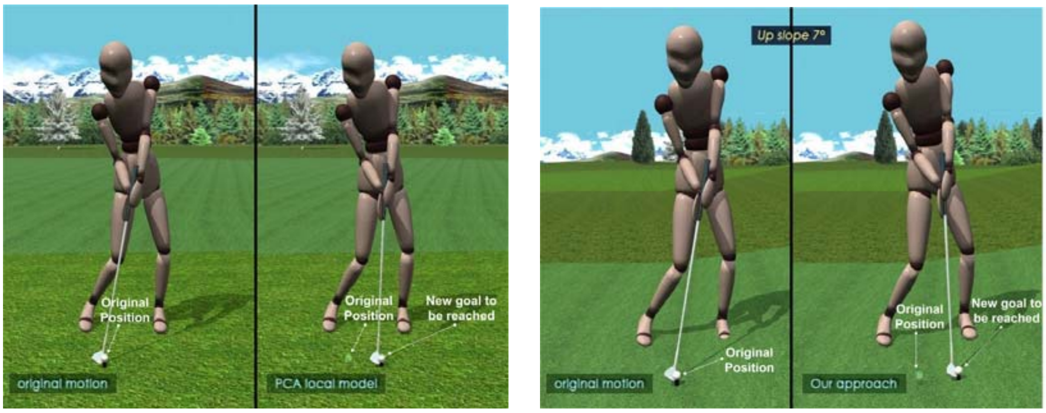
In this section, we perform experiments to compare our approach against a prioritized constraint-based technique that performs deformations in the joint space [Callennec and Boulic, 2006], regarding performance, robustness, simplicity, continuity and realism, by synthesizing golf swing motions executed on three different types of terrain: flat, up and down slope grounds. Editing golf swings confronts the system with challenging problems, such as: dealing with different styles of swings, time precision (e.g., the hitting position of the golf club head) and the swing is normally executed in high speeds [Raymond, 2003, Gehrig et al., 2003, Urtasun et al., 2005]. Hence, any artifact is clearly noticeable in the solution movement. Later, we have experimented with reaching and walking jump motions.

Number of constraints	End-effector location	Priority rank
1	right hand	1
2	left toe, right hand	1,1
2	left toe, right hand	1,2
3	right and left toes, right hand	1,1,1
3	right and left toes, right hand	1,1,2

Table 6.2: The editing is gradually refined from the top to the bottom lines: 1 (top priority).

6.4.1 Local vs. Single Models

We verified the convergence performance of the LPIK for situations in which the animator may need to increase the behavioral space. This is sometimes necessary, in order to either satisfy different constraint configurations or edit distinct motion patterns. For that, we have considered the editing of a golf swing played on a flat ground to achieve two distinct goals: (1) shifting the hit position (Figure 6.10(a)); (2) adapting the swing to an up slope with 7° (Figure 6.10(b)).



(a) (b)

Figure 6.10: The types of considered editing for the flat ground swing. (a) Golf club head adjustment. (b) Retargeting to an up slope with 7°. In both figures the left side is the original pose and the right side the modified one.

We then consider three key-frame constraint configurations:

1. attaching one constraint near the golf club head to synthesize a flat ground swing just by shifting the hit position;
2. attaching three constraints on the feet with highest priority and one constraint near the golf club head with a lower priority;
3. attaching three constraints on the feet and one constraint near the golf club with the same priority.

The hit position was the same in all cases. We built three motion models: two local models, one learned from flat ground swing motions and another from up slope swing motions, each database contain 16 data samples; and one single model from the combined flat, up and down slopes swings containing 48 samples. We have used a database percentage $\rho(m) = 98\%$, which gave a latent space with $m = 15$ and $m = 27$ dimensions for the local and combined models, respectively. The character has $n = 93$ DoFs. In the first case study, the three constraint configurations were experimented, but in the second one we used just the second and third setups because it is impossible to retarget the swing played on a flat ground to an up slope controlling just the position of the club head. Figure 6.11 shows a typical constraint configuration used in our system to edit a golf swing.

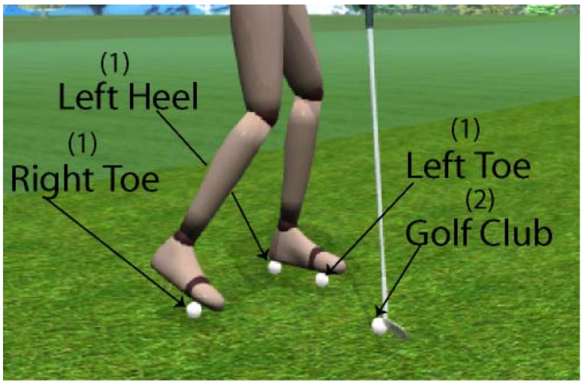


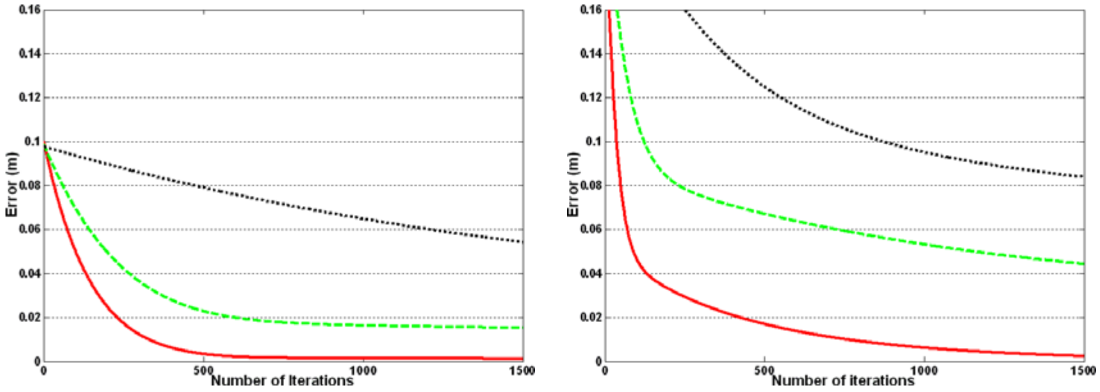
Figure 6.11: A typical constraint configuration used in our system to edit a golf swing. 1 means the highest priority level.

We can see in Figure 6.12 that the LPIK's convergence is faster when local models of the same behavior were used. Giving different levels of priority for each set of constraints also produces a faster convergence than giving the same importance for all the constraints. We observe that the error norm $e(\alpha)$, decreases as expected. We also noticed that, the convergence provided by the LPIK (Algorithm 2) depends on many parameters: the choice of a specific motion model, the priority given for each constraint, and the intrinsic parameters of the optimizer (i.e., the damping factor, $\xi = 10$ and the number of iterations, which we set to 1500). It took our system at most 5.3 milliseconds per iteration.

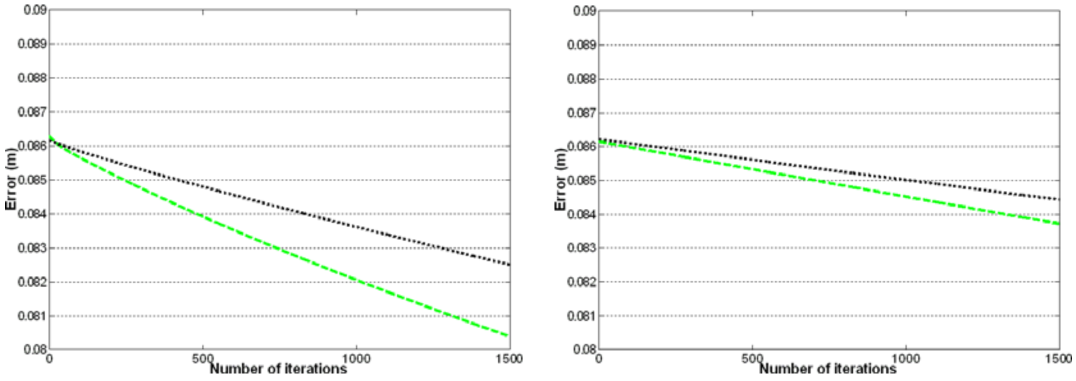
To summarize. The aim of mixing different behaviors for constructing combined models, is to give animators the possibility of editing different motion patterns without the need of changing the model, but at a price of a slower convergence.

6.4.2 Joint Space vs. Latent Space

In this section is shown the editing results for optimizations performed in the joint space (or simple PIK) and in the latent space (or simple LPIK).



(a) Local model flat ground. (b) Single model.



(c) Local model up ground. (d) Single model.

Figure 6.12: LPIK convergence for the three constraint configurations, first (solid red), second (dashed green) and third (dotted black).

6.4.2.1 Performance, Robustness, Simplicity and Realism

For a fair comparison between both techniques, in the joint constraint-based system, we set the parameters of the optimizer as suggested by the authors, first line of table 6.3, to obtain the best tradeoff between natural-looking motions and computing time performance. We made the same with our system, second line of table 6.3, and we used local models due to its faster convergence.

Flat ground	**Up slope ground**	**Down slope ground**
NoI = 100, $\xi = 10$	NoI = 100, $\xi = 10$	NoI = 100, $\xi = 10$
NoI = 25, $\xi = 1.5$	NoI = 400, $\xi = 0.8$	NoI = 400, $\xi = 0.8$

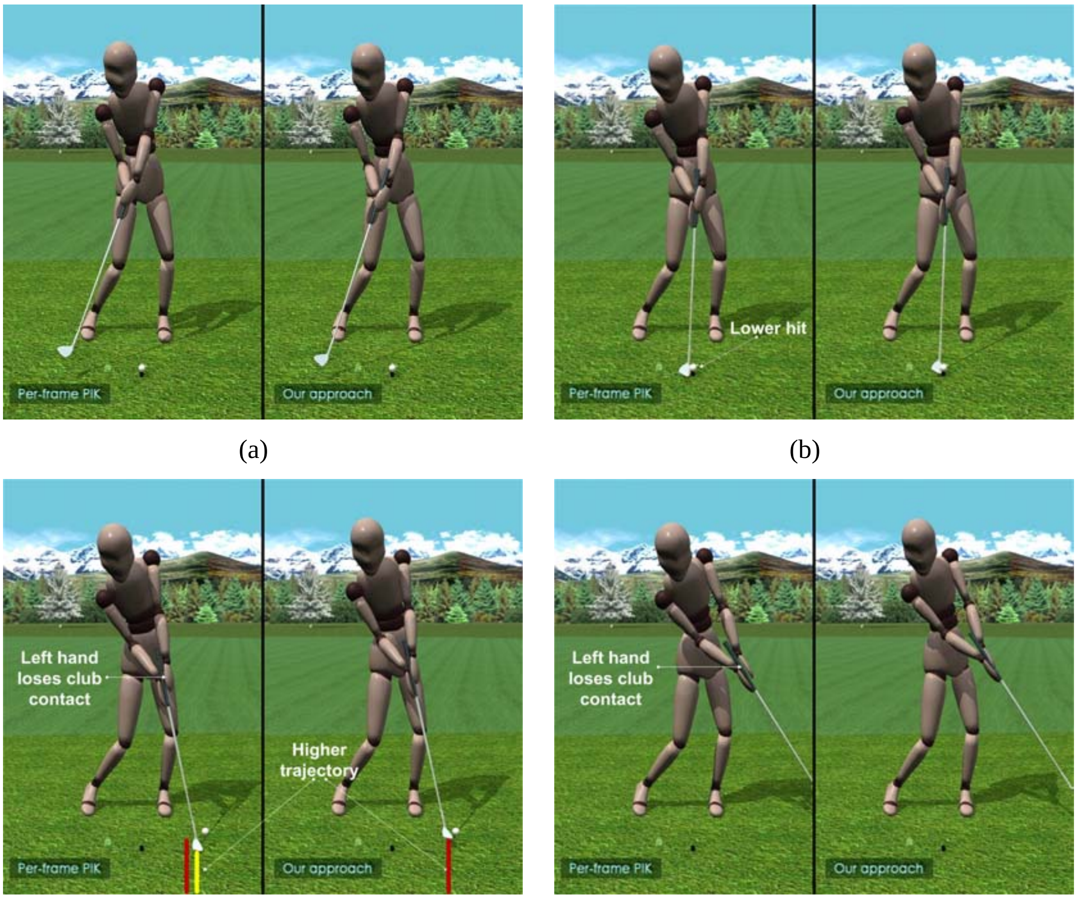
Table 6.3: Optimizer parameters values: number of iterations (NoI) and damping factor (ξ).

For these experiments, we have edited the same golf swing motion, with 132 frames, executed on a flat ground to achieve three different goals:

1. shifting the hit position of the golf club head, Figure 6.13;

2. retargeting to a down slope ground with 7° clockwise, by adjusting the feet position and by shifting the hit position of the golf club head, Figure 6.14;
3. retargeting to a up slope ground with 7° anti-clockwise, by adjusting the feet position and by shifting the hit position of the golf club head, Figure 6.15.

It is important to mention that, the 7° slope was not learned by the model. The adjusted posture, at frame 94, and the position of the golf club with respect to the hands were the same in all experiments. In all deformations, key-frame constraints were used for the data-driven approach. Key-trajectory constraints were used instead, as the joint approach cannot handle key-frame constraints.



(a) (b)

(c) (d)

Figure 6.13: Synthesis comparison joint vs. latent space: golf swing flat ground. joint and latent space results are shown in the left and right sides, respectively.

Figure 6.13 shows the deformation results for the golf swing played on the flat ground. The motion editing system based on the joint optimization generated a less realistic swing motion compared to one based on the latent space. We observed that, the character's left hand lost the contact with the golf club and the club head described a lower trajectory for postures after the hit pose. Moreover, by using the joint approach, we needed to attach five constraints: three on the feet with highest priority (activated frames $1 - 132$), to prevent foot sliding; one to control the center of mass (activated frames $1 - 132$) with lower priority, to prevent unbalance poses and one near of the golf club head (activated frames $90 - 99$) with middle priority, to control the hit position, as shown in Table 6.4. On the contrary, with the proposed approach, only one constraint was necessary - the golf club head constraint - to achieve a globally continuous motion. The computing performance needed to generate the deformed motion and the necessary number of end-effectors used per each technique is shown in the first row of Table 6.5.

Priority level	**Effectors**
1 (highest priority)	left toe, left heel, right toe
2	golf club head
3 (lowest priority)	center of mass

Table 6.4: Shows the end-effectors used to edit the golf swing motions and its priorities.

Figures 6.14 and 6.15 show the deformation results for the golf swings played on the down and up slope grounds. In both cases, the joint approach also generated less natural-looking motions compared to the latent space. Regardless hand problems, we can see in Figure 6.15(d) that the club trajectory is an issue for up slope deformations in the joint space: the club head penetrated the up slope ground. The computing performance needed to generate the down and up slope motions and the necessary number of end-effectors used per each technique is shown in the second and third columns of Table 6.5. We also experimented to generate animation by activating all frames on the motion (joint approach), but the results did not change significantly.

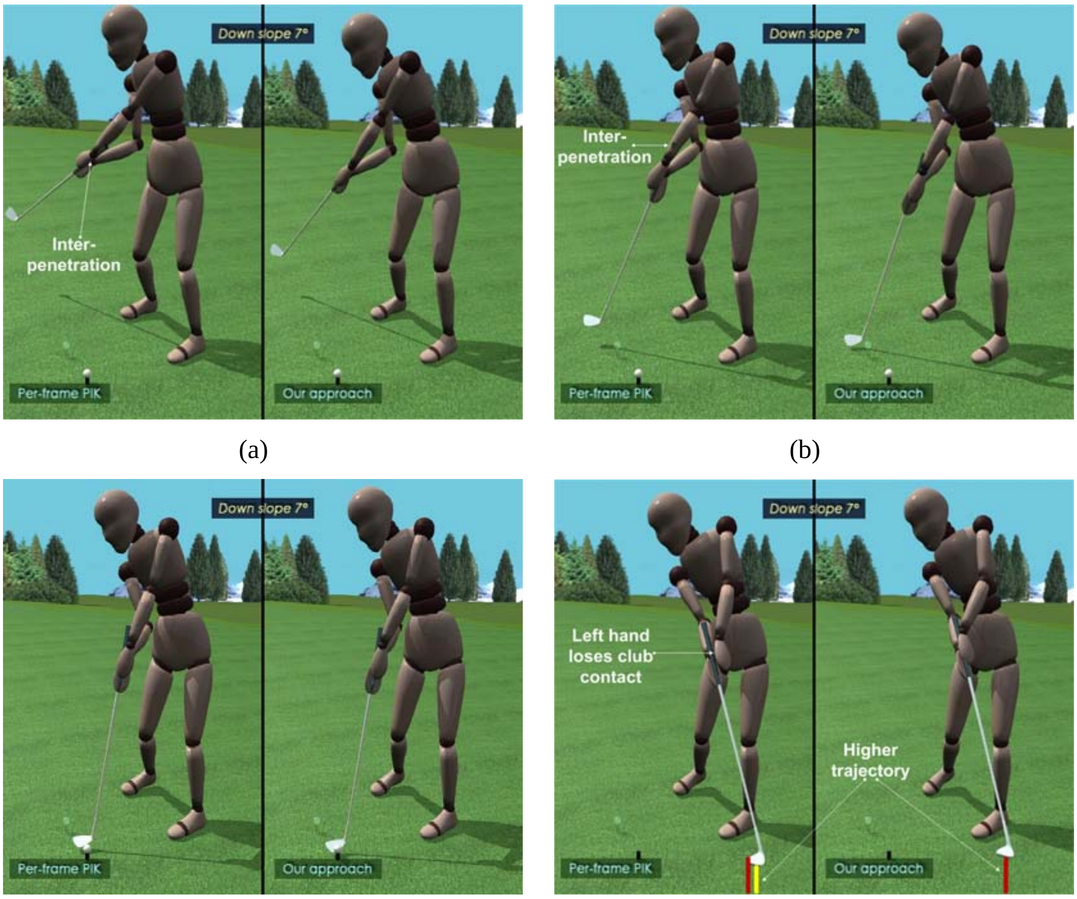
Deformation task	**Joint space**		**Latent space**	
Flat ground	$80s$	5	$0.2s$	1
Down slope	$102s$	5	$3.0s$	4
Up slope	$102s$	5	$3.0s$	4

Table 6.5: Computation performance of each editing task for both approaches.

In summary, the motions generated with the joint approach showed either inter-penetration between the arms and the golf club (e.g., Figure 6.14(a)) or with the ground (e.g., Figure 6.15(d)). In some situations, the character's left hand lost the contact with the golf club (e.g., Figures 6.14(d) and 6.15(d)), and always noticed the presence of discontinuities (e.g., body balance) around the hit frame.

6.4.2.2 Continuity

In this section, we compare the continuity techniques used to handle deformations in the latent and joint spaces, by editing walking jump and reaching motions. We used a motion model learned from 89 normalized animations, such that each motions has 26 poses. We used a

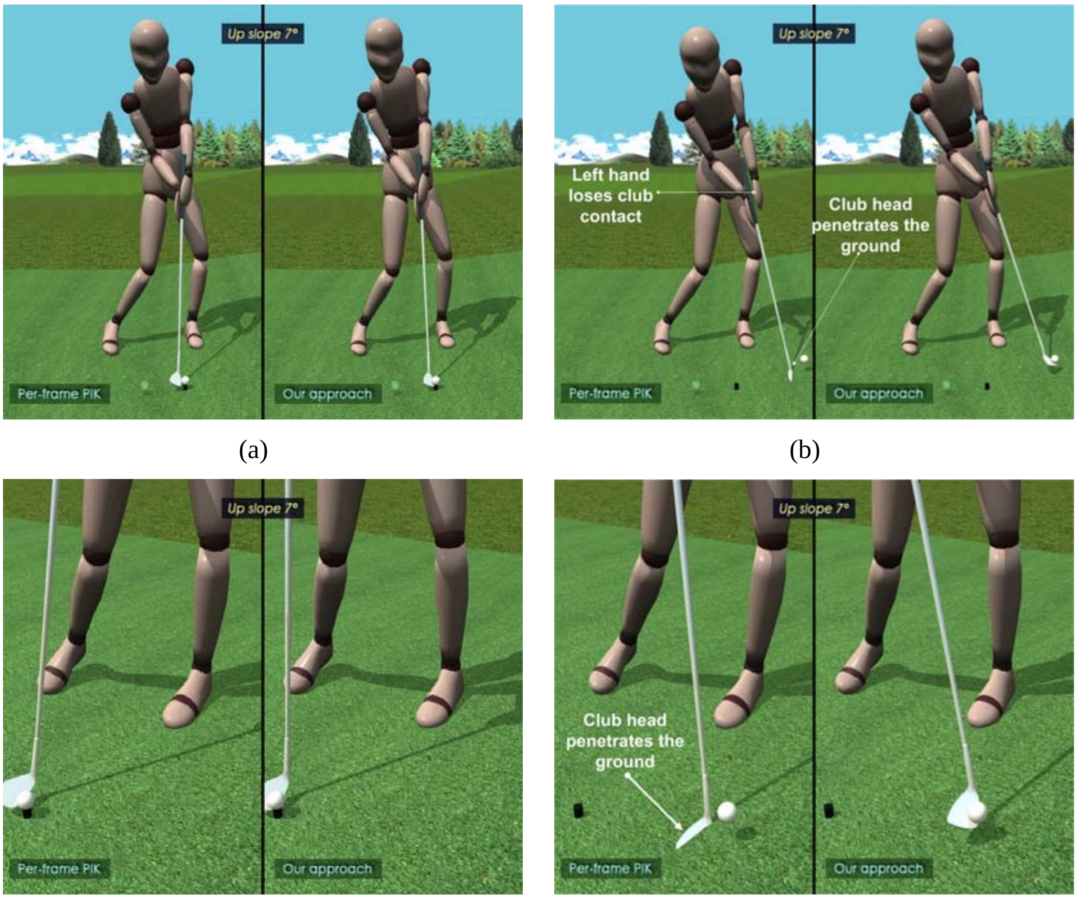


(a) (b) (c) (d)

Figure 6.14: Synthesis comparison joint vs. latent space: golf swing down slope ground. joint and latent space results are shown in the left and right sides, respectively.

database percentage $\rho(m) = 95\%$, which gave $m = 30$ dimensions. For the optimizer parameters, we have kept the same value for the regularization factor, $\xi = 10$, and the number of iterations, which we set to 100, in all the experiments. For reaching motions, We used a motion model learned from 16 normalized training motions, such that each motion has 50 frames. We considered a database percentage $\rho(m) = 98\%$, which gave $m = 15$ dimensions.

In the first case study, a walking jump of $0.8m$ was modified to produce a greater jump, by attaching one key-constraint on the character root node, at frame 7, which is the time where the character starts the jump, and move the constraint slightly forward in the jump direction. By using the joint approach, we attached one constraint on the root, at frame 7, activated from frames 1 to 26. The end-effector's final goal was the same in both systems. Figures 6.17 and 6.16 show the generated motions from key-trajectory (joint space) and key-frame (latent space) constraints. The PIK-based system always produced motions with foot sliding and the generated jumps were basically of the same length of the original one, as shown in Figure 6.17. On the contrary, the LPIK-based system generated a greater jump without foot sliding, as we



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

Figure 6.15: Synthesis comparison joint vs. latent space: golf swing up slope ground. joint and latent space results are shown in the left and right sides, respectively.

can see in Figure 6.16. Note that, a greater jump requires more speed to be performed, although speed is not measured, we can perceive its influence because the character tilts the trunk slightly forward to increase the jump length.

The aim of the second case study is to produce a higher walking jump motion. To achieve this goal, the same walking jump of $0.8m$ was modified, by also attaching one constraint on the character root node, at frame 15, which is the time of the apex of the jump, and moving the constraint to a higher location. The end-effector's final goal was the same in both systems. Figure 6.18 shows the generated motions from key-trajectory and key-frame constraints. The PIK-based system produced a higher jump motion, but by moving the constraint to a higher position made the character lose ground contact, which is not desired because its not natural. On the other hand, LPIK-based system not only generated a higher jump, but also the resulted animation kept ground contact.

In the third case study, we compared the techniques used to impose continuity from key-trajectory constraints in the latent and joint spaces. We considered a simple task: the character

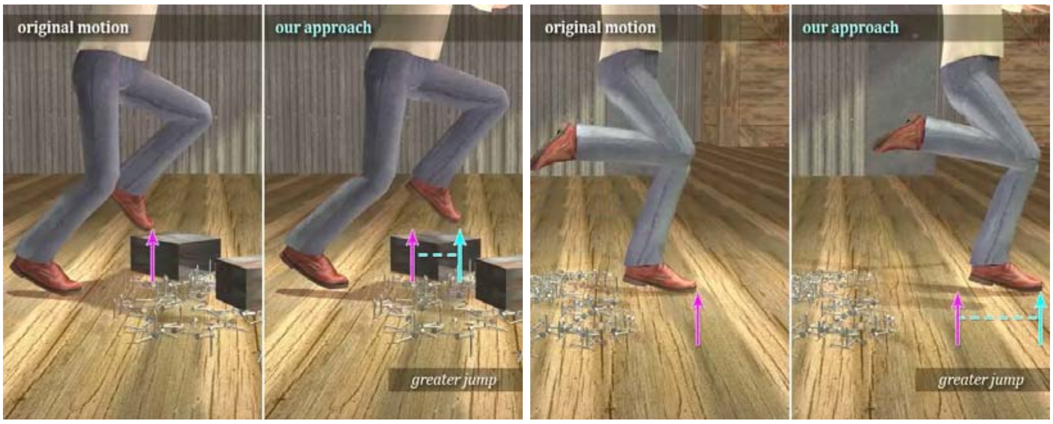


Figure 6.16: The latent approach generated a greater jump without foot sliding.

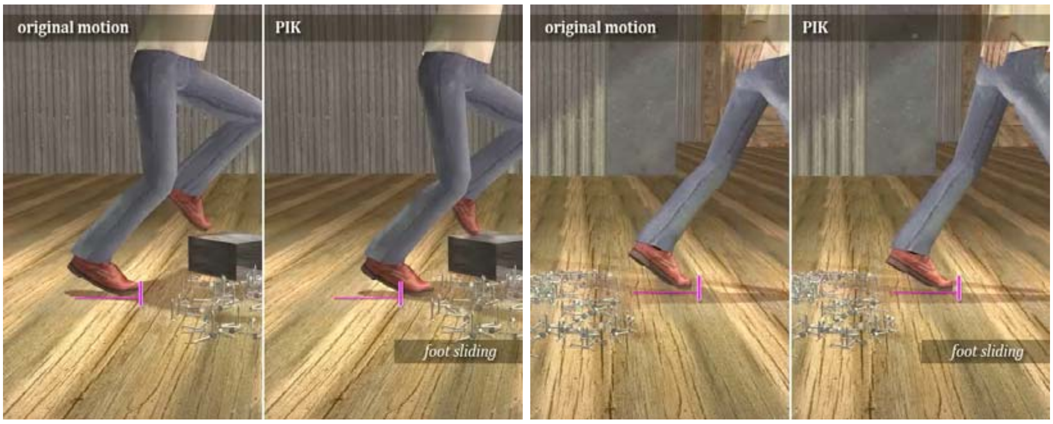
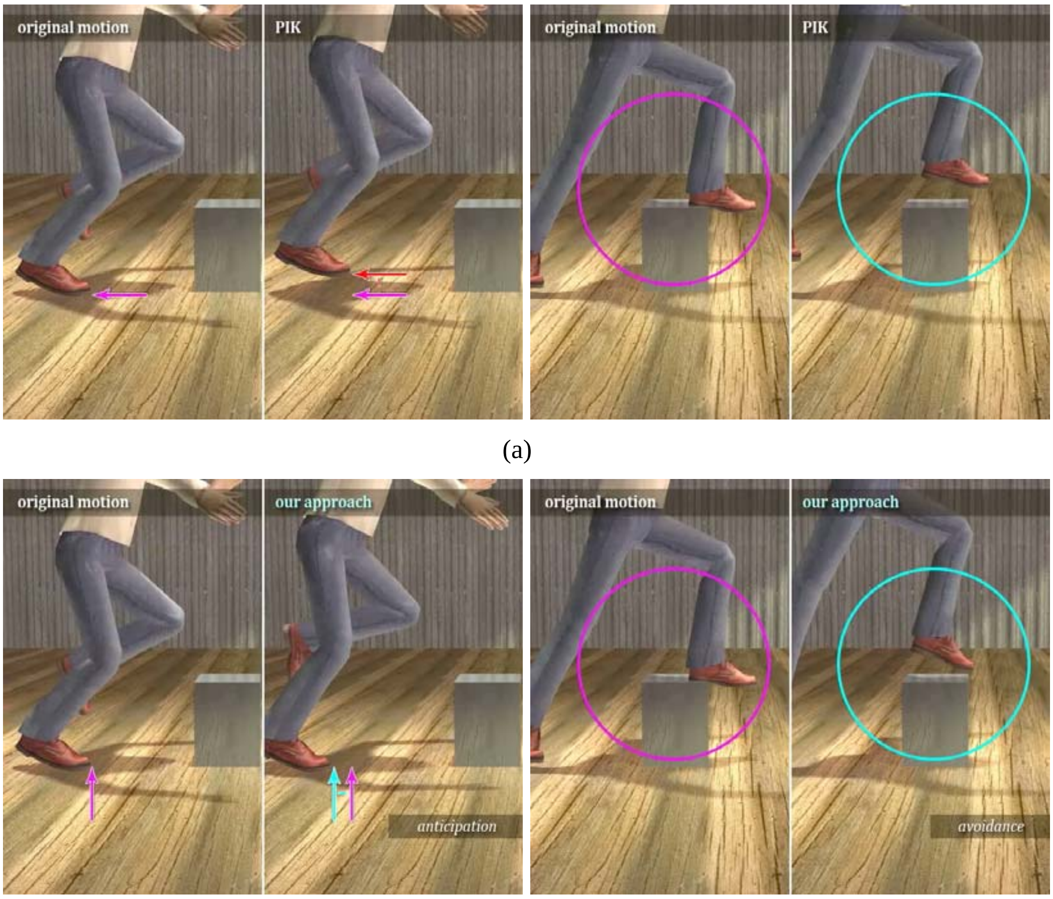


Figure 6.17: The joint approach generated a smaller jump with foot sliding.

has to reach an object. We then attached one constraint on the character's hand, at frame 50, to drive the reach in the direction of the goal. The constraint was activated over all frames because the character had to follow the path from start to end. This constraint configuration was used in both systems. We observed in Figure 6.19 that the PIK-based system generated a motion in which the character slides to reach the green bar. On the other hand, the synergy encapsulated in the LPIK solver generated an animation without introducing motion artifacts (i.e., foot slides.).

In this section, we analyzed the strengths of model-based against goal-direct continuity. We verified, for motions that need to be edited at a specific time, that continuity from key-frame constraints can generated better quality motions with less user tuning compared to continuity from key-trajectory constraints. For example, to edit a walking jump, the user has just to move a root constraint forward to say: perform a greater jump. Maybe, by using the joint approach, the system can generate the same natural-looking motions, but the user may have to specify a very detailed set of constraints over the entire motion, which can be cumbersome. Moreover, the model-based key-trajectory continuity technique demonstrate improved results, compared

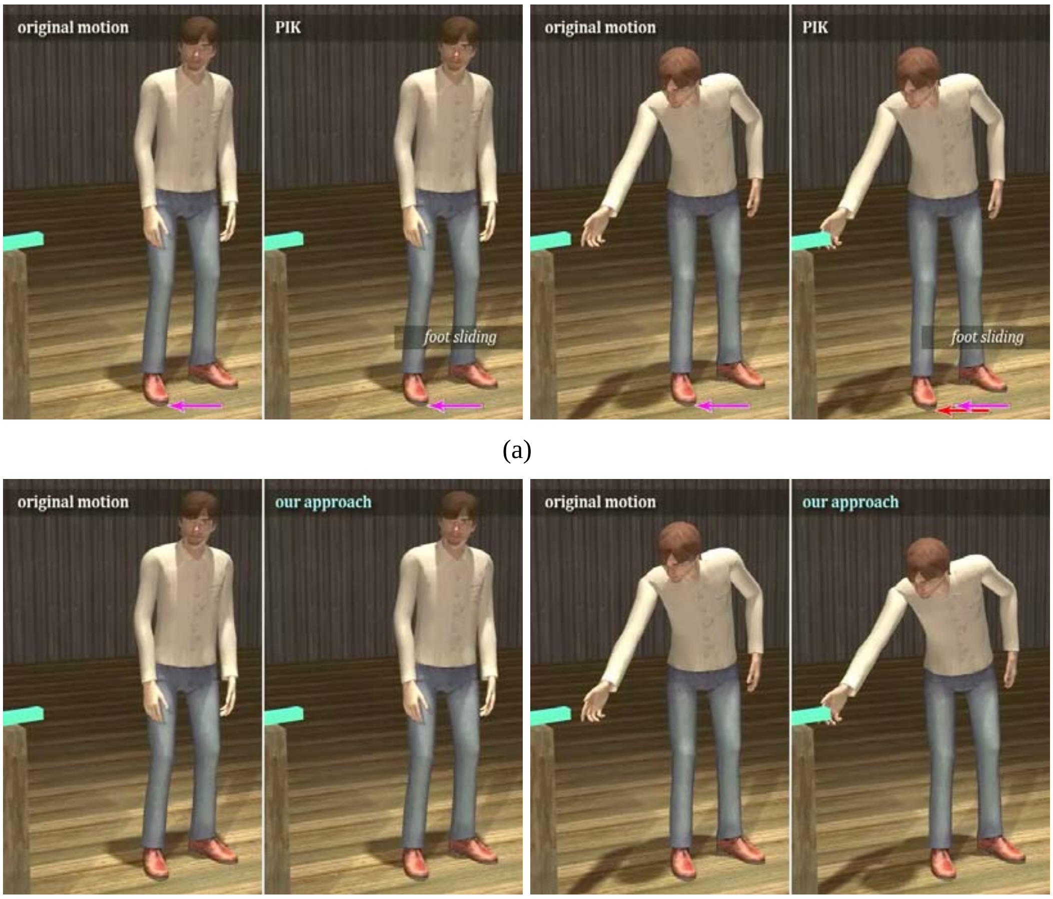


(a)

(b)

Figure 6.18: Synthesis comparison joint vs. latent space: higher walking jump. (a) Input and generated motion by using the PIK-based system. (b) The same for the proposed approach.

to the joint space technique, generating more natural-looking motions with less user-defined constraints.



(a)

(b)

Figure 6.19: Synthesis comparison joint vs. latent space: reach. (a) Input and generated motion by using the PIK-based system. (b) The same for the LPIK-based approach.

6.5 Case Study

Aiming to explore the usability of the proposed method in a more realistic context, we have elaborated a case study to simulate editing tasks that can occur in practice. In some cases, the animator has to adapt a sequence of pre-recorded movements to a new environment. For example, the animator could be asked to retarget a sequence of reaching motions to an environment containing a shelf surrounded by obstacles that should be avoided. We chose the reaching example because this type of movement can be edited with both key-frame and key-trajectory constraints. Thereby, the evaluation regarding the handling of both constraint types can be more accurate.

In order to have an environment as close as possible of a real situation, we asked an experienced designer to build a shelf with objects to be reached and add obstacles between the trajectories and the goals. We show in Figure 6.20 the designed environment, the green cubes represent the goals to be reached. The environment was built based on the pre-recorded reaching motions illustrated in Figure 6.21. Each motion is 2.0 seconds long, which gives 50 frames. For this database, we considered a percentage $\rho(m) = 98\%$, which gave $m = 15$ dimensions, and we set the optimizer's parameters to 15 iterations and $\xi = 5$. When we used key-frame constraints, the spline parameters were set to: $(1.0, 0.0, 0.0)$ for the tension, continuity and bias. We chose these values because we want a smooth trajectory, as illustrated in Figure 5.5 in section 5.3.2.

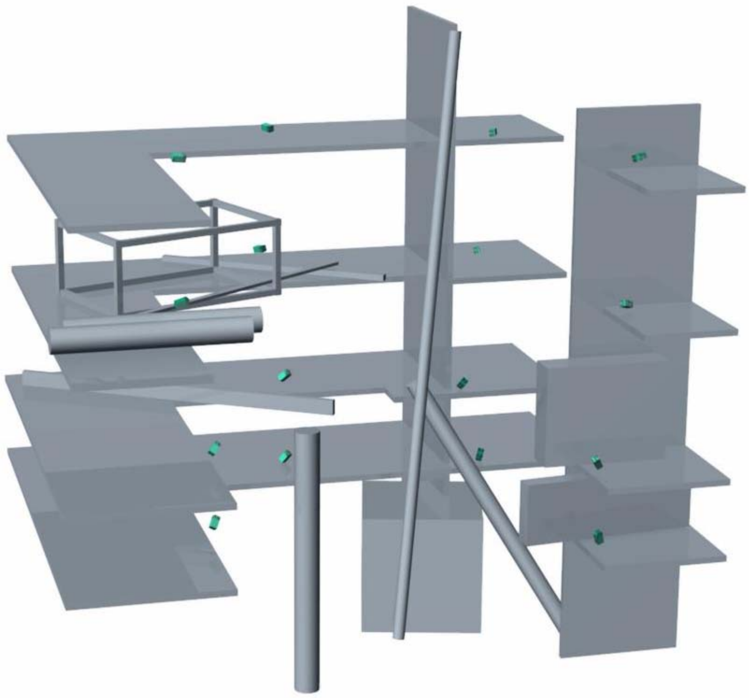


Figure 6.20: Reaching scenery.

Our task consisted in producing new motions in which the character had to avoid the ob-

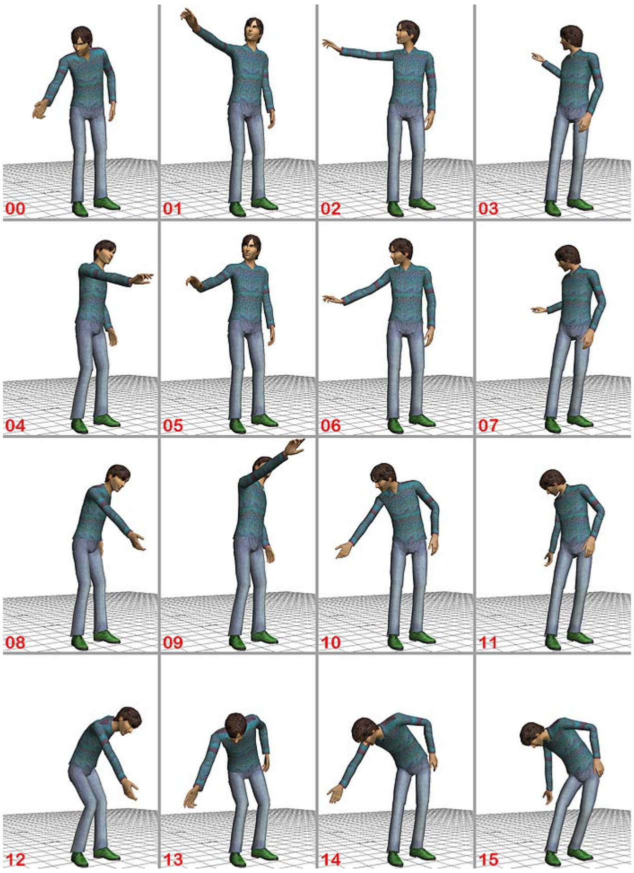
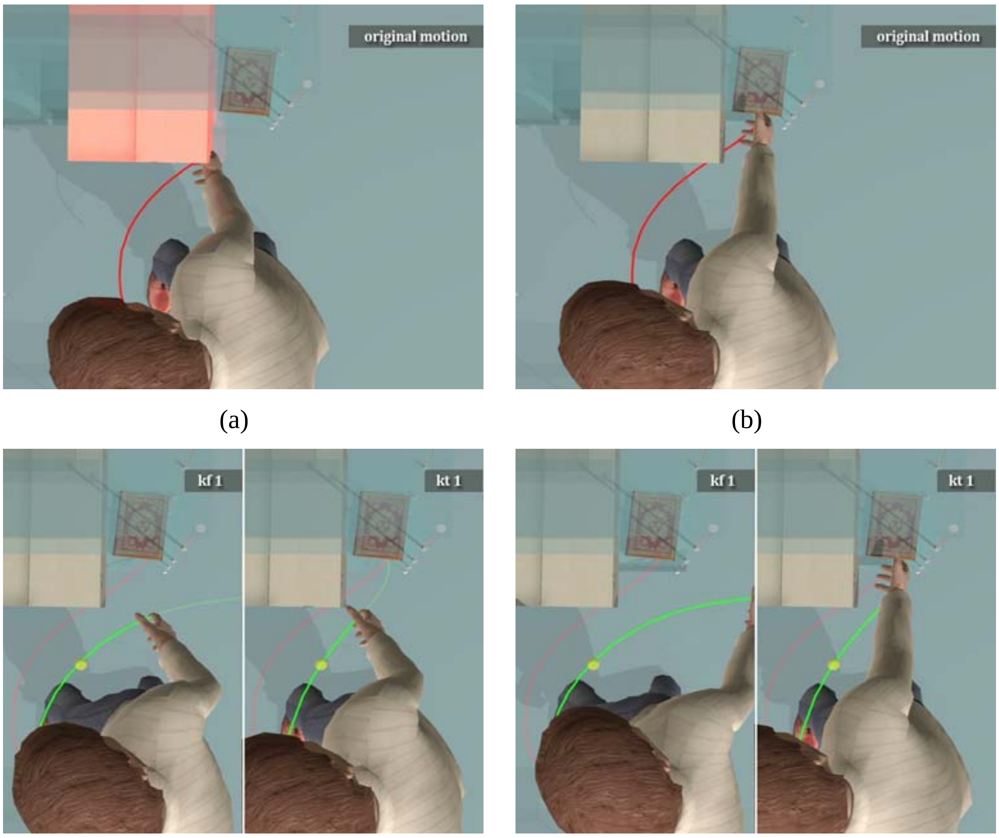


Figure 6.21: Reaching database. Final pose of each motion.

stacles while reaching the desired goal. To perform each editing, we started using a reduced set of constraints, for example, one key-trajectory constraint attached on the character's hand to guide the reaching towards the goal. If one constraint was insufficient to make the character both avoid the obstacle and reach the goal, we slightly increased the number of constraints for tuning the trajectory. In situations in which foot sliding were observed, we created foot constraints. The purpose was to produce the requested animation using the minimum amount of constraints.

The average time taken editing each motion was approximately 6 minutes. Most of the time was spent testing different constraint configurations, for example, constraining different key poses, modifying effector's goal, changing the duration of trajectories - in the case of key-trajectory constraints - and visualizing the naturalness the resulted animations. The objective was to investigate strengths and possible limitations of the proposed approach.

For the given scenery, we could edit most of the animations by using two key-trajectory constraints, one attached on the character's hand and another on the right foot. The results were natural-looking. From our experience, for reaches that needed just the addition of one constraint to be edited, key-trajectory constraints were more efficient and intuitive to manipulate compared to key-frame constraints. In fact, the major advantage of using key-trajectory



(a) (b)

(c) (d)

Figure 6.22: (a)(b) The red line represents the original right hand's trajectory. (c)(d) Yellow ball represents the constraint's position. The right side shows the respective deformed trajectory, yellow line, resulted from the application of one key-trajectory constraint. The hand avoided the obstacle and reached the goal. The left side shows the final hand's trajectory obtained from the application of one key-frame constraint. The character avoided the obstacle, but was unable the reach the goal. From left to right keys 35 (obstacle avoidance) and 50 (goal to be reached).

constraints is that they can generate the complete trajectory in just one step, reducing the work of the animator. We show in Figure 6.22 an editing task that required the use of just one key-trajectory constraint to generate the desire movement. Figure 6.22(a) and 6.22(b) show the poses of the original motion in which the character hits the obstacle while reaching. The right hand's trajectory is represented by the red line. The right side of Figures 6.22(c) and 6.22(d) shows the respective deformed poses and the right hand's trajectory (yellow line) resulted from the application of one key-trajectory constraint at frame 25. The character avoided the obstacle and reached the goal. Unfortunately, this benefit is lost when one key-frame constraint is used instead, because the dynamics of the movement tends to follow the local modifications. We can see in the left side of Figures 6.22(c) and 6.22(d) the same example edited with one key-frame constraint applied at key 25: the character's hand followed the direction of the adaptation. Although, key-trajectory constraints provide this practical solution, sometimes it was difficult to

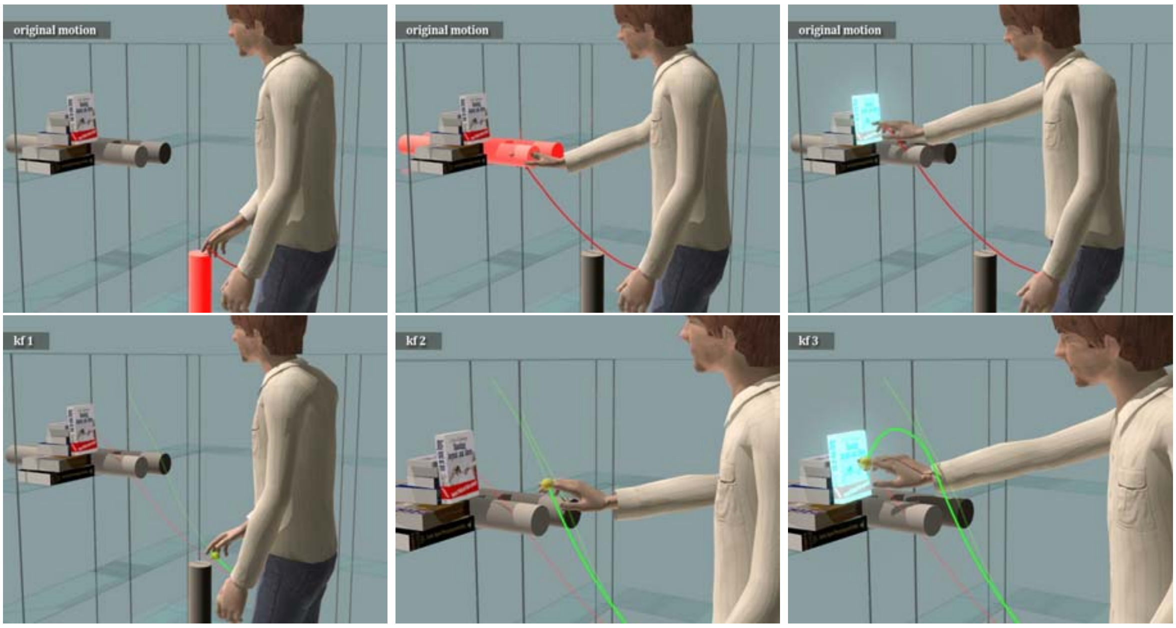


Figure 6.23: Multiple key-frame editing. Top row shows the input motion. The read line represents the original right hand's trajectory. Bottom row shows the deformed motion. The yellow lines show the trajectories obtained from the application of one, two and three key-frame constraints. From left to right keys 16, 32 and 50.

find a smooth path controlling three trajectories attached to the same effector, for example, for situations in which the character had to avoid two obstacles while reaching the final goal.

On the other hand, for motions that required more tuning (e.g., editing multiple poses) key-frame constraints demonstrated a more intuitive handling compared to key-trajectory. Essentially, we needed simply to sketch the poses for obstacle avoidance and reaching, for example, by attaching constraints on the character's hand and dragging them to new positions. Then set the spline's parameters to obtain the desired full-body animation. In particular, we show in Figure 6.23 one example that required more tuning. The character has to avoid two obstacles and reaches the book in the end of the reach. To produce the desire animation, we attached two key-frame constraints on the character's hand at frames 16 and 27 for obstacle avoidance and another at frame 50 to reach the goal. In addition, we also attached one key-frame constraint with higher priority on the right foot at frame 16, for removing foot sliding. In general, a foot constraint attached on one pose was sufficient to clean up foot sliding from all other poses. The proposed framework was capable to generate natural-looking full-body poses according to these local modifications. Finally, the full-body motion was automatically released respecting the three key-frame constraints, using the spline interpolation schema described in section 5.3.2.

6.6 Conclusion

In this chapter, we have given several examples that illustrate the use of a model-based PIK framework, which incorporates a task-priority strategy to deal with conflicting tasks in the latent space. The priority concept is interesting because it is intuitive. In other words, everybody has a clear idea of what “priority ”means: it occurs in many situations of everyday life. An important feature of the algorithm is that while respecting the priority order, it also provides least-squares solutions for each task, which means that the solution is optimal (for example, it minimizes the distance between the end-effector and its goal).

The integration of a motion model brought compelling advantages for constraint-based motion editing systems:

- the system could release natural-looking motions from a reduced set of user-defined constraints favoring less experienced users;
- the performance of the LPIK solver is more related with the number of user-specified constraints and the model dimensions. This favors the use of more complex skeletons without compromising performance.
- for motions that need to be edited at a specific time, the key-frame constraints approach demonstrated a more intuitive editing strategy compared to the key-trajectory one.

The system behaves well and robustly, as long as the model has sufficient information to handle the user-specified constraints. Table 6.6 summarizes the strengths/drawbacks of the techniques and models investigated in this thesis. These information are based both on the results reported in this chapter and on our personal experience with the proposed motion editing framework.

Experiment	Strengths	Drawbacks
Local models	Provides faster convergence compared to global models.	Constraints a single motion activity.
Global models	Capable of handling more motion activities.	Provides slower convergence compared to local models.
Joint space	Capable of constraining a wider range of user-specified constraints compared to the latent space. Do not need a motion database.	Sometimes provides a lack of naturalness when it comes to reproduce human activities adding motion artifacts.
Latent space	Provides more natural-looking solutions. Releases faster motion solutions compared to the joint space. Requires less user-specified constraints.	Reliability of the solution depends of the database.
Key-frame	Requires less computation time to release a full-body animation. More intuitive for editing multiple keys.	The dynamics of the movement follows the local modifications. This may be undesirable for some motion activities, for example, reaching.
Key-trajectory	Generate a complete trajectory in one step.	Sometimes cumbersome when tuning multiple connected trajectories.

Table 6.6: Summarizes the strengths/drawbacks of the techniques and models investigated in this thesis.

Chapter 7

Conclusions and Future Work

In this thesis, we have proposed a new approach for data-driven constraint-based motion editing. Our technique is based on the link between linear motion models such as Principal Component Analysis (PCA) and Prioritized Inverse Kinematics (PIK). The connection of both techniques allowed us to construct a Low-dimensional Prioritized Inverse Kinematics (LPIK) framework. The solver handles deformation problems within the latent space of some underlying motion pattern. By making use of the pattern space, we increased the possibilities of performing IK in the space of feasible poses.

We have shown that by constraining and adjusting just one key frame our approach provided faster and more fluid results without the need of considering ease-in/ease-out time intervals. We have seen that this feature is important for motions that need to be edited at a specific time. Also, we have demonstrated that, building *motion models* instead of *pose models* demonstrated two important advantages: (1) our system could automatically synthesize the movement as a whole without introducing motion artifacts, and (2) continuity between frames was imposed in a much straightforward way improving the performance of the system. Furthermore, system performance could also be improved by using local models instead of multi-pattern models and by giving constraints different priorities

We have demonstrated that our method achieves a degree of generality beyond the motion capture data. For example, we have retargeted a golf swing motion to 7° up and down slopes using constraints that cannot be satisfied directly by any motion in the database, and have found that the quality of the generated motion was believable. We have seen that our approach is well-suited to deal with deformations and retargeting problems. We have focused our study on modifying motions that need a great precision (for example, golf swings and walking jumps) to demonstrated the robustness of our approach, deforming these type of movements, compared to constraint-based techniques that perform optimizations in the joint space. Furthermore, by using our approach, the user can also make adjustments in the motion by constraining multiple frames. In this mode, the animator can carry out motion adaptations from effector trajectories or from individual key-frame adjustments.

The addition of one key-frame constraint on the character's foot was generally sufficient to clean up foot sliding from all other poses. However, in some situations we have observed that one constraint applied on a specific pose was insufficient to completely remove foot sliding of all poses. To handle this issue, we have used key-trajectory constraints instead.

Another issue noticed during the experiments was that, although we have used database always containing balanced and natural-looking motions, we have observed that some solutions produced either unbalanced animations or poses that violated joint limits. To perform the desired task, we handled this problem by modifying the setup of the constraints, that is:

- by constraining different sets of frames on the motion, or
- by increasing/decreasing the activation/desactivation of effector trajectories - in the case of key-trajectory constraints - or
- by decreasing the displacement of effectors to reduce the risks of extrapolations.

In situations where constraints could not be easily satisfied due to model specialization (e.g., models learned from database containing walking jumps of one length), we have increased the style variability of the database by adding more training data (e.g., jumps with different lengths). It is important to stress that, building the appropriate model is an assignment related with the task, which can be time consuming if a good database is not available.

The motion normalization process described in this thesis may present some limitations for handling complex human activities such as getting into a car. Such a type of movement presents at least two challenges. First, subjects perform a large variety of movements that depends on both the experience of each person and the geometrical characteristics of the vehicle. Therefore, a global motion normalization that would not take into account the individual strategies, could produce incorrect outcome resulting from the combination of uncorrelated strategies. This issue can be alleviated by first identifying clusters of movements performed according to the same strategy. The second challenge is linked to the normalization stage per se for which a unidimensional set of keyframe is used to align the movement according to a single pattern as in [Dufour and Wang, 2005, Urtasun et al., 2005]. Such an approach can face some limitations when the events associated to the keyframes are triggered by differing body part (e.g. feet and hands). Indeed, it may happen that, within a movement cluster characterizing a given strategy, these events are not always ordered into the same sequence. Further studies are necessary to address these data structuration issues.

The next necessary research step is the creation of new algorithms for real-time motion control. A direct consequence of that should be the possibility of applying the techniques developed in this thesis in applications that need real-time responses such as motion capturing, games, robot navigation and real-time crowd simulation. Another extension should be the integration of joint limits to handle the range of motion of the character's joints.

Another kind of application that our LPIK solver can improve is model-based human body tracking. The simplest approach to combine both techniques would be to use the tracker to send an initial estimation of the principal coefficients and a set of geometrical constraints, and the task of the solver would be to release the final principal coefficients satisfying all the constraints. We have seen in section 6.4.1 that the system took 5.3 milliseconds per iteration and in section 6.5 that the proposed framework have released believable motions with 12 iterations. Thereby, in the present moment the proposed framework is capable to compute approximately $13fps$.

Another area that requires further investigation is the extrapolation of the model parameters, to verify how far the user-specified constraints can deviate from the motions in the database before the quality of the resulting animation declines to an unacceptable level. We also pretend to investigate the combination of key-frame and key-trajectory constraints. This connection can improve the animator's work, for example, he/she can constraint all the poses for removing foot sliding with one key-trajectory constraint and then edit individual frames separately using key-frame constraints. Finally, another important issue that needs improvements is the development of a more intuitive user interface to visualize the deformation results in the latent space. An interface like that can guide users improving their constraints because they could visualize the optimized principal coefficients path and see in which motion cluster they belong.

Appendix A

Background Mathematics

A.1 Principal Component Analysis

A.1.1 Optimal Reconstruction

One important property of PCA deals with the reconstruction of M from $\mathbf{E}$. Because the rows of α are orthonormal vectors, then $\alpha^{-1} = \alpha^{\mathrm{T}}$, and any vector $\boldsymbol{\Psi}$ of M can be reconstructed from $\mathbf{E}$ by using the expression from Eq. 3.10:

$$\mathrm{M} = \alpha\mathbf{E} + \boldsymbol{\Psi}_{\circ}. \tag{A.1}$$

Now suppose that instead of using all the eigenvectors of $\widetilde{\mathbf{S}}$ we construct a matrix α_m from the m largest eigenvectors corresponding to the m largest eigenvalues, leading a transformation matrix of order $(d \times m)$. Therefore, the $\mathbf{E}_m$ vectors would them be $(m \times D)$ dimensional, and the reconstruction given in Eq. A.1 would no longer be exact. The vector reconstructed by using α_m is

$$\widehat{\mathbf{M}} = \alpha_m\mathbf{E}_m + \boldsymbol{\Psi}_{\circ}. \tag{A.2}$$

By manipulating Eqs. 3.9 and A.1, and considering that the last $(d - m)$ vectors of $\mathbf{E}_m$ and α_m are zero, it can be shown that the mean square error between M and $\widehat{\mathrm{M}}$ is given by the expression [Gonzalez and Woods, 2002]:

$$
\begin{aligned}
MSE &= \sum_{i=1}^{d} [\mathbf{M} - \widehat{\mathbf{M}}]^2 \\
&= (\mathbf{M} - \widehat{\mathbf{M}})(\mathbf{M} - \widehat{\mathbf{M}})^{\mathrm{T}} \\
&= \alpha \mathbf{E}\mathbf{E}^{\mathrm{T}} \alpha^{\mathrm{T}} - \alpha \mathbf{E}\mathbf{E}_m^{\mathrm{T}} \alpha_m^{\mathrm{T}} - \alpha_m \mathbf{E}_m \mathbf{E}^{\mathrm{T}} \alpha^{\mathrm{T}} + \alpha_m \mathbf{E}_m \mathbf{E}_m^{\mathrm{T}} \alpha_m^{\mathrm{T}} \\
&= \sum_{i=1}^{d} \lambda_i - \sum_{i=1}^{m} \lambda_i - \sum_{i=1}^{m} \lambda_i + \sum_{i=1}^{m} \lambda_i \\
&= \sum_{i=1}^{d} \lambda_i - \sum_{i=1}^{m} \lambda_i \\
&= \sum_{i=m+1}^{d} \lambda_i. \qquad \text{(A.3)}
\end{aligned}
$$

The next to last line of Eq. A.3 indicates that the error is zero if $m = d$ (i.e., if all the eigenvectors are used in the reconstruction). As the λ_i‘s decrease monotonically, Eq. A.3 also shows that the error can be minimized by selecting the m eigenvectors associated with the largest eigenvalues. Thus the PCA is optimal in the sense it minimizes the mean square error between the vectors $\mathbf{M}$ and its approximations $\widehat{\mathbf{M}}$.

Appendix B

HAnim

B.1 Level of articulation

We show the two levels of articulation used to construct the two human figures. Table B.2 shown the joints used to model the character with 90DoFs and Table B.1 the joints used to model the character with 222DoFs. Each joint is modeled by a single 3-dimensional exponential map.

HumanoidRoot	HumanoidRoot Pos	sacroiliac	l hip	l knee
l ankle	l metatarsal	l heel	r hip	r knee
r ankle	r metatarsal	r heel	vl5	vl3
vl1	vt10	vt6	vt1	vc4
vc2	skullbase	l shoulder	l elbow	l wrist
l hand	r shoulder	r elbow	r wrist	r hand

Table B.1: Level of articulation: 29 joints.

HumanoidRoot	HumanoidRoot Pos	sacroiliac	l hip	l knee
l ankle	l subtalar	l midtarsal	l metatarsal	l heel
r hip	r knee	r ankle	r subtalar	r midtarsal
r metatarsal	r heel	vl5	vl3	vl1
vt10	vt6	vt1	l sternoclavicular	l acromioclavicular
l shoulder	l elbow	l wrist	l index0	l index1
l index2	l index3	l middle0	l middle1	l middle2
l middle3	l pinky0	l pinky1	l pinky2	l pinky3
l ring0	l ring1	l ring2	l ring3	l thumb1
l thumb2	l thumb3	r sternoclavicular	r acromioclavicular	r shoulder
r elbow	r wrist	r index0	r index1	r index2
r index3	r middle0	r middle1	r middle2	r middle3
r pinky0	r pinky1	r pinky2	r pinky3	r ring0
r ring1	r ring2	r ring3	r thumb1	r thumb2
r thumb3	vc4	vc2	skullbase	

Table B.2: Level of articulation: 73 joints.

Bibliography

[Abe et al., 2004] Abe, Y., Liu, C. K., and Popović, Z. (2004). Momentum-based parameterization of dynamic character motion. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 173–182, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[Alexa and Muller, 2000] Alexa, M. and Muller, W. (2000). Representing animations by principal components. *Computer Graphics Forum*, 19(3):411–418.

[Arikan and Forsyth, 2002] Arikan, O. and Forsyth, D. (2002). Interactive motion generation from examples. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 483–490, New York, NY, USA. ACM.

[Baerlocher, 2001] Baerlocher, P. (2001). *Inverse Kinematics Techniques of The Interactive Posture Control of Articulated Figures*. Phd thesis, École Polytechnique Fédéral de Lausanne (EPFL) - IC School of Computer and Communication Sciences.

[Baerlocher and Boulic, 1998] Baerlocher, P. and Boulic, R. (1998). Task-priority formulations for the kinematic control of highly redundant articulated structures. In *International Conference on Intelligent Robots and Systems*, Practice and Applications (Cat. No.98CH36190), pages 323–329.

[Baerlocher and Boulic, 2004] Baerlocher, P. and Boulic, R. (2004). An inverse kinematic architecture enforcing an arbitrary number of strict priority levels. *The Visual Computer*, 20(6).

[Beck and Chaffin, 1992] Beck, D. J. and Chaffin, D. B. (1992). An evaluation of inverse kinematics models for posture prediction. In *Computer Applications in Ergonomics, Occupational Safety and Health, Elsevier*, volume ., pages 329–336.

[Ben-Israel and Greville, 1974] Ben-Israel, A. and Greville, T. N. E. (1974). *Generalized Inverses: Theory and Applications*. Wiley.

[Boulic and Thalmann, 1992] Boulic, R. and Thalmann, D. (1992). Combined direct and inverse kinematic control for articulated figure motion editing. *Computer Graphics Forum*, 11(4):189–202. Comput. Graphics Lab., Swiss Federal Inst. of Technol., Lausanne, Switzerland.

[Boulic et al., 1990] Boulic, R., Thalmann, N. M., and Thalmann, D. (1990). A global human walking model with real-time kinematic personification. *Visual Computer*, 6(6):344–58. Comput. Graphics Lab., Swiss Federal Inst. of Technol., Lausanne, Switzerland.

[Boulic et al., 2004] Boulic, R., Ulicny, B., and Thalmann, D. (2004). Versatile Walk Engine. *Journal Of Game Development*, 1(1):29–52.

[Boullion and Odell, 1971] Boullion, T. and Odell, P. (1971). *Generalized Inverse Matrices*. John Wiley & Sons.

[Bruderlin and Williams, 1995] Bruderlin, A. and Williams, L. (1995). Motion signal processing. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 97–104, New York, NY, USA. ACM.

[Calinon and Billard, 2005] Calinon, S. and Billard, A. (2005). Recognition and Reproduction of Gestures using a Probabilistic Framework combining PCA, ICA and HMM. In *22nd International Conference on Machine Learning*, pages 105–112.

[Callennec and Boulic, 2006] Callennec, B. L. and Boulic, R. (2006). Interactive motion deformation with prioritized constraints. *Graphical Models*, 68(2):175–193. Special Issue on SCA 2004.

[Carvalho et al., 2007] Carvalho, S., Boulic, R., and Thalmann, D. (2007). Interactive low-dimensional human motion synthesis by combining motion models and pik. *Computer Animation & Virtual Worlds*, 18. Special Issue of Computer Animation and Social Agents (CASA2007).

[Chai and Hodgins, 2005] Chai, J. and Hodgins, J. (2005). Performance animation from low-dimensional control signals. *ACM Trans. Graph.*, 24(3):686–696.

[Chai and Hodgins, 2007] Chai, J. and Hodgins, J. (2007). Constraint-based motion optimization using a statistical dynamic model. *ACM Trans. Graph.*, 26(3):8.

[Chiaverini, 1997] Chiaverini, S. (1997). Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. In *IEEE Transactions on Robotics and Automation*, volume 13, pages 398–410.

[Choi and Ko, 2000] Choi, K. and Ko, H. (2000). On-line motion retargetting. *Journal of Visualization and Computer Animation*, 11:223–235.

[CMU, 2009] CMU (2009). Carnegie Mellon University (CMU) Graphics Lab Motion Capture Database. http://mocap.cs.cmu.edu/.

[Cohen, 1992] Cohen, M. F. (1992). Interactive spacetime control for animation. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 293–302, New York, NY, USA. ACM.

[Craig, 1986] Craig, J. J. (1986). *Introduction to Robotics*. Addison-Wesley, Reading MA.

[Dufour and Wang, 2005] Dufour, F. and Wang, X. (2005). Discomfort assessment of car ingress/egress motions using the concept of neutral movement. In *SAE International conference and exposition of Digital Human Modeling for Design and Engineering*.

[Engin and Chen, 1986] Engin, A. E. and Chen, S. M. (1986). Statistical data base for the biomechanical properties of the human shoulder complex–i: Kinematics of the shoulder complex. *Journal of Biomechanical Engineering*, 108(3):215–221.

[Farrally et al., 2003] Farrally, M., Cochran, A., Crews, D., Hurdzan, M., Price, R., Snow, J., and Thomas, P. (2003). Golf sscience research at the beginning of the twenty-first century. *Journal of Sports Sciences*, 21:753–765.

[Fleishman and Endler, 2000] Fleishman, L. J. and Endler, J. A. (2000). Some comments on visual perception and the use of video playback in animal behavior studies. *Acta ethologica*, 3(1):15–27.

[Gehrig et al., 2003] Gehrig, N., Lepetit, V., and Fua, P. (2003). Golf club visual tracking for enhanced swing analysis tools. In *British Machine Vision Conference, Norwich, UK*.

[Glardon et al., 2004a] Glardon, P., Boulic, R., and Thalmann, D. (2004a). A coherent locomotion engine extrapolating beyond experimental data. In *Proceedings of CASA*, pages 73–84.

[Glardon et al., 2004b] Glardon, P., Boulic, R., and Thalmann, D. (2004b). A Coherent Locomotion Engine Extrapolating Beyond Experimental Data. In *Computer Animation and Social Agents (CASA)*, pages 73–84.

[Glardon et al., 2006a] Glardon, P., Boulic, R., and Thalmann, D. (2006a). Dynamic obstacle avoidance for real-time character animation. *Vis. Comput.*, 22(6):399–414.

[Glardon et al., 2006b] Glardon, P., Boulic, R., and Thalmann, D. (2006b). Robust on-line adaptive footplant detection and enforcement for locomotion. *Vis. Comput.*, 22(3):194–209.

[Gleicher, 1997] Gleicher, M. (1997). Motion editing with spacetime constraints. In *In Proceedings of SI3D '97*, pages 139–ff., New York, NY, USA. ACM Press.

[Gleicher, 1998] Gleicher, M. (1998). Retargetting motion to new characters. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 33–42, New York, NY, USA. ACM.

[Gleicher, 2001] Gleicher, M. (2001). Comparing constraint-based motion editing methods. *Graphical models*, 63(2):107–134.

[Gonzalez and Woods, 2002] Gonzalez, R. C. and Woods, R. E. (2002). *Digital Imaging Processing*. Tom Robbins, second edition.

[Grassia, 1998] Grassia, F. S. (1998). Practical parameterization of rotations using the exponential map. *Journal of Graphics Tools*, 3(3):29–48.

[Grochow et al., 2004] Grochow, K., Martin, S. L., Hertzmann, A., and Popovi, Z. (2004). Style-based inverse kinematics. *ACM Trans. Graph.*, 23(3):522–531.

[Grzeszczuk and Terzopoulos, 1995] Grzeszczuk, R. and Terzopoulos, D. (1995). Automated learning of muscle-actuated locomotion through control abstraction. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 63–70, New York, NY, USA. ACM.

[H-Anim, 2009] H-Anim (2009). Humanoid Animation Work Group.

[Hanafusa et al., 1981] Hanafusa, H., Yoshikawa, T., and Nakamura, Y. (1981). Analysis and control of articulated robot with redundancy. In *IFAC, 8th Triennal World Congress*, volume 4, pages 1927–1932.

[Hastie et al., 2009] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning Data Mining, Inference, and Prediction*. Number XXII in Springer Series in Statistics. Springer, second edition.

[Herda, 2003] Herda, L. (2003). *Using Biomechanical Constraints to Improve Video-based Motion Capture*. Phd thesis, École Polytechnique Fédéral de Lausanne (EPFL) - IC School of Computer and Communication Sciences.

[Ikemoto et al., 2009] Ikemoto, L., Arikan, O., and Forsyth, D. (2009). Generalizing motion edits with gaussian processes. *ACM Trans. Graph.*, 28(1):1–12.

[Jain et al., 1999] Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323.

[Jolliffe, 1986] Jolliffe, I. T. (1986). *Principal Component Analysis*. Springer-Verlag.

[Kallmann, 2008] Kallmann, M. (2008). Analytical inverse kinematics with body posture control. *Computer Animation and Virtual Worlds*, 19(2):79–91.

[Klein and Huang, 1983] Klein, C. A. and Huang, C. H. (1983). Review of pseudoinverse control for use with kinematically redundant manipulators. In *IEEE Transactions on Systems, Man and Cybernetics*, volume 13, pages 245–250. IEEE Systems, Man, and Cybernetics Society.

[Kochanek and Bartels, 1984] Kochanek, D. H. U. and Bartels, R. H. (1984). Interpolating splines with local tension, continuity, and bias control. *SIGGRAPH Comput. Graph.*, 18(3):33–41.

[Korein, 1985] Korein, J. U. (1985). *A Geometric Investigation of Reach*. MIT press.

[Kovar and Gleicher, 2003] Kovar, L. and Gleicher, M. (2003). Flexible automatic motion blending with registration curves. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 214–224, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[Kovar and Gleicher, 2004] Kovar, L. and Gleicher, M. (2004). Automated extraction and parameterization of motions in large data sets. *ACM Trans. Graph.*, 23(3):559–568.

[Kovar et al., 2002] Kovar, L., Gleicher, M., and Pighin, F. (2002). Motion graphs. *ACM Trans. Graph.*, 21(3):473–482.

[Kulpa et al., 2005] Kulpa, R., Multon, F., and Arnaldi, B. (2005). Morphology-independent representation of motions for interactive human-like animation. In *EUROGRAPHICS*, volume 24, pages 343–352.

[Lee and Shin, 1999] Lee, J. and Shin, S. (1999). A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of ACM SIGGRAPH*.

[Liegeois, 1977] Liegeois, A. (1977). Automatic supervisory control of the configuration and behavior of multibody mechanisms. In *IEEE Transactions on Systems, Man and Cybernetics*, volume 7, pages 868–871. IEEE Systems, Man, and Cybernetics Society.

[Liu and Popović, 2002] Liu, C. K. and Popović, Z. (2002). Synthesis of complex dynamic character motion from simple animations. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 408–416, New York, NY, USA. ACM.

[Liu et al., 2006] Liu, L., Zhao-qi, W., Deng-Ming, Z., and Shi-Hong, X. (2006). Motion edit with collision avoidance. In *Proceedings of the WSCG*, pages 303–310.

[Liu et al., 1994] Liu, Z., Gortler, S. J., and Cohen, M. F. (1994). Hierarchical spacetime control. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 35–42, New York, NY, USA. ACM.

[Maciejewski, 1990] Maciejewski, A. (1990). Dealing with the ill-conditioned equations of motion forarticulated figures. In *Computer Graphics and Applications, IEEE*, volume 10, pages 63–71.

[Maciejewski and Klein, 1988] Maciejewski, A. and Klein, C. (1988). Numerical filtering for the operation of robotic manipulators through kinematically singular configurations. In *Journal of Robotic Systems*, volume 15, pages 527–552.

[Maciejewski and Klein, 1985] Maciejewski, A. A. and Klein, C. A. (1985). Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. *International Journal of Robotic Research*, 4(3):109–117.

[Monzani et al., 2000] Monzani, J. S., Baerlocher, P., Boulic, R., and Thalmann, D. (2000). Using an Intermediate Skeleton and Inverse Kinematics for Motion Retargeting. *Computer Graphics Forum*, 19(3):11–19.

[Mukai and Kuriyama, 2005] Mukai, T. and Kuriyama, S. (2005). Geostatistical motion interpolation. *ACM Trans. Graph.*, 24(3):1062–1070.

[Nakamura and Hanafusa, 1986] Nakamura, Y. and Hanafusa, H. (1986). Inverse kinematic solutions with singularity robustness for robot manipulator control. *Journal of Dynamic Systems, Measurement, and control*, 10(8):163–171.

[Nakamura et al., 1987] Nakamura, Y., Hanafusa, H., and Yoshikawa, T. (1987). Task-priority based redundancy control of robot manipulators. *International Journal of Robotics Research*, 6(2):3–15.

[Nashed, 1976] Nashed, Z. (1976). *Generalized Inverses and Applications*. Academic Press.

[Ngo and Marks, 1993] Ngo, J. T. and Marks, J. (1993). Spacetime constraints revisited. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 343–350, New York, NY, USA. ACM.

[Ortega and Rheinboldt, 1970] Ortega, J. and Rheinboldt, W. (1970). *Iterative solution of nonlinear equations in several variables*. New York, Academic Press.

[Park et al., 2002] Park, S., Shin, H., and Shin, S. (2002). On-line locomotion generation based on motion blending. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 105–111, New York, NY, USA. ACM.

[Paul, 1981] Paul, R. P. (1981). *Robot Manipulators: Mathematics, Programming and Control*. MIT press.

[Popović and Witkin, 1999] Popović, Z. and Witkin, A. (1999). Physically based motion transformation. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 11–20, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

[Press et al., 1992] Press, W., Teukolsky, S., Vetterling, W., and Flannery, B. (1992). *Numerical Recipes in C*. Cambridge University Press, second edition.

[Raunhardt and Boulic, 2009] Raunhardt, D. and Boulic, R. (2009). Motion constraint. *Vis. Comput.*, 25(5-7):509–518.

[Raymond, 2003] Raymond, A. (2003). The physics of golf. *Reports on Progress in Physics*, 66(2):131–171.

[Rose et al., 1998] Rose, C., Cohen, M., and Bodenheimer, B. (1998). Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, 18(5):32–40.

[Rose et al., 1996] Rose, C., Guenter, B., Bodenheimer, B., and Cohen, M. F. (1996). Efficient generation of motion transitions using spacetime constraints. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 147–154, New York, NY, USA. ACM.

[Rose and Gamble, 1994] Rose, J. and Gamble, J. G. (1994). *Human Walking*. Willians & Wilkins, second edition.

[Safonova and Hodgins, 2005] Safonova, A. and Hodgins, J. K. (2005). Analyzing the physical correctness of interpolated human motion. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 171–180, New York, NY, USA. ACM.

[Safonova et al., 2004] Safonova, A., Hodgins, J. K., and Pollard, N. S. (2004). Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Trans. Graph.*, 23(3):514–521.

[Shin and Lee, 2006] Shin, H. and Lee, J. (2006). Motion synthesis and editing in low-dimensional spaces. *Comput. Animat. Virtual Worlds*, 17(3&dash;4):219–227.

[Shoemake, 1985] Shoemake, K. (1985). Animating rotation with quaternion curves. In *SIGGRAPH '85*, pages 245–254, New York, NY, USA. ACM Press.

[Siciliano and Slotine, 1991] Siciliano, B. and Slotine, J. J. E. (1991). A general framework for managing multiple tasks in highly redundant robotic systems. In *International Conference on Advanced Robotics*, pages 1211–1216.

[Tolani et al., 2000] Tolani, D., Goswami, A., and Badler, N. I. (2000). Real-time inverse kinematics techniques for anthropomorphic limbs. *Graph. Models Image Process.*, 62(5):353–388.

[Unuma et al., 1995] Unuma, M., Anjyo, K., and Takeuchi, R. (1995). Fourier principles for emotion-based human figure animation. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 91–96, New York, NY, USA. ACM.

[Unzueta et al., 2008] Unzueta, L., , Peinado, M., , Boulic, R., and Suescun, A. (2008). Full-body performance animation with sequential inverse kinematics. *Graph. Models*, 70(5):87–104.

[Urtasun et al., 2005] Urtasun, R., Fleet, D., and Fua, P. (2005). Monocular 3-d tracking of the golf swing. In *CVPR*, volume 1, pages 932–939.

[Urtasun et al., 2006] Urtasun, R., Fleet, D., and Fua, P. (2006). Temporal motion models for monocular and multiview 3d human body tracking. *Comput. Vis. Image Underst.*, 104(2):157–177.

[Urtasun and Fua, 2004] Urtasun, R. and Fua, P. (2004). 3d human body tracking using deterministic temporal motion models. In *European Conference on Computer Vision, Prague, Czech Republic*.

[Urtasun et al., 2004] Urtasun, R., Glardon, P., Boulic, R., Thalmann, D., and Fua, P. (2004). Style-based motion synthesis. *Computer Graphics Forum (CGF)*, 23(4):799–812.

[VAL, 2009] VAL (2009). Visual Agent Laboratory (VAL), Department of Information and Computer Sciences, Toyohashi University of Technology. http://www.val.ics.tut.ac.jp/project/geostat/.

[Verbeek, 2004] Verbeek, J. (2004). *Mixture Models for Clustering and Dimension Reduction*. Phd thesis, Universiteit van Amsterdam.

[Verriest et al., 1994] Verriest, J. P., Rezgui, M. A., and Wang, X. G. (1994). Experimental validation of arm reach movement simulation. In Health, O. and Safety, International Ergonomics Association, T. C., editors, *Proceedings of IEA*, volume 2, pages 342–344.

[Wang et al., 2008] Wang, J., Fleet, D. J., and Hertzmann, A. (2008). Gaussian process dynamical models for human motion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(2):283–298.

[Wang and Verriest, 1998] Wang, X. and Verriest, J. P. (1998). A geometric algorithm to predict the arm reach posture for computer-aided ergonomic evaluation. *The Journal of Visualization and Computer Animation*, 9(1):33–47.

[Whitney, 1969] Whitney, D. E. (1969). Resolved motion rate control of manipulators and human prostheses. In *IEEE Trans. Man-Mach. Syst*, volume 10, pages 47–53.

[Witkin and Kass, 1988] Witkin, A. and Kass, M. (1988). Spacetime constraints. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 159–168, New York, NY, USA. ACM.

[Witkin and Popović, 1995] Witkin, A. and Popović, Z. (1995). Motion warping. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 105–108, New York, NY, USA. ACM.

Curriculum Vitae

Name	Schubert Ribeiro de Carvalho
Date of birth	October 9th, 1975, in Belém-PA, Brazil
Nationality	Brazilian
Mother tongue	Portuguese
Languages	English, French

Education

Ph.D. in Computer Science (Fall 2005 - Fall 2009)
Computer, Communication and Information Sciences
École Polytechnique Fédérale de Lausanne (EPFL) - Switzerland
Thesis: Data-Driven Constraint-Based Motion Editing

Master in Computer Science (Spring 2002 - Winter 2005)
Institute of Computer
State University of Campinas (UNICAMP) - Brazil
Thesis: A Vergence Algorithm for Attention Control of Virtual Humans

Bachelors in Computer Science (Fall 1996- Fall 2001)
Federal University of Para (UFPA), Brazil

Professional Activities

- **Research assistant** (Fall 2005 - Present)
  At VRLab-EPFL, Lausanne, Switzerland.
- **Teaching Assistant** (Spring-Summer 2008)
  In Advanced Computer Graphics at EPFL, Lausanne, Switzerland.
- **Teaching Assistant** (Spring-Summer 2007)
  In *Introduction à la Programmation* at EPFL, Lausanne, Switzerland.
- **Lecturer** (Spring-Summer 2005)
  In Computer Graphics at Hoyler College, São Paulo - Brazil.
- **Trainee (support)** (Spring 2000 - Spring 2001)
  At Data Processing of Para (PRODEPA) - Brazil.
- **Trainee (network administrator)** (Winter 1998 - Spring 2000)
  At Department of Data Processing, Bank of Amazônia - Brazil.

Publications

- Carvalho, S.R., Boulic R., Thalmann D., *Motion Pattern Encapsulation for Data-Driven Constraint-Based Motion Editing*. Second International Workshop on Motion in Games (MIG2009), LNCS 5884, pp. 116–127. Springer, Heidelberg (2009).
- Carvalho S.R., Boulic R., Thalmann D., *Interactive Low-Dimensional Human Motion Synthesis by Combining Motion Models and PIK*. Journal of Computer Animation & Virtual Worlds. Copyright © 2007 JohnWiley & Sons, Ltd.
- Carvalho, S.R., Boulic R., Thalmann D., *Motion Pattern Preserving IK Operating in the Motion Principal Coefficients Space*. In the 15th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2007 (WSCG 2007).
- Carvalho, S.R., Gonçalves, L.M, *Realistic Simulation of Humanoids based on Computer Vision and Robotics*. Scientia. São Leopoldo, RS: Vol.13 (2), 2003.
- Aranha, C.C., Carvalho, S.R., Gonçalves, L.M. *Câmbio: A Realistic Simulated Robot for Vision Algorithms*. The 10-th Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (Journal of WSCG'2003), Vol.11(1), 2003. ISBN 80-903100-2-8.
- Carvalho, S.R., Gonçalves, L.M., *Realistic Simulation of Humanoids Based on Computer Vision and Robotics*. Workshop of Thesis and Dissertations on Computer Graphics and Image Processing, 2002, Fortaleza, CE. Proceddings of I WTDCGPI. São Leopoldo, RS : Gráfica da UNISINOS, 2002. v. 1. p. 42-45.
- Aranha, C.C., Carvalho, S.R., Gonçalvez, L.M.G., *Cambio: Realistic Three-dimensional Simulation of Humanoids Based on Computer Vision and Robotics* Brazilian Symposium on Computer Graphics and Image Processing, 2002, Fortaleza. Procedings of SIBGRAPI 2002. Los Alamitos, CA : IEEE Computer Society Press, 2002. v. 1. p. 388-395.