UNIVERSITE DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité « Mathématiques et Informatique »

Arrêté ministériel: 7 Août 2006

Présentée et soutenue publiquement par

MAXIME TOURNIER

le 14 Octobre 2011

RÉDUCTION DE DIMENSION POUR L'ANIMATION DE PERSONNAGES

Thèse dirigée par FRANÇOIS FAURE

JURY

M.	DAVID GUIRAUD	Directeur de Recherche - LIRMM, Montpellier	Rapporteur
M.	FRANCK MULTON	Professeur - Université de Rennes 2	Rapporteur
Mme.	MARIE-PAULE CANI	Professeur - Université de Grenoble	Examinatrice
M.	PAUL KRY	Assistant Professeur - Université de McGill, Montréal	Examinateur
M.	FRANÇOIS FAURE	Professeur - Université de Grenoble	Directeur
M.	LIONEL REVERET	Chargé de Recherche - INRIA Rhône-Alpes, Grenoble	Co-Directeur

Thèse présentée au sein du Laboratoire JEAN KUNTZMANN et de l'Ecole Doctorale MSTII

Résumé

Dans cette thèse, nous proposons de nouvelles representations pour les poses du mouvement humain, apprises sur des données réelles, en vue d'une synthèse de nouveaux mouvements en temps-réel. Dans une première partie, nous exploitons une méthode statistique adaptée aux groupes de Lie (Analyse en Géodésiques Principales, AGP) pour approximer la variété des poses d'un sujet en mouvement, à partir de données de capture de mouvement. Nous proposons un algorithme de cinématique inverse exploitant cette paramétrisation réduite, permettant par construction de synthétiser des poses proches des données initiales. Nous validons ce modèle cinématique par une application à la compression de données de mouvements, dans laquelle seules quelques trajectoires des extrémités des membres du squelettes permettent de reconstruire une bonne approximation de l'ensemble des données initiales.

Dans une deuxième partie, nous étendons cette approche à l'animation physique de personnages virtuels. La paramétrisation réduite par AGP fournit les coordonnées généralisées de la formulation Lagrangienne de la mécanique. Nous dérivons un intégrateur temporel explicite basé sur les intégrateurs variationnels. Afin d'en améliorer la stabilité, nous proposons un modèle d'amortissement inspiré de l'algorithme de Levenberg-Marquardt. Nous présentons également une méthode géométrique d'apprentissage des limites angulaires sur des données de capture de mouvement, ainsi que leur application comme contraintes cinématiques.

Dans une troisième partie, nous abordons le problème du contrôle du mouvement. En formulant les étapes de la simulation physique d'une part, et de la cinématique inverse d'autre part comme deux programmes quadratiques, nous proposons un algorithme de pseudo-contrôle par interpolation des métriques, permettant un compromis intuitif entre simulation physique non-contrôlée, et cinématique inverse. Cette approche faisant intervenir des forces externes, nous proposons une formulation alternative, utilisant uniquement les forces associées à la paramétrisation réduite des poses. Cette formulation est obtenue par relaxation du problème théorique de contrôle sous contraintes unilatérales, non-convexe, en un programme quadratique convexe. Ces algorithmes sont évalués sur des contrôleurs d'équilibre et de suivi.

Abstract

In this thesis, we propose novel, data-driven representations for human poses, suitable for real-time synthesis of novel character motion. In the first part, we exploit Lie group statistical analysis techniques (Principal Geodesic Analysis, PGA) to approximate the pose manifold of a motion capture sequence by a reduced set of pose geodesics. We propose an inverse kinematics algorithm using this reduced parametrization to automatically produce poses that are close to the learning set. We demonstrate the efficiency of the resulting pose model by an application to motion capture data compression, where only a few end-effector trajectories are used to recover a good approximation of the initial data.

In the second part, we extend this approach to the physically-based animation of virtual characters. The PGA-reduced parametrization provides generalized coordinates in a Lagrangian formulation of mechanics. We derive an explicit time integrator by approximating existing variational integrators, and propose a damping model based on the Levenberg-Marquardt algorithm. We also describe a geometric, data-driven, angular limit learning algorithm, and the associated kinematic constraints.

In the third part, we reach the problem of task-space motion control. By formulating both physical simulation and inverse kinematics time stepping schemes as two quadratic programs, we propose a simple pseudo-control algorithm that interpolates between the two metrics. This allows for an intuitive trade-off between uncontrolled simulation and kinematic manipulation. Since this approach makes use of external forces, we propose an alternate formulation using only the generalized forces associated to the pose parametrization. A control algorithm is obtained by the relaxation of the exact, non-convex control problem under unilateral constraints, into a convex quadratic program. These algorithms are evaluated on simple balance and tracking controllers.

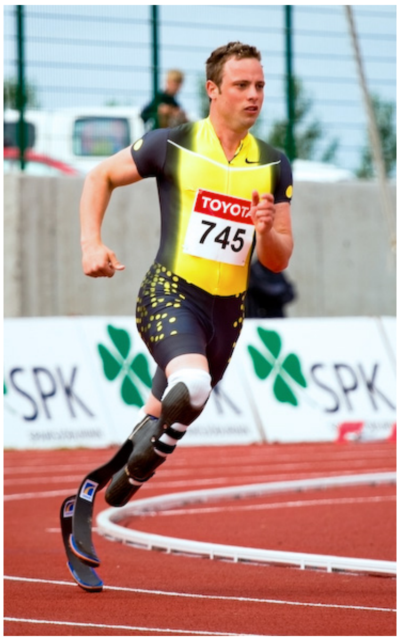
Preface

The study of human motion is a broad topic. To begin with, there are several *levels* of understanding: from pure mechanics to cognitive science, not to forget biological aspects, this problem spans a whole range of sub-problems in several scientific areas. In order to properly reason on these problems, several *models* for human motions have been proposed in the past, describing its geometry, temporal behavior or statistical variability, among many others. Eventually, each of these models has to confront real motion data for validation. Luckily, advanced motion capture techniques have been developed in the last decades, enabling the creation of several computational tools to record, analyze and synthesize motion data. With the advent of these techniques, our knowledge of human motion grew substantially, and is still growing today.

But why would one want to know about human motion in the first place? As the last decades showed, applications are numerous, ranging from medical domain to artistic creation: prosthesis improvements, sports gear design, robotic manipulation, virtual characters in movies, and more generally whenever there is a need to (re)create motion. One of the most striking successes of this knowledge is the case of South African Paralympic runner Oscar Pistorius. The leg prosthesis he uses[1] are the result of decades of research and development in bio-mechanics. This prosthesis raised controversy (see [WB10] and subsequent articles) as to whether the mechanics involved with it gave Pistorius an advantage even over normal-legged athletes. While he is not able to use his calves to push the ground when accelerating, the prosthesis could allow for a much better restitution of elastic energy when running. The very possibility of such a controversy highlights the progresses made by our understanding of how humans move.

The process of understanding an aspect of human motion goes through the design of a *motion model*, that accounts for the features one seeks to explain while abstracting out the unneeded complexity. By modeling a particular phenomenon, one assume some underlying *structure* in the motion, which is what a motion model should capture. In this thesis, an important part of

[1] Cheetah, by Össur (http://www.ossur.com/)



(CC BY Elvar Freyr)

Figure: Athlete Oscar Pistorius, wearing controversial leg prosthesis, 2007

our contributions consists in capturing *geometric* aspects of human motion, through dedicated dimension reduction techniques. While our bodies exhibit numerous degrees of freedom for producing motion, bio-mechanical studies showed they tend to be actuated in a highly-correlated way, resulting in similarly coordinated body motion.

This redundancy offers an opportunity to reduce the total complexity of animating a virtual character, by capturing this structure in a simple geometric model. It allows to generate better-looking animations at a lower computational cost.

Since human motion results from a physical process, we were naturally led to use our geometric model in a physically-based animation context, in which dynamic effects are described by the equations of motion, another type of structure in the motion.

Finally, as recent biomechanic studies show, the muscle activations in the human body seem to be performed in a highly-coordinated way. Having derived a data-driven, geometric description of motion correlations, it seemed natural to investigate whether this reduced pose coordinates could provide a working actuation basis for motor control.

Organization of the Manuscript

The introduction *(part I)* presents historical and practical aspects on motion capture techniques, for the reader to become familiar with the vocabulary and concepts used throughout the manuscript.

Part II provides background on the mathematical modeling of a human skeleton, along with relevant geometric and theoretical tools, most notably concerning the *Lie groups* theory. Without going too much into details, we try to expose important results and concepts by favoring intuition over proofs. In particular, we illustrate how Lie groups connect intuitive geometric reasoning to matrix computations.

Part III *(kinematics)* proposes an algorithm for learning and approximating the pose manifold of a motion capture, and the associated Inverse Kinematics procedure. A validation of this algorithm is proposed as a motion compression algorithm.

Part IV *(dynamics)* extends these ideas in the context of physically-based animation of characters. The reduced pose parametrization derived in the first part serves as the basis for deriving discrete equations of motion for the animated character. A data-driven angular limits learning algorithm, and a dedicated damping model are derived.

Part V *(control)* further extends the physical modeling to the motion control problem. Two algorithms are proposed, with or without external forces. In the latter, the reduced coordinates are used as whole-body actuators in a task-space control framework. Balance and tracking controllers are presented.

We conclude in part VI, with perspectives on future works.

Contents

Part I

Introduction

Chapter 1

History of Motion Study

The study of human motion has a long history, therefore we will only sketch it quickly here, with an emphasis on the different data sources available. The first known book on biomechanics is *On the Motion of Animals*, written by Aristotle (384-322 B.C.). For the first time, animals were viewed as a mechanical system (similar to puppets) and their motion studied as such, in terms of joints and levers. He notably performed a geometric description of walking.

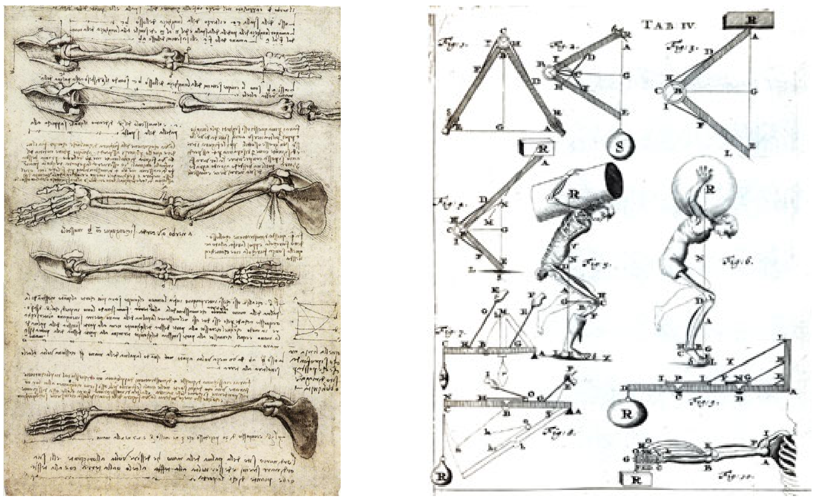


Figure 1.1: Drawings by Leonardo Da Vinci *(left)* and Borelli *(right)*, among the first known works in biomechanics.

Leonardo Da Vinci (1452-1519) identified muscles and nerves of the human body, and the mechanics involved in various human activities. Interestingly, he tried to inspire from animal body structures to design better machines, or to better adapt machines to animal bodies. Analysis on the forces exerted in the human body went further with works of Giovanni Alfonso Borelli (1608-1679), when he applied mechanic principles to the study of the human body. However precise were these hand-drawn studies, it's only with the invention of photography in the 19th century that visual data on the human body became reliable.

1.1 Chronophotography

Quickly after the first photography processes were created (between 1826 and 1840 and through the works of Joseph Nicéphore Niépce and Fox Talbot, among many others) and the photography technique was setting up, photographers began studying motion by decomposing it into a series of shortly time-spaced snapshots of a moving subject. The first known work in *chronophotography*, as it is called today, was produced by Eadweard Muybridge in 1878 and depicted a galloping horse (*Sallie Gardner at a Gallop*).

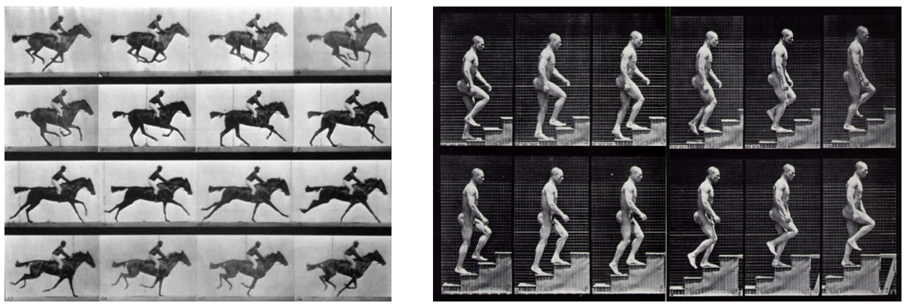


Figure 1.2: Muybridge works: Sallie Gardner at a Gallop *(1878, left)*, Man ascending stairs *(1884-1885, right)*.

This work is interesting because it was used as a scientific proof to settle a popular question by horsing fans at that time: is there a moment during which all four of a horse's hooves leave the ground during a gallop? As it can be seen on the picture (1.2), there is indeed one such moment. Muybridge also studied human motion, as it can be seen on *Man ascending stairs* (1.2). Background shows white lines, used to indicate proportions and distances for subsequent measures. While Muybridge used several cameras to capture a subject, Étienne Jules de Marey, around the same years, set up a way of capturing several pictures with only one camera. He used the device for his research on biomechanics, and produced detailed studies of animal motion (figure 1.3), as well as the human motion.

After the invention of the *cinematograph* device by the Lumière brothers in the 1890s, film techniques developed quickly but the quality was somewhat lacking. The *rotoscope* device, by Max Fleicher in 1915, allowed animators to draw directly over captured performance to improve character animation in cartoons. Meanwhile, photography techniques were enhanced and Harold

Eugene Edgerton proposed to use stroboscopic equipment to achieve the unprecedented speed of 120 flashes per second, allowing for a much more detailed capture of motion. He used this technique to record high speed phenomena such as bursting balloons, or water flows.

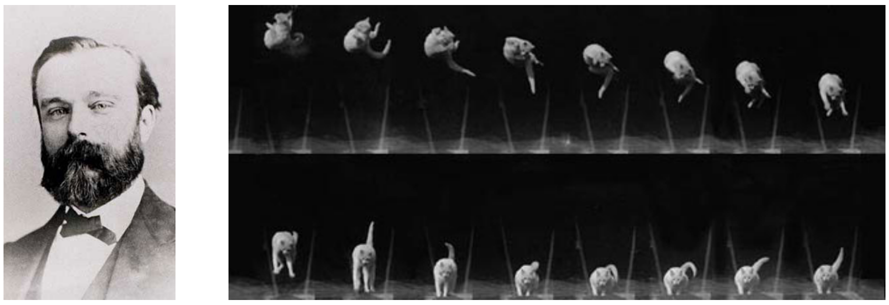


Figure 1.3: Étienne Jules de Marey, and his famous *Falling Cat* (1890).

1.2 Motion Capture

The first motion capture experiments in the modern sense were made by Lee Harrison and his team, in the early 1960s. For the first time, the real motion of a human character was recorded using a mechanical capture suit, to be further processed by a computer to finally generate an animation. This process, called *digital puppetry*, developed further with Waldo C. Graphic, a character created in 1988 that was captured and rendered to screen in real time, allowing for live performances. The same year, Brad DeGraf and his partner Michael Wahrman presented *Mike the talking head* to the SIGGRAPH conference. It featured detailed facial motion capture as well as a set of high-level parameters for controlling character expressions on run-time.

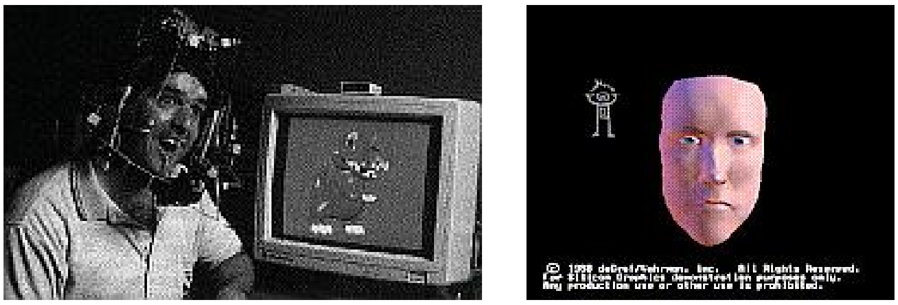

© 1988 deGraf/Wahrman, Inc. All Rights Reserved.
For Silicon Graphics demonstration purposes only.
Any production use or other use is prohibited.

Figure 1.4: Mike the talking head [Rob88] featured speech recognition system that translated phonemes into facial expression of a digital puppet.

First computer-vision based approaches to performance capture developed quickly after, in the 90's. The principle is to equip an actor with a set of reflective markers, that are filmed by multiple calibrated cameras, allowing to reconstruct the three-dimensional trajectories of these markers (see section 2.1). The ease of capture they provided for actors was somewhat mitigated by the technical difficulties raised by occluded markers, and these occlusions often had to be taken care of by hand, resulting in lots of tedious work by expert animators.

The motion picture industry started relying more and more on motion capture techniques, as it made mixing real performance with computer generated backgrounds a lot easier. Notable examples include *Final Fantasy - The Spirits Within* as being the first full-length movie to exploit motion capture techniques to animate all virtual humans, whereas previous computer graphic movies relied on talented animators for adjusting every animation parameters in dedicated software. More recently, the movie *Avatar* set up new standard in performance capture techniques, allowing its director James Cameron to visualize a live (though simplified) version of a complete virtual world, including background scenery, virtual actors and their facial expression, when filming an action.

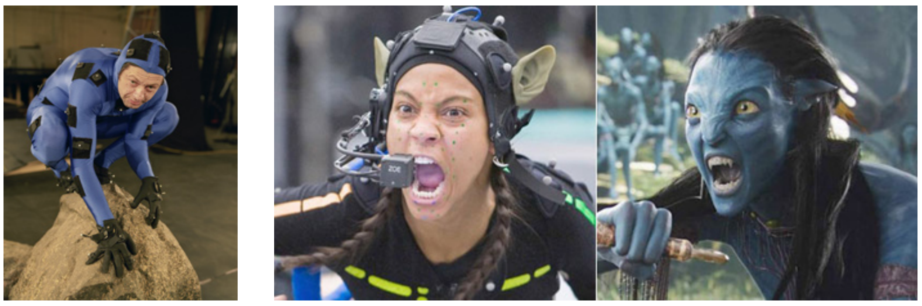


Figure 1.5: Lord of the Rings actor Andy Serkis *(left)*, and Avatar actress Zoe Saldana *(right)*, in capture suit.

Nowadays, motion capture data is used virtually everywhere high quality human motion data is needed: biomechanics, motion pictures, user interfaces, video games . . . This ubiquity comes at a cost: it is becoming more and more difficult to store, transmit and index large motion capture databases. Even if the spectrum of uses is large, the high dimensionality of these data is a recurrent problem, having multiple implications as we will see in the next chapter.

Chapter 2

Motion Analysis

As we have seen, motion capture data is used in a wide range of contexts, with motion data coming from different kinds of devices. In this chapter, we present various motion capture related concepts: device classes, capture principle and algorithms, and the challenges arising when dealing with these data. Through different applications in motion analysis and synthesis, we stress the importance of a good *pose model* for producing better results at a lower computational cost, at each stage from capture to synthesis.

2.1 Motion Capture

2.1.1 Device Classes

Mechanical Devices

Mechanical motion capture devices are amongst the oldest and most precise capture devices available. Not only do they directly measure position or joint angles with precision, but they are sometimes also able to provide force measures as well.

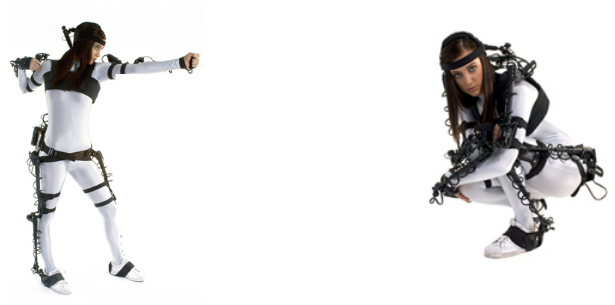


Figure 2.1: Gypsy 5, by Animazoo, an example of a *mechanical* motion capture device.

Such devices usually come in the form of an exoskeleton that is put on an actor during the capture, which restricts the allowed range of motions to that of the exoskeleton. Most of these devices are now wireless, allowing for a virtually unlimited *capture volume*. By their design, these devices can only record relative positions and angles of a skeleton, therefore they need an additional global positioning system, for correctly placing the subject in the surrounding space. Such information can also be integrated from a known initial position without additional global information, but this often lead to a drift in skeleton placement due to numerical errors during the integration.

Magnetic Devices

Magnetic devices work by measuring the magnetic flux at the markers, producing in orientation/position data for all markers. These devices are highly subject to magnetic and electric field disturbances, therefore are not suitable in an environment containing metal objects, or computers. Most of the time the markers are wired, preventing an actor from performing extreme actions. Finally, they can only function reliably in a relatively small capture volume, where the field distortion is kept low.

Optical Devices

Optical systems are the most widespread device class, and come in several variants: passive markers, active markers, and markerless. The basic principle is the following (*cf.* figure 2.2):

1. The captured subject is filmed by multiple, calibrated cameras
2. Relevant features (usually, markers) are detected in camera views, then tracked over time
3. Feature trajectories are reconstructed in 3D using vision algorithms and *epipolar*[1] geometry
4. Optionally, an additional post-processing pass is applied to correct remaining artifacts

Passive systems work by placing *reflective* markers on an actor, whose performance is captured by multiple calibrated cameras (often up to 24). The more camera available, the less likely marker occlusions will happen, at the expense of computational cost. Each marker is tracked in time on camera views, and its three-dimensional position is recovered using computer vision

[1]The epipolar geometry describes the relations between scene points and their projections on multiple camera images. These relations serve as constraints to reconstruct 3D points from multiple 2D views.

algorithms. The tracking part is difficult due to potential marker occlusions or collisions. Passive markers are usually lit by an external (usually infrared) light source, therefore they can not be captured unless they are sufficiently close to both the light source and the cameras.

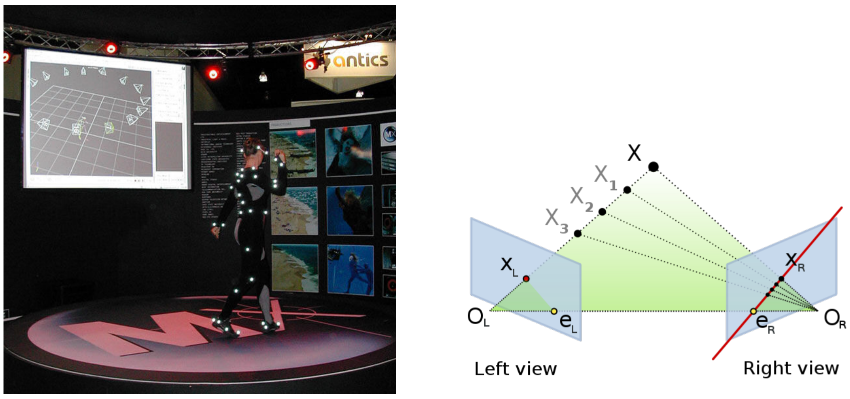


Figure 2.2: An optical motion capture system *(left)*, with the reconstructed scene in the background showing cameras. The epipolar geometry of calibrated cameras *(right)* allows to reconstruct the three-dimensional position of point x through camera views.

Active markers, on the contrary, emit their own light, allowing for more capture space. Furthermore, they can emit light in a way that is synchronized with the cameras, so that only one marker will be lit at a time. This highly reduces the problems related to marker tracking, at the expense of the capture rate.

Recent advances in computer vision algorithms allowed the development of *markerless* capture devices, in which the actor is simply filmed by multiple calibrated cameras. The actor silhouette is usually separated from the background on each camera, then the multiple silhouettes are merged into a three-dimensional description of the character (polygonal mesh, skeleton pose...) as seen on figure 2.3.

Even though this kind of equipment can be less expensive than other solutions (one only needs multiple cameras), they tend to suffer from changing light conditions and are subject to the camera frame rate and resolution limitations.

2.1.2 Challenges

Optical devices are by far the most widespread technology for capturing motion, and this popularity is ever increasing as low-cost solutions hit the video games market (see figure 2.4). The tendency seems to be in favor of lower-end capture devices, backed-up by both better capture algorithms and increasing

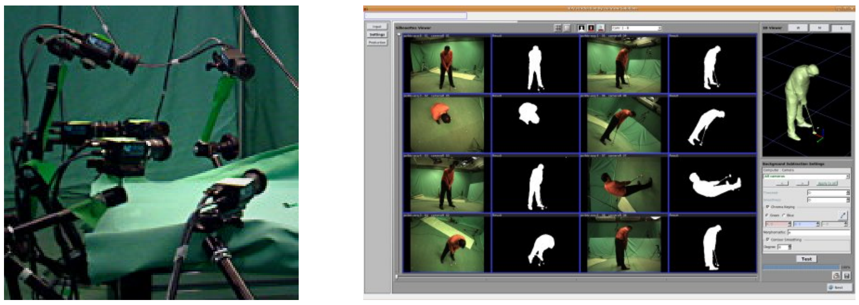


Figure 2.3: A commercial markerless capture system by 4DViews, based on work by [AMR+07]. The subject is isolated from the background *(right)* in camera view. A 3D reconstruction is computed on the fly, with textures projected from camera views.

computing power (notably multi-core CPUs, by affecting a core to a specific task). Therefore, there is an increasing demand for vision algorithms allowing robust motion capture even on modest capture hardware.

Figure 2.4: Major 3 video game systems all have a more-or-less evolved version of a motion-based controller. From left to right: Nintendo Wii, Sony Playstation Move, Microsoft Kinect.

Achieving robustness in motion capture algorithms is usually possible using some kind of *prior* over the captured situation: is there any known structure in the captured motion that could be exploited to recover a loss of information? Obviously, the knowledge of tracking an almost rigid skeleton can (and was) used as such a prior. But recent years showed growing interest for *learned* motion models (see [MG01] for a good overview), using machine learning algorithms on motion capture data. For instance, if we acquire a golf swing motion, there is only a few subset of all the possible skeleton poses that is relevant during the capture. With such a prior over the motion model, known marker positions can be used to recover occluded ones. Another example of prior is to consider that the captured motion satisfies a sufficient degree of smoothness, preventing too large displacements between two frames. This can be exploited in tracking tasks.

2.1.3 Motion Capture Data

Once the motion data have been captured, they are made available under an agreed-upon file format for further treatment. Most common formats are targeted at skeleton animation (*e.g.* Acclaim ASF, Biovision BVH), therefore provide a geometric description of a skeleton alongside motion data. In essence, motion data represent the pose taken by a character, sampled across time. The storage needed for such data scales as $O(m.n)$, where m is the number of frames, and n is the number of degrees of freedom of the character skeleton.

Databases In order to cover a wide range of human behaviors, it is often necessary to manage motion captures in a *motion database*, that associates semantic or descriptive information to a large collection of captures, and allows easy indexing and retrieval. Such databases are in wide use in motion pictures studios, or by video games developers. However, labeling and organizing such large collections by hand is a tedious task, hence ongoing research try to exploit motion data structure in order to automatically cluster similar motions, as for example [KPZ+04]. Here, pose models are used for indexing motion collections.

Compression As these databases grow large, there have been a recent interest in compressing them, again exploiting motion structure to detect sparsity or correlations appearing in motion data (see chapter 9). For instance, temporal coherence can be exploited to interpolate between relevant key-frames, cyclic motions can be treated in the frequency domain, or similar motions can be regrouped together under a same pose model. Such analysis can be used to reduce the storage problems of such databases, while potentially increasing their indexing power.

Pose Space Letting large databases and redundancies between similar motions aside, we can observe that poses taken by a character for a single capture tend to lie in a relatively well-defined subspace of all the possible poses, especially when performing a specific action. This suggests that there may exist a more compact parametrization of the poses space, depending on the action performed by the character, that exhibit fewer degrees of freedom while still spanning the pose space for the task.

In other words, with a good motion model one should be able to compress one motion sequence efficiently, because it accounts for most of the structure present in the data. We argue that the compression problem actually provides a relevant framework for *evaluating* motion models, by directly measuring how compactly they can encode a motion. This subject will be developed in more details in part III.

Models We see that just like motion models allow to ease the motion *capture* process by resolving ambiguities, they can also be used for the *storage* of the resulting data, by enabling motion compression.

2.2 Motion Data Processing

Since acquiring and storing motion data is a costly and a generally complicated process, several authoring tools have been developed to permit its modification without having to re-capture a full sequence (*cf.* figure 2.5).

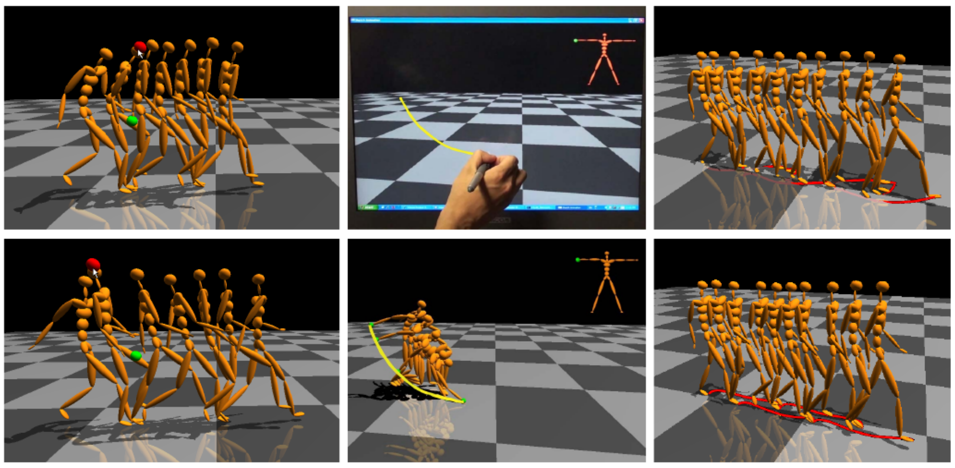


Figure 2.5: A state-of-the-art motion editing interface proposed by [MCC09].

Two main approaches can be distinguished:

1. Motion *editing*, the process of adjusting an existing motion capture to satisfy given constraints
2. Motion *synthesis*, where original motion data is synthesized without previous reference capture

As one can expect, motion processing tools are subject to the following competing challenges:

- Result quality
- Control over final result
- Computational-cost

Let us now briefly describe the two classes of motion processing methods, and motivate the need for a fast and compact motion model.

2.2.1 Motion Editing

Due to the burden and cost of acquiring motion capture data, it is often desirable to edit existing captures in case the capture is not satisfactory. It might be the case for a virtual character whose feet are badly placed in a virtual terrain, or when hitting a ball at different location. Rather than solving for the full motion that satisfies edition constraints, which is computationally-expensive, it is often easier to simply adjust existing captures to the situation. A wide range of algorithms tackle this problem (reviewed in part III) but most of the time, these techniques are limited to small edits around given motion. Here again, a suitable pose model could effectively restrict possible edits to match the reference motion structure, therefore enabling motion edition to extrapolate outside reference motion.

2.2.2 Motion Synthesis

At the broadest level, the process of sythesizing motion data can (and is in practice, see part III) be cast as a constrained optimization problem: find the best motion frames, in the sense of a given metric, satisfying given constraints. Typical choices of constraint terms include smoothness, boundary conditions, keyframe interpolation, laws of dynamics, or geometric constraints. The error metric usually penalizes the energy consumption, or the distance to a reference motion/pose.

Such formulations primarly suffer from the *huge* dimension of the search space, not even mentioning the convexity of the objective function. Every degree of freedom plays a role in the optimization, which can result in severely ill-posed problems.

In this context, it seems that a reduced pose model could efficiently improve the computational cost of these approaches, by both reducing the dimension of the search space and the possible candidate solutions to the optimization problem.

2.2.3 Muscle Synergies

In fact, recent research works in biomechanics formulate the hypothesis of highly correlated muscle activation pattern, known as *muscle synergies*, that could explain a wide variety of human activities (see [TJ09] for an excellent introduction). Proponents of the theory argue that muscles activations are *a priori*clustered in so-called synergies that effectively reduce the number of degrees of freedom for solving the task. On the other hand, opponents consider the observed correlations to be task-dependent, as they only reflect the best activation subspace for accomplishing the task. One possibility does not completely rule out the other, as individual Degree of Freedom (DOF) control can always be cast in terms of highly specialized synergies. On the other hand, muscles synergies could be a kind of low-level muscle activation prior, with more advanced control strategies taking relay when needed.

While biomechanic interpretations may vary, both sides agree that muscle actuation is perfomed in a highly correlated way, thus producing highly-correlated angular motion at the joints. This plays in favor of the construction of a task-specific, reduced pose model for character animation.

2.3 Dimensionality

In both of these approaches, the dimensionality of the human skeleton raises the following problems:

- Motion synthesis algorithms provide control, but work on high-dimensional, often ill-posed problems.
- Editing a motion is fast but often limited to small changes for quality results
- Biomechanics studies suggest that a reduced pose parametrization can be found

In the last decades, several works have been proposed to apply machine learning algorithms to automatically build motion models from motion capture data (see part III). However, such models are often very computationally-expensive themselves, both when learning and synthesizing motion data, due to the high non-linearity of joint orientation data.

In this work, we propose to derive a data-driven character pose model that is adapted to the geometry of rotations, while at the same time retaining good computational cost. The following chapter summarizes the above issues, and presents our contributions in the construction of new representations for human poses.

Chapter 3

Motivations - Contributions

After this rapid overview of the challenges associated with motion capture and synthesis, let us now summarize the aforementioned issues, and present the motivations and goals of this work in a more systematic way.

3.1 Pose Model

We have seen that motion data is used a wide variety of contexts, all of which could benefit from an adapted *pose model* for improving the overall quality of the results.

3.1.1 Capture

The most common motion capture systems are optical systems, which inherently suffer from marker occlusion related problems. These problems are even more pronounced in the case of lower-end capture system, whose market is in rapid growth. In the absence, or incompleteness, of visual clues, the only way to allow motion to be captured is to rely on a pose *prior* that can compensate for the loss of information.

3.1.2 Processing

In order to edit a motion capture sequence without altering its stylistic properties, a pose model can guarantee that edited motion lies in the same task-specific subspace as the original motion, preventing undesired edits.

Many motion synthesis tools use optimization problem formulation. Since many possible solutions may exist, a pose model provides a natural way of disambiguating between candidates, potentially improving optimization behavior.

3.1.3 Control

As recent biomechanics studies suggest, the human body is actuated in a highly-coordinated, task-specific way, reflected in the final motion. Physically-based motion controllers could thus benefit from a data-driven actuation basis in order to produce more convincing results.

3.2 Dimension Reduction

The dimensionality involved in processing human motion capture data often raises performance issues. By providing a more compact description of pose spaces, pose models enable dimension reduction that can lower the impact of dimensionality.

3.2.1 Compression

Motion data can be made more compact by encoding correlations existing in motion. Motion databases can be clustered by motion types, thus factoring common behaviors. Ideally, a good pose model could encode such correlations in a compact way as well.

3.2.2 Optimization

The search space dimension in several motion synthesis problems can be prohibitively high. Reducing the number of degrees of freedom, and thus the search space dimension can provide significant speed gains.

3.2.3 Real-time Simulations

By restricting the number of degrees of freedom, dimension reduction techniques enable real-time simulation of complex mechanical structures. This strategy could be applied to character animation as well.

3.3 Goals

From the preceding section, it appears that many stages of the character animation pipeline could benefit from a *data-driven pose-model*, with the following features:

- Ability to perform **dimension reduction**, to improve both computational time and result quality
- **Compactness** of the learning data, allowing its use in compression applications

- Possible integration with other existing motion synthesis tools, ideally through a **smooth** forward kinematics pose parametrization with reasonable computational cost

3.4 Contributions

3.4.1 Kinematics

The research for such a pose model is the subject of part III. We review previous work in structure-preserving statistic analysis, and propose a character pose model based on Principal Geodesic Analysis (PGA). This model is evaluated on a motion compression application. Our contributions in this part are the following:

- A PGA based skeleton pose model, with the associated forward kinematics and pose Jacobian computations (8.2).
- A data-driven, real-time, full-body Inverse Kinematics (IK) algorithm using our reduced pose model (8.2.2)
- A motion compression algorithm exploiting the PGA-based IK algorithm (9), for evaluating the pose model performances

3.4.2 Dynamics

Since character animation derives from a physical process, better animations can be obtained by incorporating dynamics into the character model. Therefore, we develop the use of our pose model in a physically-based simulation in part IV. This also provides a benchmark on real-time performances of our model.

Our contribution in this part are the following:

- A velocity/impulse explicit time integrator for our pose model (11.1), based on variational geometric integrators
- An *ad hoc* damping model based on the Levenberg-Marquardt algorithm, and an extension using kinetic energy to prevent instabilities (11.2)
- A geometric, data-driven algorithm for learning angular limits, and their associated kinematic constraints (12)

3.4.3 Control

Lastly, since biomechanics studies suggest that human actuation is performed in a highly coordinated way, we investigate the use of our pose model in physically-based motion controllers in part V. Our contributions are the following:

- A simple quadratic programming pseudo-control framework, providing trade-off between physical-simulation and inverse kinematics by metric interpolation (14.1)
- A more advanced, feature-based motion control framework using the reduced coordinates as actuators. A convex relaxation of the complete control problem under unilateral constraints is proposed (14.2)

We conclude this manuscript in part VI with a synthesis of our approach, and propose possible future work directions.

Part II

Geometry for Character Animation: Background

Chapter 4

Character Modeling

In this part, we give a quick introduction to the notations and concepts that will be used throughout this thesis.

A reader with experience in skeleton kinematics and Lie group theory may skip directly to part III, as the present part only recalls background information on the subject. Nonetheless, a quick glance at the notations used for left and right trivialized tangent maps, described in 5.2.5, will be helpful for the remaining of this work.

We begin with a review of the mathematical representations for modeling an animated articulated character. This will mainly be a pretext to introduce non-linear configuration spaces, and more specifically *Lie groups*, as several parts of this work rely on their use. We conclude this part by reviewing classical articulated skeleton kinematics algorithms.

Under their most basic form, motion data are simply a time series of motion capture frames, *i.e.* the output of the motion sensors presented in a more-or-less evolved form. Capture software usually process these data in order to provide a more user-friendly version of them, for instance as the absolute marker trajectories in the case of an optical capture system. These marker trajectories can serve as the basis for more evolved representations, which we review in this chapter.

4.1 Articulated Body

Depending on the level of details needed for a given application, it might not be necessary to model the entire human body (including muscles, skin . . .) to obtain satisfying character *motion* models. Indeed, the overwhelming majority of existing works in the field only model the human body as a collection of connected, perfectly rigid bodies approximating the character bones, which simplifies computations considerably.

From a perceptive point of view, [Joh73] showed that only a few bright spots (around 10-12) placed at the skeleton principal *joints* are needed for conveying the impression of human motion for activities such as walking, dancing or running. This suggests that the skeletal approximation of a character is usually sufficient to recreate a convincing motion.

In fact, even the body surface motion can be considered approximately rigid around the closest bone. This observation has led to the classical *skinning* algorithm (see [LCF00] for an introduction), in which surface element positions are obtained by weighting perfectly rigid approximations. When the skeleton is animated, the skin automatically follows the skeleton motion and is therefore animated too. More faithful modelings have been proposed to account for the physical interactions between bones, muscles and skin, but they usually require a lot of processing power and are therefore restricted to movie production, or medical simulations. Here again, the animation is usually achieved by animating the skeleton first (*cf.* figure 4.1).

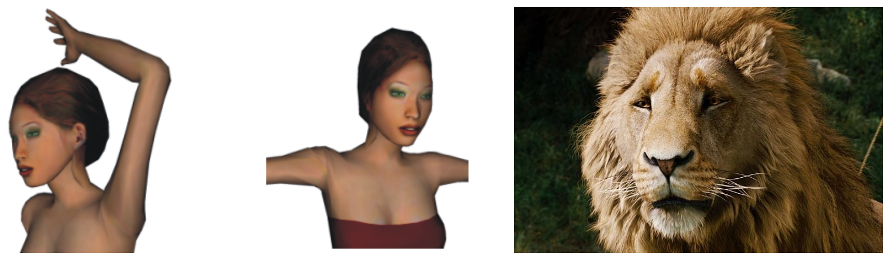


Figure 4.1: Skinning of a virtual character using *dual quaternions* [KCZO08] *(left)*, and an example of physically-based skinning in the movie Narnia by Disney *(right)*.

4.2 Joints, Topology

The human skeleton bones are connected through *joints*, most of which are traditionally approximated by ideal revolute or ball-and-socket mechanical joints. While more advanced strategies have been proposed to better model knee or shoulder joints (*e.g. spline joints* [LT08]), the vast majority of animation models use idealized joints between rigid bones.

Depending on the application and the level of detail needed, we may consider a more-or-less simplified version of the skeleton: the configuration of each hand phalanx is probably not relevant when performing a running motion. In the same spirit, the approximate modeling of the shoulder range of motion, using only ball-and-socket joints, can be achieved by adding a fake joint near the clavicle.

The connections between bones give the skeleton a tree topology, and we will consider that such a tree structure is always implicitly given. In practice, it is usually desirable to choose the *root* of the skeleton such as the height of the resulting tree is minimal, in order to accumulate as few numerical errors as possible in hierarchical computations.

4.3 Configuration Space

Modeling a tree-like articulated skeleton is traditionally achieved using different approaches:

- Using motion capture marker positions in the world frame, with additional constraints to enforce rigidity between markers belonging to the same bone *(marker positions)*
- Considering the configuration of each bone with respect to the world reference frame *(absolute configurations)*
- Absolute configuration of the root together with configurations of other bones relative to their respective parents *(relative configurations)*

Each of these modelings has an associated *configuration space*, *i.e.* a mathematical space *uniquely* representing each possible configuration of the model. Such configurations describe the DOFs of the model. Different modelings correspond to different configuration spaces, with different mathematical structures, which we describe now.

4.3.1 Marker Positions

The most straightforward approach is to model the skeleton directly using motion capture marker positions (*cf.* figure 4.2). Such markers are usually placed where the skin deformation is minimal, so that these markers can be considered as being placed directly on the underlying bone. Mathematically, this corresponds to representing a skeleton configuration as an element of the usual n-dimensional Euclidean space $\mathbb{R}^n$, where $n \in \mathbb{N}$ is the number of markers.

It is quite obvious that such a description of a skeleton does not account for the *rigidity* of the bones, that is the fact that the Euclidean distance between points of a same bone are constant. Therefore, some additional kinematic constraints will have to be enforced to maintain rigidity between markers on the same bone, otherwise one may end up with inconsistent skeleton representations. A better approach would be to directly encode this constraint *in* the configuration space structure.

Another, less obvious, drawback of this modeling is that it does not provide an easy way of comparing body *poses*: if the same character is standing

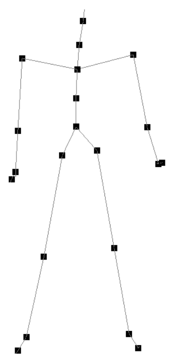


Figure 4.2: Motion capture marker positions for a simple skeleton model, with the tree topology in gray.

in two different places, facing different directions but with the very same attitude, or pose, it is not trivial to compute this similarity using this modeling. Said differently, this approach lacks a simple way of comparing configurations *up to* (or *modulo*) a global rigid transformation of the whole skeleton.

4.3.2 Absolute Configurations

The non-rigidity problem of the marker position approach can easily be corrected by only allowing each bone to transform in a way that preserves distances between any pair of its points. Such *isometries* are called rigid, or Euclidean, transformations, and their space is noted $SE(3)$. We may model a skeleton configuration by giving the rigid transformation of each of its bones with respect to some absolute reference frame (*cf.* figure 4.3). In other words, we view a skeleton configuration as an element of $SE(3)^n$, where $n \in \mathbb{N}$ is the number of bones.

We see that restricting the admissible transformations to the ones preserving the rigidity invariant allows to get rid of the additional constraint found in the previous modeling. The counterpart is that a transformation space such as $SE(3)^n$ has a more complicated mathematical structure than the Euclidean structure of $\mathbb{R}^n$, as we will see. However, this approach has a new drawback, namely it does not prevent joints from disconnecting.

Furthermore, this representation is still not very convenient to compare poses up to an absolute rigid transform. However we can already see that if we apply the *same* rigid transformation to each skeleton bone, the *relative* rigid transformation between two bones does not change. This suggests that

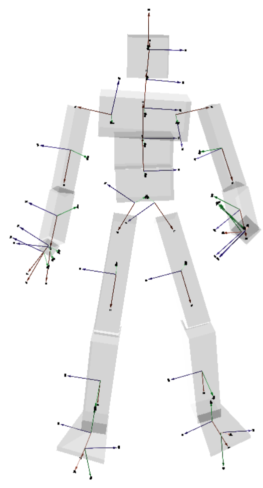


Figure 4.3: Each bone of this character has its own rigid frame, and associated configuration. The visual envelope does not represent all the bones in the skeleton.

such a relative skeleton representation allows to compare poses more easily, since it is *invariant* to global rigid transformation.

4.3.3 Joint Configurations

The tree topology allows us to represent a skeleton using the world configuration of the root bone ($\in SE(3)$), together with the transformation of each other bone relative to its parent. By restricting the set of allowed relative transformations, it is possible to keep the joints from disconnecting, for example by only considering transformations of the form:

$$g_{p,c} = j_p.r.{j_c}^{-1} \in SE(3)$$

where $j_p \in SE(3)$ is the transformation from the parent bone frame to the parent joint frame, $j_c \in SE(3)$ is the transformation from the child bone frame to the child joint frame, and $r \in SO(3)$ is the joint configuration, here restricted to be a pure rotation to model a ball-and-socket joint. Since all joint frames j_p, j_c with respect to their bone are constant, the relevant variables needed to describe a skeleton pose will be the set of *joint configurations*. We will only consider ball-and-socket joints for the human skeleton, hence each of the corresponding joint configurations will be a rotation. The complete configuration space for this modeling is thus:

$$G = \underbrace{SE(3)}_{\text{root}} \times \underbrace{SO(3)^n}_{\text{pose}}$$

where $n \in \mathbb{N}$ is the number of *joints* in the skeleton (see figure 4.4).

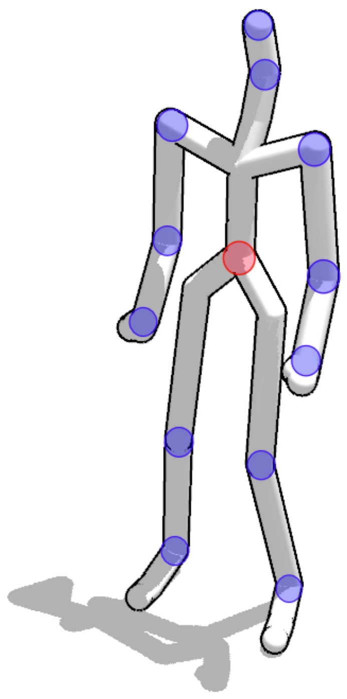


Figure 4.4: The configuration of the root bone *(red)* is a rigid transformation, whereas the internal joint configurations *(blue)* are rotations. The total configuration space is thus $SE(3) \times SO(3)^n$.

We see that using this approach, it is much easier to compare skeleton configuration up to a rigid transform since we only need to compare the pose part of the configuration. Compared to the marker positions approach, we see that we progressively encoded the initial constraints directly into the configuration space *structure*. Instead of describing points of the skeleton and to restrict their possible motions, we tried to represent the set of allowed transformations for the skeleton bones. Like other transformations, these can be chained together by applying one then another, or reversed, leading to the concept of a transformation *group*. As in the previous modeling, the mathematical structure of the configuration space is no longer Euclidean, as it were for the marker positions, but something more complicated called a *Lie group*. Fortunately, the geometry of such objects is well-understood and remarkably rich.

Chapter 5

Lie Groups

This chapter is intended to be an as-concise as possible introduction to the basics of Lie groups, their related concepts and notations. We feel that despite their wide use in robotics, Lie groups are not as widespread as they ought to be in the computer graphics community. Therefore, we will only give an introduction to the theory by mentioning key ideas and intuitions. Our goal here is to introduce basic Lie group *calculus*, left and right *trivialized tangent maps*, and the *exponential* mapping. We also mention basic Riemannian geometry concepts (metric, geodesics) as they will be required in the remaining of this thesis. The interested reader can find a much more solid introduction to these subjects in the excellent (robotics) book by [MSZ94], or a more in-depth treatment in [DK00].

5.1 Definition, Examples

Informally, a *Lie group* is a group that is also a smooth manifold, and for which the algebraic operations are smooth. The relations between algebraic and differentiable structures gives these objects a particularly rich geometry. In this subsection, we quickly review the basics of groups and of smooth manifolds. Let us begin with the group definition:

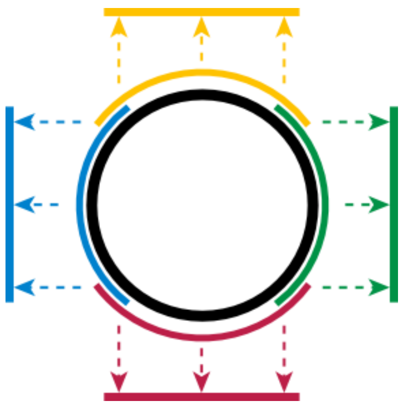
Definition 1 (Group). *A group is a set G, together with a binary operation $\circ$ such as:*

1. *G is closed under $\circ$*
2. *$\circ$ is associative*
3. *$\exists e \in G$ such as $\forall g \in G, e \circ g = g \circ e = g$ (identity element)*
4. *$\forall g \in G, \exists g^{-1} \in G$ such as $g \circ g^{-1} = g^{-1} \circ g = e$ (inverse)*

A group can be thought of as the abstraction of a set of transformations: the transformation that does nothing is the identity, transformations can be composed to produce another transformation, or reversed, and so on. For multiplicative groups, the group operation will be denoted by a dot $(.)$. Let us now recall the definition of a manifold :

Definition 2 (Manifold)**.** *A (real) manifold M of dimension $n \in \mathbb{N}$ is a topological space that is locally homeomorphic*[1] *to the Euclidean space $\mathbb{R}^n$.*

This definition means that a manifold is an object we can locally map to $\mathbb{R}^n$ using bi-continuous *charts*. A collection of charts that covers the manifold is called an *atlas*. Figure 5.1 shows a visual representation of an atlas for the circle S^1.



(CC BY-SA KSmrq)

Figure 5.1: The circle S^1 is a one-dimensional differentiable manifold. A covering atlas of charts is represented in colors, as 4 overlapping mappings from different parts of the circle to a line segment.

Let ϕ, ψ be two charts around a point $p \in M$ and $U \subset M$ be an open set containing p in the intersection of their domains. Let now $V = \phi(U)$ and $W = \psi(U)$ be the images of U by the two charts, *i.e.* open subsets of $\mathbb{R}^n$. Since both ϕ and ψ are invertible, let us consider the following *transition maps*:

$$\phi^{-1} \circ \psi : V \to W$$

$$\psi^{-1} \circ \phi : W \to V$$

These transition maps are usual functions from and to open subsets of $\mathbb{R}^n$, and are the basis for the *smooth* manifold definition:

[1] An homeomorphism is an invertible, continuous application, with its inverse being also continuous.

Definition 3 (Smooth manifold)**.** *A* smooth manifold M *of dimension* $n \in \mathbb{N}$ *is a manifold for which all transition maps are smooth* (i.e. *of class* $\mathcal{C}^\infty$*)*

Informally, charts are used to locally extend traditional Euclidean, differential calculus to the smooth manifold case: a function $f : M \to \mathbb{R}$ will be differentiable at a point if it is in coordinates around this point (*i.e.* if $f \circ \phi^{-1}$ is differentiable in the Euclidean sense). In the same spirit, a function *between* two smooth manifolds $f : M \to N$ will be differentiable if it is in coordinates (consider $\psi \circ f \circ \phi^{-1}$). The definition of a smooth manifold guarantees that each of these definitions is chart-invariant.

5.1.1 Lie groups

Now that we defined both groups and smooth manifolds, we are ready to define Lie group:

Definition 4 (Lie group)**.** *A* Lie group *is a group and a smooth manifold, such as the group operations are smooth.*

A wide range of examples can be given, most of which are widespread in computer graphics applications:

- $\mathbb{R}^n$ with vector addition,
- Complex numbers $\mathbb{C}$, unit complex numbers S^1 with complex multiplication,
- The quaternions $\mathbb{H}$ and unit quaternions S^3 with quaternion multiplication,
- Any subgroup of $GL(n)$ with the matrix multiplication, which includes rotations, affine, rigid and projective transformations,
- Positive, symmetric definite matrices $\mathbb{S}_+(n)$ with log-product [PFA06],
- Any direct product of these.

Most of the above examples can be seen as subgroups of the general linear group of $n \times n$ invertible, real matrices $GL(n)$. Such subgroups of $GL(n)$ are called *matrix* groups and are probably the most frequently encountered Lie groups in practice. For example, the additive group $(\mathbb{R}^n, +)$ can be seen as a multiplicative subgroup of $GL(n+1)$ using the classical homogeneous coordinates representation for translations. For this reason, we will only consider matrix groups in the remaining of this work.

One particular advantage of Lie groups is that once an algorithm is expressed in the framework of Lie groups, it is automatically available to a wide range of spaces. Examples of this include interpolation [Sho85], averaging [Moa02], splines [KKS95], multi-resolution analysis [LS01, RDS+05] or statistical analysis [Pen06, FLJ03].

5.2 Tangent Space

When animating virtual characters, we will encounter *tangent vectors* to Lie groups, since they are used to represent velocities. Let us now quickly introduce these objects.

5.2.1 Tangent Vectors

Intuitively, a tangent vector to a manifold M at some point $p \in M$ of the manifold is the derivative of some *curve* of M at p, as illustrated on figure 5.2:

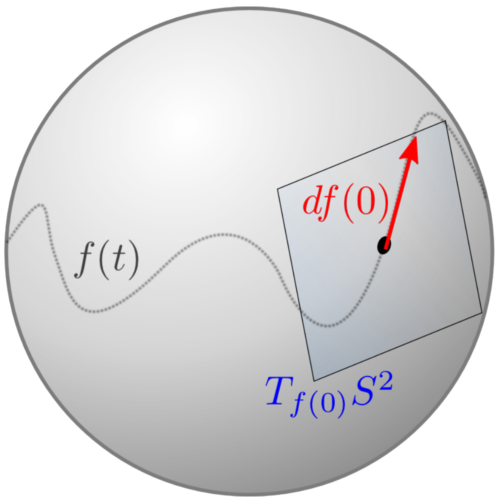


Figure 5.2: The tangent space to the sphere S^2 at $x = f(0)$ *(blue)*, as the set of all possible curve derivatives *(red)* at this point of the sphere. f is an example of such curve.

Formally, it is slightly more complicated, mainly due to the fact that we did not assume manifolds to be surrounded by a Euclidean space. Let $\phi : U \to \mathbb{R}^n$ be a chart around p, we now consider the following curve:

$$\gamma : \mathbb{R} \to M$$
$$\gamma(0) = p$$

Since $\phi \circ \gamma : \mathbb{R} \to \mathbb{R}^n$, it is possible to compute the usual differential of the curve *in coordinates*:

$$d(\phi \circ \gamma)(0) \in \mathbb{R}^n$$

Now, if we identify all the curves sharing the same differential at 0 in coordinates, we obtain an *equivalence class*[2] that is classically defined as a *tangent vector* at p:

[2] Using equivalence classes is a practical way to make the definition chart-invariant.

Definition 5 (Tangent vector)**.** *A* tangent vector *to an n-dimensional smooth manifold M, at a point $p \in M$, is an equivalence class for the equivalence relation:* "having the same tangent in coordinates at 0", *defined on curves of M passing through p at 0.*

The quotient space for this equivalence relation is called the tangent space*, denoted by T_pM, and has a vector space structure of the same dimension as the manifold.*

Notation

In practice, a tangent vector at p implicitly defines a smooth curve γ, passing through p at 0, whose equivalence class (*i.e.* tangent vector at 0) is the considered tangent vector. This tangent vector will be noted $d\gamma(0)$. Most of the time, the derivation at 0 will be implied, unless stated otherwise, so that $d\gamma := d\gamma(0)$.

Under this notation, writing $d\gamma \in T_pM$ both defines the tangent vector *and* the curve γ. When the context is clear, we will sometimes abusively write $dp \in T_pM$ to make the base point p more obvious in computations.

Lie Algebra For a Lie group G, the tangent space at the identity element T_eG plays a special role and will be noted $\mathfrak{g}$. Together with an additional natural operator (the Lie bracket, which we will not discuss here), it is called the *Lie algebra* of the group.

Tangent Bundle The *disjoint* union of all the tangent spaces to an n-dimensional manifold M is called the *tangent bundle* of M, denoted by TM. It is also a smooth manifold, of dimension $2n$.

5.2.2 Differential

The tangent vector definition might seem overly formal, but it allows to see how smooth functions act on tangent vectors. Let $f : M \to N$ be a smooth function between smooth manifolds M and N, and let $dp \in T_pM$ be a tangent vector to M, using the above notation conventions. Consider the following curve on N:

$$f \circ p : \mathbb{R} \to N$$

The associated tangent vector at 0 is thus: $d(f \circ p) \in T_{f(p)}N$. Computations in coordinates show that f induces a linear map between tangent spaces, called the *differential* of f at p and noted $df(p)$, satisfying:

$$\begin{aligned} df(p) : T_pM &\to T_{f(p)}N \\ dp &\mapsto df(p).dp = d(f \circ p) \end{aligned}$$

Said differently, smooth functions between smooth manifolds induce *linear tangent maps* between tangent spaces. It is immediate to verify that the *chain rule* still applies to smooth functions between smooth manifolds.

Jacobian Matrix As a linear mapping between real vector spaces, the differential can be written in terms of the coordinates in the respective basis as a matrix, called the *Jacobian* matrix. For a mapping $f : M \to N$, we will denote the Jacobian matrix of f at p as $J_f(p) \in \mathcal{M}_{n,m}(\mathbb{R})$, for $m = \dim(M)$ and $n = \dim(N)$.

5.2.3 Pullback

The way the differential of a function f acts on tangent vectors is sometimes referred to as the *pushforward* of tangent vectors by f: the function *pushes* vectors from the source tangent space to the destination tangent space. A dual notion can be defined, the *pullback* of linear forms by the function f, from the destination *cotangent* space to the source *cotangent space*. Let us first define the cotangent space:

Definition 6 (Cotangent space)**.** *The* cotangent *space to a manifold M at a point $p \in M$ is the linear dual of the tangent space T_pM,* i.e. *the space of linear forms over T_pM. It is denoted by T_p^*M.*

Let us now consider a smooth function $f : M \to N$, a point $p \in M$ and its image by f: $q = f(p) \in N$. A cotangent vector (also called a *covector*) $d^*q \in T_q^*N$ can be *pulled back* by f as a covector $d^*p \in T_p^*M$, defined by:

$$\forall dp \in T_pM, \quad d^*p.dp = d^*q.\underbrace{df(p).dp}_{\in T_qN} \in \mathbb{R}$$

In order to better see what is happening, one may consider matrix notation instead, by replacing dual signs * with transpose operators ^{T}. The pulled-back covector d^Tp (row-vector) is simply obtained by left multiplying the function Jacobian matrix $J_f(p)$ by the original covector d^Tq (row-vector):

$$d^Tp = d^Tq.J_f(p)$$

The pullback of a covector by f at p defines a linear operator between cotangent spaces, noted $d^*f(p) : T^*_{f(p)}N \to T^*_pM$. Its application to the covector $d^*q \in T^*_{f(p)}N$ is given by:

$$\forall dp \in T_pM, \quad \underbrace{(d^*f(p).d^*q)}_{\in T_p^*M}.dp = d^*q.\underbrace{(df(p).dp)}_{\in T_{f(p)}N} \quad \in \mathbb{R}$$

The matrix associated with this operator is simply the transpose of the Jacobian matrix of f at p. The pullback of covectors will be used to compute generalized forces in the Lagrangian formulation of mechanics.

Lie Coalgebra The cotangent space to a Lie group G at the identity is called its Lie *coalgebra*, denoted by $\mathfrak{g}^*$.

5.2.4 Group Translations

While the preceding definitions were given for general smooth manifolds, we now focus on the differential geometry of Lie groups. Let us first define two important maps that are the *left* and *right* translations:

Definition 7 (Translations)**.** *The left and right translations by an element $h \in G$ are defined by:*

$$\begin{aligned} L_h : G \to G, &\quad g \mapsto h.g \quad \textit{(left)} \\ R_h : G \to G, &\quad g \mapsto g.h \quad \textit{(right)} \end{aligned}$$

From the definition of a Lie group, these operations are smooth and invertible (*i.e.* diffeomorphisms). Let us consider their differentials at the identity element:

$$\begin{aligned} dL_h(e) : \mathfrak{g} \to T_hG \\ dR_h(e) : \mathfrak{g} \to T_hG \end{aligned}$$

These differentials provide two *invertible* linear maps (*i.e.* isomorphisms) between the tangent space at the identity $\mathfrak{g}$, and the tangent space at *any* point T_hG. For matrix groups, these isomorphisms are trivial: let $de \in \mathfrak{g}$ be a tangent vector at the identity, we have:

$$\begin{aligned} dL_h(e).de = d(h.e).de = h.de \\ dR_h(e).de = d(e.h).de = de.h \end{aligned}$$

This means that for any tangent vector $dh \in T_hG$, there are two canonical ways of representing dh by an element of $\mathfrak{g}$, using the inverse mappings:

$$\begin{aligned} dL_h(e)^{-1}.dh = dL_{h^{-1}}(e).dh = h^{-1}.dh \in \mathfrak{g} \\ dR_h(e)^{-1}.dh = dR_{h^{-1}}(e).dh = dh.h^{-1} \in \mathfrak{g} \end{aligned}$$

These two representations of a tangent vector on the Lie algebra are known respectively as the *body* and *spatial* velocities.

5.2.5 Body and Spatial Velocities

Given a tangent vector $dg \in T_gG$, we define the corresponding *body* velocity $d^bg \in \mathfrak{g}$ and *spatial* velocity $d^sg \in \mathfrak{g}$ by the following relation:

$$dg = dL_g(e).d^bg = dR_g(e).d^sg$$

For matrix groups, this simply reduces to:

$$dg = g.d^bg = d^sg.g$$

In practice, these expressions are interesting since they allow to obtain a compact description of tangent vectors. If we store the coordinate derivatives of a rotation matrix $g \in SO(3)$, we need $3 \times 3 = 9$ coefficients. If we store the corresponding body or spatial velocity coordinates instead, we only need 3 coefficients since the Lie algebra $\mathfrak{so}(3)$ is 3-dimensional.

Adjoint

From the above relation, we see that it is possible to convert between body and spatial velocities. This mapping is an isomorphism of the Lie algebra to itself (*i.e.* an automorphism), and is known as the *adjoint* representation of g over its Lie algebra.

$$Ad_g = d(L_g \circ R_{g^{-1}})(e) : \mathfrak{g} \to \mathfrak{g}$$

Again, for matrix Lie groups, the expression is simpler:

$$\begin{aligned} Ad_g(d^bg) &= g.d^bg.g^{-1} \\ &= d^sg \end{aligned}$$

We will also need the *coadjoint* automorphism, defined as:

$$Ad_g^* = d^*(L_g \circ R_{g^{-1}})(e) : \mathfrak{g}^* \to \mathfrak{g}^*$$

In practice, the coadjoint matrix is given by the transpose matrix of the adjoint operator.

Body and Spatial Differentials

Let us consider a smooth function between two Lie groups G and H:

$$f : G \to H$$

This function has a differential at $g \in G$, as seen previously:

$$df(g) : T_gG \to T_{f(g)}H$$

If we express the input and output tangent vectors in their respective Lie algebras $\mathfrak{g}$ and $\mathfrak{h}$ through *body* velocities, we obtain the *body-fixed* differential, also called the *left-trivialized tangent*, noted $d^b f(g)$ and defined by:

$$\begin{aligned} d^b f(g) : \mathfrak{g} &\to \mathfrak{h} \\ d^b g &\mapsto f(g)^{-1}.df(g).g.d^b g \end{aligned}$$

This definition is perhaps more explicit using this diagram:

$$\begin{array}{l} \mathfrak{g} \to T_g G \longrightarrow T_{f(g)} H \to \mathfrak{h} \\ d^b g \mapsto \underbrace{g.d^b g}_{dg} \mapsto \underbrace{df(g).dg}_{dh} \mapsto d^b h = d^b f(g).d^b g \end{array}$$

The idea is simply to compose the body-velocity representations at both ends of the differential, in order to obtain a mapping between Lie algebras. In the same spirit, one can define the *right-trivialized* tangent:

$$\begin{array}{l} \mathfrak{g} \to T_g G \longrightarrow T_{f(g)} H \to \mathfrak{h} \\ d^s g \mapsto \underbrace{d^s g.g}_{dg} \mapsto \underbrace{df(g).dg}_{dh} \mapsto d^s h = d^s f(g).d^s g \end{array}$$

Again, in practice, using left or right trivialized tangents usually simplifies computations considerably as they can be expressed in the canonical basis of the respective Lie algebras. Left and right trivialized tangents are related using the corresponding adjoints.

Notation In all this work, a $.^b$ superscript will indicate a *body* quantity, while a $.^s$ superscript will indicate a *spatial* quantity. This convention will be used for all tangent vectors/covectors and differential/pullbacks, but also for Jacobian matrices.

5.3 Exponential, Logarithm

As a smooth manifold, a Lie group can be described using an atlas of charts. However, certain charts have particularly interesting properties, as it is the case for the exponential map.

5.3.1 Vector Fields, Integral Curves

A vector field is simply a function that assigns a tangent vector at the point it is evaluated, in a smooth way:

Definition 8 (Vector field). *Let M be a smooth manifold. A* vector field *on M is a smooth function X such as:*

$$\begin{aligned} X : M &\to TM \\ p &\mapsto X(p) \in T_pM \end{aligned}$$

Said differently, a vector field is an Ordinary Differential Equation (ODE) on a smooth manifold. An *integral curve* for X passing through $p \in M$ is a solution of this ODE for the initial value $y(0) = p$, that is a curve γ satisfying:

$$\begin{aligned} \gamma : U \subset \mathbb{R} &\to M \\ d\gamma(t) &= X(\gamma(t)) \\ \gamma(0) &= m \end{aligned}$$

for some real open set U containing 0. Intuitively, an integral curve can be pictured as the trajectory of a particle flowing under the velocity field described by X, starting at $p \in M$ for a given amount of time.

5.3.2 Left-Invariant Vector Fields

Among all possible vector fields of a Lie group G, let us consider the ones for which all the tangent vector have the *same* body velocity: such vector fields are called *left-invariant*:

Definition 9 (Left-invariant vector field). *A left-invariant vector field X on a Lie group G satisfies:*

$$X(g) = dL_g(e).X(e)$$

where $e \in G$ is the identity element. In terms of the body velocity $X^b(g)$, this simply means that:

$$X^b(g) = X(e)$$

Similarly, one can define *right-invariant* vector fields. We see that invariant vector fields are entirely described by their value at the identity. For this reason, invariant vector fields are very compact and easy to manipulate.

5.3.3 Exponential

It turns out that both left and right-invariant vector fields of a Lie group share the same integral curves[3]. If we let $v \in \mathfrak{g}$ be velocity at the identity, and γ_v

[3] Left and right-invariant vector fields can be identified using the inverse mapping.

be the integral curve for the associated left- (or right-) invariant vector field, starting at the identity, we are now ready to define the *exponential* map:

$$\begin{aligned}&\exp : \mathfrak{g} \to G\\ &\exp(v) = \gamma_v(1)\\ &\exp(0) = e\end{aligned}$$

Intuitively, the exponential is obtained by flowing from the identity along a left- (or right-) invariant velocity field, for a unit of time. Fortunately, for matrix groups, this mapping has more familiar expression:

Definition 10 (Exponential map)**.** *For matrix groups, the exponential map is given by the usual power series:*

$$\forall v \in \mathfrak{g}, \quad \exp(v) = \sum_{i=0}^{\infty} \frac{v^i}{i!}$$

For most matrix groups, this series has a much more compact expression. Note that the exponential is not injective in general: for example, the exponential for the unit complex circle is the usual complex exponential $e^{i.\theta}$, which is 2π-periodic. However, it does cover all of the connected component of the identity[4]. Its differential at $0 \in \mathfrak{g}$ is the identity mapping:

$$d\exp(0) = Id_{\mathfrak{g}}$$

It is also worth noting that even if the group is not commutative, elements along an exponentiated vector line still commute [5]:

$$\forall (s,t) \in \mathbb{R}^2, \quad \exp\left((s+t)\,.v\right) = \exp(s.v)\ \exp(t.v)$$

A useful consequence of this fact is that: $\exp(-v) = \exp(v)^{-1}$. Even if the power series formula is all that is needed in practice, the geometric intuition conveyed by invariant vector fields is much more effective at describing the behavior of this function.

5.3.4 Logarithm

The logarithm is simply the inverse of the exponential map, when defined. Since the exponential might be periodic, one usually defines the logarithm using only the period neighboring the origin. Let $v \in \mathfrak{g}$ and $g = \exp(v)$, we call:

[4] To see this, consider $(\mathbb{R}^*, .)$: one can not reach negative numbers using the exponential, but every positive number can be reached.

[5] The exponential is called a *one-parameter subgroup*.

$$t^\star = \operatorname*{argmin}_{\exp(t.v)=g} |t|$$

the smallest absolute value of time for which the exponential flow reaches g for the initial velocity v (including negative values). The logarithm is then defined by:

$$\log(g) = t^\star v \in \mathfrak{g}$$

Intuitively, the logarithm is used to obtain the tangent *direction* of a group element, as seen from the identity, whereas the exponential follows this direction to reach the corresponding element.

5.3.5 Practical Interest

In practice, the exponential provides a natural chart around the identity of a Lie group. This chart can in fact be used anywhere by left (resp. right) translation: $L_g \circ \exp$ will be a chart around g. Compact, closed-form formula are available for the groups we use in this work (see [BM95]).

The exponential can be seen as a mean of moving *inside* a Lie group given tangent vectors. For example, the classical Spherical Linear Interpolation (SLERP) algorithm [Sho85] for rotation interpolation can be expressed in terms of the exponential map. Let $p, q \in SO(3)$ and $\alpha \in \mathbb{R}$:

$$SLERP(p, q, \alpha) = p.\exp\left(\alpha \log\left(p^{-1}q\right)\right)$$

The algorithm first obtains the tangent direction of q as seen from p, by computing $\log\left(p^{-1}q\right)$, then performs linear interpolation in the tangent space, and finally maps the result back to $SO(3)$ using the exponential.

5.3.6 Riemannian Geometry

Compact Lie groups such as $SO(3)$ can be given the additional structure of a *Riemannian* manifold, which is needed to perform consistent measures of angles, lengths or curvatures in a smooth manifold.

Definition 11 (Riemannian manifold)**.** *A* Riemannian manifold *is a smooth manifold M equipped with a* Riemannian metric*: a smoothly varying collection of inner products over the tangent spaces of M. The Riemannian metric at $p \in M$ is thus a symmetric, definite, positive, bilinear form on T_pM:*

$$\langle ., . \rangle_p : T_pM \times T_pM \to \mathbb{R}$$

The Riemannian metric is also known as the metric tensor*.*

Using this metric, it is possible to measure the length of a curve γ on the interval $[a, b]$ by summing the norms of tangent vectors along γ:

$$L_a^b(\gamma) = \int_a^b ||d\gamma(t)||_{\gamma(t)} .dt$$

Note that at each point $\gamma(t)$, the metric $\langle ., . \rangle_{\gamma(t)}$ is used to compute the tangent vector norm $||d\gamma(t)||_{\gamma(t)}$.

Geodesics

On a Riemannian manifold, curves of minimal length are of special interest since they provide a *metric* generalization of straight lines in the vector case. Formally, geodesics are defined by *locally length-minimizing* curves, parametrized with *constant velocity*[6]. For a given starting point $p \in M$ and initial velocity $v \in T_pM$, there is a unique geodesic curve starting at p with initial velocity v. Moreover, this geodesic has constant velocity:

$$||d\gamma_v(t)|| = ||v||$$

The geodesics of a *connected* Riemannian manifold turn it into a *metric space*, with a distance function:

$$d : M \times M \to \mathbb{R}$$

returning the smallest length of geodesic curves between two points.

Riemannian Exponential

The Riemannian geometry also has an exponential map, but this one is related to geodesics: if γ_v is *the* geodesic curve starting at $p \in M$ with initial tangent vector $v \in T_pM$, the Riemannian exponential at p is defined as:

$$\exp_p(v) = \gamma_v(1)$$

Intuitively, this means that we start from the point p and flow along the geodesic γ_v for a unit of time. Since the velocity has constant norm, the following is true for small enough $v \in T_pM$:

$$\begin{aligned} d\left(p, \exp_p(v)\right) &= \int_0^1 ||d\gamma_v(t)|| .dt \\ &= ||v|| \end{aligned}$$

The inverse function (when defined) is known as the Riemannian logarithm, and can be used to compute the Riemannian distance:

[6] *i.e.* such as the length between $\gamma(a)$ and $\gamma(b)$ is proportional to $|b - a|$

$$d(x,y) = ||\log_x(y)|| = ||log_y(x)||$$

The Riemannian logarithm is illustrated on figure 5.3.

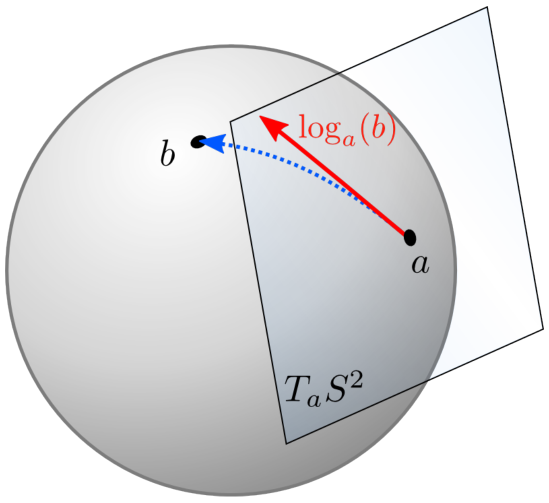


Figure 5.3: The Riemannian logarithm of b at a *(red)*, and the corresponding geodesic curve *(blue)*: $b = \exp_a(\log_a(b))$.

Relation to Lie Groups

As mentioned above, compact Lie groups have a bi-invariant Riemannian metric, that is a Riemannian metric that is both left- and right- invariant. In this case, the two exponential functions coincide, meaning it is possible to compute *geodesics* using the Lie exponential, *i.e.* without performing costly minimization procedures but directly using a closed-form expression.

This fact will be of great help when performing statistical analysis of skeleton poses. Indeed, as a compact Lie group, the pose manifold $SO(3)^n$ possesses a natural Riemannian metric that enables the use of several non-linear statistical tools. The compatible Lie group structure has the added benefit of allowing simple closed-form expressions for computing geodesics.

Chapter 6

Skeleton Kinematics

In this chapter, we recall basic skeleton kinematics that will be used throughout the remaining chapters. We quickly review the Lie groups $SO(3)$ and $SE(3)$, forward kinematics and classical Inverse Kinematics (IK) algorithms.

6.1 Euclidean Transformations

This section quickly reviews the two Lie groups $SO(3)$ and $SE(3)$ and provides visual interpretation of related concepts. Most of this section has been adapted from [MSZ94].

6.1.1 Definitions

Special Orthogonal Group

The rotation group, or *special orthogonal group* is the space of linear maps of $\mathbb{R}^3$ to itself that preserve the euclidean norm and orientation:

Definition 12 (Rotation group). *The* rotation group *is the set of* 3×3 *real, orthogonal matrices with determinant* $+1$*, together with the matrix multiplication:*

$$SO(3) = \left\{ g \in GL(3) / \quad g^T.g = g.g^T = I, \ \det(g) = 1 \right\}$$

Its Lie algebra is given by the set of antisymmetric matrices:

$$\mathfrak{so}(3) = \omega \in \mathcal{M}_{3,3}(\mathbb{R}) / \quad \omega^T = -\omega$$

If we consider the coordinates of $\mathfrak{so}(3)$ elements in the canonical basis through the following mapping:

$$\begin{aligned}\widehat{\ }: \mathbb{R}^3 &\to \mathfrak{so}(3)\\ \begin{pmatrix}x\\y\\z\end{pmatrix} &\mapsto \begin{pmatrix}0 & -z & y\\ z & 0 & -x\\ -y & x & 0\end{pmatrix}\end{aligned}$$

we obtain the usual notion of *angular velocity vector*. The adjoint mapping verifies:

$$Ad_g(\widehat{\omega}) = \widehat{g.\omega}$$

Thus, the conversion between body and spatial velocities corresponds converting angular velocity vector coordinates from the body frame to the spatial frame (hence the name).

Special Euclidean Group

The *special euclidean group* is the space of orientation-preserving *isometries* of $\mathbb{R}^3$ *i.e.* that preserve the euclidean norm between any pair of points. It can be shown that its elements (the *rigid motions*) are affine maps whose linear part lies in $SO(3)$. It is usually represented with 4×4 matrices using homogeneous coordinates:

Definition 13 (Special Euclidean group). *The Special Euclidean group $SE(3)$ is the space of 4×4 matrices of the form:*

$$g = \begin{pmatrix}r & t\\ 0 & 1\end{pmatrix}$$

with $r \in SO(3)$, $t \in \mathbb{R}^3$. Its Lie algebra is given by 4×4 matrices of the form:

$$\kappa = \begin{pmatrix}\omega & v\\ 0 & 0\end{pmatrix}$$

$\omega \in \mathfrak{so}(3)$ is called the angular *velocity and $v \in \mathbb{R}^3$ the* linear *velocity.*

Elements of the Lie algebra $\mathfrak{se}(3)$ are sometimes called *twists*, and elements of the coalgebra $\mathfrak{se}(3)^*$ called *wrenches*. We extend the definition of the $\widehat{\ }$operator to $\mathfrak{se}(3)$ as follows:

$$\begin{aligned}\widehat{\ }: \mathbb{R}^6 &\to \mathfrak{se}(3)\\ \begin{pmatrix}\omega\\v\end{pmatrix} &\mapsto \begin{pmatrix}\widehat{\omega} & v\\ 0 & 0\end{pmatrix}\end{aligned}$$

The coordinates of a body velocity twist correspond to the intuitive interpretation: ω represents the coordinates of the angular velocity expressed in the body frame, and v is the linear velocity of the frame origin, also expressed in the body frame. However, the *spatial* velocity twist interpretation is a bit counter-intuitive, since its linear part is *not* the linear velocity of the frame origin, expressed in the spatial frame as one could expect. Instead, it is the linear velocity of the *world* origin, *as if it were part of the rigid body*, expressed in the world frame. The angular part of the spatial velocity twist is the angular velocity expressed in the world frame. See [MSZ94] for details.

6.1.2 Torques, Wrenches

While the tangent *vectors* represent the notion of *velocity*, the notion of *force* is represented by *cotangent* vectors. For example, *conservative forces* arise in the Euler-Lagrange equations as the differential of the potential energy $V : Q \to \mathbb{R}$, which is a cotangent vector:

$$dV(q) \in T_q^*Q$$

The natural action of *cotangent* vectors on *tangent* vectors defines the *instantaneous work*, sometimes called *virtual work*. Letting $f_q \in T_q^*Q$ and $v_q \in T_qQ$, the instantaneous work δW is simply:

$$\delta W = f_q.v_q \in \mathbb{R}$$

The term *natural action* emphasizes that this operation does *not* depend on a metric structure defined on the tangent space, such as the kinetic energy tensor. It is simply the application of a linear form to a vector, unlike the inner product of two vectors.

All these definitions obviously apply to Lie groups (as smooth manifolds) and in particular to $SO(3)$ and $SE(3)$, where generalized forces are respectively called *torques* and *wrenches*. Just like tangent vectors can be expressed as body and spatial velocities, cotangent vectors can be expressed as body and spatial forces through left and right translations.

For a tangent covector to a Lie group $f_g \in T_g^*G$, the equivalent body force is obtained by *pulling* f_g over the Lie coalgebra, by the left translation mapping L_g:

$$f_g^b = d^*L_g(e).f_g \tag{6.1}$$

To better see what is happening, one can always decompose this operation through the instantaneous work to obtain an *equivalent* body force. For any tangent vector $v_g \in T_gG$, we have:

$$\begin{aligned}\delta W &= f_g.v_g \\ &= \underbrace{f_g.dL_g(e)}_{f_g^b}.v_g^b\end{aligned}$$

which is precisely equation (6.1). For matrix groups such as $SO(3)$ and $SE(3)$, the expression is, again, simpler:

$$f_g^b = f_g.g$$

Using the same reasoning, we obtain that the spatial and body forces are related through the *coadjoint* mapping:

$$\begin{aligned}\delta W &= f_g^b.v_g^b \\ &= \underbrace{f_g^b.Ad_{g^{-1}}}_{f_g^s}.v_g^s\end{aligned}$$

Thus:

$$f_g^s = Ad_{g^{-1}}^*.f_g^b$$

We now illustrate how the adjoint and coadjoint mappings can be used to express velocities and forces in different frames.

6.1.3 Frame Change

The adjoint transformation not only allows to convert between body and spatial velocities, it is also used to express velocities in different frames. We illustrate this on $SE(3)$, but the same is true with other Lie groups.

Let us consider the situation described by figure 6.1, where rigid transformations $a, b, c \in SE(3)$ are related by $c = ab$. If frames a and c move together, *i.e.* if b is constant, then the following hold between body velocities, with respect to the reference frame:

$$\begin{aligned}dc = da.b \Rightarrow d^b c &= c^{-1}.dc \\ &= b^{-1}a^{-1}.da.b = Ad_{b^{-1}}.d^b a\end{aligned}$$

This formula is illustrated on figure 6.2. Notice that velocities are pushed between frames in the *opposite* way, compared to the relative transform. The previous relation on velocities can be used to transport generalized forces from one frame to another. Let $f_c^b \in \mathfrak{g}^\star$ be a body force acting on c, the equivalent body force $f_a^b \in \mathfrak{g}^\star$ acting on frame a is given by:

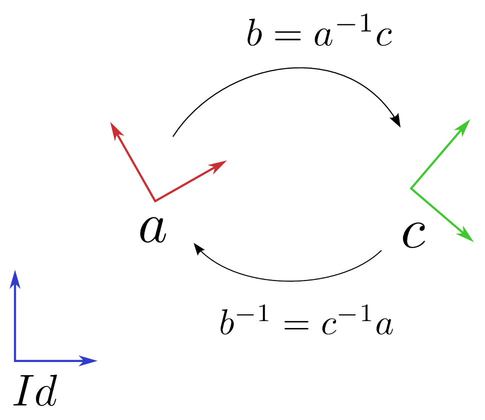


Figure 6.1: The group identity is the reference frame *(blue)* relative to which frames a and c are defined *(red, green)*. The configuration of c relative to a is given by b, so that $c = ab$.

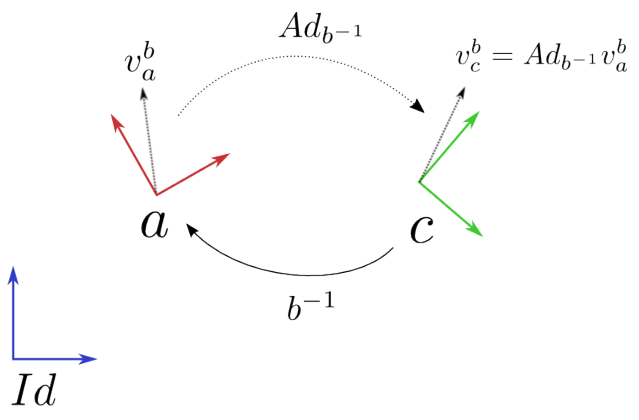


Figure 6.2: The effect of frame change over velocity vectors. If $b = c^{-1}a$ is constant, the adjoint gives the transformation between the body velocities at frames a and c

$$f_c^b . v_c^b = \underbrace{f_c^b . Ad_{b^{-1}}}_{f_a^b} . v_a^b$$

Thus:

$$f_a^b = Ad_{b^{-1}}^* . f_c^b$$

This formula is illustrated on figure 6.3. Notice that the forces are pulled between frames in the *same* way as the relative transform.

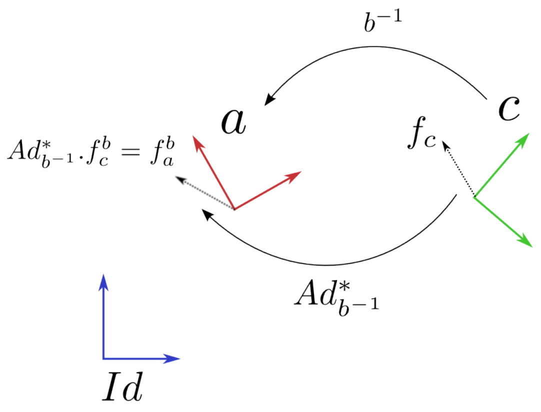


Figure 6.3: The effect of frame change over cotangent vectors. If $b = c^{-1}a$ is constant, then the adjoint transpose gives the transformation between cotangent vectors expressed in frames c and a.

6.2 Forward Kinematics

Let us now provide a quick introduction to the problems of forward and inverse kinematics. Given an n-joints skeleton parametrized by its *configuration manifold* Q, the *forward kinematics* problem aims at computing the function:

$$f : Q \to SE(3)^{n+1}$$

where f maps skeleton DOFs to absolute bone configurations. For an articulated skeleton consisting only of ball-and-socket joints, the configuration manifold is a Lie group:

$$G = SE(3) \times SO(3)^n$$

describing the rigid configuration of the root bone and the relative orientation of each joint with respect to its parent. We denote by $a_i \in \mathbb{N}$ the index of the parent bone of the bone with index $i \in \mathbb{N}$. Additionally, we refer to parent and child joint offsets respectively as $p_i \in SE(3)$ and $c_i \in SE(3)$, as described on figure 6.4:

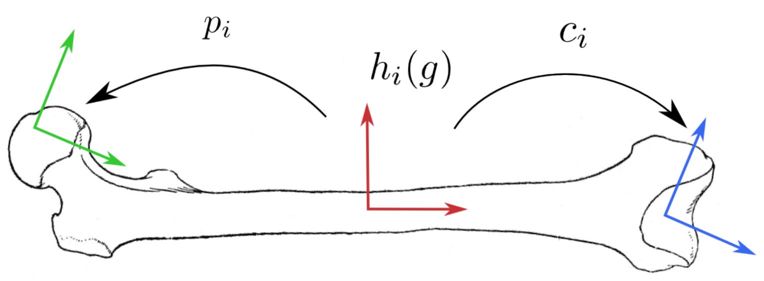


Figure 6.4: Each bone (here, a femur) has a world configuration $f_i(g)$ obtained by forward kinematics *(red)*. p_i and c_i respectively describe the parent *(green)* and child *(blue)* frame transformations with respect to the bone frame.

The forward kinematics mapping is compactly expressed using a recursive formula. Let $g = (q, r) \in G$ and $f(g) = (f_i(g))_{i \leq n-1}$, assuming that the root bone has index 0 and that the joint between bones a_i and i has index $i - 1$, this mapping is defined as:

$$\begin{aligned} f_0(g) &= q \\ f_i(g) &= f_{a_i}(g).c_{a_i}.r_{i-1}.{p_i}^{-1} \end{aligned}$$

This function only involves Lie group operations, it is thus smooth. Letting $dg = (dq, dr) \in T_gG$, its differential is given by:

$$\begin{aligned} df_0(g).dg &= dq \\ df_i(g).dg &= df_{a_i}(g).dg.c_{a_i}.r_{i-1}.{p_i}^{-1} \; + \; f_{a_i}(g).c_{a_i}.dr_{i-1}.{p_i}^{-1} \end{aligned}$$

The corresponding left-trivialized tangent is:

$$\begin{aligned} d^b f_0(g).d^b g &= d^b q \\ d^b f_i(g).d^b g &= f_i(g)^{-1}.df_i(g).g.d^b g \\ &= Ad_{\left(c_{a_i} r_{i-1} {p_i}^{-1}\right)^{-1}}.d^b f_{a_i}(g).d^b g \; + \; Ad_{{p_i}^{-1}}.d^b r_i \\ &= Ad_{\left(p_i {r_{i-1}}^{-1} {c_{a_i}}^{-1}\right)}.d^b f_{a_i}(g).d^b g \; + \; Ad_{{p_i}^{-1}}.d^b r_i \end{aligned}$$

From the above algorithm, we see that the pushforward of a tangent vector can be computed in $O(n)$ operations. The algorithm presented here is fully general in the sense that any feature computed from a bone configuration can be expressed as a function of the DOFs, by composition. For instance, if we are interested in the absolute position of an end-effector belonging to character hand, we may chain the above kinematics mapping with the following:

$$\begin{aligned} h : SE(3)^n &\to \mathbb{R}^3 \\ (x_i)_{i\leq n} &\mapsto x_k(p) \end{aligned}$$

where $p \in \mathbb{R}^3$ describes the local coordinates of the end-effector with respect to the hand frame, and $k \in \mathbb{N}$ is the hand index. In most cases, the forward kinematics problem is only concerned with the computation of the composed mapping $h \circ f$, for features defined by the function h.

6.3 Inverse Kinematics

While the forward kinematics problem is straightforward, the inverse problem of finding the skeleton configuration satisfying a given set of constraints, if any, is much harder. This is mostly due to the fact that the IK problem is usually ill-defined, meaning that zero, one or more configuration might satisfy the required constraints.

We quickly review the classical IK algorithms and associated optimization methods, as they will be used in the remaining of this work. A much more detailed introduction to these methods can be found in [Bus04]. In specific cases, an analytical solution can be found (such as Paden-Kahan subproblems [MSZ94]) but in the general case, including the methods presented here, the IK problem is cast an optimization problem, possibly with constraints.

In the remaining of this section, we will consider a given smooth *feature* function h, describing an abstract set of features in function of the skeleton degrees of freedom $G = SE(3) \times SO(3)^n$:

$$h : G \to H$$

where H is an Euclidean space for simplicity. The *target* features will be denoted by $h^\star \in H$, and the *error* by $e(g) = h^\star - h(g)$.

6.3.1 Jacobian Transpose

The most straightforward approach to IK is to apply the classical Gradient Descent algorithm to the following optimization problem:

$$g^\star = \underset{g\in G}{\operatorname{argmin}} \quad ||h^\star - h(g)||^2 =: c(g)$$

The cost function $c : G \to \mathbb{R}$ has a differential $dc(g) \in T_g^*G$ defined by:

$$dc(g).dg = 2\langle h^\star - h(g), dh(g).dg\rangle \in \mathbb{R}$$

The gradient descent algorithm uses the corresponding gradient $\nabla c(g) \in T_g G$ as the descent direction, using the exponential mapping to remain inside the group:

$$g_{k+1} = g_k . \exp\left(-\alpha . \nabla^b c(g_k)\right)$$

where $\alpha \in \mathbb{R}^+$ is the step size, usually computed using a *line search* algorithm. The name *Jacobian transpose* comes from the expression of the gradient. If we assume a canonical inner product on H, then the gradient expression is, in matrix notation:

$$\nabla^b c(g) = 2.J_h^b(g)^T . \left(h^\star - h(g)\right)$$

where $J_h^b(g)$ is the (body-fixed) Jacobian matrix of h at g. This method suffers from all the classical drawbacks of the gradient descent, the principal being its slow convergence.

6.3.2 Pseudo-Inverse

The *pseudo-inverse* method is the application of the classical Gauss-Newton algorithm for non-linear least-squares, to the IK problem. The optimization problem is, again:

$$g^\star = \underset{g \in G}{\operatorname{argmin}} \quad ||h^\star - h(g)||^2$$

The Gauss-Newton algorithm looks for a configuration step $\Delta g \in \mathfrak{g}$ that solves the *approximate* optimization problem:

$$\Delta g^\star = \underset{\Delta g \in \mathfrak{g}}{\operatorname{argmin}} \left|\left| h^\star - \left(h(g) + d^b h(g).\Delta g\right) \right|\right|^2$$

obtained by locally linearizing the function h using the exponential map:

$$h\left(g.\exp(\Delta g)\right) \approx h(g) + d^b h(g).d^b \exp(0).\Delta g = h(g) + d^b h(g).\Delta g$$

The above approximate least-squares problem is a *linear least squares* problem, which can be solved by the mean of a pseudo-inverse matrix. If we call $J = J_h^b(g)$ the body Jacobian matrix of h at g, the *normal equations* for above problem are:

$$J^T J.\Delta g = J^T.e(g)$$

This means that $\Delta g = \left(J^T J\right)^{-1} J^T.e(g) = J^+.e(g)$, where J^+ is the Moore-Penrose pseudo-inverse of matrix J. The *damped pseudo-inverse* prevents too large values of Δg by solving the following problem instead:

$$\Delta g^\star = \underset{\Delta g \in \mathfrak{g}}{\operatorname{argmin}} \left|\left| h^\star - \Big(h(g) + d^b h(g).\Delta g \Big) \right|\right|^2 \quad + \quad \lambda \left|\left|\Delta g\right|\right|^2$$

where $\lambda \in \mathbb{R}$ is a damping parameter. This is still a *linear* least squares problem, whose normal equations are:

$$\left(J^T J + \lambda I\right).\Delta g = J^T.e(g)$$

The damped pseudo-inverse algorithm is also known as Levenberg's algorithm. When the damping parameter λ is set high, this method approaches the gradient descent and conversely. This can be seen as a Tikhonov regularization [TA77] of the Gauss-Newton system.

6.3.3 Levenberg-Marquardt

Marquardt's insight was to *selectively* damp dimensions according to the *curvature* of the above approximate cost function. The intuitive idea is that strongly curved directions, corresponding to strong end-effector velocities, should not rely on the linear approximation and use gradient descent instead. Conversely, directions where the cost function is almost flat, corresponding to small end-effector velocities, should use the tangent approximation, which also avoids slow gradient convergence.

The *Hessian* matrix of the approximate cost function is given by $H = J^T J$, whose diagonal values indicate the curvature of the (approximate) cost function in the corresponding dimension. Therefore, the resulting normal equations become:

$$\left(J^T J + \lambda.\text{diag}\left(J^T J\right)\right).\Delta g = J^T.e(g)$$

The resulting method is known as the Levenberg-Marquardt algorithm. We propose a damping model for physically-based character animation inspired by this algorithm in 11.2.2.

Part III

Kinematics

Chapter 7

Previous Work

In this part, we derive a reduced-dimension, data-driven kinematic model for an articulated skeleton, based on existing non-linear statistical analysis. After reviewing previous work in character animation in general, and reduced dimension models in particular, we present an existing statistical analysis algorithm, named Principal Geodesic Analysis, and its application to character animation in more details. Then, we evaluate the interest of this approach with a novel *motion compression* algorithm based on this reduced kinematic model.

We begin this chapter with a general review of existing character animation techniques. We then motivate the need for a simple, data-driven, reduced kinematic pose model. Finally, we review previous work in the field of non-linear data analysis for suitable dimension reduction techniques.

7.1 Character Animation

In this section, we propose a general overview of the main existing character animation techniques. These can be partitioned in three principal classes, though some overlap may exist. We begin with *kinematic methods*, which mainly use skeleton kinematics described in section 6.3 to synthesize poses under constraints. *Motion reuse* techniques, on the contrary, rely on existing motion capture data to generate new motion. Finally, *machine learning* techniques build motion or pose models based on existing data, which are then used to synthesize new motions.

7.1.1 Kinematic Synthesis

Kinematic motion synthesis can be seen as an extension of the problem of the IK problem to the temporal domain: given a skeleton and its geometry, the goal is to find a series of skeleton configurations that satisfy geometric constraints across time.

Inverse Kinematics

An immediate solution is to employ inverse kinematics for each frame, and to animate end-effectors instead. This approach is available in major commercial animation software such as Alias Maya[1] or 3D Studio[2]. They generally use some form of analytic IK solver (*e.g.* Kahan-Paden sub-problems, see [MSZ94]) for optimal speed. Jacobian pseudo-inverse solvers [GM85] were also used, or their regularized variant Levenberg-Marquardt described in 6.3. Joint limits, or specific limb configurations can be enforced by solving a non-linear program [ZB94].

Since the IK problem is ill-posed, several solutions may exist for given target end-effector positions, which may lead to popping artifacts on the resulting animation. This is usually corrected by a smoothness term biasing the current solution towards the one at the previous time step.

Even if the synthesized motion is smooth, all these techniques usually result in poor quality human animations that exhibit a "robotic" aspect as well as physical inconsistencies, unless special care is taken, such as the large number of end-effector key-frames used by artists during motion pictures creation.

Constrained Optimization

The principle of *continuous constrained optimization*, also known as *space-time constraints* is to globally optimize pose configurations for the whole motion sequence, so that higher-level invariants (*e.g.* physics laws, temporal smoothness of the solution), or features (*e.g.* having the character hand reaching a particular object at a given time), can be enforced as constraints in an optimization problem.

A pioneer work in this field is [WK88], in which Sequential Quadratic Programming (SQP) is used to solve for the minimum actuation of a Luxo lamp (*cf.* figure 7.1), under key-frame and physics laws constraints. A similar framework was used for the simpler problem of interpolating between existing clips [RGBC96]. [Gle97] used space-time constraints to edit existing motion, or to adapt the motion of one character to another, a technique known as *motion re-targeting* [Gle98]. Continuous optimization was also used in bio-mechanics [AP99].

While these methods provide impressive results, they tend to be computationally expensive due to the dimensionality of the problem. This usually forces the use of simplified models to represent the character, at the expense of quality. Furthermore, these methods are usually subject to local extrema problems which are difficult to deal with in a robust way. In particular, these techniques usually require a good initial guess.

[1] http://www.autodesk.com/maya/
[2] http://www.autodesk.com/3ds-max/

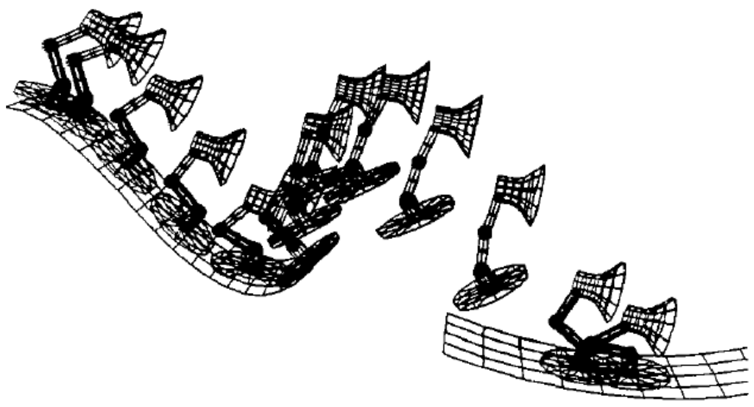


Figure 7.1: The famous Luxo lamp used in *spacetime constraints* [WK88].

Addressing these issues, [PW99] propose a physically-based motion editing tool, featuring space-time constraints on a simplified character model for editing an existing capture, which attenuates local extrema problems. [LP02] propose to enforce simpler linear and angular momentum constraints to avoid solving for skeleton actuation, speeding up the optimization while still producing plausible results. [FP03] restrict the optimization to a set of *features* to speed-up differential quantities computations. [SHP04] proposes a dimension reduction process to bias the optimization towards a low-dimensional subspace. Yet, the computational cost of all these methods remains too high to allow their use for real-time applications.

Interestingly, [LS99] propose a multi-resolution approach to the problem by encoding a motion as a set of *level of details*, using hierarchical B-splines. A custom IK solver then enforces constraints on this representation, with details added when needed. By replacing a single large optimization problem by many smaller, localized ones, the authors achieve interactivity in their motion editing system. However, the performance of this approach is highly dependent on the time length of constraints. Furthermore, spatial correlations existing in input data are not taken into account.

7.1.2 Motion Reuse

A natural alternative to produce highly-detailed motion data is to reuse existing, real motion data, and modify it so that it satisfies user constraints. An obvious, greedy method would consist in having a sufficient quantity of data covering all the possible cases for the constraints, and then select relevant data from this motion database, based on clustering techniques. Of course, much more evolved techniques have been proposed.

Motion Warping

Motion data can also be seen as a time signal that can be edited as any other time series. A very simple way to do so consists in scaling or offsetting the signal in the time domain. This allows simple transition, overlap or blending of similar motions, techniques collectively known as *motion warping* [WP95]. More advanced signal processing techniques can also be applied, either in the time domain [BW95] (low pass, displacement map, gains, time warps,...), or in the frequency domain where Fourier coefficients are interpolated to transition between walk and run motions [UAT95] (*cf.* figure 7.2).

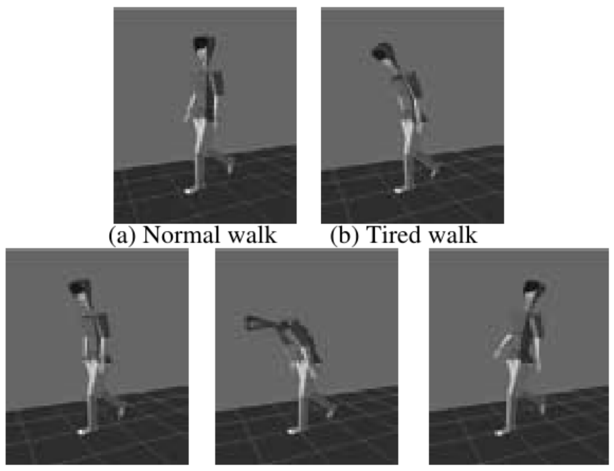


Figure 7.2: Example animations obtained by modifying Fourier coefficients to obtain different motion styles [UAT95].

Fourier coefficients difference between motions may also be transferred to another motion. Unfortunately, these techniques require carefully aligned motions in both time and space to produce convincing results.

Motion Blending, Interpolation

The blending, or interpolation, of several animations has also received attention, since it allows a high-level control at a reasonable computational cost. [RCB98] use Radial Basis Functions (RBFs) to interpolate between both different behaviors *(verbs)* and styles within the behaviors *(adverbs)*. This method relies on hand-chosen key events to temporally align motions before interpolation. [KG04b] proposed a method to automatically parametrize variations in similar motions for a whole motion database, featuring similar motion clustering and automatic time registration. Such technique allows to automatically parametrize a family of reaching motion by the end-effector position, speeding up IK computation. [SH05] proposed interpolation strategies to enforce correctness of physical quantities such as linear or angular momentum, frictional behavior. Fast linear blending strategies have been proposed in [WB08].

The motion blending methods are generally restricted to mixing existing motion clips. Despite their speed, they are usually limited to the problem of transitioning between existing motion clips, and therefore lack the diversity and control of other motion synthesis methods.

Motion Graphs

Initial version of *motion graphs* [KGP02] detected possible transitions frames in motion capture data, organized transition points in a graph structure allowing the continuous synthesis of an animation by traversing the graph (*cf.* figure 7.3). This graph walk can be driven by external events, allowing interactive synthesis of character animations in a video game for example. Various improvements have been proposed, by improving both the quality and the number of transitions [ZS09].

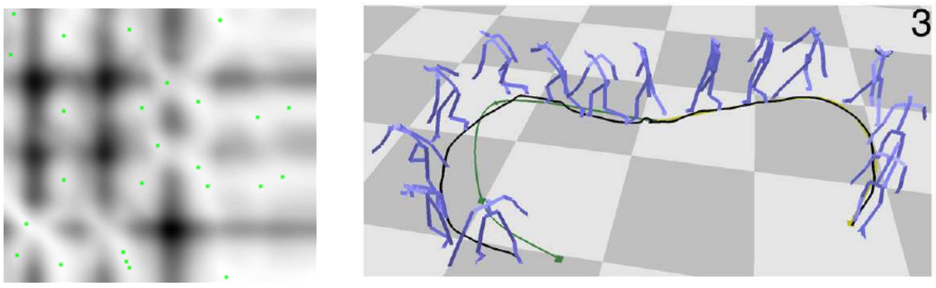


Figure 7.3: Motion Graphs as originally proposed in [KGP02]. The error function *(left, green)* minima describe valid transition points between frames i and j. A new motion path can then be recomputed from the corresponding graph *(right)*.

Motion graphs also allow for advanced off-line path selection strategies: [SH07] used A^* search to find an optimal solution given path constraints. [TLP07] used a motion graph-like motion engine driven by a reinforcement learning controller to select near-optimal transitions in real-time.

In general, motion graphs offer very little editing facilities over resulting animations, as they only provide suitable transition points between existing motion sequences.

7.1.3 Machine Learning

Recent advances in the machine learning theory allowed to design systems that capture geometric features as well as modeling time evolution on existing motion capture data, allowing to synthesize plausible motion while still retaining good control.

Interpolation Space

The idea of using machine learning to compute a higher-level parametrization of a motion given real examples is appealing. To this end, several classes of algorithms have been tried. [RCB98] used RBFs to learn an interpolation space from examples. [MK05] proposed to use geo-statistics to perform better interpolation in a control space.

Brand and Hertzmann [BH00] propose a cross-entropy optimization framework to learn Hidden Markov Models (HMMs) for both structure and style from motion capture data. Style-specific parameters are then de-correlated using Principal Component Analysis (PCA), allowing a low-dimensional style transfer: from a motion y in a different style, the structure $S(y)$ and style parameters are extracted, then a new motion y' for different style parameters v is synthesized by calculating the maximum-likelihood path:

$$\underset{y'}{\operatorname{argmax}} \quad p\left(y' | v, S(y)\right)$$

While promising, this method requires an expansive optimization, and the model is tied to the training set, making the generation of completely new motions difficult.

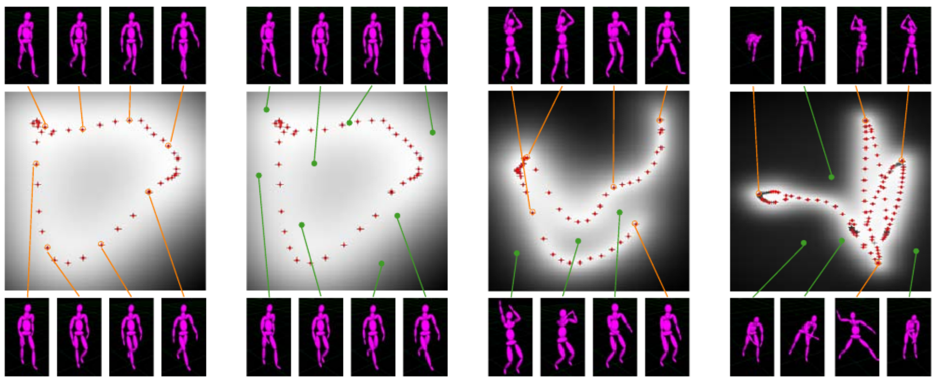


Figure 7.4: Low-dimensional latent variables extracted using GPLVM on various animations [GMHP04].

[GMHP04] cast the IK problem as the optimization of the likelihood of a Probability Density Function (PDF) over poses, knowing constraints. They propose to learn this PDF using Gaussian Process Latent Variable Model, a generalization of PCA and RBF model that is computed on motion features (angles, velocities ...). Then, the corresponding pose likelihood function is optimized under geometric constraints (*cf.* figure 7.4).

All these methods exploit ever more complex non-linear statistical techniques to represent a statistical distribution over poses, possibly using regression against a feature space or directly optimizing the likelihood in which case the computational cost is high. However, simpler models have also shown good results even in the context of performance capture.

Motion Models

[CH05] retrieve a set of frames satisfying on-line marker positions from a motion database, then build local predictive models around these data to reconstruct full-body poses in real-time, using only a few marker position. However, the local models are only valid in a relatively small neighborhood and the whole system requires a motion database.

[UFF06] showed that while complex non-linear statistic *pose* models have been used in the past, the use of simpler PCA over aligned motions provides efficient *motion* models, to be used in a motion tracking context. The authors highlight the fact that such models yield a smooth parametrization, allowing for standard optimization techniques. However, the authors indicate that a high number of examples is required, due to the misfit of linear probabilistic models to inherently non-linear orientation data.

[LWS02] use Linear Dynamic Systems (LDSs) to learn dynamic models for motion segments, together with transition probabilities and key poses (forming so-called *textons*). A two-level motion synthesis is then performed, by first choosing a texton path for the task, then using textons to generate motion between key poses. Here again, linear motion models are only valid locally (*cf.* figure 7.5).

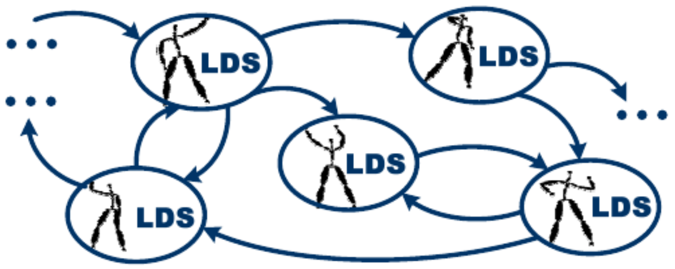


Figure 7.5: Two-level motion synthesis proposed by [LWS02]. LDS describe local dynamic behaviors in a graph, which is traversed to generate motion between key poses.

In the same spirit, [LM06] detect motion segments in which marker positions are linearly dependent using clustered PCA, then use these local linear models to compression motion data. Using several linear models requires transitions between models to be done in a smooth way. Moreover, linear models for marker positions inherently produce limb elongations.

Space-time

The constrained continuous optimization techniques have been extended to exploit information on existing motions. [SHP04] used PCA on several similar motion to build a low-dimensional linear subspace approximating the pose manifold of the motion. This subspace is then used to regularize a space-time optimization by biasing the results towards this subspace. However, the computational benefit of dimension reduction is not fully exploited, since it is only enforced as a penalty term.

[LHP05] propose to learn physical parameters of a virtual human by using Non-linear Inverse Optimization. A reference motion capture sequence is assumed optimal for energy consumption under geometric constraints, allowing to optimize physical parameters giving rise to this solution. These parameters are then used to synthesize new motion again using space-time constraints.

[CH07] use PCA to reduce the number of degrees of freedom in a motion database, then learn a statistical dynamic model on motion data. Motion is then synthesized by computing the maximum likelihood of a motion under kinematic constraints and the motion prior. DOF reduction speeds up the learning of the dynamic model. This work reformulates space-time constraints in a probabilistic framework, with the difference that the dynamic model is inferred from existing data. However, this system generates poor results when the set of constraints is sparse, and is computationally expensive.

7.2 Observations

From what precedes, a certain number of observations are possible on the following topics. Generally, the above classes of methods can be described as follows:

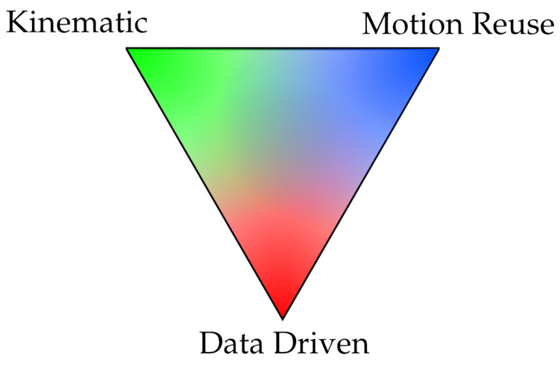


Figure 7.6: 3 classes of character animation techniques.

- Kinematic methods offer a high degree of control, at the expense of expressiveness.
- Motion reuse offers very limited control, but produces high quality at a low computational cost.
- Data-driven techniques provide an interesting trade-off, but tend to be computationally expensive during both pre-processing and on-line phases.

Apart from these general comments suggesting that an expressive, controllable and computationally inexpensive character animation technique is obviously desirable, more specific issues can be raised.

Inverse Kinematics

Several works are targeted at learning a mapping relating end-effectors to skeleton pose, while retaining features from existing motion. For sufficiently short parts of the motion, this mapping can be linearly approximated with good results with respect to the learning set. However, more global approaches usually necessitate costly optimizations to both construct and to use the regression model, due to the highly non-linear nature of the underlying mapping.

Data-driven space-time methods tend to necessitate a large number of key-frames, which suggests that simpler IK between key-frames could generate satisfying results. However, style-based IK techniques are expensive and are not guaranteed to synthesize smooth animations.

Dimension Reduction Benefits

Most of these works use a dimension reduction technique like PCA to obtain a pose prior, but few of them transport subsequent calculations in the reduced space, which could yet significantly improve computation time. This could be due to the prior not being of sufficient quality to enforce it as a hard constraint, except locally around training data. On the other hand, advanced non-linear probabilistic dimension reduction are usually very costly and difficult to integrate in a non-probabilistic framework.

So it seems that simple linear methods are of too poor quality to be trusted globally, while complex non-linear methods are much slower and difficult to integrate with. Surprisingly, none of the above references are concerned with the special structure and topology of the configuration space of the articulated skeleton.

Non-linearity

While problems due to the use of Euler angles for animation have been known for years, several authors use linear models based on them to animate virtual characters. While this may not be a problem for one-dimensional joints like the knee, this can cause serious issues for 3-dimensional joints like the elbows or the hips which have a reachable space, increasing the risk of the infamous *gimbal* lock.

A pose reconstructed by linearly combining Euler angles can be surprisingly different from expectations, for the very same reason Euler angles can not be interpolated well. However, few researchers consider the use of reconstruction rules adapted to the special structure of rotations (*e.g.* by using SLERP [Sho85]).

To summarize, using certain rotation parametrization may impose the need for complex non-linear machine learning algorithm to perform satisfying dimension reduction. We argue that taking the special geometry of the rotation group into account during the statistical analysis can improve both the quality and the simplicity of the algorithm. Let us now review existing works in the field of smooth manifold statistics.

7.3 Manifold Statistical Analysis

As mentioned in the previous section, the configuration space of an articulated character, parametrized by its joint orientations, is not a vector space. We also saw in 5.3.6 that the pose manifold $SO(3)^n$ possesses a natural Riemannian structure, for which generalizations of Euclidean algorithms have been proposed, including multi-resolution analysis, interpolation, and statistics. We review the latter in this section.

The lack of vector-space structure prevents the use of linear statistical quantities such as the empirical mean of samples, traditionally obtained by linear combination. However, the metric or pseudo-metric induced by a Riemannian metric allows to generalize relevant geometric properties of the mean. [BF01] propose an algorithm to compute spherical means as a minimization process. [Moa02] generalize these ideas to the rotation group. [PFA06, Pen06] give a general form for *intrinsic*, or Fréchet, mean computation using the Riemannian exponential. The intrinsic mean $\mu \in M$ is defined only in terms of the Riemannian metric and as such, it does not depend on the choice of embedding. It minimizes the sum of square Riemannian distances to all m samples $(x_i)_{i \leq m}$:

$$\mu = \operatorname*{argmin}_{y \in M} \sum_i d(y, x_i)^2$$

where $d(x, y)$ is the Riemannian distance from x to y. It can be computed using a simple gradient descent algorithm [Pen06]:

$$\mu_{k+1} = \exp_{\mu_k} \left(\sum_{i=1}^{m} \log_{\mu_k}(x_i) \right)$$

In contrast, the *extrinsic* mean would minimize the sum of distances in the sense of a surrounding space, such as matrix norm. The definition of the intrinsic mean requires the manifold to be connected and geodesically complete [3], a requirement met for $SO(3)$. [Pen06] also generalize variance and covariance definitions by developing the manifold over the tangent space at the Fréchet mean of the data, and give a definition of a Gaussian distribution. These non-linear tools are used for the statistical study of diffusion tensors generated by Magnetic Resonance Imaging (MRI), through a Riemannian structure.

[FLJ03, FLPJ04] proposed an extension of PCA to the case of Riemannian manifolds, first in the case of Lie groups having a compatible Riemannian structure, then on symmetric Riemannian manifolds. Known as Principal Geodesic Analysis (PGA), it generalizes the familiar PCA by exploiting the Riemannian structure: vector lines are replaced by *geodesics* (see 5.3.6) and the Riemannian metric is used to compute (and maximize) geodesically projected data variance on principal geodesics. The authors propose a first-order approximation that reduces to standard PCA over linearized data using the exponential map at the intrinsic mean. The projection over a geodesic is generally obtained by a minimization algorithm, though closed form exist for $SO(3)$ [SCLS07]. The differences between linearized and exact PGA are discussed in detail in [SLN10, SLHN10].

⋆ ⋆ ⋆

We have found the PGA algorithm to be a natural, simple dimension reduction technique, suitable for analyzing rotational pose data. We will therefore describe it in more details in the next chapter, and show how it meets most of the requirements we expressed in 3.3.

[3] *i.e.* in which geodesic curves can be extended indefinitely, having the important consequence that there exist a length-minimizing geodesic between any two points [FLPJ04]

Chapter 8

Principal Geodesic Pose Model

This chapter reviews the PGA algorithm in more details, and how it can be applied to motion capture data for dimension reduction. We derive a smooth PGA-based kinematic pose model, and the associated forward-kinematics. An IK algorithm is presented, and an application to motion compression is proposed for evaluation.

8.1 Principal Geodesic Analysis

The PGA algorithm extends the classical PCA algorithm by only considering the *metric* structure of a space. Such structure is available on geodesically complete Riemannian manifolds such as $SO(3)$. After describing the algorithm itself, we develop its practical implementation and approximations.

8.1.1 Algorithm

We begin with a quick review of PGA algorithm as described in [FLPJ04, FLRS06], starting with the linear (PCA) case.

Principal Component Analysis

The PCA seeks to find nested linear subspaces that maximize the projected variance of m samples $x_j \in \mathbb{R}^n$ or equivalently, that minimize sample projection errors. Assuming zero mean for observations, the i^{th} principal component v_i is defined by:

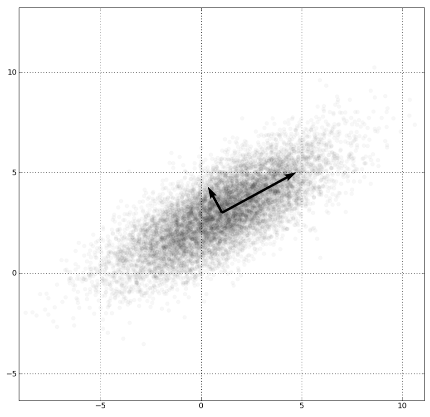
$$v_i = \underset{||v||=1}{\operatorname{argmin}} \sum_j ||x_j - \pi_i(x_j)||^2$$

where π_j is the orthogonal projection over the $i \in \mathbb{N}$ first principal components, $i-1$ of which have already been computed:

$$\pi_i(x) = \sum_{k=1}^{k=i} \langle x, v_k \rangle \, v_k = \pi_{i-1}(x) + \langle x, v_i \rangle \, v_i$$
$$\pi_0(x) = 0$$

This definition implies that the v_i form an orthogonal basis of principal subspaces, the last of which being all of $\mathbb{R}^n$. In practice, the eigen decomposition of the covariance matrix $X^T X$, where X is a $m \times n$ matrix whose rows are the m samples x_i^T, gives principal components as the eigenvectors sorted by decreasing eigenvalues. The corresponding eigenvalues (squared) indicate the projected variance on each component.

PCA is commonly used as a dimension reduction technique, as it allows to only keep the first principal components (in the sense of decreasing corresponding eigenvalues) to obtain a good linear approximation of the data set.



(CC BY-SA Ben FrantzDale)

Figure 8.1: The PCA of a multivariate Gaussian distribution of samples. The two orthogonal eigenvectors have been scaled by their corresponding eigenvalue.

Geodesics, Lie Groups

The above definition relies on one primitive: the projection over a principal component. The idea behind PGA is to extend this primitive to Riemannian manifolds, by the use of geodesics instead of vector lines, and the geodesic distance instead of the Euclidean one.

As seen briefly in 5.3.6, geodesics in a Riemannian manifold are given by the (Riemannian) exponential map. Let $\gamma_{a,v}(\alpha) = \exp_a(\alpha\, v) \in M$ be the geodesic curve starting at $a \in M$ with initial tangent vector $v \in T_a M$. The projection of $x \in M$ on the geodesic $\gamma_{a,v}$ is obtained by the following non-linear minimization problem:

$$\pi_{a,v}(x) = \underset{y=\exp_a(\alpha v)}{\operatorname{argmin}}\ d(x,y)^2 \tag{8.2}$$

where $d(x,y)$ is the geodesic distance given by $d(x,y) = ||\log_x(y)||_x$. This projection operation is illustrated on figure 8.2.

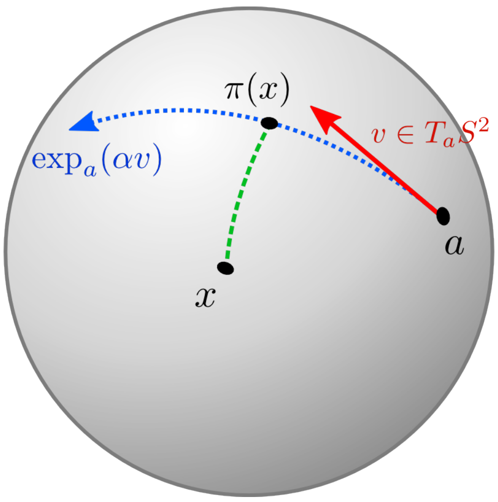


Figure 8.2: The geodesic projection of point x over the geodesic $\exp_a(\alpha v)$ *(blue)* corresponds to the point where the geodesic segment in green has minimal length.

In the case of a Lie group with a bi-invariant Riemannian metric such as $SO(3)$, the metric and group exponential coincide. Given such a Lie group G, the geodesic starting at $a \in G$ with initial (body) tangent vector $v \in \mathfrak{g}$ is given by $\gamma_{a,v} = a.\exp(\alpha\, v)$, for $\alpha \in \mathbb{R}$. The geodesic distance between two points a and b of G is given by:

$$\begin{aligned} d(a,b) &= ||\log_a(b)|| = ||\log_b(a)|| \qquad \textit{(Riemannian)} \\ &= \left|\left|\log(a^{-1}b)\right|\right| = \left|\left|\log(b^{-1}a)\right|\right| \qquad \textit{(Lie)} \end{aligned}$$

In terms of Lie group operations, the projection of $x \in G$ on a geodesic starting at $a \in G$ with body tangent vector $v \in \mathfrak{g}$ is thus:

$$\pi_{a,v}(x) = a.\exp(\alpha^\star v) \quad \text{with } \alpha^\star = \underset{\alpha \in \mathbb{R}}{\operatorname{argmin}} \left|\left|\log\left(x^{-1}.a.\exp(\alpha v)\right)\right|\right| \tag{8.3}$$

Intrinsic Mean

As in the PCA case, the first step to compute the PGA decomposition is to somehow *center* data around their mean. In the absence of a vector space structure, one can still define the mean from a metric point of view as [Pen06]:

Definition 14 (Fréchet mean)**.** *The* Fréchet*, or* intrinsic *mean of m points x_i of a Riemannian manifold M is defined as:*

$$\mu^\star = \underset{\mu \in M}{\operatorname{argmin}} \sum_{i=1}^{m} d(\mu, x_i)^2$$

This requires the points to be sufficiently localized to guarantee the uniqueness of the solution [Pen06].

The intrinsic mean can be computed efficiently using the following fixed-point iteration [Pen06]:

$$\mu_{k+1} = \exp_{\mu_k}\left(\frac{1}{m}\sum_{i=1}^{m} \log_{\mu_k}(x_i)\right)$$

Intuitively, this algorithm uses the exponential map to linearize the samples around the current mean candidate μ_k, performs a Euclidean mean in the tangent space, then advances to the corresponding point of the manifold μ_{k+1} until the tangent mean becomes close enough to zero. In the case of a Lie group with a bi-invariant metric, this gives:

$$\mu_{k+1} = \mu_k \exp\left(\frac{1}{m}\sum_{i=1}^{m} \log\left({\mu_k}^{-1} x_i\right)\right)$$

Principal Geodesics

Once the intrinsic mean of the data has been computed, the PGA seeks to compute a set of $k \in \mathbb{N}$ geodesics that best represent the data, using the same metric criterion as in the linear case. Namely, the first geodesic direction is given by the solution of the following minimization problem:

$$v_1 = \underset{||v||=1}{\operatorname{argmin}} \sum_j d\left(x_j, \pi_{\mu,v}\left(x_j\right)\right)^2 \tag{8.4}$$

To compute the subsequent geodesic directions, several approaches have been proposed, depending on the geometry of the manifold. For the general Riemannian case, [FLPJ04, SLN10] propose to incrementally compute an orthonormal basis v_k of $T_\mu M$ spanning k linear subspaces $V_k = \mathrm{span}\,(v_1, \dots, v_k)$, which correspond to k *geodesic* submanifolds $H_k = \exp_\mu(V_k)$. The projection of a point on this geodesic submanifold is given by:

$$\pi_{\mu,V}(x) = \underset{y \in \exp_\mu(V)}{\mathrm{argmin}}\ d(y, x) \tag{8.5}$$

Having defined the projection over a geodesic submanifold, the k^{th} geodesic direction is computed by maximizing projected variance:

$$v_k = \underset{||v||=1, v \in V_{k-1}^{\perp}}{\mathrm{argmax}} \sum_j d\left(\mu, \pi_{\mu,V_k}(x_j)\right)^2 \tag{8.6}$$

As pointed out by [SLN10], this is no longer equivalent to minimizing the square reconstruction errors, as it was in the vector case. Thus the following scheme will produce different geodesic directions:

$$v_k = \underset{||v||=1, v \in V_{k-1}^{\perp}}{\mathrm{argmax}} \sum_j d(x_j, \pi_{\mu,V_k}(x_j))^2 \tag{8.7}$$

The differences between the two is illustrated on figure 8.3.

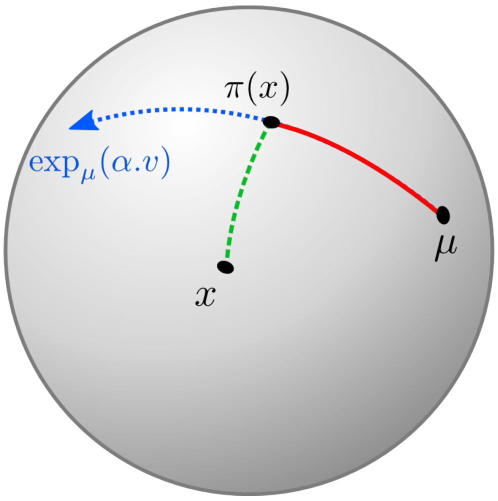


Figure 8.3: In a curved space, maximizing the projected variance *(red)* over a geodesic *(blue)* is not equivalent to minimizing the reconstruction error *(green)* for multiple points.

However, the authors found this latter scheme to be more stable than the former and are currently investigating differences between them. In the case of a Lie group with bi-invariant metric, [FLJ03] proposed a different construction scheme, taking the simpler form of the exponential into account:

$$\begin{aligned} x_j^{(0)} &= \mu^{-1} x_j && \text{(8.8a)} \\ v_k &= \underset{||v||=1}{\operatorname{argmin}} \sum_j d\left(x_j^{(k-1)}, \pi_v\left(x_j^{(k-1)}\right)\right)^2 \\ x_j^k &= \pi_{v_k}\left(x_j^{(k-1)}\right)^{-1} x_k^{(k-1)} \end{aligned}$$

The first step centers all samples around the intrinsic mean, so that the base point is now the group identity. Then, the first geodesic direction is computed by minimizing square reconstruction errors. Finally, the projections over the last geodesic are removed from the samples by left multiplication. Step 2 and 3 are then iterated until the k geodesic directions have been computed. Letting $p_k = \pi_{v_k} x^{(k)}$, we see that this scheme decomposes the samples in the following way:

$$x = \mu p_1 \dots p_k x^{(k+1)}$$

That is, the samples are reconstructed by iteratively composing geodesic paths, one starting at the end of the previous in a different geodesic direction. In contrast, the previous schemes approximate a sample as one geodesic starting at μ, resulting from a linear combination of k orthogonal directions of $T_\mu M$. These different schemes generate different parametrizations of the data manifold, classically known as the canonical coordinates of *first* and *second* kind.

Definition 15 (Canonical coordinates)**.** *Given a Lie group G of dimension $n \in \mathbb{N}$, a basis (x_i) of its Lie algebra $\mathfrak{g}$, and an element $g \in G$ in the neighborhood of the identity $e \in G$, one can define the two following coordinates:*

The Canonical Coordinates of the First Kind (ccfk) of $g \in G$ are n scalar $(\lambda_i)_i$ such as:

$$g = \exp \sum_{i=1}^{n} \lambda_i x_i$$

The Canonical Coordinates of the Second Kind (ccsk) (ccsk) *of $g \in G$ are n scalar $(\lambda_i)_i$ such as:*

$$g = \prod_{i=1}^{n} \exp(\lambda_i x_i)$$

These two coordinate systems are diffeomorphism in the neighborhood of the origin.

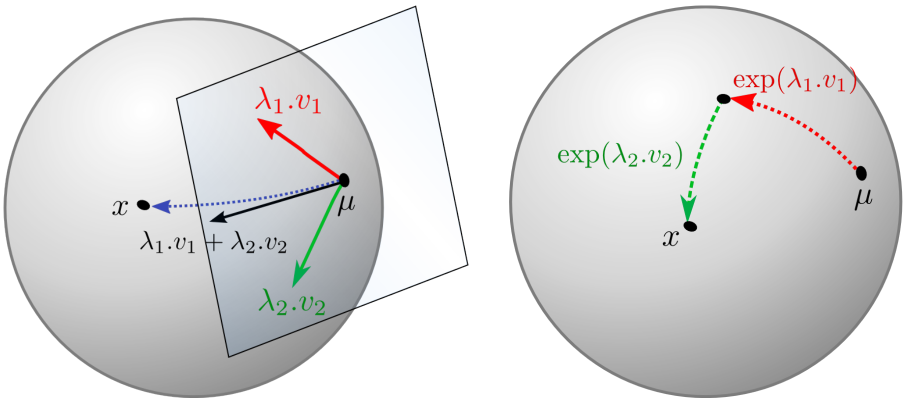


Figure 8.4: Illustration of the differences between the canonical coordinates of the first *(left)* and second *(right)* kind. In the case of a non-commutative group, the order matters for ccsk.

The geometric difference between the two coordinates is illustrated on figure 8.4. Interestingly, the differential of the exponential is not needed when differentiating the ccsk[1], contrary to the ccfk [MSZ94]. We will develop these coordinate systems and their derivative in more details in the next section.

Coordinate-Invariance It should be noted that one of the most appealing features of the PGA is that its results only depend on the Riemannian structure of the sample space, and not on a given coordinate system. To illustrate this fact, let us consider the statistical analysis of a set of rotations, parametrized by their Euler angles. Depending on the conventions chosen for the Euler angle sequence, and possible offsets in the parametrization (*e.g.* to better reflect a rest pose), a PCA on such data will produce different results for each coordinate choice. On the contrary, the PGA will *first* reconstruct the corresponding rotations, *then* perform statistics in a coordinate-agnostic way, therefore resulting in a coordinate-invariant analysis.

As an added benefit, the resulting modes for the PGA correspond to geodesics of the manifold, whereas a PCA on Euler angles will produce modes that suffer from the same drawbacks as the linear interpolation of Euler angles (*e.g.* possible gimbal lock, complex interpolated path, ...).

8.1.2 Exact vs. Linearized PGA

The PGA algorithm seems promising in theory. In practice however, one has to compute several nested optimization problems, and the convexity of the objective function is not guaranteed. [SCLS07] proposed a closed-form

[1] Each term of the product is a one-parameter subgroup: $d^b \exp(\alpha\, v).d\alpha = v\; d\alpha$

formula for the projection operator in the case of $SO(3)$ using unit quaternions, but in the general case no such formula exist. To remedy this situation, [FLJ03, FLPJ04] proposed an approximation of the projection operator, using a linearization of data in Lie algebra by the exponential map:

$$d(x, \exp(\alpha\, v)) \approx ||\log(x) - \alpha\, v||$$

Under this approximation, the *approximate* projection $\widetilde{\pi_v}$ on a geodesic starting at the identity is simply given by the *Euclidean* projection of log-linearized samples in the Lie algebra, mapped back to the group using the exponential. Assuming $||v|| = 1$, the approximate projection is thus:

$$\widetilde{\pi_v}(x) = \exp\left(\langle \log(x), v\rangle\; v\right) \tag{8.9}$$

This implies that the first geodesic direction is the same as the first principal component for log-linearized data around the mean μ, that is the eigen vector associated with the largest eigenvalue of the covariance matrix. If we choose the first reconstruction scheme for geodesics, as in equation (8.5), then the whole computation effectively reduces to standard PCA over the log-linearized data at the mean. If we choose instead the second reconstruction scheme, as in equation (8.8), the subsequent iterations still apply, but the projection step is replaced by the approximate, tangent-Euclidean, projection.

In practice, we found that motion data are usually sufficiently localized around their mean for the two schemes to produce quite similar *modes*, but the approximated algorithm is much faster to compute.

As for the reconstruction, using the canonical coordinates of the first and second kind produce nearly identical results near the coordinates origin, but start to differentiate the further the coordinates are from the origin, due to the non-commutativity of the group. As described in 8.2.2, an interesting aspect of the ccsk are that the exponential differential is not needed when differentiating the coordinates.

8.1.3 Summary

Let us now recapitulate the most salient features of the PGA:

- It extends PCA to Riemannian manifolds, using geodesics instead of vector lines.
- The use of the Riemannian structure results in a *coordinate-invariant* analysis.
- It generates a *smooth* parametrization of geodesic submanifolds in terms of the canonical coordinates.
- Several different algorithms have been given in the literature [FLJ03, FLPJ04, SLN10].

- In its most simple, approximate form, the PGA reduces to a PCA in the tangent space at the intrinsic mean of the data.

8.2 Pose Model

In this section, we describe our use of the PGA algorithm to build a smooth skeleton pose parametrization, learned on existing motion capture data. We present examples of intrinsic mean and principal poses for real motion data. Then, we propose the corresponding forward kinematics mapping, together with an efficient computation of the associated pose Jacobian. Finally, we present results of real-time, data-driven, full-body Inverse Kinematics using this model.

8.2.1 Pose Parametrization

We begin with an existing motion capture sequence represented as a set of m skeleton configurations $(g_i)_{i \leq m} \in G$ sampled over time. Each configuration is a pair $(c, p) \in G = SE(3) \times SO(3)^n$, where $n \in \mathbb{N}$ is the number of joints, c is the rigid transformation describing the configuration of the root bone, and p contains the relative orientation of each joint with respect to its parent.

We will denote the **pose** space of relative joint orientations by $P = SO(3)^n$. As a compact Lie group, P has a bi-invariant Riemannian metric as described previously. We then apply the PGA on the poses $(p_i)_{i \leq m}$ to obtain the intrinsic mean $\mu \in P$, and a set of $k \in \mathbb{N}$ geodesic directions $v_j \in \mathfrak{p}$ describing the poses. With this decomposition, the new pose DOFs are k scalars $(\lambda_j)_{j \leq k} \in \mathbb{R}$ describing coordinates over the k principal geodesics.

We define the *reduced* pose parametrization as a smooth function $f : \mathbb{R}^k \to P$ that reconstructs a pose given k geodesic coordinates. As seen previously, two canonical coordinate can be chosen:

$$f_1(x) = \mu \ \exp\left(\sum_{j=1}^{k} \lambda_j . v_j\right) \qquad \textit{(first kind)}$$

$$f_2(x) = \mu \ \prod_{j=1}^{k} \exp(\lambda_j . v_j) \qquad \textit{(second kind)}$$

We can remark that when $(\lambda_j)_j = (\delta_{i,j})_j$, that is when only the i$^{\text{th}}$ coordinate is non-zero, the two parametrizations give the same pose. A pose obtained along only one geodesic will be called a *principal pose* for the motion data. Figure 8.5 shows examples of principal poses and intrinsic mean for motion capture data.

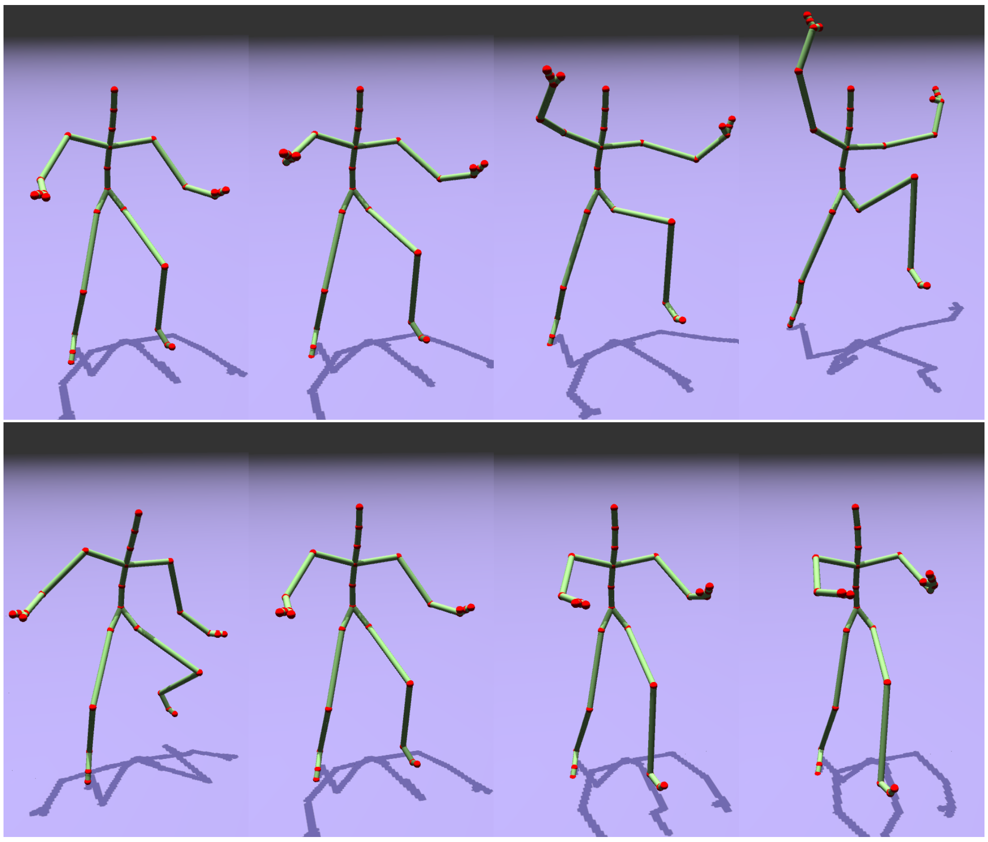


Figure 8.5: Examples of intrinsic mean *(top-left)* and the two first principal geodesics for motion capture data *(top, bottom)*. Notice how full-body joint correlation are captured.

While the principal poses are the same for both parametrizations, they are combined in a different way to produce a final pose:

- f_1 composes principal pose directions in a linear way, *then* reconstructs a pose with $\exp$
- f_2 reconstructs principal poses with $\exp$, *then* composes them using group structure

Choosing Reduced Dimension

The non-linear nature of PGA has counter-intuitive consequences: [SCLS07] showed that when the exact principal geodesic analysis is used for $SO(3)$, the number of principal directions needed to reconstruct a set of samples is not *a priori*bounded by the dimension of the space. [SLN10] require the geodesic directions to be all orthogonal to each other, thus removing the problem.

In the linear case, the cumulative projected variance over principal components is classically used to indicate the percentage of total data variance accounted by the reduced model. Letting $\sigma_j^2 \in \mathbb{R}$ be the eigenvalues of the sample covariance matrix (*i.e.* the projected variance on the corresponding principal components), the cumulative variance over the first j principal components η_j is given by:

$$\eta_j = \frac{\sum_{l=1}^{j} \sigma_l^2}{\sum_{l=1}^{n} \sigma_l^2} \quad \in [0,1] \tag{8.11}$$

It is the sum of j first projected variances, normalized by the total variance of the data. The cumulative variance gives a simple criterion for choosing the dimension $k^\star \in \mathbb{N}$ of the reduced dimension space, given a target variance percentage $\epsilon \in [0,1]$:

$$k^\star = \inf_{\eta_k > \epsilon} k$$

It is simply the smallest integer for which the cumulative variance is greater than the given threshold ϵ. A common heuristic is to choose $\epsilon = 0.9$ [Fuk90]. For the non-linear case, the projected variance is given [FLJ03] by the sum of squared geodesic distances of projected samples to the mean:

$$\sigma_j^2 = \frac{1}{m} \sum_{i=1}^{m} || \log \pi_{v_j}(\mu^{-1} x_i) ||^2$$

A similar cumulative variance quantity can then be computed from it. Of course, if the approximated PGA is performed as in our examples, the (tangent) linear criterion (8.11) is used since the projection approximation is linear. In practice, it is common to require that 99% of the variance should be represented by the reduced model. Figure 8.6 shows cumulative variance graphs and the dimension of the reduced space for different motion capture sequences. Depending on the variability found in the data set, we can observe that the dimension reduction can be sometimes significant, as for example highly coordinated motions such as walking.

As shown on figure 8.7, the number of principal geodesics needed to represent 99% of the input data variance is generally inferior to 20. For motions with stronger correlations, such as walking motions, 10 geodesics are in most cases enough to express 95% of the input variance.

8.2.2 Kinematics

We now describe the forward kinematics mapping for the reduced pose model. It is obtained by composing the standard articulated forward kinematics, presented in section 6.2, with the reduced pose parametrization. The differential

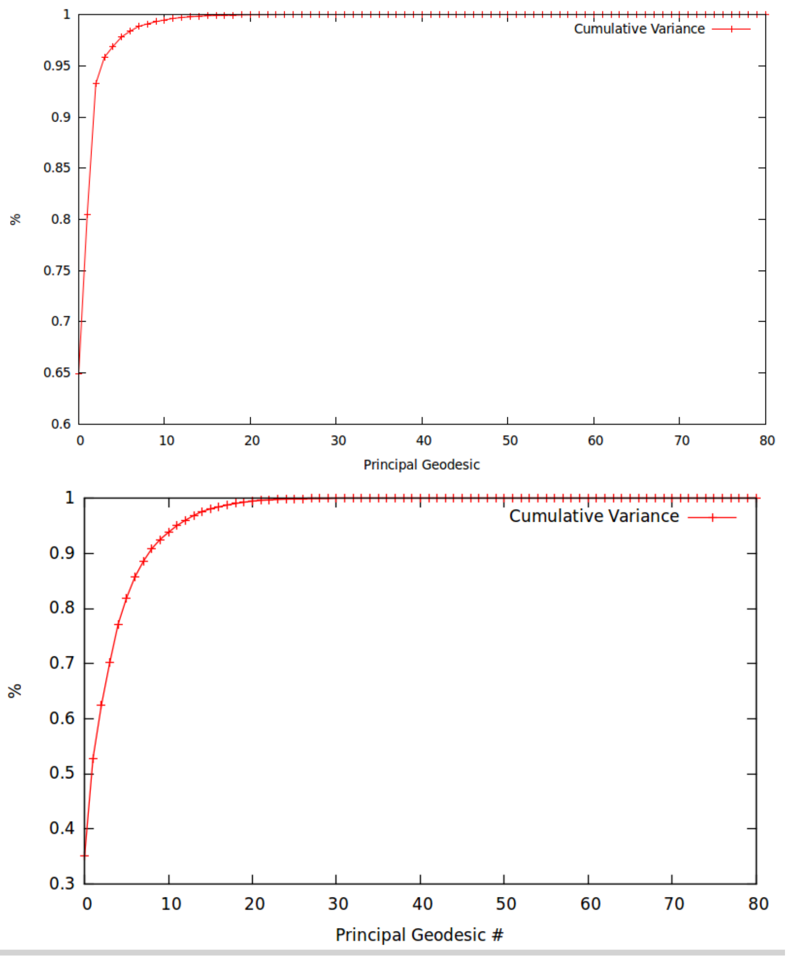
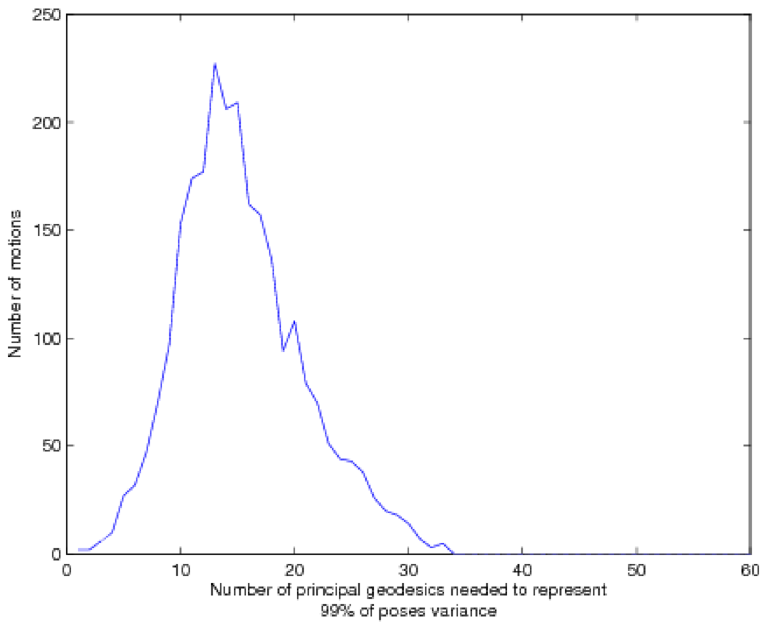


Figure 8.6: Cumulative variance diagrams for two motion capture sequences: short walk *(top, CMU:02-01)*, and long breakdance *(bottom, CMU:85-01)*.

of this mapping is derived, both for the first and second canonical coordinates, as well as the efficient computation of its Jacobian matrix. Then, we present full-body IK results obtained using our model and the classical IK algorithms presented in section 6.3. We conclude with a discussion of our results.

Forward Kinematics

So far, we obtained a reduced pose model as a smooth function $f : \mathbb{R}^k \to P$. In order to obtain the full forward kinematics, we compose this function with the standard forward kinematics mapping for an articulated body with configuration space $G = SE(3) \times SO(3)^n$, described in 6.2:



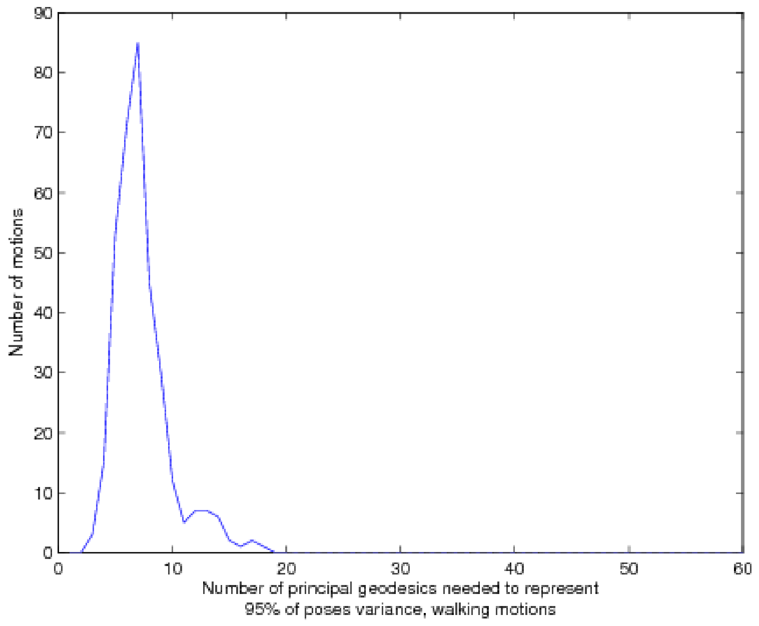


Figure 8.7: Histograms showing the numbers of principal geodesics *sufficient* to represent a target variance, using approximate PGA. Top: 99% of variance for the whole CMU database. Bottom: 95% of variance for only walking motions.

$$h : G \to SE(3)^{n+1}$$

We form the reduced forward kinematics mapping by:

$$\begin{aligned} r : Q = SE(3) \times \mathbb{R}^k &\to SE(3)^{n+1} \\ (x, \alpha) &\mapsto h(x, f(\alpha)) \end{aligned}$$

The differential is given by the chain rule. With $dx \in T_x SE(3)$ and $d\alpha \in T_p\mathbb{R}^k = \mathbb{R}^k$, we have:

$$dr.(dx, d\alpha) = dh.(dx, df.d\alpha)$$

As this differential map is to be used intensively by pose optimization algorithms, we need to give details on the efficient computation of its differential and Jacobian matrix.

Differential

We have seen in 6.2 that $dh(g).dg$ can be computed in $O(n)$ operations. We now describe how to compute the full Jacobian matrix of r, also in $O(k.n)$ operations. This can be achieved by computing the Jacobian matrix of the ccfk and ccsk in $O(k.n)$, instead of $O(k^2.n)$ for the naive algorithm:

$$J_f = \left(\frac{\partial f}{\partial \alpha_i}\right)_{i \leq k}$$

Once J_f has been computed, we apply $dh(g)$ to the resulting k tangent vectors to obtain a total number of operations, for the full Jacobian matrix:

$$O(\underbrace{k.n}_{J_f} + \underbrace{k.n}_{k \times dh(g)} + \underbrace{6.n}_{\text{root dofs}}) = O(k.n)$$

Let us now describe how to compute the Jacobian of the coordinate maps efficiently.

First Kind Recall that $f_1(\alpha) = \exp\left(\sum_{i=0}^k \lambda_i.v_i\right)$. The differential is obtained by the chain rule:

$$df_1(\alpha)d\alpha = d\exp\left(\sum_{i=0}^k \lambda_i.v_i\right).\left(\sum_{i=0}^k v_i.d\lambda_i\right)$$

For a general tangent vector $d\lambda$, this expression can be computed in $O(k.n)$:

- Forming $\sum_{i=0}^k v_i.d\lambda_i$ requires $O(k.n)$ operations

- Forming $\sum_{i=0}^{k} \lambda_i.v_i$ also requires $O(k.n)$ operations
- Applying the final $d\exp$ is a $O(n)$ operation (term-wise $d\exp_{SO(3)}$)

The final cost for a single tangent vector $d\alpha$ is $O(k.n + k.n + n) = O(k.n)$ in the general case. The *full* Jacobian matrix can also be computed in $O(k.n)$ as follows:

- Form $\sum_{i=0}^{k} \lambda_i.v_i$ in $O(k.n)$ operations (only once)
- The Jacobian matrix of $\sum_{i=0}^{k} v_i.d\lambda_i$ is already given by $(v_i)_{i\leq k}$
- Apply $d\exp$ to the k tangent vectors $(v_i)_{i\leq k}$, in a total of $O(k.n)$ operations

The final cost for the full Jacobian is thus $O(k.n + k.n) = O(k.n)$ operations.

Second Kind Recall that $f_2(\lambda) = \prod_{i=0}^{k} \exp \lambda_i.v_i$. The body-fixed directional derivative is given by [MSZ94]:

$$d^b f_2(\lambda).d\lambda = \sum_{i=0}^{k} Ad_{p_i^{-1}}.v_i.d\lambda_i \tag{8.12}$$

where $p_i = \prod_{j=i}^{k} \exp(\lambda_i.v_i)$ is the reconstruction using only the last $k - i$ geodesics. The Jacobian $J_{f_2}(\lambda)$ can thus be computed using the following algorithm:

$$\begin{aligned} p_k &= \exp(\lambda_k.v_k) && O(n) \\ p_{i-1} &= \exp(\lambda_{i-1}.v_{i-1}).p_i && O(n) \\ \frac{\partial f_2}{\partial \lambda_i} &= Ad_{p_i^{-1}}.v_i && O(n) \end{aligned}$$

In total, $O(k.n + k.n) = O(k.n)$ operations are required for computing $J_{f_2}(\lambda)$. However, contrary to the *ccfk* case, this algorithm can no longer be parallelized due to the dependencies involved in the computation of p_i.

Inverse Kinematics

Once the forward kinematics Jacobian has been derived, it can be used to implement an IK algorithm as a non-linear least-squares problem, using the algorithms presented in 6.3. In our experiments, we used a Levenberg-Marquardt solver. Figure 8.8 and 8.9 show some of the resulting poses we obtained in a real-time application.

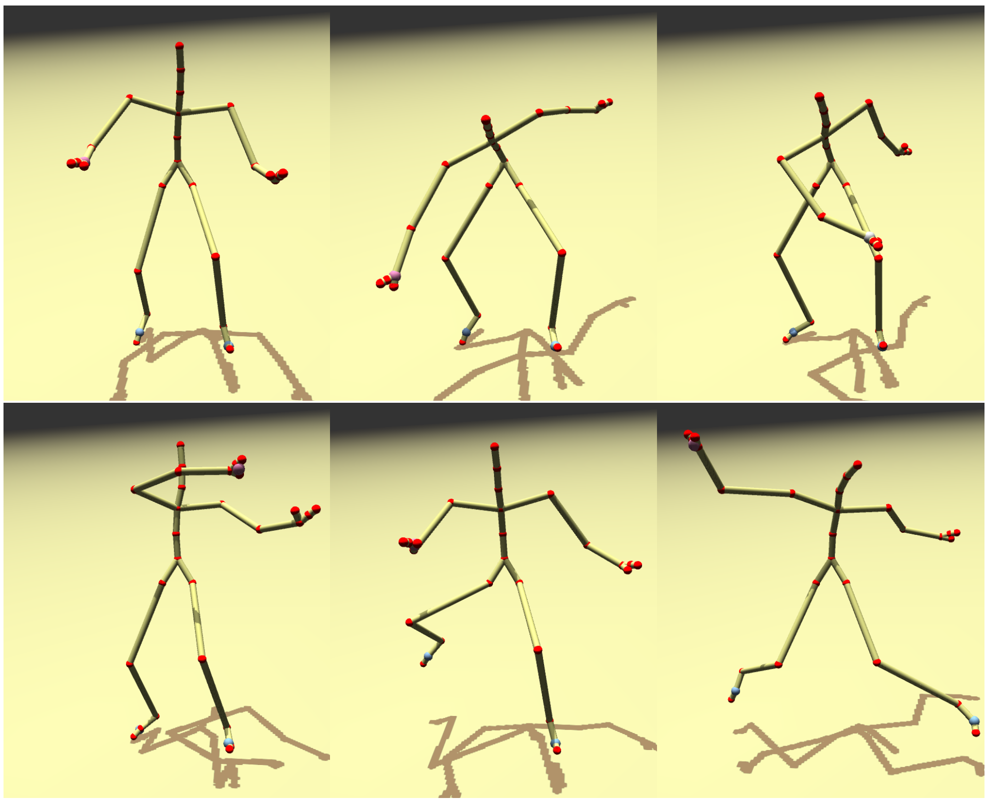


Figure 8.8: Our PGA-based IK solver used in real-time manipulation. The input motion is a *break dance* sequence from the CMU database. There are three IK handles in this example: one on each foot and one on the right hand. The optimization is done using 10 geodesics.

Anticipating on part IV, we mention that using a different metric in Gauss-Newton-like algorithms (such as Levenberg-Marquardt) can produce interesting results. The linear least squares solved at each step by the *damped* Gauss-Newton algorithm is (see 6.3):

$$\Delta x^\star = \underset{\Delta x \in \mathfrak{q}}{\operatorname{argmin}} \underbrace{\left|\left|y^\star - \left(y(x) + d^b y(x).\Delta x\right)\right|\right|^2}_{\text{constraint error}} \quad + \quad \underbrace{\lambda \left|\left|\Delta x\right|\right|^2}_{\text{step damping}}$$

where $x \in Q$ are the DOFs and $y : Q \to Y$ are the optimized features, with desired value $y^\star \in Y$ for some abstract, Euclidean feature space Y. If the canonical metric on $\mathfrak{q}$ is used in the rightmost term, this means that every DOF receives the same amount of damping. In particular, root DOFs

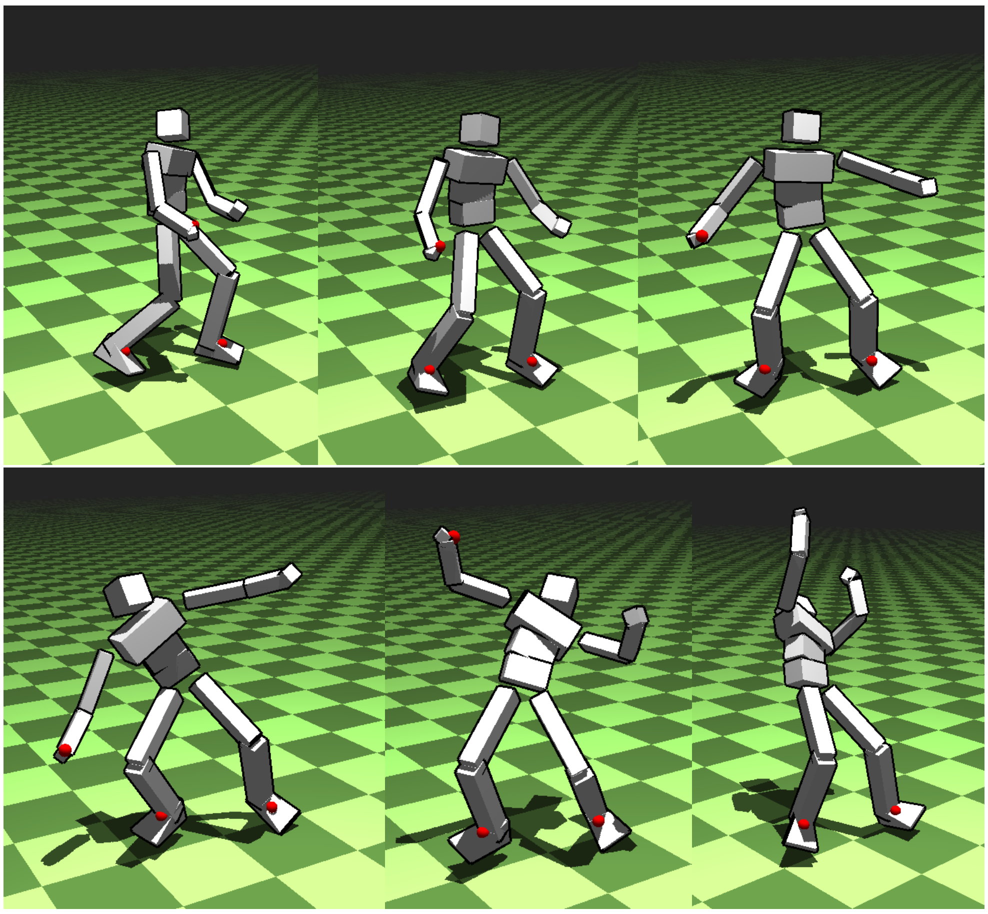


Figure 8.9: Another example of IK manipulation (10 geodesics, break-dance) using our system. Whole-body correlations are effectively captured by the model: the left arm automatically moves even when not constrained.

are damped exactly as any other joint DOFs, even though they induce much more significant overall motion.

If we replace this canonical metric by the *kinetic energy* metric, we instead favor low-energy displacements Δx, meaning the optimization will tend to displace the lower-mass character limbs *before* displacing the whole body. We have found this simple strategy to result in more stable behavior and visually more convincing poses, as illustrated on figure 8.10.

8.2.3 Discussion

Let us now discuss the pros and cons of the PGA-based IK solver.

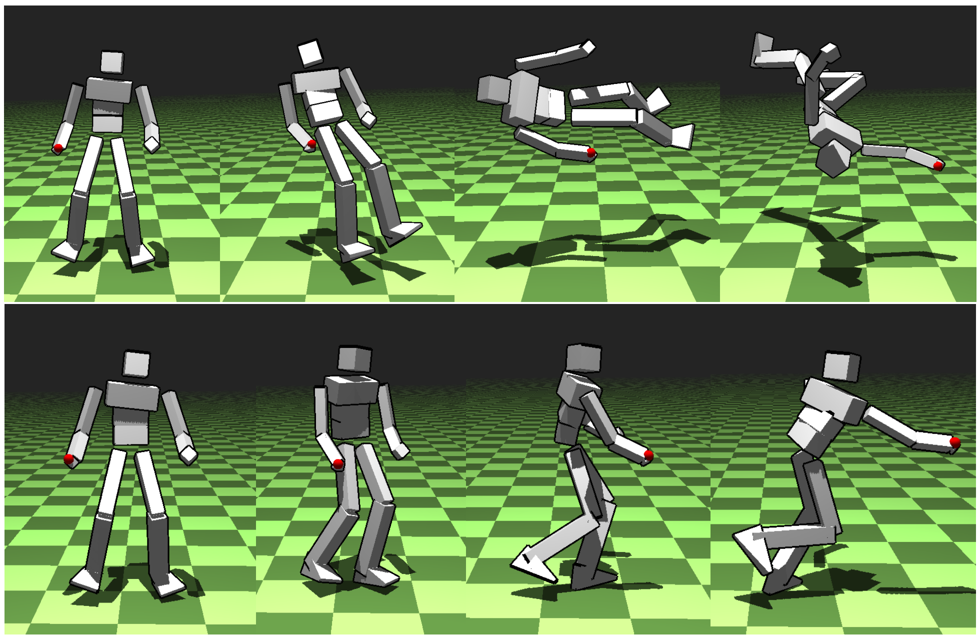


Figure 8.10: Comparison of the canonical *(top)* and kinetic *(bottom)* metrics in the optimization, for a single end-effector *(red)*. The undesired rotation around the root bone *(top)* is automatically corrected by the metric change.

Benefits

While the presented IK algorithm can not compete with closed-form solvers in terms of speed, it still exhibits several interesting features:

- Low-dimensional search space for the optimizer
- Automatic full-body correlations, keeping the synthesized poses close to the learning set (*cf.* figure 8.11)
- Efficient Jacobian computation
- Interesting effects by changing metric

On a modern machine (Quad Core, 64 bits), this algorithm easily reaches real-time performance (more than 200 FPS) since the most computationally intensive operation at each optimization step is the inversion of a $(6+k) \times (6+k)$ matrix, with usually $5 \leq k \leq 15$. In contrast, a complete articulated skeleton contains between 75 and 90 degrees of freedom, which results in much slower matrix inversions without the benefits on resulting poses.

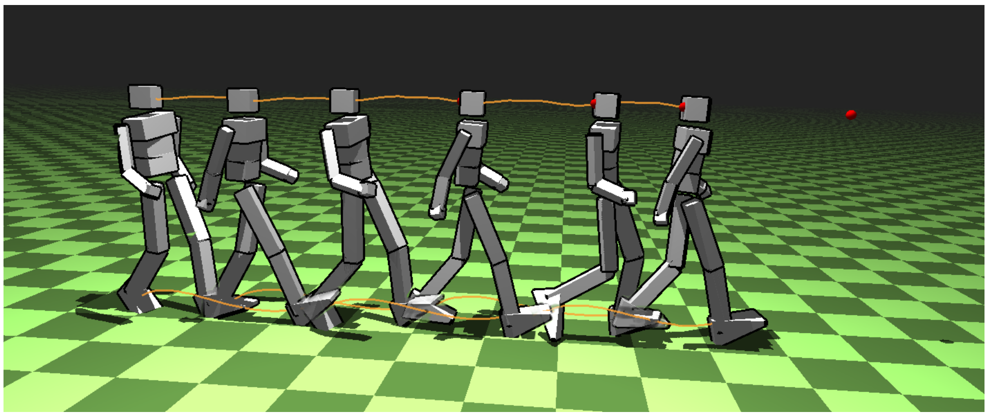


Figure 8.11: Tracking of motion capture end-effectors for a highly correlated motion, walking. Only 6 geodesics are used. 3 end-effectors constraints are present on both feet and head *(orange)*, automatically inducing arm-swing. An additional head orientation constraint *(looking at the rightmost red spot)* is present to keep the character orientation straight.

Potential Issues

Local Extrema Since the exponential map is periodic[2] on $SO(3)$, the proposed pose model is inherently subject to local extrema problems. However, for such a situation to occur when optimizing, some skeleton joint has to perform a complete turn. It is thus possible to eliminate this problem by enforcing joint limits during the optimization process (*cf.* chapter 12).

A softer way to prevent this issue is to add an energy term to the optimization based on the geodesic coordinates λ: $E(\lambda) = ||\lambda||^2$. In practice, for a Gauss-Newton-like solver, this amounts to adding a weighted equation $\lambda = 0$ to the least-squares problem. Such potential energy will act as a spring preventing the geodesic coordinates from going too far from the origin, which corresponds the mean pose. We have found this approach to be sufficient when solutions are close to the poses in the training set (as in the compression application presented in chapter 9), but not in general IK manipulation where angular limits are needed. Of course, a rest pose different from the mean pose can also be defined this way.

In the case of an ill-posed IK problem (less constraints than DOFs), this potential energy provides a simple and efficient regularization term, by dragging the solution towards the mean pose.

[2] $\exp\left((\alpha + 2\pi).v\right) = \exp(\alpha.v)$

Jacobian Singularities Since the Jacobian for standard articulated body is singular at some configurations[3], the reduced pose model necessarily exhibits these singularities. Near singular points, the Jacobian matrix exhibits very small, but non-zero singular values, corresponding to directions difficultly reachable due to structural constraints. If we write the Singular Value Decomposition (SVD) of the Jacobian matrix as $J = USV^T$, where U, V are orthogonal matrices and S is positive diagonal, the normal equations become:

$$\underbrace{VS^2V^T}_{K}.\Delta x = J^T.e(g)$$

This writing shows the instabilities caused by small eigenvalues in S, since the inverse matrix $K^{-1} = VS^{-2}V^T$ may exhibit very large eigenvalues. The Tikhonov regularization of the above system (damped-pseudo inverse, see 6.3) alleviates this problem, resulting in an inverse matrix of the form:

$$K^{-1} = V\big(I + S^2\big)^{-1}V^T$$

It is worth noting that in practice, enforcing angular joint limits can efficiently decrease instabilities, since most joints in the skeleton can not get past singularities (think of knees or elbows). Angular limits will be treated in more details in chapter 12. Still, in order to cope with this problem, we have found the Levenberg-Marquardt strategy of damping to be effective in order to prevent the character reduced state from reaching singular points.

[3] *e.g.* when the arm and the forearm become aligned

Chapter 9

Application to Motion Compression

We now propose an evaluation of the PGA-based IK algorithm using a novel motion compression algorithm. The compression problem provides an insightful way of measuring the expressive power of a motion model, as it both evaluates how compactly a pose manifold can be encoded, and how well it can reconstruct actual motions.

9.1 Previous Work

We begin this chapter with the previous works in the motion compression domain. We then develop existing motion metrics for evaluating motion quality. Finally, we review non-linear multi-resolution data analysis.

9.1.1 Motion Compression

Though recent works on motion capture data compression can be found, the problem of motion compression has been mainly focused on animated meshes compression so far: those high-dimensional data often present high spatial and temporal coherence that can be exploited to reduce the data size. [Len99] detects parts of the mesh with rigid motion to encode only the transformation and the residuals. Correlations that may exist in parts of the moving object have also been exploited through the use of PCA [SSK05] to compress the mesh vertices.

Skeleton motion also exhibits such cross-limbs, or *spatial* correlations. These are mainly exploited in optimization frameworks as they allow for search space dimension reduction. [SHP04] apply PCA on a group of similar motions in order to synthesize motion close to the learning space. Unfortunately, the computational benefits of dimension reduction are not exploited during the IK phase, contrary to our work. [GMHP04] use a probabilistic latent variable space to perform inverse kinematics that preserves stylistic properties. This method requires heavy computations, both during pre-processing and on-line. [LM06] detect motion segments in which joints positions lie in a reduced linear subspace and use PCA to reduce the dimensionality for compression purposes. However, transitions between models might result in artifacts unless special care is taken.

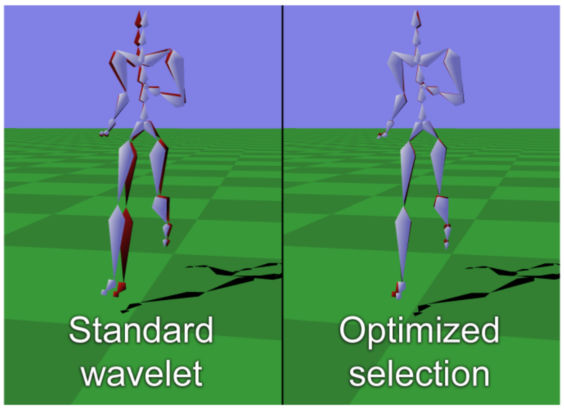


Figure 9.1: Automatic wavelet basis selection for motion capture compression, by [BPP07].

Motion capture data inherently possess temporal coherence which can be exploited to achieve compression: [LM06] use spline key-framing to compress the PCA projections of markers, per motion segments. [Ari06] uses splines to represent global markers trajectories. The control points for a whole motion database are then compressed using clustered PCA. In both cases, working with global marker positions requires an additional pass of optimization to keep the bone length constant across the synthesized motions. Other methods use rotational data: [BPP07] adapt standard wavelet compression on joint angles by automatically selecting the basis elements in a way that minimizes reconstruction error (*cf.* figure 9.1). However, high compression ratios can result in undesirable reconstructed paths due to the use of Euler angles. All these works perform an additional quantization pass to further improve compression ratios.

Any lossy motion capture compression method, being orientations-based or positions-based, introduces errors that are likely to result in various perceptual artifacts, such as *foot skating*, which greatly penalizes the visual quality of synthesized motions. This artifacts are usually corrected using IK techniques. However, doing so might alter the stylistic properties of initial motion, therefore a *style-based* IK algorithm [GMHP04] might be required. Unfortunately, these algorithms are computationally expensive and require significant amounts of data.

9.1.2 Non-Linear Signal Processing

As mentioned earlier, two natural ways of compressing motion data are to exploit both temporal and spatial coherence in the motion of parts of the skeleton. To achieve this, one typically uses multi-resolution and dimension reduction techniques. While well-known theoretical frameworks for these are available in the case of data lying in a linear space (such as wavelets, PCA), their extension to non-linear spaces (for instance, the space of rotations $SO(3)$ is not trivial and is a recent field of research. Unfortunately, to our knowledge, no motion compression method addressed the special geometry of the rotation group during data processing. Since we have already reviewed non-linear statistic tools in 7.3, let us now turn to multi-resolution analysis.

Early examples of structure-preserving computations in the rotation group include the well-known unit quaternion SLERP [Sho85] used for interpolating rotations, where interpolation is performed along great circles of the quaternion unit sphere S^3. [KKS95] proposes a generalization of Euclidean splines for unit quaternions, by reformulating classical algorithms in terms of Lie group operations.

[LCR+02] present a construction scheme for general time-domain filters, again respecting the unit quaternion sphere structure. [LS01] derive a *multi-resolution* scheme that allows editing, blending and stitching of motion clips. A potential application to motion compression is mentioned, though not developed. [RDS+05] generalize this scheme to symmetric Riemannian manifolds using exponential and logarithmic maps. This scheme can be seen as a special case of the *lifting scheme* [Swe98] algorithm, an alternate formulation of classical wavelets. We will give more details on this in 9.2.2

9.1.3 Motion Evaluation Metrics

As with any lossy compression system, a central problem with motion capture data compression is the *error metric* used to evaluate the quality of the results. The problem in this case is that the metric should take *perceptual* features into account. While it is commonly accepted that the standard L^2 norm over markers positions is a weak indicator of the perceptual similarity of two animations, few works proposed efficient, alternative metrics. [RP03] propose a study of user sensitivity to errors considering only ballistic motions. [RPE+05] try to evaluate the natural aspect of an animation. To do so, 3 classes of metrics are distinguished:

- Heuristic rules, that penalize the score of an animation when violated (*e.g.* physical laws)
- Perceptual metrics that highlight artifacts noticed by users (*e.g.* foot skating)

- Classifiers-based metrics trained on large data sets

The first two usually fail to quantify the natural aspect, or the style of an animation, but are good at detecting precise artifacts. The latter is based on the assumption that a human will perceive a motion as natural if it has already been seen a lot of times. On the contrary, an unusual motion will be perceived as unnatural. Such metrics often detect stylistic closeness successfully, but are highly dependent on the data set used for the training: they will fail for a natural motion that is not in the data set. Moreover, local physical anomalies or artifacts are often not detected. As a matter of fact, finding an accurate and robust metric for human motion perception remains, to the best of our knowledge, an open problem.

9.2 Proposed Method

9.2.1 Motivations - Overview

We now give an overview of our motion capture data compression method. Most approaches to human motion compression exploit global markers positions to achieve compression. While this has some advantages, such as speed and the use of well-known frameworks, the biggest drawback is that the constant bone-length of the skeleton cannot easily be guaranteed, which can introduce undesired limbs deformations. A post-processing pass is needed for this constraint to be enforced. Yet, this additional process can itself introduce artifacts. We want to address this problem by working on orientations rather than positions. However, because of the hierarchical nature of the skeleton, even slight errors in reconstructed orientations can lead to significant positions errors for end-joints. The most notable artifact of this kind is probably foot skating, which greatly penalizes the perceptual quality of synthesized animations.

We intend to work around this by building a PGA-based pose model from an animation clip: this model allows us to synthesize poses that match given end-joints constraints, while staying close to the input data, using the PGA-based IK algorithm presented in 8.2.2. All that remains to do is to compress the end-effector and root trajectories, using temporal coherence. Since both the root trajectory and orientation are to be compressed, we use the multiscale representation by [RDS+05] for simplicity, as it applies to both vector and rotational data.

Starting from an uncompressed motion capture composed of $m \in \mathbb{N}$ samples, we compute the PGA of the pose samples, keeping only the leading principal geodesics. The data associated with this reduced model are the pose intrinsic mean and the k leading principal geodesics, where k is user-selected or automatically adjusted based on reconstruction error.

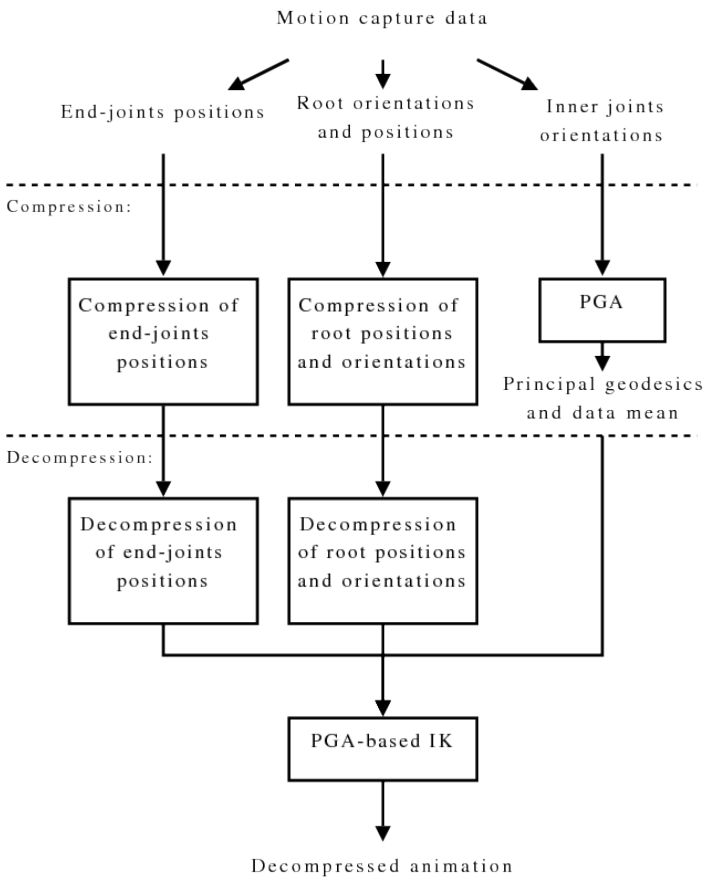

Motion capture data
End-joints positions
Root orientations and positions
Inner joints orientations
Compression:
Compression of end-joints positions
Compression of root positions and orientations
PGA
Principal geodesics and data mean
Decompression:
Decompression of end-joints positions
Decompression of root positions and orientations
PGA-based IK
Decompressed animation

Figure 9.2: Flow diagram for the compression pipeline

This model is then used with the IK algorithm presented in to synthesize poses that both match end-joints constraints *and* are close to the input data. Given this pose model, we only have to store the compressed end-joints trajectories as well as the root joint positions and orientations, also compressed, in order to recover the (approximate) motion using IK. The whole compression/decompression pipeline is presented on figure 9.2.

9.2.2 Multi-Resolution for Trajectory Compression

Let us now describe the multi-scale representation used for trajectory compression. As a particular case of the *lifting scheme* [Swe98], the multi-scale representation for manifold data introduced by [RDS+05] can be summed up as follows. In the general case, let D be the set of data we want to represent in multi-scale:

- D is first **partitioned** into two sets, A and B

- Elements of A are used to **predict** the data in B, using a prediction operator
- The **differences** between the *predictions* B_A and the actual B elements form the *details*

The process is then iterated over the set A until there are no data remaining. By doing so, one creates a collection of decreasing-size Levels of Details (LODs). This "pyramid" is the multi-scale representation of the original data. For this representation to be useful in compression, one generally wishes to partition the data so that the prediction step is as accurate as possible. In that case, the details needed to correct the prediction are small and can hence be omitted with few errors. For instance, in the case of data presenting temporal coherence, the partition can be achieved by a simple sub-sampling, and the prediction using a smooth interpolation.

Reconstruction Given a LOD pyramid, the initial data can be recovered by successive prediction/correction steps from the coarsest to the finest LODs, possibly omitting finest LODs in compression applications.

Rotations

In the case of time-dependent orientation data, D can be represented by a collection of rotations $q = (q_i)_{1 \leq i \leq m}$, where m is the number of samples. In order to exploit the temporal coherence, we simply sub-sample the data by a factor 2 to partition the data.

Instead of using the tangent spline interpolation described in [RDS+05] for the prediction step, we use the quaternion splines proposed by [KKS95]. We have found the latter to produce more consistent results since the interpolated path does dot depend on the point considered to perform the tangent interpolation, as it its the case in [RDS+05]. [KKS95] initially transform an Euclidean spline:

$$f(t) = \sum_{i=1}^{n} p_i . b_i(t)$$

where $b_i : \mathbb{R} \to \mathbb{R}$ are the spline basis functions, to the equivalent *cumulative* form:

$$f(t) = \sum_{i=1}^{n} \Delta p_i . \widetilde{b_i}(t)$$

where $\Delta p_i = p_i - p_{i-1}$, $\Delta p_o = p_0$, and the modified basis functions $\widetilde{b_i}$ are defined accordingly. This serves as the basis for the following quaternion curves:

$$f(t) = \prod_{i=1}^{n} \Delta q_i^{\widetilde{b_i}(t)}$$

where $\Delta q_i = {q_{i-1}}^{-1} q_i$, $\Delta q_0 = q_0$, and $q^x = \exp(x \log(q))$ for $x \in \mathbb{R}$. It should be noted that these quaternion splines only make use of the Lie group structure on the unit quaternion sphere S^3, therefore they can be extended to other Lie groups such as $SO(3)$ and $\mathbb{R}^3$ in a straightforward way.

Coming back to the multi-scale representation of the root orientations, we use such rotation splines with Catmull-Rom basis function between consecutive points at the lower LOD to predict the points from the upper LOD. The prediction error (*i.e.* details) is stored as a 3-vector:

$$\log\left({q_{\text{pred}}}^{-1} . q_{\text{real}}\right)$$

In the end, the root orientations are decomposed as a pyramid of 3-vector, with the finest LODs possibly omitted for compression.

We could also have used simple SLERP [Sho85] prediction between samples (as in [LS01]). However, this leads to a piecewise SLERP reconstructed signal when omitting LODs, which presents discontinuities of the first derivatives that strongly penalize the visual quality of the result. Instead, the use of quaternion spline interpolation results in a smooth reconstructed signal even in the case of missing data.

9.2.3 Putting Everything Together

After the principal geodesics have been extracted from the input motion using approximate PGA, global end-joints trajectories can be compressed using any linear compression method. The root orientation is eventually compressed using the multi-scale representation presented in 9.2.2. For the sake of consistency, our implementation uses the presented multi-scale scheme for both orientations and positions, but any suitable temporal coherence-based compression technique could work.

Decompression The decompression phase consists in decompressing the global trajectories as well as the global root orientation, then expressing the end-joints positions in the root joint frame, and eventually performing the PGA-based IK algorithm to recover poses.

Data Size

Let us now give an estimation of the data size needed to store an animation using our technique. Each of the k geodesics kept after the PGA is a vector of $\mathbb{R}^{3n}$, which is roughly the size of one motion frame. The mean value of the inner joints can also be stored as a vector of $\mathbb{R}^{3n}$ using the exponential map. The data needed for the PGA reconstruction can hence be stored in a $s_{PGA} = (k+1) \times 3n$ matrix. In most of our experiments,

The global root orientations and positions as well as the 5 end-joint positions are compressed by a factor 2^p by omitting p levels of details: each time we remove one level, we divide the data size by two. All those trajectories together can be encoded in a $s_{traj} = 3(2+5) \times \frac{m}{2^p}$ matrix. In most of our experiments, due to the frequency of motion capture sampling (usually 120 Hz), using $p = 3$ (thus compressing trajectories by a factor 8) did not introduce noticeable errors on end-effector trajectories, unless very quickly changing behaviors were present in the motion. Of course, for lower sampling rate, the temporal coherence is lower as well, and so is the expected temporal compression.

Given an initial animation with size:

$$s_{orig} = m \times 3(n+1) = O(m \times n)$$

whereas the compressed version using our algorithm, keeping k geodesics, will have the size:

$$s_{compressed} = s_{PGA} + s_{traj} = O(m+n)$$

Examples of compression ratios obtained using our technique can be found in section 9.3. Before presenting the results obtained with our method, let us explicit what we are the main benefits of an approximate PGA over a PCA of the standard exponential maps for poses parametrization, since the two techniques might seem similar. Even in its approximate form, using PGA leads to a coordinate-invariant and distortion-minimized linearization of the data at the intrinsic mean. This property improves the quality of the statistical analysis since it only depends on the input motion, and not on its *parametrization* (*e.g.* the choice of the reference pose).

Intuitively, even if the considered data are inherently non-linear, there exists a data-driven chart of the manifold in which the situation is much improved. In this chart, using linear statistic approximation makes sense.

9.3 Results

We present here the compression rates of our algorithm on selected motions from the Carnegie Mellon University (CMU) Graphic Lab motion capture database available online[1]. We chose motions with different characteristics of length, diversity, and dynamics as shown on table 9.1.

As stated in section 9.1.3, no really robust and efficient metric is available to assess the quality of the reconstructed animations. However, for the sake of results comparisons, we used a distortion rate as the one defined in

[1] http://mocap.cs.cmu.edu/

ID	Subject/Trial	Description	#Frames
1	09/06	Running, short	141
2	17/08	Walking, slow	6179
3	15/04	Various, dancing, boxing	22948
4	85/12	Breakdance	4499
5	17/10	Boxing	2783

Table 9.1: Motion capture clips used in our experiments

[KG04a] and [LM06] to evaluate the quality of the reconstruction. This distortion rate is defined as:

$$d = 100 \frac{\|A - \widetilde{A}\|}{\|A - E(A)\|}$$

where A is the $m \times 3n$ matrix containing absolute marker positions at each frame for the original motion, $\widetilde{A}$ is the same matrix for the decompressed animation, and each row of $E(A)$ contains the mean marker positions with respect to time.

Table 9.2 shows the obtained compression ratios and distortion rates for different combinations of geodesics numbers and trajectories levels of details. Table 9.3 shows the results obtained by [LM06], who holds the best compression rates at the time of writing. Note that we always used 5 end-joints in our tests, but more could be used if a higher quality is required. As expected, our method works best when the spatial and temporal coherence is high: a rather slow walking motion can be compressed around 180 times with few reconstruction errors, whereas a highly dynamic breakdance sequence is only compressed 118 times for about the same distortion rate. This table shows that our technique allows substantial compression rates improvement over existing techniques, with limited distortion.

9.3.1 Discussion

We proposed a compression algorithm that takes the special geometry of orientation data into account. The compression is achieved by storing a compact, reduced pose-model alongside compressed end-effector and root trajectories. Poses are recovered using our reduced-dimension IK algorithm with decompressed end-effector trajectories. This method allows to obtain high compression ratios, without resorting to quantization as it is the case in concurrent works.

ID	1	2	3	4	5
#Geodesics	6	12	17	15	12
#LOD (root)	4/9	8/14	12/16	9/14	8/13
#LOD (end-joints)	4/9	8/14	12/16	9/14	9/13
Compression ratio	1:18	1:182	1:69	1:97	1:61
Distortion rate (%)	0.36	0.049	1.55	0.56	0.49
Decompression time (msec/frame)	7.88	16.2	30.6	20.42	15.97

Table 9.2: Compression rates, distortion errors and decompression times for the selected motions using our technique. Different combinations of geodesics numbers and trajectories level of details are presented

Sequence	Compression ratio	Distortion rate	Decompr. time (msec/frame)
Jumping, bending, squats	1:55.2	5.1	0.7
Long breakdance sequence	1:18.4	7.1	0.7
Walk, stretches, punches, drinking	1:61.7	5.1	0.7
Walk, stretches, punches, kicking	1:56.0	5.4	0.7

Table 9.3: Compression rates presented in [LM06] using PCA on motion segments. We achieve substantial compression rates improvement with fewer distortion, though our decompression pass takes longer.

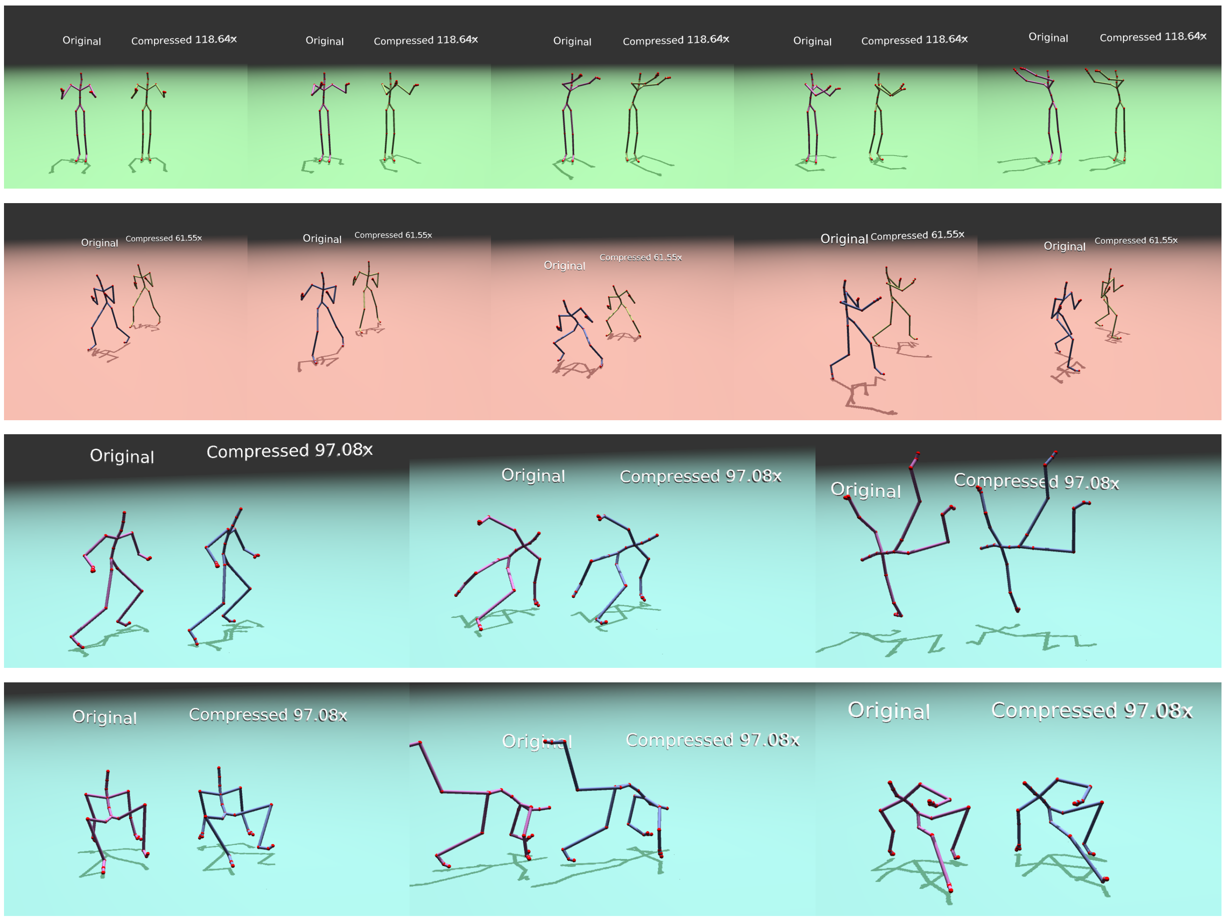


Figure 9.3: 3 animations from the CMU database compressed using our technique

Pose Model When too few geodesics are used in the IK, the reachable pose space gets too small and the synthesized poses sometimes fail to match all the given constraints. On the contrary, once enough geodesics have been selected, further increases on that number only result in a slower optimization time. In our experiments, 10 to 15 geodesics are sufficient in most cases to yield a large enough pose space. For long motions in which clear distinct motion behaviors occur, a segment-based approach similar to [LM06] may be used. This could improve the accuracy of each different pose model, resulting in more natural poses for each behaviors, and possibly better conditioning the Jacobian matrix used in the optimization. The transition between different models would have to remain smooth however, which is not an easy task.

End-Effectors As expected, when the compression for the end-joints trajectories is set too high, artifacts start to appear as the feet contacts on the ground are smoothed too much. In the same way, too high compression over the root joint positions and orientations causes the skeleton to slide, as hung in the air. The acceptable compression ratio for those trajectories highly depends on the dynamics found in the animations. In any case, quantization may be used to compress the trajectories reconstruction errors while controlling additional overhead.

Outliers Since we are performing a statistical analysis to represent the motion poses, strong outlier poses will be more difficult, if not impossible, to reconstruct correctly. In practice, such a situation could arise if the character holds a very specific pose for a very short amount of time with respect to the duration of the motion. We did not encounter such cases in our experiments.

Computational Cost The compression time depends on the length of the animation, as it only involves the intrinsic mean of pose data calculation, and a PCA of the linearized rotations in the tangent space at that point. In practice, it is very inferior to the decompression time, during which an optimization is performed for each frame to reconstruct poses given end-joints constraints. Our implementation was realized in C++ on a Dell workstation, with dual 2.6 Ghz CPU and 4 GB memory.

9.3.2 Conclusion

We presented a novel method for human motion capture data compression exploiting both temporal and spatial coherence, to achieve high compression ratios with few perceptual distortion. Our experiments show that the use of a compact pose model allows to successfully recover poses given only end-joints positions. As the end-joints and root joint trajectories present high temporal coherence, they can also be compressed in order to further improve compression rates. A particularly appealing aspect of our technique is that the pose model may also be used for editing compressed motions by employing the very same IK algorithm.

Though the inverse kinematics algorithm we presented is able to run in real-time on a modern machine, the decompression times are still longer than for other motion capture data compression techniques. However, our implementation could still be improved. The compression technique used for end-joints trajectories could also be enhanced to better reconstruct sharp features, such as foot contacts. The use of a suitable wavelet compression could lead to better results. We also did not exploit the linear correlations present in the end-joint positions: applying a compression technique similar to the one presented in [LM06] to these markers could even improve compression performances, either allowing to further reduce data size, or to increase the number of constraint joints in the IK. If more quality is required, quantization could also be employed to improve the reconstruction of joint trajectories by compressing the errors with controllable size overhead.

⋆ ⋆ ⋆

Finally, using different metrics in the optimization, as presented in the previous chapter, could lead to improved results by penalizing excessive kinetic energy, or angular momentum. In order to produce better quality animations, some dynamics elements could be automatically inferred by the animation algorithm by formulating and simulating the laws of physics for our reduced pose model. This is the subject of the next part of this thesis.

Part IV

Dynamics

Chapter 10

Previous Work

Having described how to use a PGA-based reduced pose model in a kinematic animation context, we now move on to the physically-based animation of a character, using this reduced model.

We begin this part with a review of existing physically-based character animation techniques, modal reductions and contacting systems in chapter 10. In chapter 11, we use the reduced pose model as the generalized coordinates in the Lagrangian formulation of mechanics, in order to obtain our model Lagrangian. We derive a velocity-level, explicit time integrator for this model in section 11.1, based on geometric variational integrators. Since our time integrator is explicit, we propose a damping scheme inspired by the Levenberg-Marquardt algorithm in 11.2, in order to improve stability. A geometric, data-driven angular limits learning procedure, and the associated kinematic constraints are proposed in 12.

10.1 Physical Models for Articulated Bodies

Modeling the *complete* physical behavior of the entire human body for animation is challenging, due to the complexity of the structures involved, and the tight physical coupling between them. Although it has some applications in biomechanics and motion pictures, the vast majority of character animation techniques builds on simplified robotics representations, and models the human body as a collection of articulated rigid bodies, connected using idealized joints.

The problem of animating such *articulated rigid bodies* has received extensive attention in the past, mostly because rigid bodies often provide a good approximation of solid objects, and the rigid approximation allows for fast numerical simulations, as we will see. We will not review the classical physics *theory* behind rigid bodies animation, as it can be found in most physics or robotics books (see [MSZ94] for instance). Instead, we will focus on the different approaches for *computing* the motion of rigid objects, in the context of animating an articulated character.

10.1.1 Problem Formulation

There are two main strategies for animating articulated rigid bodies:

- By assuming that all the articulated bodies are *a priori*independent, then computing and applying *constraint forces* in order to restrict their global degrees of freedom
- By assuming that a complete description of the degrees of freedom of the system is available (taking mechanical constraints into account) and expressing the laws of dynamics in this setting

The first alternative starts from the Newton-Euler equations of motion for rigid-bodies, and seeks constraint forces through the computation of *Lagrange Multiplier*, giving their name to this approach. The second is known as the *reduced coordinates* approach, and follows the formalism of the *Lagrangian* mechanics. We quickly present these two strategies and how suitable they are for our purposes.

10.1.2 Maximal Coordinates

The Newton-Euler equations describe the equations of motion for a single rigid body. They are given by [MSZ94]:

$$\begin{pmatrix} \mathcal{I} & 0 \\ 0 & m.I \end{pmatrix} \begin{pmatrix} \dot{\omega}^b \\ \dot{v}^b \end{pmatrix} + \begin{pmatrix} \omega^b \times \mathcal{I}\omega^b \\ \omega^b \times m.v^b \end{pmatrix} = \begin{pmatrix} \tau \\ f \end{pmatrix} \tag{10.1}$$

where m is the total mass of the rigid body, and $\mathcal{I}$ is the body-fixed inertia tensor. ω and v are respectively the angular and linear part of the body-velocity, and τ, f are respectively the angular and linear part of the body-fixed net external force. In computer applications however, equation (10.1) is often integrated using an *explicit* time stepping scheme of the form [Bar92]:

$$M_t \dot{v}_{t+1} = f_t \tag{10.2}$$

where v_{t+1} gathers the velocity at time $t+1$ of each body, M_t is the (block-diagonal) mass-inertia tensor, and f_t gathers the external and inertial forces at the previous time-step. We indicate the time step corresponding to each quantity with subscripts, however we will sometimes omit them for clarity, once they have been introduced.

Under this formalism, it is possible to enforce kinematic constraints $J_t\, \dot{v}_{t+1} = b_t$ on the acceleration, where J_t is a constraint matrix, and b_t holds the corresponding constraint values, both computed at time t. To do so, the associated *constraint forces* $J_t^T \lambda_{t+1}$ are added to the system, where λ_{t+1} are new unknowns called *Lagrange Multipliers*. This leads to the following system of *constrained* equations, where $\dot{v}_{t+1}$ and λ_{t+1} are the unknowns:

$$\begin{aligned} M_t\, \dot{v}_{t+1} &= f_t + J_t^T \lambda_{t+1} \\ J_t\, \dot{v}_{t+1} &= b_t \end{aligned}$$

[Bar96] showed that using this formulation and under certain hypothesis on the constraints, this system can be solved in linear-time. More precisely, when the *constraint graph* for the system is *acyclic*, a sparse Cholesky factorization and solving of this linear system can be performed in $O(m+n)$, where $n \in \mathbb{N}$ is the number of degrees of freedom and $m \in \mathbb{N}$ is the number of scalar constraints.

Working at the force/acceleration level has a number of drawbacks, the most notables being:

- The J matrix and b vector can be difficult to compute as they usually involve second-order derivatives even for simple holonomic constraints (*i.e.* of the form $h(q) = 0$, where q are the DOFs)
- The system may fail to have a solution in the presence of frictional contact constraints (*cf.* the Painlevé paradox, see [AH04])

A common way to work around these issues is to formulate the system dynamics at the velocity/impulses level instead, as in [AP97, CP03, Erl07]. This is generally achieved by using forward finite differences:

$$v_{t+1} \approx v_t + \dot{v}_{t+1}.dt$$

Plugging this equation into the dynamics (10.2), and expressing constraints at the velocity level $(Jv = b)$ leads to a system of the following form:

$$\begin{aligned} Mv &= f + J^T \lambda \\ Jv &= b \end{aligned}$$

Here, f is an *impulse* containing the sum of the *momentum* and the external/inertial impulses at the previous time step:

$$f = Mv_t + dt.f_t^{ext}$$

and the $J^T\lambda$ are now constraint *impulses*, with λ accounting for the dt.

Due to the fact that the constraints are not enforced directly by the system degrees of freedom, a *numerical drift* can (and most probably will) happen as the constraints are only maintained at the discrete time samples. Therefore, a constraint *stabilization* pass usually has to be performed in order to compensate for this problem, as for instance the post-stabilization method [CP03] for holonomic constraints (*cf.* figure 10.1).

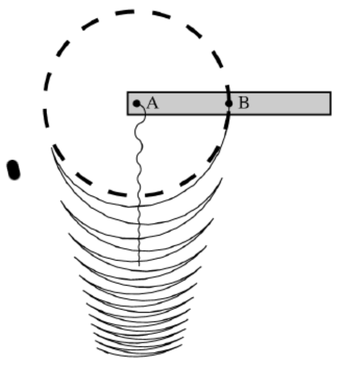


Figure 10.1: Unstabilized simulation of a swinging pendulum. Constraint post-stabilization is used to correct such constraint drift *(from [CP03])*.

10.1.3 Generalized Coordinates

Instead of formulating the Newton-Euler equations for the individual bodies, then applying constraint forces to it, the *generalized coordinates* approach *directly* formulates the system dynamics in a coordinate system encoding the constraints, as for instance an angular parametrization of a pendulum. In this setting, the equations of motion are derived from a *variational principle* describing how the system DOFs behave between time-steps, in relation to energy variations. Letting Q be the configuration space of the system, the total system energy is summarized by a smooth function called the *Lagrangian*:

$$\mathcal{L} : TQ \to \mathbb{R}$$

This Lagrangian is used to obtain the equations of motion in a systematic way, usually through the Euler-Lagrange equations. We quickly review this process now.

Lagrangian

This approach tightly follows the *Lagrangian* formulation of classical mechanics, which generalizes the Newton-Euler equations of motion in any coordinate system. To do so, the kinetic (T) and potential (V) energies are grouped together forming the *Lagrangian*, expressed in terms of the *generalized coordinates* $q \in Q$, and *generalized velocities* $v \in T_q Q$ by:

$$\mathcal{L}(q, v) = T(q, v) - V(q, v) \in \mathbb{R} \tag{10.5}$$

Euler-Lagrange Equations

The *Hamilton principle* states that in the absence of external forces, the motion of the system on the time interval $[t_1, t_2]$ is given by the critical points of the following *action* functional S:

$$S(q) = \int_{t_1}^{t_2} \mathcal{L}(t, q(t), \dot{q}(t)).dt \tag{10.6}$$

Equivalently, these critical points can be expressed as a differential equation, known as the *Euler-Lagrange* equations:

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}} = \frac{\partial \mathcal{L}}{\partial q}$$

Several numerical schemes have been proposed to integrate this differential equations, with different orders of accuracy and computational efficiency.

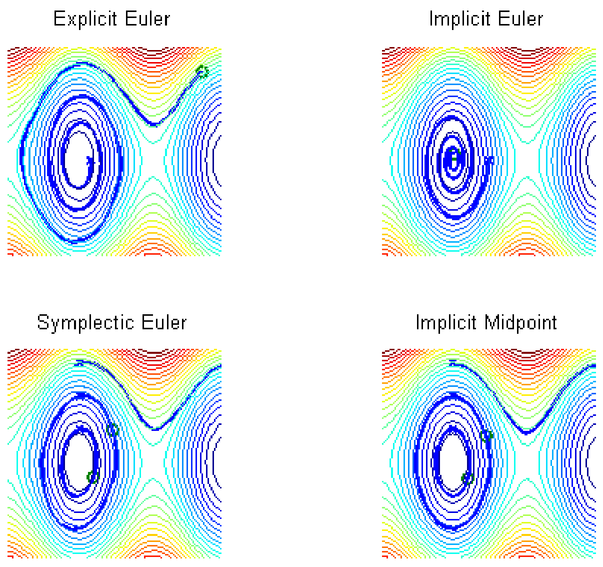
For articulated rigid-bodies, one of the most widespread method for solving the associated time-discretized equations of motion is the linear-time algorithm for acyclic articulated chains by [Fea87]. However, adapting this algorithm for our purposes is far from immediate.

More generally, using the Euler-Lagrange equations automatically results in second-order differential equations, which implies that several mappings have to be derived twice. Instead, Hamiltonian formulations of mechanics result in more equations, but only first-order. Recent advances in geometric integrators provide such formulations.

Geometric Integrators

Geometric integrators for a differential equation preserve a certain property of the exact solution, or flow. This property usually comes as some kind of invariant, or symmetry, such as for instance energy conservation in a closed system.

Symplecticity In the case of the Euler-Lagrange equation, *symplecticity* is an important property of the exact flow that is preserved by *symplectic* integrators. Without going into details, symplectic flows preserve the volume of any part of the *phase space*, which is a coordinate change from position/velocity to position/momentum. This volume conservation is related to the energy conservation in a closed system: by design, symplectic integrators almost preserve the energy of such system (for a time-independent Lagrangian), while non-symplectic ones often exhibit so-called *energy drift*, as it is the case for the explicit/implicit Euler integrator (see figure 10.2).



(CC BY-SA Maksim)

Figure 10.2: Phase space (position, momentum) trajectories for a simple pendulum, using different integrators. *Symplectic Euler* and *Implicit Midpoint* are symplectic integrators, resulting in stable, cyclic orbits for this conservative system, unlike *Explicit/Implicit Euler* (*i.e.* energy drift).

Symplectic integrators, even explicit ones, usually allow for larger time steps during simulation, as they better capture the structure of the underlying differential equation.

Discrete Hamilton Principle While deriving a symplectic integrator for the Euler-Lagrange equation is not an easy task in general, Marsden *et al.* [BRM09, KYT+06, KCD09] showed that symplectic integrators with any order of accuracy can be obtained by discretizing the Hamilton *principle* instead of the Euler-Lagrange *differential equations*. The basic idea is to derive an approximating *quadrature* L^d of the action functional:

$$L^d(q_k, v_{k+1}) \approx \int_{t_k}^{t_{k+1}} \mathcal{L}(t, q, \dot{q}).dt$$

Applying the Hamilton principle to this *discrete* action actually *defines* a symplectic integrator. So instead of a non-symplectic integrator integrating the Euler-Lagrange equations for the exact Lagrangian, this method systematically obtains a symplectic integrator for an approximation of the action functional. The accuracy of this scheme depends on the order of accuracy of the quadrature rule.

Hamilton-Pontryagin Principle Lagrangian and Hamiltonian formalisms can be unified through the Hamilton-Pontryagin (HP) variational principle [KYT+06], which treats position and velocities as *a priori*independent quantities, using momentum p as Lagrange multipliers to enforce the constraint $\dot{q} = v$:

$$\delta \int_o^T L(q, v) + p^T \left(\dot{q} - v\right).dt = 0 \tag{10.7}$$

The same methodology (approximating quadrature) can be applied to this variational principle in order to obtain symplectic integrators. The Hamiltonian approach avoids second-order differential equations as it is the case in the Lagrangian formalism, but results in (usually, twice) more first-order equations. The Legendre transform classically needed to derive the Hamiltonian formulation from the Lagrangian one, is implicit in the above HP principle.

Variational HP integrators systematically provide symplectic update rules for q, v, p as a non-linear system, that can be cast as an optimization problem, under some assumptions on the Lagrangian [KYT+06]. In contrast, the Euler-Lagrange equations are second-order and must be discretized using finite differences to obtain non-linear, and usually non-symplectic, $q, \dot{q}$ update rules.

Finally, this integrator family naturally generalizes to Lie group configuration spaces, as shown in [BRM09, KCD09], which will be useful for our purposes.

10.1.4 Constraint Systems

Here we quickly review the general problem of solving kinematic constraints and contacts, since these are essential feature of a physical simulation, and will be used in the next part on character control. In all this section, we will assume that the dynamics of our system are given by the following velocity-level linear time-stepping scheme:

$$Mv = f \tag{10.8}$$

assuming that M and f are given, M is a Positive Semi-Definite (PSD) matrix, and we are solving for the velocity v.

Bilateral Constraints

As we have seen in 10.1.2, the bilateral velocity constraints $Jv = b$ can be enforced by solving the following linear system:

$$\begin{aligned} Mv &= f + J^T\lambda \\ Jv &= b \end{aligned} \tag{10.9}$$

When J has full rank and M is invertible, the above system can be reduced to the smaller, following system [Bar96]:

$$JM^{-1}J^T\lambda = b - JM^{-1}f$$

One can notice that the system (10.9) describes a Lagrange point for the following convex Quadratic Program (QP):

$$v^\star = \underset{Jv=b}{\operatorname{argmin}} \quad \frac{1}{2}v^TMv - f^Tv$$

This QP actually reduces to a linear-least squares problem, which has perhaps a much more intuitive geometric interpretation when written as:

$$v^\star = \underset{Jv=b}{\operatorname{argmin}} \quad ||v - M^{-1}f||_M^2$$

In other words, the bilateral constraint problem is equivalent to projecting the *unconstrained* velocity $M^{-1}f$ on the (convex) linear space $Jv = b$, according to the kinetic metric $||v||_M^2 = v^TMv$ (*cf.* figure 10.3).

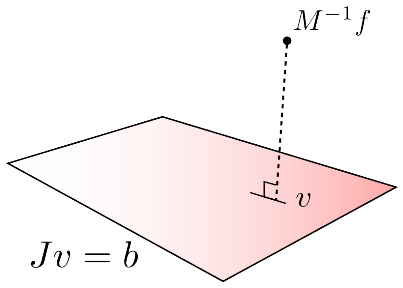


Figure 10.3: Bilateral constraints solving as a M-orthogonal projection of the unconstrained velocity $M^{-1}f$ on the feasible set $Jv = b$.

Another equivalent formulation is to say that the velocity *correction* $v_c = M^{-1}J^T\lambda = v - M^{-1}f$ should have a minimal kinetic energy.

Unilateral Constraints

The case of unilateral constraints $Jv \geq b$ differs in that the system to solve is no longer linear, but exhibits a *complementarity* constraint, ensuring that no impulse can occur when the contacting bodies are separating [Bar94, AP97]. Conversely, no relative velocity should occur when a contact is active (*i.e.* has a non-zero contact impulse, given by the corresponding λ):

$$\begin{aligned} Mv &= f + J^T\lambda \\ Jv &\geq b \\ 0 \leq Jv - b \quad &\perp \quad \lambda \geq 0 \end{aligned}$$

The same manipulations as in the bilateral case show that λ must satisfy:

$$0 \leq JM^{-1}J^T\lambda - b + JM^{-1}f \quad \perp \quad \lambda \geq 0$$

which is known as a Linear Complementarity Problem (LCP) [Cot09]. LCPs usually arise as the Karush-Kuhn-Tucker (KKT) conditions for QPs [BV04]. Actually, the above LCP characterizes *the* KKT point for the following convex QP:

$$v^\star = \underset{Jv \geq b}{\operatorname{argmin}} \quad \frac{1}{2}v^TMv - f^Tv$$

Again, this QP can be reformulated in a somewhat more intuitive fashion:

$$v^\star = \underset{Jv \geq b}{\operatorname{argmin}} \quad \left|\left|v - M^{-1}f\right|\right|_M^2$$

Here again, we look for the M-projection of the unconstrained velocity onto the convex set of admissible velocities.

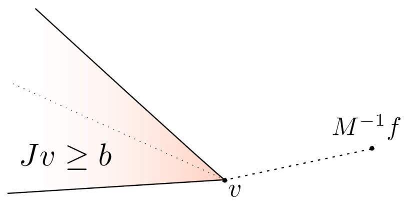


Figure 10.4: Unilateral constraints as the M-projection of the unconstrained velocity $M^{-1}f$ on the feasible set $Jv \geq b$ (a convex polytope).

Frictional Contacts

Energy dissipation is commonly expressed through a *viscous* damping, *i.e.* a force in $-\alpha v$ where $\alpha \in \mathbb{R}$ is the damping factor. However, while this kind of damping is very easy to compute and apply, it does not model *dry*[1]

[1] Dry friction models attempt to describe the two distinct *sticking* and *sliding* phases.

frictional behavior correctly. Therefore, the well-known Coulomb's friction law is usually preferred, which unfortunately considerably complicates the dynamics computation.

Coulomb's Law Dry frictional contacts following Coulomb's law are much more difficult to handle than all that precedes, due to the disjunction of cases it models. Coulomb's law states that the reaction force F should oppose the motion at the contact point, and be part of the so-called *friction cone*:

$$F_T \leq \mu F_N$$

where F_T, F_N are respectively the tangential and normal component of the contact force, and $\mu \in \mathbb{R}^+$ is the material-dependent friction coefficient determined experimentally. There has been extensive literature on this problem for the last decades, both from the theoretical and practical points of view.

While recent research work [BDCDA11] proposed to solve the *exact* Coulomb's law friction problem, by formulating the solution as a non-smooth root-finding problem, the overwhelming majority of authors propose to use a linearized approximation of Coulomb's cone, enabling the use of Linear (LP) and Quadratic Programming algorithms instead of the more expensive Second Order Cone Programming (SOCP) ones.

Linearized Friction Cone When using a linearized friction cone, the frictional contact problem can be modeled as a (non-symmetric) LCP [AP97]. This asymmetry can be explained by the coupling between normal and tangential responses, which makes the full problem *non-convex*. This is a serious theoretical issue since it suggests a potentially exponential time for solving is needed (see [KSJP08] for details).

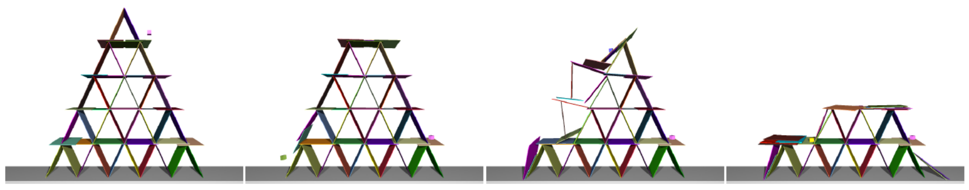


Figure 10.5: House of cards simulation obtained by [KSJP08]. Maintaining initial equilibrium for the whole structure is challenging due to the normal/tangent reaction coupling induced by Coulomb friction.

Instead of solving both tangential and normal responses together in the same optimization problem, [KSJP08] proposed to treat them separately, solving only tangential response assuming normal response is given, and conversely. Each sub-problem is this time a convex QP, (*i.e.* a projection on a convex polytope), and the authors showed that this alternate projection sequence has the full frictional problem solution as a fixed-point. Though no convergence proof was given, it was reported experimentally even on challenging contacting situations (card house, *cf.* figure 10.5).

10.2 Modal Analysis

The benefits of dimension reduction have been exploited in the context of physically-based animation, since they allow to reduce computational requirements while preserving most dynamic features of the full-dimension simulation. As an added benefit, these techniques sometimes provide a *mechanically* orthogonal description of a system which is particularly interesting for computational purposes.

10.2.1 Modal Dynamics

[PW89] pioneered the use of modal analysis in the context of computer graphics. The basic idea of this approach is to decompose a mechanical system with numerous, coupled mechanical degrees of freedom into an equivalent set of mechanically-independent one-dimensional degrees of freedom, usually called *modes*. The set of all modes forms the *modal basis* which can be seen as an alternate representation of the system DOFs.

Mechanically independent modes allow to easily compute the whole system response as the sum of each modal response, which are typically very fast to compute since they are one-dimensional. Modes mechanical independence also permits aggressive parallel computations, and is therefore of significant interest for computer animation where highly parallel Graphical Processing Units (GPUs) are widely available.

Finally, modes can be selected according to their energetic contribution to the whole system response, usually allowing for significant dimension reduction with respect to the original DOFs, while still providing convincing dynamic behavior.

In contrast to these benefits, the main bottleneck of this approach is usually the modal basis computation, which must usually be performed offline. In [PW89], a Finite Element Model (FEM) is linearized around some rest position, providing mass tensor M and stiffness tensor K, both being PSD matrices. A bi-diagonalization of M and K is performed, resulting in a modal basis U making modes *mechanically* independent. Since this process is costly, it is usually done only once around a rest position, imposing only small displacements during the simulation.

The dimension reduction is achieved by selecting modes based on the corresponding eigenvalues, allowing to get rid of unwanted, high frequency modes often responsible for instabilities in explicit time integration schemes.

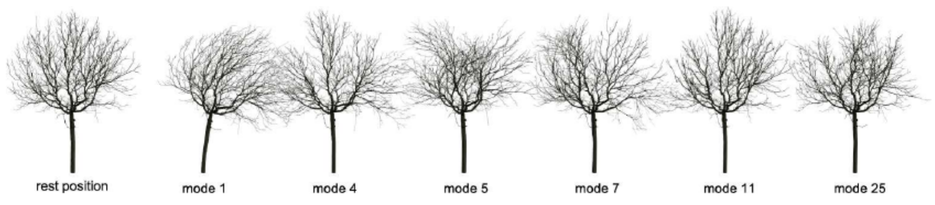


Figure 10.6: Several vibration modes for a tree structure computed using modal analysis [DRBR09]

Due to its numerous advantages, modal analysis has spawned a wide variety of research works ([JP02, JBP06, KJP02, DRBR09], among others) building on this basic principle, and can even be seen as the basis of more recent work on spectral mesh processing [LZ09]. Unfortunately, such decompositions are limited to small linear displacements around some rest pose.

10.2.2 Modal Locomotion

[KRFC09] proposed an interesting extension of this technique to articulated rigid bodies: rather than performing the modal analysis on a mesh-supported, linearized FEM, the authors applied the modal decomposition to the dynamics of an articulated rigid body around a rest pose, obtaining new mechanically-independent angular degrees of freedom around a rest pose. This was the basis for animating locomotion behaviors as combinations of natural vibration modes.

The core assumption of this work is that animal bodies are structured in such a way that their natural vibration modes can be easily actuated to produce common motion patterns, such as locomotion. Exploiting the *passive* dynamics (*i.e.* without actuation) provided by their bodies allows animals to obtain their most common gaits at a minimal energy cost.

Even if this kind of analysis is, again, restricted to small displacement around a rest pose, the fact that displacement are *angular* allows for much larger global character motion without noticeable *visual* artifacts, such as finite elements elongation in [DRBR09]. As for the dynamics however, the mass tensor can only be considered constant in a neighborhood of the rest pose, due to the significant non-linearities found in the forward kinematics.

Furthermore, the basis resulting from the modal analysis is highly dependent on the stiffness and damping parameters chosen for the articulated

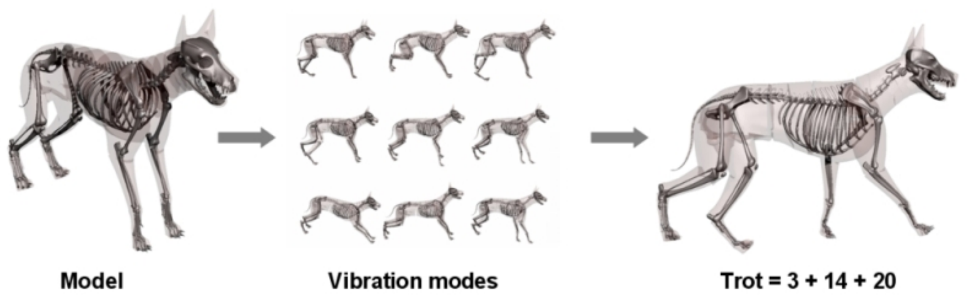


Figure 10.7: From left to right: a dog model at the rest pose, vibration modes extracted from the modal analysis, and modes recombination to obtain a trot animation [KRFC09]

model. These values are difficult to set up manually, even though some automatic estimation procedures were proposed by [LHP05] when motion capture data is available.

10.2.3 Conclusion

Most existing physically-based dimension reduction techniques are based on the modal analysis, which has two major drawbacks:

- The modal decomposition is costly and must therefore be precomputed, making the resulting model only suitable for small-displacements, *e.g.* locomotion.
- In the case of articulated rigid bodies, the joint stiffness must be known for the modal analysis to make sense.

Instead, expressing the dynamics in our reduced pose parametrization is not restricted to small displacements. Besides, the reduced basis we obtained in 8 is not dependent on hand-chosen stiffness/damping parameters, but only on motion capture data.

10.3 Statistical Analysis of Torques

An alternate approach featuring dimension reduction for character animation is the one followed by [YL08]. The goal of this work is to construct a data-driven basis of torques from motion capture data, then extract the coordinates corresponding to the *least* actuated torques. This *near-unactuated* basis is later used to synthesize upper-body, physically-based perturbations on top of existing motion capture animations. The main assumption is that the unactuated coordinates tend to be much more compliant, in the case of external disturbances, than the actuated ones.

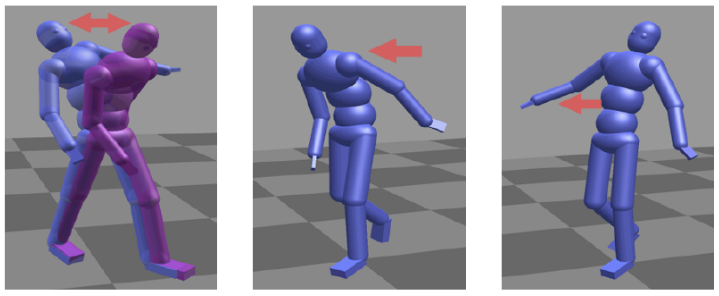


Figure 10.8: Example perturbations obtained by [YL08].

While this technique seems promising, it has several drawbacks compared to expressing the dynamics in the PGA-reduced pose space:

- It requires inverse dynamics, which is challenging to setup in the case of contacts (explaining the upper-body limitation).
- It is limited to motion perturbations.
- The statistical analysis does not provide nor exploit dimension reduction.

As we have seen, all previous methods exploiting dimension reduction in a physically-based animation context possess inherent limitations (small displacements, need for inverse dynamics), which leads us to express the laws of dynamics directly for our reduced kinematic pose model instead. We describe this process in the following chapter.

Chapter 11

PGA Dynamics

In this chapter we expose how we use the PGA-reduced pose parametrization in a physical simulation in more details. We begin by deriving the discrete equations of motions based on the variational integrators described in the previous works, to obtain an explicit time integrator. We then describe how to include kinematic constraints in this model, and propose a new formulation for joint limit constraints. Finally, we draw the parallel between our physical modeling and the classical Gauss-Newton optimization scheme and propose an adaptive damping scheme based on Levenberg-Marquardt algorithm to improve the stability of our explicit integrator.

11.1 Equations of Motion

In this section we motivate and describe a time-stepping scheme for physically animating a virtual character parametrized using PGA. After a discussion on the use of maximal vs. reduced coordinates, we derive a geometric integrator based on [KCD09]. Finally, we propose an approximated, explicit time-integrator, for use in a real-time simulation.

11.1.1 Motivation

As seen in the related work section, we are faced with two main approaches for simulating articulated rigid bodies: maximal versus reduced coordinates. We quickly sketch the methodology behind each of these choices, in the context of a reduced pose parametrization.

Maximal Coordinates

The use of maximal coordinates seems easier at first sight as it involves well-known discrete equations of motion for rigid-bodies. As we have seen, enforcing bilateral constraints results in the following linear system:

$$\begin{aligned} Mv &= f + J^T\lambda \\ Jv &= b \end{aligned}$$

where M is the block-diagonal inertia tensor, and J are kinematic constraints forcing the state of the system to belong to the reduced manifold computed using PGA. The major problem here is that we do not have a *holonomic* parametrization of the reduced manifold, *i.e.* as $M = f^{-1}(0)$ for some smooth function f. It is thus difficult to obtain the J matrix as the differential of f.

Even if we had a good candidate for the J matrix, it is likely to be dense as the very purpose of PGA parametrization is to encode *full-body* pose correlations. Thus the resulting linear system will be both high-dimensional *and* dense: if n is the dimension of the unconstrained system, and k is the (low) number of remaining degrees of freedom, we must solve the following $(n-k) \times (n-k)$ dense linear system:

$$JM^{-1}J^T\lambda = b - JM^{-1}f$$

What is more, as the number of constraints raises, the probability of having linearly dependent constraints (J rows) increases, which usually results in an infeasible problem. Devising an automatic relaxation strategy can be difficult and computationally expensive.

Another issue related to not having a holonomic description of the PGA parametrization is that it is much more difficult to correct constraint drift in a systematic way (as for instance using post-stabilization [CP03]). The only solution is then to project poses onto the reduced manifold in an energy-agnostic way, which is very likely to produce noticeable visual artifacts: the human eye is sadly very good at noticing energy-incorrect corrections.

Reduced Coordinates

The use of reduced coordinates seems natural in our context since we already have the reduced coordinates pose mapping as the PGA pose parametrization. In this case the constraint forces are implicit, and no constraint drift can occur by design. However, this formalism imposes to derive dedicated equations of motion, and notably to compute the reduced mass tensor and Coriolis forces according to the reduced coordinates mapping. The Coriolis forces are especially challenging to compute since they involve the second derivative of the Jacobian.

In this context, the low-dimensionality of the PGA parametrization will be an advantage since it will keep the computational cost reasonable while still enforcing natural poses.

One major drawback of reduced formulations for articulated bodies is that errors add along the skeleton hierarchy, potentially leading to stability issues for the bottom-most nodes. This issue is especially aggravated around singular configurations (*i.e.* where the articulated body loses one degree of freedom, as for instance when an arm is fully extended), where the numerical stability of the Jacobian (and thus of the reduced mass tensor) is usually poor as already discussed in 8.2.3.

Discussion

We summarize the pros and cons of each approach in the following tables 11.1 and 11.2

Maximal Coordinates	
Pros	Cons
• Well-known, standard rigid body integrators • Block-diagonal, constant mass-matrix • Explicit link forces available	• High dimensionality, dense constraint system • Numerical instability for large amounts of constraints • Constraint drift • No holonomic parametrization of reduced manifold

Table 11.1: Maximal Coordinates

Reduced Coordinates	
Pros	Cons
• No constraint drift by design (when applicable) • Low dimensionality • Fits naturally with the reduced pose parametrization	• Requires a dedicated physical model and integrator • Numerical instability near singular configurations • Inertia forces computation, dense mass tensor

Table 11.2: Reduced Coordinates

From these tables we conclude that in our context, the maximal coordinates approach is flawed with several critical issues, the most serious one being that it does not take advantage of the reduced dimension properly. We therefore opt for a reduced coordinate approach, addressing the above issues.

11.1.2 Geometric Integrator Derivation

We now derive the discrete equations of motion for a PGA-parametrized virtual character. For this, we use the geometric, variational integrator methodology presented in [KCD09] as they generalize those by [KYT+06] to the case of Lie groups.

Unfortunately, the results in [KCD09] assume that the Lagrangian is left-invariant, which is not the case in our context. Furthermore, the derivations leading to this result are a bit involved, though insightful, in our opinion. Therefore, we now derive equations for the general case in this part.

Lagrangian Definition

We suppose that we are given a skeleton parametrization in terms of the following reduced configuration space:

$$G = SO(3) \times \mathbb{R}^3 \times \mathbb{R}^k$$

Its elements represent the orientation and position of the root joint, and the k geodesic coordinates for the reduced pose model described in part III. Note that we use $SO(3) \times \mathbb{R}^3$ instead of $SE(3)$ for the root configuration for practical reasons, since the Lie group integrators involve the derivative of the exponential map. This expression is much easier to compute on $SO(3) \times \mathbb{R}^3$ than it is on $SE(3)$ [BM95].

Kinetic Energy We assume that the body-fixed inertia tensors for the $n+1$ bones composing the skeleton are given as a PSD, block-diagonal matrix $\widetilde{M}$ of size $6(n+1)$. Let $f : G \to SE(3)^{n+1}$ be the forward kinematics, mapping reduced coordinates to world bone configurations (*cf.* 8.2.2). The kinetic energy is defined as:

$$T(g, v) = \frac{1}{2} v^T J_f^b(g)^T.\widetilde{M}.J_f^b(g).v$$

where $g \in G$ and $v \in \mathfrak{g}$ is a body velocity. The above expression defines the generalized mass matrix, or inertia tensor as:

$$M(g) = J_f^b(g).\widetilde{M}.J_f^b(g)$$

Discrete Hamilton-Pontryagin Principle

Let us first recall the HP principle in the vector case:

$$\delta \int_0^T \mathcal{L}(q,v) + p^T \left(\dot{q} - v\right) .dt = 0$$

where $p, q, v \in \mathbb{R}^n$. For a Lie group configuration space G, we can restate this principle in terms of body-fixed velocities:

$$\delta \int_0^T \mathcal{L}(g,v) + p^T \left(\dot{g}^b - v\right) .dt = 0$$

where $g \in G, \quad \dot{g}^b, v \in \mathfrak{g}, \quad p^T \in \mathfrak{g}^*$, with $\dot{g}^b$ being the body-fixed velocity, and using the notation $p^T v$ for the natural pairing between $\mathfrak{g}$ and $\mathfrak{g}^*$.

[KCD09] propose a discrete approximation of this integral (a *quadrature*) in terms of a *group difference* map $\tau : G \to \mathfrak{g}$ (usually $\tau = \log$):

$$\int_0^T \mathcal{L}(g,v)+p^T \left(\dot{g}^b - v\right) .dt \approx \sum_{k=0}^{N} h.\mathcal{L}(g_k, v_{k+1})+p_{k+1}^T \left(\tau({g_k}^{-1} g_{k+1}) - h.v_{k+1}\right)$$

where h is the time step. We see that the group difference map is used as an approximation of the body-velocity between two consecutive configurations, so the logarithm seems like a natural choice[1]. We end up with the *discrete* Hamilton-Pontryagin principle:

$$\delta \sum_{k=0}^{N} h.\mathcal{L}(g_k, v_{k+1}) + p_{k+1}^T \left(\tau({g_k}^{-1} g_{k+1}) - h.v_{k+1}\right) = 0 \qquad (11.2)$$

Taking variations in g, v and p with fixed endpoints for g, and expressing the optimality criterion results in a symplectic update rule for g, v and p. This is what we describe now.

Variations

The calculus of variations essentially deals with deriving *functionals* with respect to *functions*, whereas usual calculus derives *functions* with respect to *variables*. While conceptually similar, the terminology changes: in the very same way that we derive functions with respect to tangent vectors, functionals are derived with respect to *variations*.

Given a smooth function of time $q : \mathbb{R} \to Q$, a *variation* of q can be seen as a smooth function $\delta q : \mathbb{R} \to TQ$ such as each $\delta q(t) \in T_{q(t)}Q$ describes a point-wise tangent vector at each $q(t)$. For some smooth function f defined on Q, we thus have:

$$\delta(f \circ q)(t) = df\left(q(t)\right).\delta q(t)$$

[1] Though for some groups, the Cayley map can be used instead as it is usually simpler to compute

Momentum Taking variations in p in equation (11.2) gives:

$$\sum_{k=0}^{N} \delta p_{k+1}^T \left(\tau({g_k}^{-1} g_{k+1}) - h.v_{k+1}\right) = 0$$

The *fundamental lemma of the calculus of variations* [Lei81] implies that each of the terms in the above sum is zero, thus for all $k \leq N$:

$$\delta p: \quad \tau({g_k}^{-1} g_{k+1}) = h.v_{k+1} \in \mathfrak{g} \tag{11.3}$$

Velocity Taking variations in v in equation (11.2) gives:

$$\sum_{k=0}^{N} \frac{\partial \mathcal{L}}{\partial v}(q_k, v_{k+1}).\delta v_{k+1} - p_{k+1}^T \delta v_{k+1} = 0$$

$$\sum_{k=0}^{N} \left(\frac{\partial \mathcal{L}}{\partial v}(q_k, v_{k+1}) - p_{k+1}^T\right).\delta v_{k+1} = 0$$

The same argument implies that for all $k \leq N$:

$$\delta v: \quad \frac{\partial \mathcal{L}}{\partial v}(q_k, v_{k+1}) = p_{k+1}^T \in \mathfrak{g}^* \tag{11.4}$$

Position Taking *spatial* variations in g (see 5.2.5), with fixed end-points (*i.e.* $\delta^s g_0 = \delta^s g_N = 0$) in equation (11.2) gives:

$$\sum_{0}^{N} h.\frac{\partial^s \mathcal{L}}{\partial g}\left(g_k, v_{k+1}\right).\delta^s g_k + p_{k+1}^T d^s\tau\left({g_k}^{-1} g_{k+1}\right).\delta^s\left({g_k}^{-1} g_{k+1}\right) = 0$$

Letting $d_{k+1} = {g_k}^{-1} g_{k+1}$, we have that:

$$\delta^s d_{k+1} = Ad_{{g_k}^{-1}}(\delta^s g_{k+1} - \delta^s g_k)$$

Regrouping terms in $\delta^s g_k$ and $\delta^s g_{k+1}$ gives:

$$\begin{aligned}\sum_{k=0}^{N} \left(h.\frac{\partial^s \mathcal{L}}{\partial g}(g_k, v_{k+1}) - p_{k+1}^T d^s\tau(d_{k+1}).Ad_{{g_k}^{-1}}\right).\delta^s g_k \\ + \sum_{k=0}^{N} p_{k+1}^T d^s\tau(d_{k+1}) Ad_{{g_k}^{-1}} \delta^s g_{k+1} = 0\end{aligned} \tag{11.5}$$

The second sum can be reformulated as:

$$
\begin{aligned}
\sum_{k=0}^{N} p_{k+1}^T d^s\tau(d_{k+1}) Ad_{g_k^{-1}} \delta^s g_{k+1} &= \sum_{k=1}^{N+1} p_k^T d^s\tau(d_k) Ad_{g_{k-1}^{-1}} \delta^s g_k \\
&= \sum_{k=0}^{N} p_k^T d^s\tau(d_k) Ad_{g_{k-1}^{-1}} \delta^s g_k
\end{aligned}
$$

where we used the fact that the endpoints are fixed in the last line. Rearranging terms in (11.5) gives:

$$
\sum_{k=0}^{N} \left(h.\frac{\partial^s \mathcal{L}}{\partial g}(g_k, v_{k+1}) - p_{k+1}^T d^s\tau(d_{k+1}).Ad_{g_k^{-1}} + p_k^T d^s\tau(d_k) Ad_{g_{k-1}^{-1}} \right) .\delta g_k = 0 \tag{11.6}
$$

Here again, this shows that for all $k \leq N$:

$$
h.\frac{\partial^s \mathcal{L}}{\partial g}(g_k, v_{k+1}) - p_{k+1}^T d^s\tau(d_{k+1}).Ad_{g_k^{-1}} + p_k^T d^s\tau(d_k) Ad_{g_{k-1}^{-1}} = 0 \tag{11.7}
$$

At this point, it is worth noticing that:

$$
\frac{\partial^s \mathcal{L}}{\partial g}(g_k, v_{k+1}) = \frac{\partial^b \mathcal{L}}{\partial g}(g_k, v_{k+1}).Ad_{g_k^{-1}}
$$

Right-multiplying (11.7) by Ad_{g_k} finally gives:

$$
\delta g : \quad h.\frac{\partial^b \mathcal{L}}{\partial g}(g_k, v_{k+1}) - p_{k+1}^T d^s\tau(d_{k+1}) + p_k^T d^s\tau(d_k).Ad_{d_k} = 0 \in \mathfrak{g}^* \tag{11.8}
$$

Finally, we remark that $d^s\tau(d_k).Ad_{d_k} = d^b\tau(d_k)$, since the adjoint Ad_{d_k} converts from body to spatial velocity, and the range space of τ is a vector space, $\mathfrak{g}$.

Variational Update Equations (11.3), (11.4) and (11.8) may be summarized in the following non-linear system relating $(g_{k+1}, v_{k+1}, p_{k+1}^T)$ to (g_k, v_k, p_k^T):

$$
\frac{\partial \mathcal{L}}{\partial v}(g_k, v_{k+1}) d^s\tau(d_{k+1}) = p_k^T d^b\tau(d_k) + h.\frac{\partial^b \mathcal{L}}{\partial g}(g_k, v_{k+1}) \tag{11.9a}
$$

$$
g_{k+1} = g_k \tau^{-1}(h.v_{k+1}) \tag{11.9b}
$$

$$
p_{k+1}^T = \frac{\partial \mathcal{L}}{\partial v}(q_k, v_{k+1}) \tag{11.9c}
$$

Though this variational update has interesting theoretical properties, its non-linearity is an issue regarding its applicability in a real-time context. Therefore, we propose two approximations to obtain a linear update.

Forcing

External forces can also be incorporated, through an extended variational principle (Pontryagin-d'Alembert principle) which states [KYT+06, KCD09]:

$$\delta \int_0^T \mathcal{L}(g,v) + p^T(\dot{g}^b - v).dt + \int_0^T F(g,\dot{g})^T.\delta g.dt = 0$$

An explicit quadrature for the forcing term is [KYT+06, KCD09]:

$$\int_0^T F(g,\dot{g})^T.\delta g.dt \approx h.\sum_{k=0}^{N} f_k^T.\delta g_k$$

where $f_k^T \in T_{g_k}^* G$ is the force applied at the k^{th} time step. Injecting this term in equation (11.7) gives:

$$h.\frac{\partial^s \mathcal{L}}{\partial g}(g_k, v_{k+1}) + h.f_k^{sT} - p_{k+1}^T d^s\tau(d_{k+1}).Ad_{g_k^{-1}} + p_k^T d^s\tau(d_k) Ad_{g_{k-1}^{-1}} = 0 \tag{11.10}$$

Since $f_k^{sT} Ad_{g_k} = f_k^{bT}$, the final update is then:

$$\frac{\partial \mathcal{L}}{\partial v}(g_k, v_{k+1}) d^s\tau(d_{k+1}) = p_k^T d^b\tau(d_k) + h\left(\frac{\partial^b \mathcal{L}}{\partial g}(g_k, v_{k+1}) + f_k^{bT}\right) \tag{11.11a}$$

$$g_{k+1} = g_k \tau^{-1}(h.v_{k+1}) \tag{11.11b}$$

$$p_{k+1} = \frac{\partial \mathcal{L}}{\partial v}(q_k, v_{k+1}) \tag{11.11c}$$

11.1.3 Approximations for Real-Time Simulation

Two non-linear terms in v_{k+1} complicate the integration update in the left hand-side of equation (11.11a). We propose two approximations to turn the update into a linear system, at the expense of symplecticity.

Quadratic Forces

The computation of so-called quadratic forces [MSZ94] $\frac{\partial^b \mathcal{L}}{\partial g}(g_k, v_{k+1})$, containing Coriolis and centrifugal forces, is problematic since this term is quadratic in v_{k+1}. We propose to approximate this term using the velocities from the previous time step, thus making it completely explicit:

$$\frac{\partial^b \mathcal{L}}{\partial g}(g_k, v_{k+1}) \approx \frac{\partial^b \mathcal{L}}{\partial g}(g_k, v_k)$$

In our simulator, we compute this term using central finite differences.

Difference Map Derivative

The second cause of non-linearity in the above variational update is the presence of $d^s\tau(d_{k+1})$ in the left-hand side of equation (11.11a), and this term is non-linearly coupled with v_{k+1}. To remedy this situation, we propose and compare two approximations to remove this non-linearity.

Small Velocities Since $d_{k+1} = \exp(h.v_{k+1})$, we may consider that $d_{k+1} \approx Id$ for small enough velocity/time-steps. Under this approximation, we have:

$$d^s\tau(d_{k+1}) \approx d^s\tau(Id_G) = Id_{\mathfrak{g}}$$

This approximation makes the left-hand side of equation (11.9a) linear in v_{k+1}. The smaller the time step, the better this approximation. The corresponding explicit velocity update is:

$$\frac{\partial \mathcal{L}}{\partial v}(g_k, v_{k+1}) = p_k^T d^b\tau(d_k) + h\left(\frac{\partial^b \mathcal{L}}{\partial g}(g_k, v_k) + f_k^{bT}\right) \tag{11.12}$$

We will refer to this approximation as the *identity approximation*.

Constant Velocities Another plausible hypothesis is to consider that $d_{k+1} \approx d_k$, which corresponds to a constant velocity. Under this approximation, the velocity update rule becomes:

$$\frac{\partial \mathcal{L}}{\partial v}(g_k, v_{k+1}) d^s\tau(d_k) = p_k^T d^b\tau(d_k) + h\left(\frac{\partial^b \mathcal{L}}{\partial g}(g_k, v_k) + f_k^{bT}\right)$$

Since $d^s\tau(d_k) = d^b\tau(d_k).Ad_{{d_k}^{-1}}$, we have:

$$\frac{\partial \mathcal{L}}{\partial v}(g_k, v_{k+1}) d^b\tau(d_k).Ad_{{d_k}^{-1}} = p_k^T d^b\tau(d_k) + h\left(\frac{\partial^b \mathcal{L}}{\partial g}(g_k, v_k) + f_k^{bT}\right)$$

Reorganizing terms, we obtain the following velocity update:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial v}(g_k, v_{k+1}) = \Bigg[& p_k^T d^b\tau(d_k) \quad + \\ & h\left(\frac{\partial^b \mathcal{L}}{\partial g}(g_k, v_k) + f_k^{bT}\right)\Bigg].Ad_{d_k}.d^b\tau^{-1}(h.v_k)\end{aligned} \tag{11.13}$$

We will refer to this approximation as the *constant approximation*.

Experimental Results

We now compare the two proposed approximations. In order to make the Lie group integration advantages more visible, we also included another integrator into the comparison, with the following form:

$$\frac{\partial \mathcal{L}}{\partial v}(g_k, v_{k+1}) = p_k^T + h\left(\frac{\partial^b \mathcal{L}}{\partial g}(g_k, v_k) + f_k^{bT}\right) \tag{11.14}$$

It is almost identical to the one derived from the identity approximation, except that the momentum is not pulled back in the correct frame. It can be seen as the naive adaptation of an Euclidean geometric integrator, approximated with explicit Coriolis/centrifugal forces for the sake of comparison.

We compare the behavior of these three integrators on the following scenario: the character is in a gravity-less environment, we apply a short impulse on its right hand, along the positive Z axis (*cf.* figure 11.1). We choose a quite large time-step for an explicit integrator: $h = 0.1s$. Figures 11.2 and 11.3 present the time evolution of the Linear and Angular Momentum (AM).

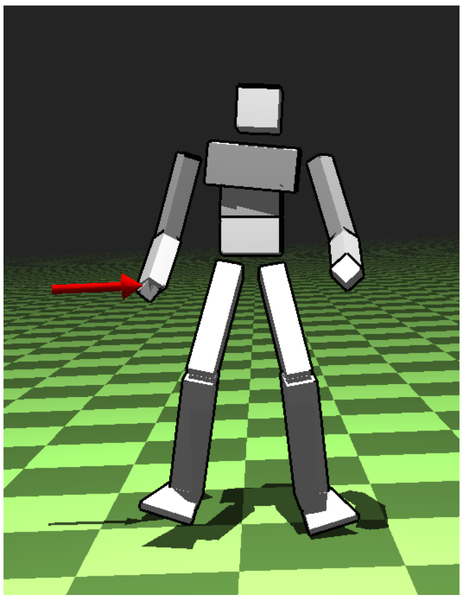
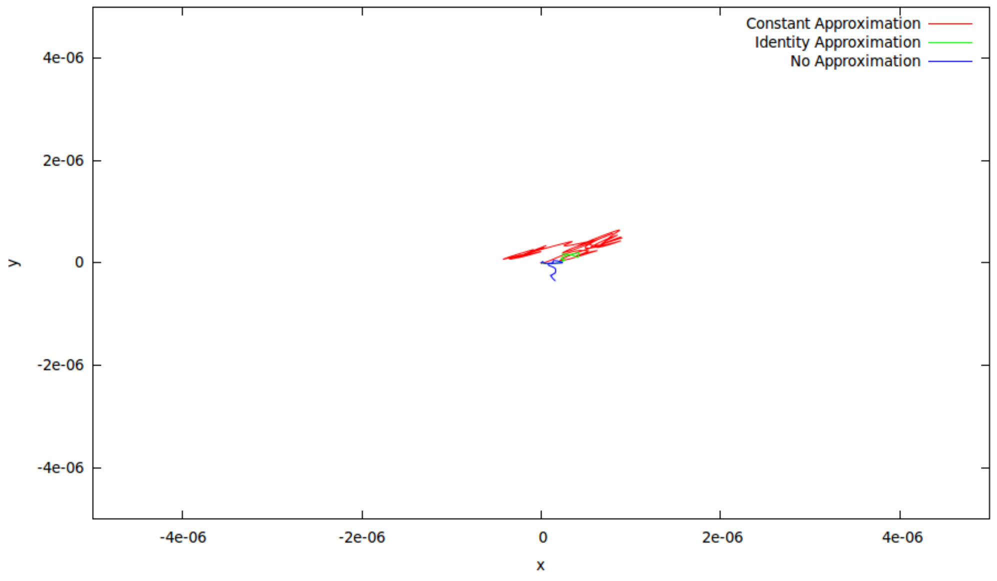


Figure 11.1: The experiment scenario for measuring angular and linear momenta. We apply a short impulse *(red)* on the right hand of the character, without gravity, then measure the linear and angular momenta.

Discussion The AM plots show that the constant approximation has interesting momentum conservation properties, for a slightly higher computational cost due to the computation of $Ad_{d_k}.d^b\tau^{-1}(h.v_k)$. The identity approximation can be seen as a special case of the constant approximation for small



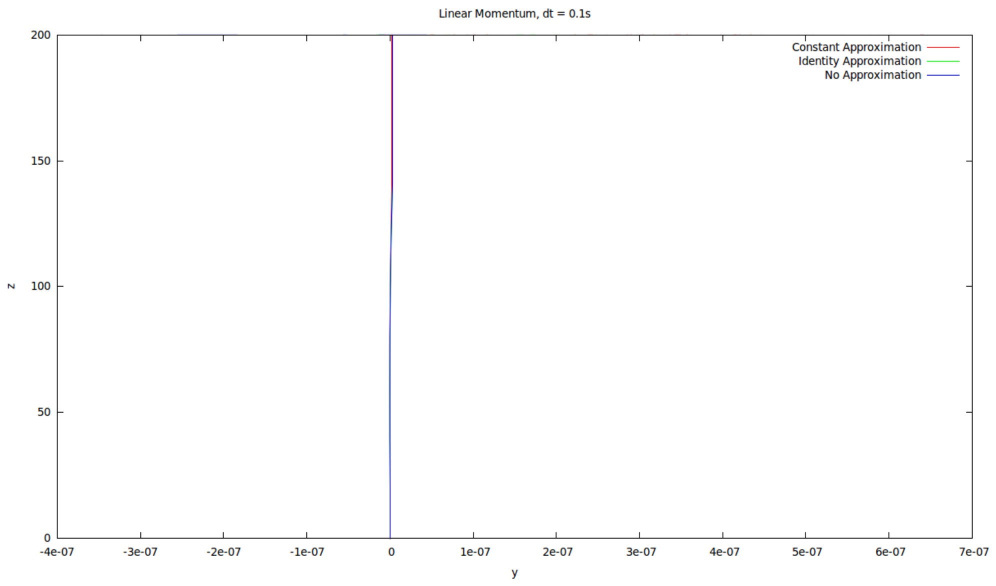
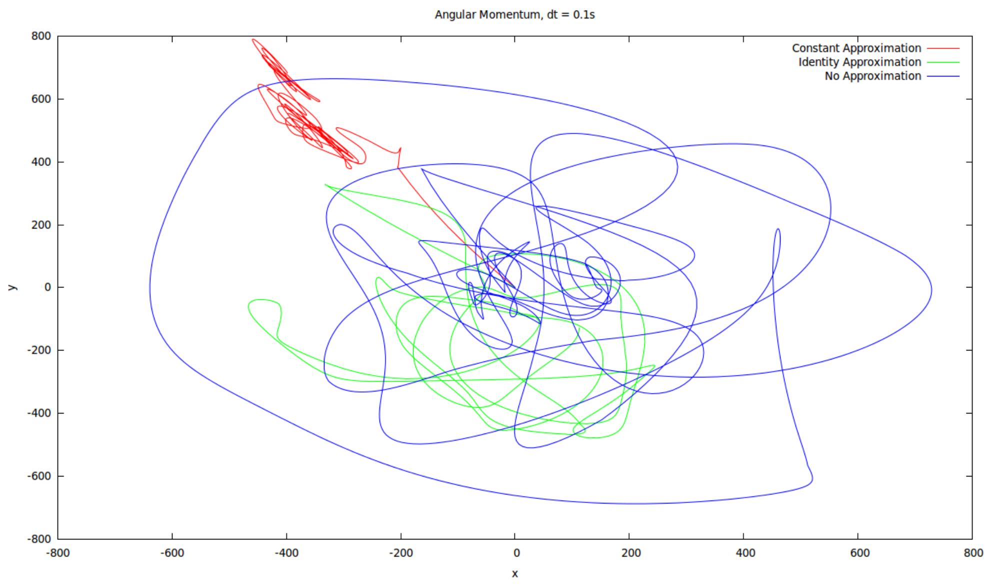


Figure 11.2: Linear momentum time evolution for the constant *(red)*, identity *(green)*, and naive *(blue)* approximations, after the initial z-impulse *(bottom)*. The three integrators nearly preserve the linear momentum. Note the small numerical scale on the top graph.

velocities since in this case, the difference between subsequent velocities will necessarily be small. Therefore, we choose the constant approximation as our explicit integrator for the remaining of this work.



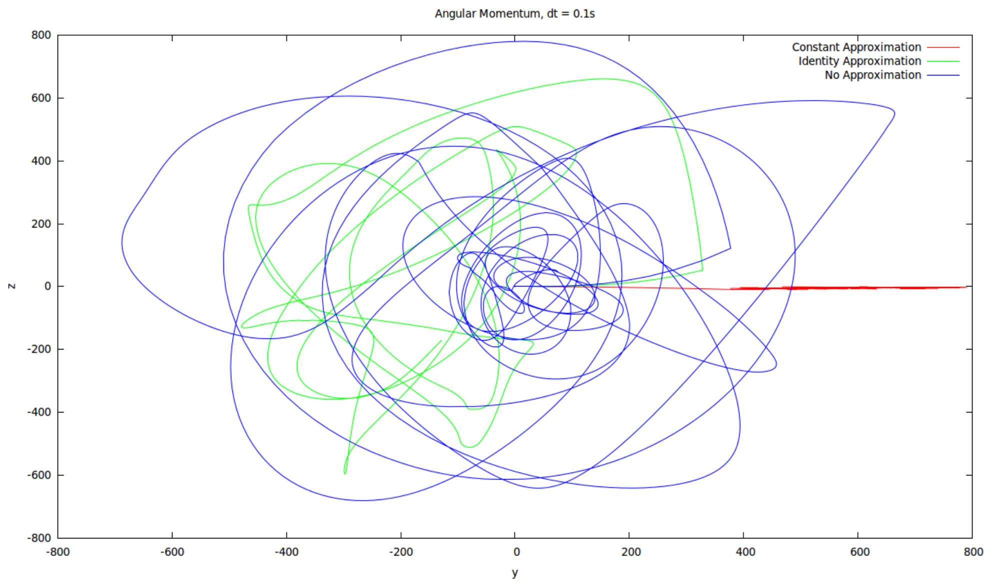


Figure 11.3: Angular momentum time evolution for the constant *(red)*, identity *(green)*, and naive *(blue)* approximations. The constant approximation preserves the AM much better than the other two approximations once the initial impulse has been applied *(top)*. In particular, the symmetries of the system are well captured: almost no AM along Z *(bottom)*. Without much surprise, the naive approximation is the worst.

11.2 Improving Stability

Explicit integrators are inherently subject to stability problems, when the time-step is too large. The Courant–Friedrichs–Lewy (CFL) necessary condition gives an upper bound on the time step to guarantee correct results [PTVF92] that is inversely proportional to the velocity. The stability can be especially poor in the case of a hierarchical representation of an articulated body, due to numerical instabilities of the Jacobian near singularities. This produces high end-effector velocities, thus requiring smaller time steps to ensure stability.

In our case, this problem is even aggravated by the fact that a single geodesic coordinate spans angular velocities on *every* inner joint of the hierarchy. The accumulation of all these angular velocities results in even higher end-effector velocities.

11.2.1 Damping

A common strategy to prevent this problem is to apply damping to the system. Let us once again write the velocity/impulse dynamics as:

$$Mv = f$$

Applying a viscous damping with parameter $\alpha \in \mathbb{R}^+$ is done by introducing the damping impulse $-dt.\alpha.v$:

$$Mv = f - dt.\alpha.v$$

Without loss of generality, we will assume that α already incorporates the time step dt and consider the following damped system instead:

$$(\alpha I + M)v = f$$

The effect of damping on the dynamics is particularly clear when one looks at the eigen-decomposition of $M = UDU^T$, with $UU^T = I$ and D positive diagonal. $M, \alpha I + M$ and $(\alpha I + M)^{-1}$ all share the same eigen-vectors U, and their eigen-decomposition is given by:

$$\begin{aligned} M &= UDU^T \\ \alpha I + M &= U(\alpha I + D)U^T \\ (\alpha I + M)^{-1} &= U(\alpha I + D)^{-1}U^T \end{aligned}$$

We see that damping the mass matrix replaces the response matrix eigenvalues ${d_i}^{-1}$ with $(\alpha + d_i)^{-1}$, reducing response velocities in the case of small

d_i. Keeping in mind that M is a mass tensor, it has the form $M = J^T\widetilde{M}J$ for some block-diagonal rigid-body inertia tensor $\widetilde{M}$ and some Jacobian J. Similarly, since f is a generalized force, it can be expressed as $f = J^T\widetilde{f}$. Written this way, the dynamics equations now strongly resemble the pseudo-inverse IK scheme presented in 6.3 :

$$J^T\widetilde{M}J.v = J^T f$$

Similarly, the damped system is reminiscent of the damped pseudo-inverse IK scheme (again, see part 6.3) :

$$(\alpha I + J^T\widetilde{M}J).v = J^T\widetilde{f}$$

In practice this damping model can effectively correct for some instabilities, but it lacks flexibility: when set too low, instabilities can still happen and when set too high, it induces an almost rigid character behavior. Since this is precisely the reason why the Levenberg-Marquardt algorithm was introduced for non-linear least-squares, we investigate a similar strategy for physically-based animation.

11.2.2 Levenberg-Marquardt Damping

The Levenberg-Marquardt works by successive linearization of a non-linear least-square problem. At each step, the non-linear error function is approximated, to the first order, into a tangent *metric*. This metric approximates the Hessian of the error function and describes its local curvature, which quantifies the confidence in the linear approximation (see 6.3.3 for more details), and drives the applied amount of damping in the corresponding dimension.

A similar strategy can be used for physically-based animation: instabilities are usually characterized by strong end-effector velocities, particularly concerning the leafs of the character topology. Therefore, we can use these end-effectors as an error function to drive the amount of damping applied to each dimension, in a Levenberg-Marquardt fashion: we will apply more damping in the directions of strong end-effector velocities (*i.e.* highly-curved error function) and conversely: this is the *kinematic* damping. Another, even simpler approach, is to simply damp dimensions according to the associated kinetic energy: this is the *energetic* damping. We now present these two approaches in more details. We will denote by $\underline{\mathrm{diag}}$ the operator extracting the diagonal elements of a matrix, except for the root indexes which are set to zero. This operator will be used to damp *internal* DOFs only.

Kinematic Damping

Assuming that a set of end-effector is chosen, we denote by J the associated Jacobian matrix. The *kinematic* damping strategy applies damping according to the magnitude of end-effector velocities. The corresponding dynamics are described by:

$$\Big(dt\ \underline{\mathrm{diag}}\left(\alpha J^T J\right) + M\Big).v = f$$

where $\alpha \in \mathbb{R}^+$ is a user-selected parameter controlling the amount of damping.

Energetic Damping

Instead of using a kinematic metric to control the damping, it is possible to use the kinetic energy metric for this purpose. Intuitively, this corresponds to damping directions not only according to the associated character velocity, but also according to the amount of mass displaced in these directions. The resulting damping scheme is:

$$\Big(dt\ \underline{\mathrm{diag}}\left(\alpha M\right) + M\Big).v = f$$

In practice, this simply reduces to multiplying diagonal elements of M by $(1 + \alpha dt)$, which is computationally less expensive that kinematic damping. We now compare experimental results for both schemes.

Experimental Results

In order to compare the amount of adaptive damping produced by both schemes, we performed an interactive manipulation of the physically-simulated character using kinematic constraints (see next section), and recorded the minimum and maximum damping coefficients generated by both schemes, on a normalized scale. The results are presented on figure 11.4.

We see that while the two schemes produce generally similar results, the maximum damping is higher with energetic damping, producing slightly more stable results. Moreover, computing the energetic damping is simply achieved by scaling the mass matrix diagonal elements, contrary to computing another Jacobian matrix as it is the case with kinematic damping. We found these two damping strategies to be generally more stable than uniform damping during interactive manipulation.

11.3 Kinematic Constraints

We now summarize how kinematic constraints are added to our physical model, and how to compensate for constraint drift using [CP03]. The velocity update in (11.13) can be rewritten as:

$$M.v = f$$

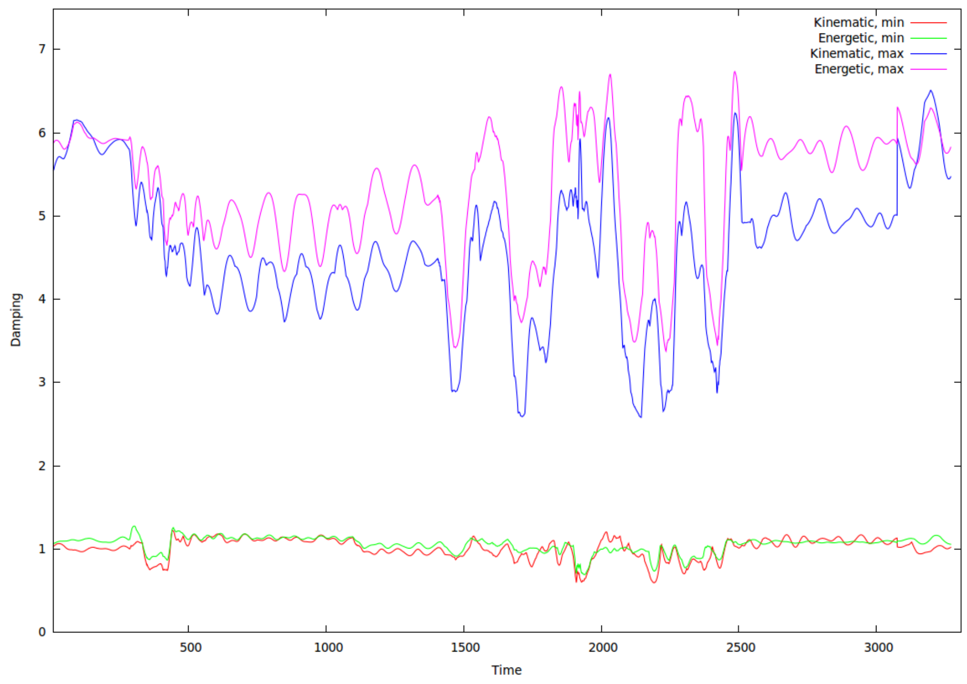


Figure 11.4: Minimum and maximum damping coefficients for the two damping schemes, normalized, during interactive manipulation. Uniform damping corresponds to a constant value of 1.

where v is the body-fixed velocity, M is the reduced mass tensor in body coordinates, and f is the body-fixed net impulse containing the previous momentum with the appropriate frame change, the (now explicit) quadratic forces, and the external forces. If we want to enforce kinematic constraints, being bilateral or unilateral, we must solve the following QP as seen in 10.1.4:

$$v = \underset{v \in \mathcal{C}}{\operatorname{argmin}} \quad \frac{1}{2} v^T M v - f^T v$$

where $\mathcal{C}$ is the feasible set for *body-fixed* velocities. For example, if we consider a *holonomic* constraint $c(g) = 0$ defined by a smooth function $c : G \to \mathbb{R}^m$, and assuming that the constraint is satisfied at the current time step, we need to enforce that c is stationary, that is:

$$d^b c(g).d^b g = 0$$

Strictly speaking, the above linear equation is expressed in the Lie algebra of G (whose elements might be matrices themselves). Using the coordinates in the canonical basis of $\mathfrak{g}$, it is expressed as:

$$J_c^b(g).v = 0$$

where $J_c^b(g)$ is the Jacobian matrix for the left-trivialized tangent $d^b c(g)$. We will express the fact that v represents the coordinates for $d^b g$ as:

$$d^b g \simeq v$$

For practical computations, we will always work in coordinates, since the resulting equations are much more compact. Since our integrator is explicit due to computational requirements, we will also treat the constraints as explicit during constraint resolution, thus the constraint differential is computed at the position from the previous time step (*i.e.* g_k if we are computing v_{k+1}). To summarize:

- Constraints are expressed on body-fixed velocity coordinates (*i.e.* using left-trivialized Jacobian matrices)
- Constraints are expressed at the position of the previous time step

We now give an example of unilateral constraint formulated in this context: a coordinate-invariant joint limits constraint.

Chapter 12

Data-Driven Angular Limits

In this short chapter, we propose an automatic, coordinate-invariant method to learn angular limits from motion capture data. This method is based on Löwner ellipsoids, or Minimum Volume Enclosing Ellipsoids (MVEEs) [KMY03]. We first show how a coordinate-invariant description of angular limits can be obtained using a *tangent* MVEE, and how to compute this tangent ellipsoid. Then, we show how to use this limit formulation to enforce kinematic constraints in a physically-based simulation. We conclude with some results and experimental feedback.

12.1 Related Works

Angular limits for character animation are usually enforced using Euler angles parametrization, by specifying intervals for pitch, yaw and roll angles. This has a number of disadvantages, including the traditional singularities, as noted in [BB01]. Furthermore, it is not clear how to translate these Euler angles limits to other parametrizations of rotation. An ellipsoid-based solution was proposed by [Gra98], where the rotation vector is decomposed along swing and twist components. [BB01] use spherical ellipses to represent admissible regions for swing.

The main drawback of these methods is that they are not *coordinate-invariant*, since they are based on a exponential maps representation of rotations and as such, are subject to singularities unless special treatment is applied (*e.g.* choose an appropriate rotation center for shoulder joints). They also require distinct treatments for revolute/ball-and-socket joints. Finally, they do not provide an automated procedure for obtaining the limits.

[HUH02] proposed an approach conceptually similar to ours, by fitting sphere unions to the quaternion vector part to obtain a tight and automatic fit of rotation subspace 12.1. Unfortunately, their method is not coordinate-invariant since they perform the fit directly on quaternion vector part. More-

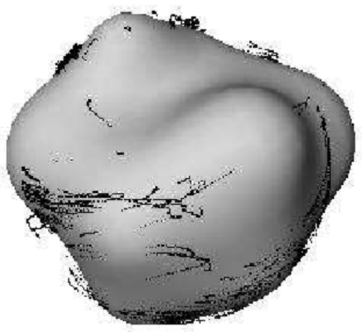


Figure 12.1: Implicit surface fitting to the quaternion vector parts from orientation data [HUH02].

over, they do not provide details regarding the corresponding kinematic constraints. We address all these problems in our proposed method.

12.2 Our Algorithm

Our idea is to use a Minimum Volume Enclosing Ellipsoids (MVEEs) in order to tightly fit a set of orientation samples, and to do so in a coordinate-invariant way. A primitive such as MVEE is attractive due to its compactness, and as we will see in the results, it fits motion capture data quite well. It also allows to formulate angular constraints naturally.

Instead of computing spherical ellipses based on a swing/twist decomposition of exponential maps [BB01], we directly fit a 3-dimensional MVEE to exponential maps. However, performing only this step would result in a coordinate-dependent definition of the mean, since any rotational offset in the data would produce different exponential maps, thus different MVEEs.

Coordinate-Invariance Our idea to enforce coordinate-invariance is inspired by the Fréchet mean algorithm for $SO(3)$ described in 7.3. The intrinsic mean [Pen06, Moa02, FLJ03] algorithm successively linearizes rotations at the current mean estimate, using the exponential map, then compute a linear average, and finally uses the exponential to progress to the next estimate, until convergence. As the limit of this iterative process, the final result no longer depends on any given initial coordinate system, but only on the repartition of the input *data*. We propose a similar procedure with the use of MVEE on linearized data.

Definitions We define a *tangent ellipsoid* as a couple $\varepsilon = (c, A) \in SO(3) \times \mathbb{S}^3_+$, where $c \in SO(3)$ is the rotation defining the center of the tangent ellipsoid, and A is a positive semi-definite matrix defining an ellipsoid in $\mathfrak{so}(3)$ as the set of vectors $v \in \mathfrak{so}(3)$, such as $v^T A v = ||v||^2_A \leq 1$. Another rotation $g \in SO(3)$ will be said *inside* the tangent ellipsoid ε if, and only if:

$$g \in \varepsilon \iff \left|\left|\log\left(c^{-1} g\right)\right|\right|^2_A \leq 1$$

Algorithm Starting with orientation samples $(x_i)_{\leq m} \in G$, and tangent ellipsoid initial guess $c^{(0)} = Id$, we begin by expressing the samples in the tangent space using the exponential map:

$$\forall i \leq m, \quad t_i^{(0)} = \exp\left(c^{-1} x_i\right)$$

We then fit a MVEE $(d^{(0)}, A^{(0)})$ on the tangent data, using the Khachiyan's first-order algorithm, described in [KMY03]. The 3-vector $d^{(0)} \in \mathfrak{so}(3) \simeq \mathbb{R}^3$ is the MVEE center and $A^{(0)} \in \mathbb{S}^3_+$ its metric. The tangent center $d^{(0)}$ is then used to move towards a new linearization point:

$$c^{(1)} = c^{(0)}.\exp\left(d^{(0)}\right)$$

We iterate this scheme until the tangent center $d^{(k)}$ is sufficiently close to the origin, or after a given number of iterations. This procedure is summarized in algorithm 1.

Algorithm 1 Tangent MVEE from m data samples $x_i \in G$

```
$c := Id_G, A := Id_{\mathfrak{g}}$
**repeat**
  {Exponential map of samples around $c$}
  **for** $i := 1$ **to** $m$ **do**
    $t_i \leftarrow \log\left(c^{-1} x_i\right)$
  **end for**
  {Compute tangent MVEE}
  $(d, A) \leftarrow MVEE(t)$
  {Advance $c$}
  $c \leftarrow c.\exp(d)$
**until** $||d|| < \epsilon$
**return** $(c, A)$
```

12.3 Results

We tested this algorithm on several motion capture sequences from the CMU motion capture database, as shown on figures 12.2, 12.3 and 12.4.

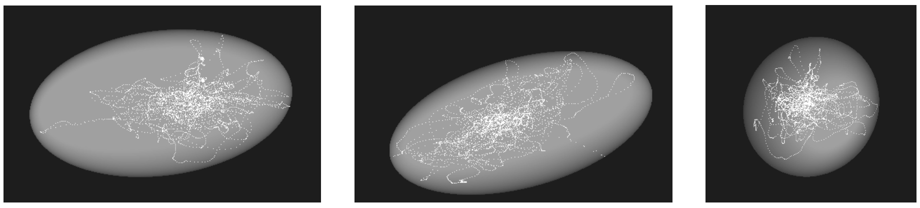


Figure 12.2: Resulting tangent ellipsoids for the left clavicle *(CMU:85-12)*. (x, y, z) axis-aligned views.

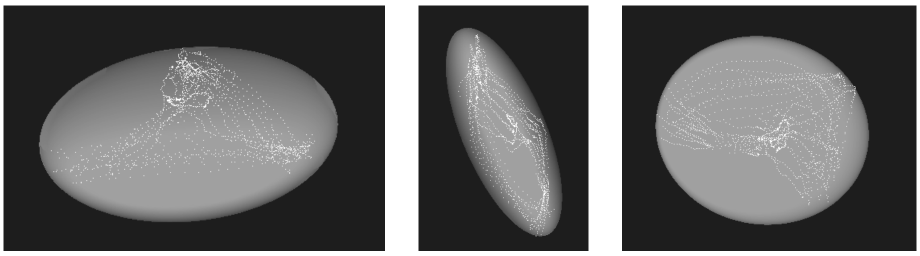


Figure 12.3: Resulting tangent ellipsoids for the right femur *(CMU:91-62)*. (x, y, z) axis-aligned views.

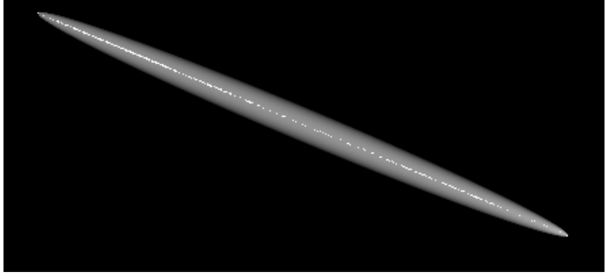


Figure 12.4: Resulting tangent ellipsoids for left radius (one-dimensional) *(CMU:49-06)*. The ellipsoid is displayed thicker to be visible.

12.3.1 Discussion

Convergence

We do not have any proof of convergence of the above algorithm, but we reported experimental convergence even for "range of motion" sequences. We experimentally observed a linear decrease of stopping criterion. As for the intrinsic mean algorithm, it is necessary for the data to lie in some sufficiently localized area of $SO(3)$, otherwise no convergence can be obtained for the same reasons as for the intrinsic mean. However, we have not found this theoretical problem to be a practical issue, as motion capture data are usually sufficiently well localized in practice, except on motions exhibiting capture artifacts.

Outliers

The noise found in low-quality motion capture data can be problematic for the MVEE algorithm, as it can significantly alter the fit of the resulting ellipsoid to the data even if only one outlier is found. In order to tackle this problem, we have found that a simple outlier detection based on the geodesic distance between consecutive orientation samples produced good results. In our tests, we used the threshold $\left|\left|\log({q_k}^{-1}q_{k+1})\right|\right|^2 \geq 1$.

A more systematic approach could be devised using the *core set* of the MVEE, which is a byproduct of [KMY03] algorithm. Intuitively, the core set is the set of samples touching the MVEE. Outliers search could be restricted to this set, *e.g.* using a Malahanobis distance criterion.

Numerical Issues

In the case of perfectly one-dimensional joints, the MVEE will be degenerated which will lead to numerical instability problems, as for example spurious active limit detections. In order to avoid this, we add a small amount of random noise to the input data before applying the algorithm. Note that the dimensionality of joints is already taken into account by the PGA decomposition, so adding noise will only slightly grow the ellipsoids but will not affect the joint degrees of freedom.

Fit Quality

In some cases, the joint orientations lie in a degenerate, bi-dimensional sub-manifold of $SO(3)$ (*cf.* figure 12.5). Should this happen, an ellipsoid fit does not seem particularly well adapted. One should keep in mind, however, that the PGA pose model already takes care of maintaining orientations in this 2D sub-manifold, so that the effective limits enforced in practice are rather given by the *intersection* of this ellipsoid with the 2D sub-manifold.

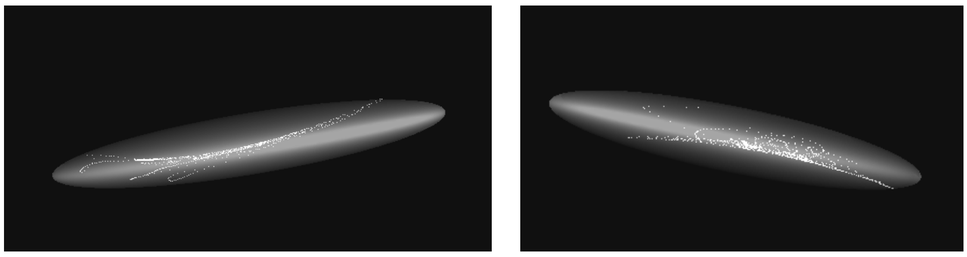


Figure 12.5: An example degenerate 2D rotation sub-manifold resembling an hyperbolic paraboloid, difficultly captured by the MVEE *(CMU:85-01, right foot)*.

12.3.2 Limit Constraints

After the limit computation, each joint is associated with its tangent ellipsoid, as a pair $\varepsilon = (c, A) \in SO(3) \times \mathbb{S}^3_+$ representing the center (in the group) and the metric of the ellipsoid.

Let us introduce the function $f_\varepsilon(q) = \log(c^{-1}q)^T A \log(c^{-1}q) \in \mathbb{R}$. The joint limit will be *active* whenever $f_\varepsilon(q) \geq 1$ and in this case, we should introduce an unilateral velocity constraint preventing the limit to be further violated. Doing so is achieved using the gradient of f_ε. A body-velocity $d^b q$ will be *admissible* when:

$$d^b f_\varepsilon(q).d^b q \leq 0 \tag{12.1}$$

Geometrically, this corresponds to enforcing the velocity to point *inside* the limit ellipsoid, with respect to the tangent plane at the contact point (*cf.* figure 12.6). The body-fixed differential for f_ε is given by:

$$d^b f_\varepsilon(q).d^b q = 2.\log(q)^T.A.d^b \log(q).d^b q$$

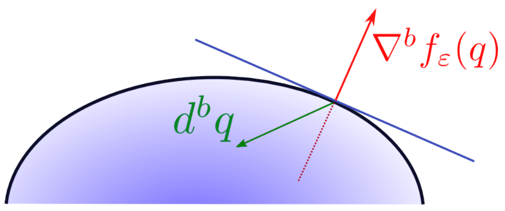


Figure 12.6: The angular limit kinematic constraint enforces that $d^b q$ should point inside the ellipsoid, to the first order: $d^b f_\varepsilon(q).d^b q \leq 0$.

Should the angular limit be violated, due for instance to constraint drift, a post-stabilization approach can be derived from this formula, by replacing the right-hand side of (12.1) by $\frac{1-f_\varepsilon(q)}{dt}$, where dt is the time step. This corresponds to ensuring that $dt.d^b q$ will lie *inside* the ellipsoid at the next time step, to the first order. An example use of these angular limits on a virtual character is presented on figure 12.7.

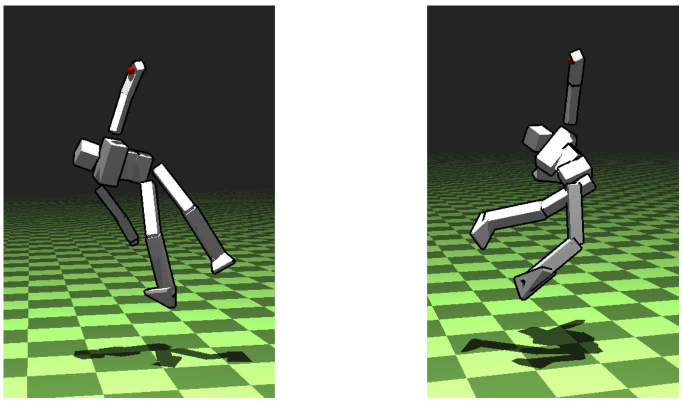


Figure 12.7: Comparison of the physical simulation with *(left)* and without *(right)* enforcing the data-driven joint limits.

Conclusion

In this part, we derived a physical model for our PGA-reduced pose representation. We used a Lagrangian formulation, and proposed an explicit integrator, obtained by approximating a geometric variational integrator. The integrator momentum conservation was evaluated for a large time-step, with promising results. For more stability, we proposed an adaptive, Levenberg-Marquardt inspired damping strategy, that is easily implemented by weighting diagonal matrix elements. Finally, we proposed a geometric, coordinate-invariant, data-driven angular limits learning algorithm, that works by successively fitting MVEE over linearized orientation data.

⋆ ⋆ ⋆

Now that a physically-based animation model has been derived, we turn to the problem of motion control, in order to investigate the relevance of our pose parametrization for actuation.

Part V

Control

Chapter 13

Previous Works

This part is focused on motion control in a physically-based simulation, for character animation. After reviewing existing work, we describe a task-space control framework. Two control solvers are presented in chapter 14, both using Quadratic Programming:

- A simple motion control strategy that simply interpolates between physically-based and kinematic animation, possibly using external forces (section 14.1)
- A more advanced motion control algorithm using only the character actuators to perform motion, (section 14.3)

We conclude this part by presenting control objectives and associated results.

⋆ ⋆ ⋆

Let us begin this part by reviewing existing work in character control. The literature in this field is extensive, especially since it covers both computer animation and robotics problematics. We will organize this chapter along the main approaches followed in computer graphics, pointing to relevant robotics work when needed.

13.1 Proportional-Derivative

A first class of methods for controlling physically-based characters is that of Proportional-Derivative (PD) joint controllers [RH91, HWBO95, FvdPT01, YLvdP07]. In this setup, every joint of the character is affected a *desired* configuration, as well as *proportional* (P) κ_p and *derivative* (D) κ_d gains, so that the actual control *torque* τ applied at the joint is given by the PD feedback loop:

$$\tau = \kappa_p.e + \kappa_d.\dot{e}$$

where e is the error between desired and actual joint configuration. When applied to every joint of a character, this basic controller prescribes a given *desired pose* and a feedback loop to achieve it. The process of controlling a character motion is thus shifted to the control of desired poses for the given task. To drive the sequence of desired poses, a Finite State Machine (FSM) is usually employed, for instance to describe the succession of stances in a walk cycle. [FvdPT01] propose an interesting modular approach on the top of this framework, by automatically identifying correct input/output states for a given controller, allowing higher-level transitions between controllers in a motion-graph spirit. FSMs can serve as a basis for a more complex controller, with the final desired pose being influenced by run-time quantities, for instance to control the swing leg or torso given the walking speed [YLvdP07].

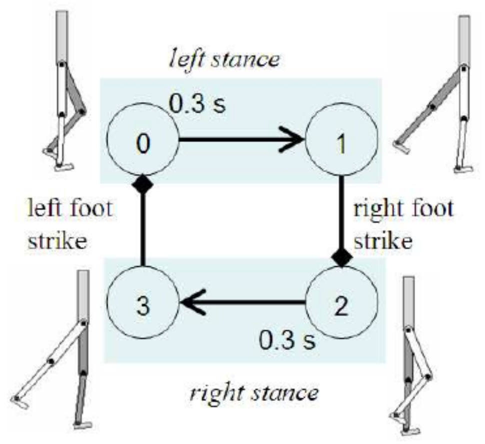


Figure 13.1: A Finite State Machine for walking, in [YLvdP07]. This FSM drives the desired position of joint PD controllers.

Despite their apparent simplicity, these controllers are extremely efficient and able to produce reactive, quality motion controllers. Their main drawback, however, is that they usually require *extensive* tweaking of various gains to achieve both correctness and robustness, and are usually only adapted to a single specific task, making transitions between them difficult. These two drawbacks can be mitigated [SKL07] by automatically learning parameters on motion capture data (or kinematic blending of animations in the case of transitions). Still, these controllers inherently lack flexibility since they are strongly task-specific.

13.2 Space-Time Optimization

This lack of flexibility has led researchers to design more generic approaches to physically-based motion synthesis. Among them, the use of space-time constraints already reviewed in part 7.1.3, allowed to express control as a series of kinematic goals, while allowing to enforce optimal energy criteria to obtain smoother and more plausible results. As we have already seen, the main drawbacks of space-time constraints are:

- Its computational cost, due to the size of the search space
- The *offline-by-design* nature of the algorithm
- Its sensitivity to local extrema, often requiring extensive key-framing

The main attractiveness of these methods are that control is no longer tied to be expressed in the joint space, but any kinematic goal can be specified either as a hard or soft constraint.

13.3 Task-Space Control

Among the most recent works in character animation, several are based on *task-space control* [MZS09, JYL09, LMH10, MLH10]. In this formulation, kinematic goals are formulated in some abstract *task-*, or *feature*-space, and the system solver is responsible for optimizing these goals using physics laws as a constraint *at each time-step*. This approach can be seen as a synthesis of the advantages of joint-space control and space-time optimization while mitigating their drawbacks: instead of solving an enormous, global optimization problem preventing interactivity, a smaller, easier problem is solved at each time-step. Besides, the burden of choosing and adapting individual control gains is shifted from the joint space to an abstract task-space, usually better adapted. Such approaches have been previously used in robotics (see [NCM+08] for an overview). In essence, task-space control algorithms automatically translate task-space goals into character actuation forces, given more-or-less detailed physical model.

[JYL09] only enforce the dynamics equations on the root of the character, allowing real-time control of balance controllers using linear momentum and torso orientation tasks. More complex behaviors such as side steps can be driven using a FSM. The contact model uses previous time-step tangential velocity information to establish static/dynamic frictional behavior (thus making static to dynamic changes impossible). Unfortunately, this method may easily produce physically invalid motions since the Lagrange equations are only enforced on the root joint.

[MZS09] uses linear and angular momentum control strategies to obtain optimal joint accelerations for the task, which are then translated to joint

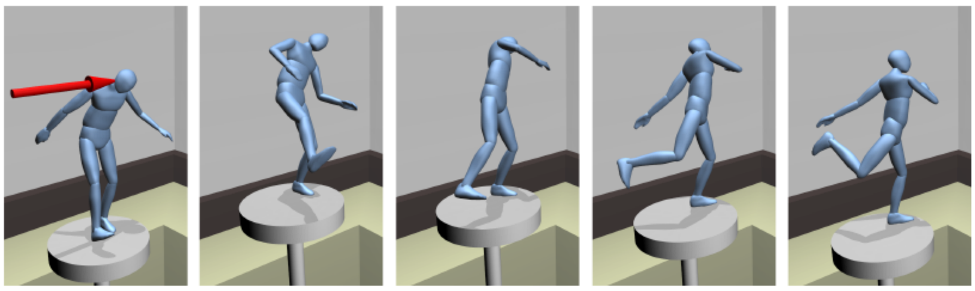


Figure 13.2: Angular momentum regulation, through Center of Pressure control, produces interesting *windmilling* motions *(source: [MZS09])*

torques using Featherstone's inverse dynamics algorithm [Fea87]. The contact model is penalty-based, thus necessitates strong stiffness (and according time-step, the integrator being explicit) in order to maintain realism. The two-pass strategy from task to accelerations, then to torques could probably be reduced to one by directly expressing tasks as a function of torques.

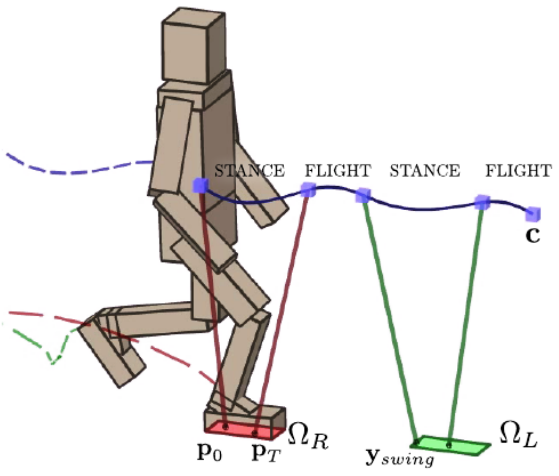


Figure 13.3: Low-dimensional planning by [MLH10]. The planning output is fed to a task-space motion controller that is responsible for computing the corresponding character actuation.

[LMH10, MLH10] propose an interesting framework for task-based character control. The kinematic goals are expressed on accelerations, by forming a quadratic objective function. This objective is optimized under the dynamic equations and approximate contact constraints (no complementarity). A *feature* priority solving strategy is proposed, preventing certain control objectives to compete with each others.

While this approach is the most general and well-grounded to date for character control, it could arguably be improved especially by the use of a velocity/impulse formulation of dynamics and features, removing the need for second derivative which can be difficult to compute. Besides, the contact formulation raises some issues as we show in 14.2. Last, but not least, a reduced dimension motion model could improve the overall efficiency of this method quite significantly. We will address these points in the remaining of this chapter.

Chapter 14

Quadratic Programming Control

In this chapter, we describe a character control framework in the same spirit of [LMH10], but at the velocity/impulse level instead of force/acceleration. First, we show that changing the metric in the physics solver presented in part IV results in a very simple control framework, allowing an easy trade-off between physically-based and kinematic animation. However, such an approach makes implicit use of spurious external forces to animate the character, which is somewhat unsatisfactory: it would be more interesting to *only* use the character actuators to animate it. In section 14.2, we present theoretical aspects of this problem, with an emphasis on unilateral constrains: the quadratic control problem under unilateral constraints can be cast as a *bi-level* quadratic program, which is unfortunately hard to solve because of its non-convexity. In a third section, we propose different relaxations of this control problem in a simpler quadratic program for use in practical applications.

14.1 Changing Metric

We initially tried controlling our virtual character using hard kinematic constraints. However, since hard constraints can not be broken by user interaction, for example in response to a projectile hit, the result was not entirely satisfying in our opinion. In this section, we show how a simple motion controller can be obtained by changing the metric used when simulating the physically-based character, producing easily controllable *soft* kinematic constraints. Let us recall that the dynamics of our character are given, in body coordinates, by the following QP:

$$v^\star = \underset{Jv \geq b}{\operatorname{argmin}} \quad \frac{1}{2} v^T M v - f^T v \tag{14.1}$$

where v describes the coordinates of the body velocity, f contains the coordinates of the external/inertial body forces, M is the body-fixed mass tensor, J and b describe unilateral constraint. We can modify the original quadratic form, using a *control* quadratic form $\frac{1}{2}v^TQv+c^Tv$, in order to obtain a different behavior:

$$v^\star = \underset{Jv\geq b}{\operatorname{argmin}} \quad \underbrace{\frac{1}{2}v^TMv - f^Tv}_{\text{dynamics}} \quad + \quad \underbrace{\frac{1}{2}v^TQv + c^Tv}_{\text{control}} \tag{14.2}$$

The control form may represent any *soft* kinematic constraint one wishes to satisfy on velocities, expressed as a quadratic form. We will see how this form is constructed in 14.1.1. In order to highlight the impact of this metric change on the dynamics, we may rewrite the above quadratic form so that the effects of the control form are more visible:

$$\frac{1}{2}v^TMv - f^Tv \quad + \quad \frac{1}{2}v^TQv + c^Tv = \frac{1}{2}v^TMv + v^T\left(\frac{1}{2}Qv + c - f\right)$$

This underlines the fact that, conceptually, adding such a control form to the dynamic form has the same effect as adding a velocity-dependent *external* control force f_c:

$$f_c = -\left(\frac{1}{2}Qv + c\right)$$

A simple interpolation parameter $\alpha \in [0,1]$ can be employed to drive the amount of control force in the simulation:

$$v^\star = \underset{Jv\geq b}{\operatorname{argmin}} \quad (1-\alpha)\left(\frac{1}{2}v^TMv - f^Tv\right) \quad + \quad \alpha\left(\frac{1}{2}v^TQv + c^Tv\right) \tag{14.3}$$

Setting $\alpha = 0$ will produce a pure dynamic simulation, at the expense of control, while $\alpha = 1$ will produce a pure kinematic animation, at the expense of physical realism. We now describe how to construct the control form in terms of control features.

14.1.1 Control Form, Features

In practice, the control form is the aggregation of several error terms, each one controlling a specific velocity-dependent *feature*. Let $\mathcal{F} \simeq \mathbb{R}^d$ be an abstract *feature space* of dimension d. We define a *feature* γ as an affine map γ

$$\begin{aligned}\gamma &: \mathfrak{g} \to \mathcal{F}\\ \gamma(v) &= \Gamma.v - \widetilde{\gamma}\end{aligned}$$

We call $\Gamma \in \mathcal{M}_{d,n}$ the feature *matrix*, and $\widetilde{\gamma} \in \mathcal{F}$ the feature *desired value*. The squared norm of a feature is a quadratic form, and will be called the feature error *error*. Given a set of $k \in \mathbb{N}$ features $(\gamma_i)_{i \leq k}$, we build a quadratic form by simply summing all the form errors:

$$\frac{1}{2} v^T Q v + c^T v := \sum_{i=1}^{k} ||\gamma_i(v)||^2$$

Features can be given more importance in the final control form by *weighting* them accordingly. Such weight is implicit in the definition of each feature, in the above formula.

14.1.2 Desired Values

While computing the feature *matrix* is usually straightforward (since we normally know *what* we need to control, as we did for kinematic constraints in 11.3), deciding of a relevant *desired value* is not so simple (*i.e. how* we want to control). We illustrate this with an example feature, controlling for example a hand position or the head orientation. Let us define a function between the configuration Lie group G and some other, feature-related Lie group H (in this example, $H = \mathbb{R}^3$ or $H = SO(3)$) by a mapping ϕ:

$$\phi : G \to H$$

This mapping expresses the hand position or the head orientation given the current configuration $g \in G$. The associated control feature, defined on the body velocity coordinates $v \simeq d^b g$, is given by:

$$\gamma_\phi(v) = J^b_\phi(g).v - \dot{\phi}^b_{\text{des}} \quad \in \mathbb{R}^3 \simeq \mathfrak{h}$$

We have the feature matrix as the body Jacobian matrix of ϕ, let us now look for the desired value $\dot{\phi}^b_{\text{des}}$. A simple control strategy is to use the first-order approximation of ϕ to express the feature matrix and desired value, in a Gauss-Newton fashion. Suppose we want to have $\phi(g) = h \in H$ at the next time-step, we obtain the corresponding condition on v as:

$$J^b_\phi(g).dt.v = \log\left(\phi(g)^{-1} h\right) \tag{14.4}$$

This condition is similar to the bilateral kinematic constraints corrections seen in part IV. Unfortunately, while enforcing this condition as a *hard* kinematic constraint would effectively result in $\phi(g) \approx h$ at the next time step, using this feature in the control form will only *minimize* its error, thus producing a *soft* kinematic constraints.

As such, control features may take time before reaching the desired configuration. In practice, this usually results in undesired *oscillations* around the target value for ϕ. A common approach to tackle this issue is to use Proportional-Derivative(PD) control on H to obtain a value for $\dot{\phi}^b_{\text{des}}$. Since

H is a Lie group, we use the PD controllers defined in [BM95], producing a control law of the form:

$$\dot{\phi}^b_{\text{des}} = -\kappa_p.\log\left(h^{-1}\phi(g)\right) \; - \; \kappa_d.J^b_\phi(g).v_t$$

where κ_p, κ_d are respectively the proportional and derivative gains, and v_t is the *current* velocity (*i.e.* not the one we will be solving for in the control problem, which is v_{t+1}). The control law (14.4) corresponds to a PD controller with $\kappa_p = dt^{-1}, \kappa_d = 0$. In practice, the derivative term tends to damp the aforementioned oscillations.

14.1.3 Results

We now present some control results obtained using our system. Figure 14.1 shows a simple user interaction using IK targets, and highlights the use of external forces in our formulation. Figure 14.2 shows an example of one-foot balance and the control of the Center of Mass. Finally, figure 14.3 shows an external perturbation caused by a ball thrown at the character.

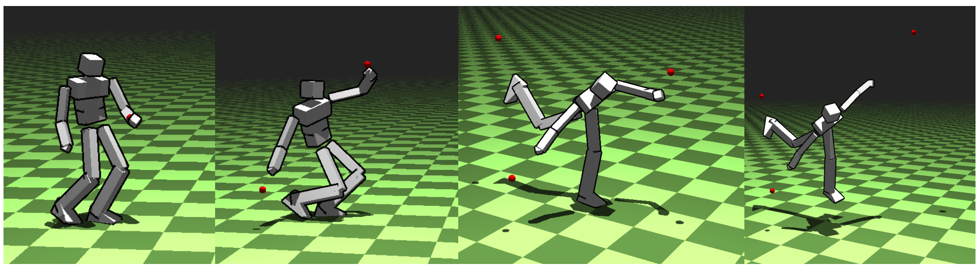


Figure 14.1: User interaction using IK targets. Soft control features allow to obtain a result even when the kinematic objective is infeasible. The use of external forces is made explicit on the last picture.

14.2 Complementarity Constraints

While the pseudo-control framework presented above can produce good visual results, it is more a physically-based *puppetry* approach than a true control strategy, since the character is *externally* actuated. A more satisfying, but also more difficult approach, is to only use the *internal* actuators of the character to achieve a given task expressed using velocity features. Unfortunately, doing this is much more difficult from a theoretical point of view, as we describe in this section.

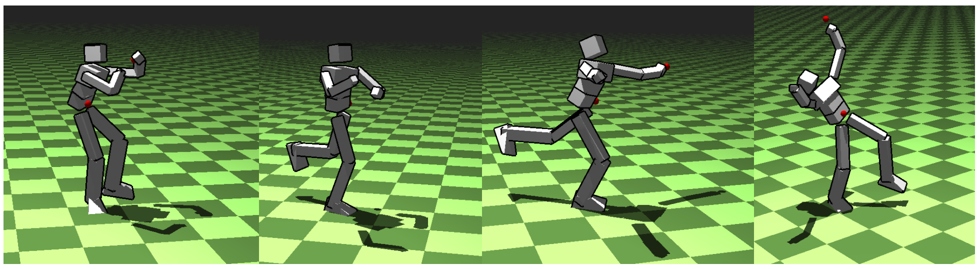


Figure 14.2: One foot balance. 3 control features are present here: right foot, Center of Mass (CoM), and left hand *(red)*. The system automatically adjusts the left leg position to maintain the CoM above the right foot while the user manipulates the left hand position.

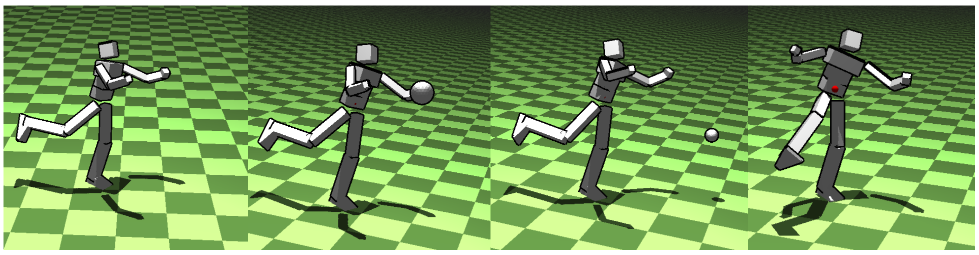


Figure 14.3: One foot balance, with a ball thrown at the character's hand. 2 control features: left foot and CoM *(red)*. Using soft control constraints allows to obtain more natural character responses to external perturbations, as for instance the right leg swing to maintain balance.

14.2.1 Control Problem

Let us first state the control problem we are interested in. We have seen that the dynamics of our virtual character, under unilateral constraints, are given by the convex QP (14.1), which can be seen as a M-projection on the feasible set $Jv \geq b$:

$$v^\star = \pi^M_{Jv \geq b}(f) =: \pi(f)$$

The control problem aims at finding the best *actuation* forces $A^T\mu$, given an (internal) actuation basis A^T, such as a given control form is minimized, under the physical unilateral constraints. This can be expressed as the following, non-convex QP:

$$\mu^\star = \underset{\mu \in \mathcal{A}}{\operatorname{argmin}} \quad \frac{1}{2} v^T Q v + c^T v \tag{14.5}$$

$$\text{s.t.} \quad v = \pi\left(f + A^T \mu\right) \tag{14.6}$$

where $\mathcal{A}$ is a convex, bounded polytope specifying the allowed actuation forces. For a PGA-reduced pose parametrization, with configuration space $Q = SO(3) \times \mathbb{R}^3 \times \mathbb{R}^k$, the actuation matrix A^T describes the generalized forces associated with the reduced pose DOFs, and has the following form:

$$A^T = \begin{pmatrix} 0 \\ Id_k \end{pmatrix} \in \mathcal{M}_{6+k,k}$$

The above QP is known as a *bi-level* QP, since one of the constraints of the *upper* QP (14.5) involves optimality for a second, *lower* QP (14.6). Unfortunately, it is easy to see that in the general case, such bi-level QPs are non-convex. Indeed, the feasible set for the upper QP is the projection of a convex polytope on another convex polytope, which can easily fail to be convex (*cf.* figure 14.4). This unfortunately makes bi-level QP hard to solve.

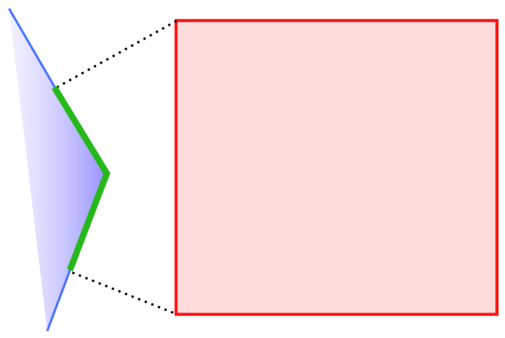


Figure 14.4: The projection *(green)* of a convex *(red)* set on another *(blue)* is generally not convex. In our case, the red area represents the allowed actuation forces, and the blue area represents the unilateral constraints. For the control problem, the red and blue areas are usually intersecting (*i.e.* actuation forces meet the kinematic constraints), but the green area remains non-convex.

Previous Work Bi-level QPs can be seen as QPs with Complementarity Constraints (QPCC), themselves beeing Mathematical Programs with Complementarity Constraints (MPCC). Several strategies have been proposed to solve bi-level QPs [Eto10], QPCCs [BM05] and more generally MPCCs (see [Ani05, FLRS06, SS00]), most of them involving Sequential Quadratic Programming (SQP) or specific relaxations [BM05]. [BM05] recalls that such problems are generally hard to solve due to the combinatorial structure resulting from the complementarity constraints.

14.3 Relaxation

Considering that existing algorithms for QPCC are computationally expensive, we examine and experimentally compare different relaxations of the QPCC (14.5) and (14.6) as a convex QP. While complementarity will no longer be enforced, we show that some relaxations produce satisfying results while still permitting the real-time simulation and control of a virtual character.

14.3.1 Removing Complementarity

Though it is based on a second-order dynamic model rather that a velocity/impulse formulation, [LMH10] propose to express the control problem as a QP similar to the following, putting aside frictional constraints:

$$\begin{aligned} \mu^\star = \operatorname{argmin} \quad & \frac{1}{2} v^T Q v + c^T v \\ \text{s.t.} \quad & Mv = f + J^T \lambda + A^T \mu \\ & Jv \geq b, \quad \lambda \geq 0 \\ & \mu \in \mathcal{A} \end{aligned} \tag{14.7}$$

In essence, this formulation is a relaxation of the full control QPCC by omitting the complementarity constraint resulting from the optimality for the lower QP:

$$0 \leq Jv - b \quad \perp \quad \lambda \geq 0$$

This complementarity condition guarantees that contact impulse are *reaction* impulses: no impulse is applied when the relative normal velocity is non-zero, and conversely. Thus, removing the complementarity constraint produces an undesirable artifact: nothing prevents contact forces to take absurdly high values in order to minimize the control form. In practice, the result is that the ground has a tendency to push the character above when trying to control the center of mass, as depicted in figure 14.5.

This problem also happens with angular limits, which can produce strong impulsive motions, should an angular limit become active. Such strong impulses can degrade the simulation stability. A workaround for this problem is to add a control objective penalizing relative tangent motion, using the following feature:

$$\gamma_J(v) = Jv - b$$

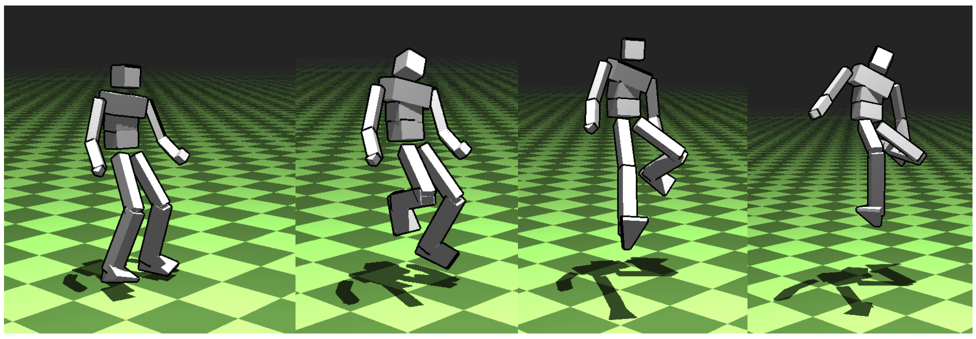


Figure 14.5: Spurious ground contact impulses pushing the character up, when the complementarity condition is removed.

This corresponds to a soft kinematic constraint turning unilateral constraints into bilateral constraints. Unfortunately, while this method produces good results for ground contacts, it greatly penalizes possible motions when applied to angular limit constraints, since these limits become "sticky" once active.

While it is possible to adjust contact stickiness according to their nature (ground, limits) by tuning the associated weights, this approach is tedious and context-dependent. Therefore, we investigated different complementarity relaxation strategies.

14.3.2 Kinetic Energy

We have seen in (14.6) that unilateral constraints minimize the correction kinetic energy, assuming μ is fixed:

$$v^\star = \underset{Jv \geq b}{\text{argmin}} \quad \left|\left|v - M^{-1}\left(f + A^T\mu\right)\right|\right|_M^2$$

Therefore, we tried to add this energy term to the control QP, as a relaxation of the complementarity conditions:

$$\begin{aligned} \mu^\star = \text{argmin} \quad & \frac{1}{2}v^TQv + c^Tv \quad + \quad \frac{1}{2}\left|\left|v - M^{-1}\left(f + A^T\mu\right)\right|\right|_M^2 \\ \text{s.t.} \quad & Mv = f + J^T\lambda + A^T\mu \\ & Jv \geq b, \quad \lambda \geq 0 \\ & \mu \in \mathcal{A} \end{aligned}$$

This QP can be rewritten in the following, more compact expression:

$$\begin{aligned}\mu^\star = \operatorname{argmin} \quad & \frac{1}{2}v^TQv + c^Tv \quad + \quad \frac{1}{2}\lambda^TJM^{-1}J^T\lambda \\ \text{s.t.} \quad & Mv = f + J^T\lambda + A^T\mu \\ & Jv \geq b, \quad \lambda \geq 0 \\ & \mu \in \mathcal{A}\end{aligned} \tag{14.8}$$

As we can see, this amounts to penalizing contact forces according to the inverse kinetic metric. We tried the above relaxation, but it resulted in overly large character motions (*cf.* figure 14.6).

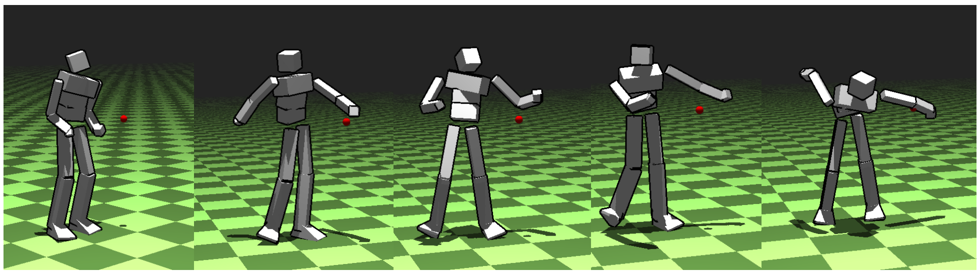


Figure 14.6: The effects of the kinetic energy relaxation: the motions are overly large in compensation for the smaller contact forces

However, if we expand the relaxation term $\frac{1}{2}\left|\left|v - M^{-1}\left(f + A^T\mu\right)\right|\right|_M^2$, we observe that it contains the following term:

$$-v^T.A^T\mu$$

This term is equal to the negative actuator forces instantaneous work. Minimizing it corresponds to maximizing the amount of instantaneous work produced by actuator forces, which introduces a lot of kinetic energy into the system. We tried to remove it from the relaxation, leaving it as (omitting constant terms):

$$\frac{1}{2}v^TMv + \frac{1}{2}\mu^TAM^{-1}A^T\mu - f^Tv + f^TM^{-1}A^T\mu \tag{14.9}$$

Note that this new relaxation term is still a convex quadratic form. Surprisingly, this simple modification produced far better results, since we were able to control the character as well as with the approach described in 14.3.1. Furthermore, the resulting motions were much smoother, and generally looked more natural. However, we had to increase the weight of control features or, equivalently, decrease the weight of the relaxation term slightly, in order to obtain similar objectives satisfaction, since the relaxation term competes with the control form. We used this relaxation strategy in our examples.

If more dissipation is needed, it is possible to add the following term to the relaxation objective (14.9):

$$\alpha.v^T A^T \mu$$

where $\alpha \in [0,1]$ is the amount of dissipation to add. This corresponds to minimizing the amount of instantaneous work performed by actuator forces, hence maximizing the dissipation. Note that α needs to stay in the $[0,1]$ range for the relaxation objective to remain convex. Applying dissipation results in slower, damped motions.

14.3.3 Control and Dynamics QPs

The two preceding relaxations techniques produce actuation forces, based on a simplification of the exact control problem. These actuation forces are then used in a dynamics simulation, where exact contact force complementarity is enforced. In practice, this means we must solve two consecutive QPs:

- The *control* QP, where actuation variables $\mu^\star$ are computed, based on a relaxation of the exact control problem, as described in 14.3.1 and 14.3.2
- The *dynamics* QP, where the computed actuation $\mu^\star$ is used to compute the dynamics of the character, this time with correct contact complementarity (*cf.* 10.1.4):

$$v^\star = \underset{Jv \geq b}{\operatorname{argmin}} \quad \frac{1}{2} v^T M v - \left(f + A^T \mu^\star\right)^T v$$

The time integration is performed using $v^\star$. Note that the control QP also produces velocity variables. While using these velocities for time integration generally results in seemingly more robust controllers, they are based on inexact contact forces, and as such do not reflect the real efficiency or robustness of the controller. However, they do produce visually satisfying animations.

In all the shown examples, the animations were produced using the dynamics QP. Let us now describe the control features we used in our experiments.

Chapter 15

Control Features

In this chapter, we describe the different features used in our simulator. We first present simple kinematic features, for performing IK and looking at a target. Then, features related to character balance are presented. Finally, we describe *energy-related* quadratic terms controlling character actuation. In the remaining of this chapter, we will denote the use of a Lie group PD controller (described in 14.1.2) by a function pd, defined on the *feature* Lie group H:

$$\begin{array}{rcl} \mathrm{pd} : H \times TH & \to & \mathfrak{h} \\ (h^\star, dh) & \mapsto & -\kappa_p . \log\left(h^{\star -1}.h\right) \; - \; \kappa_d . d^b h \end{array}$$

where $\kappa_p, \kappa_d \in \mathbb{R}$ are respectively the proportional and derivative gains, $d^b h$ is the current feature state, and $h^\star$ the target value for the controller. In our framework, the output of the function pd will be chosen as the corresponding feature desired value.

We now describe the different control feature we used in our examples. In the following, we assume that the character pose is parametrized using the PGA kinematics model described in 8.2.2, with configuration space $Q = SE(3) \times \mathbb{R}^k$, where $k \in \mathbb{N}$ is the number of principal geodesics. Features will be expressed in function of the body velocity $v \in \mathfrak{q}$.

15.1 Tracking

We begin this chapter with simple tracking tasks: Inverse Kinematics and looking at a point target.

15.1.1 Inverse Kinematics

The case of IK is the direct application of the example presented in 14.1.2. We simply express the forward kinematics for a point belonging to the skeleton as a function $f : Q \to \mathbb{R}^3$ expressing the *absolute* position of this point in the world frame. The feature geometry is simply given by $J_f^b(g)$, hence the complete feature is defined as:

$$\gamma_{IK}(v) = J_f^b(g).v \; - \; \mathrm{pd}\left(p^\star, \dot{f}_{\text{curr}}\right) \in \mathbb{R}^3$$

where $p^\star \in \mathbb{R}^3$ is the target end-effector position, and $\dot{f}_{\text{curr}}$ is the current time-derivative of the end-effector position. An end-effector tracking example is shown on figure 15.1.

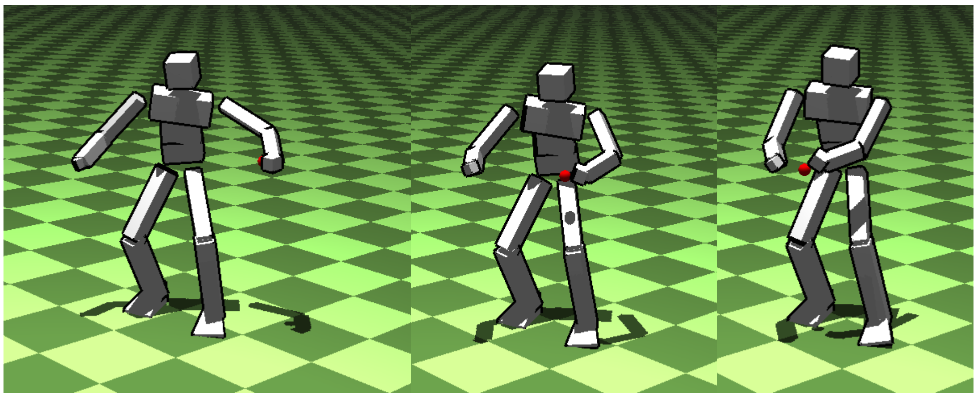


Figure 15.1: Simple IK control feature. The desired position *(red)* is interactively manipulated by the user while the character maintains balance.

15.1.2 Looking at Target

Due to the full-body correlations captured by the PGA pose model, we have found the head rotational motion to be sometimes too important and thus penalizing for visual quality. We thus implemented the following target tracking feature in order to gain an intuitive control on the head orientation.

Yet again, we simply express the *absolute* head orientation (in the world frame) as a function $f : Q \to SO(3)$. The target value for the orientation is computed by building an orthogonal basis from the up $u \in \mathbb{R}^3$ (pointing upwards) and view $w \in \mathbb{R}^3$ (pointing in the view direction) vectors of the character, expressed in the world frame:

- The first basis vector e_1 is simply the view vector w, constructed as the difference between the view target t and the head position h:

$$e_1 \sim t - h = w$$

- The second basis vector e_2 is the projection of the up vector u on the plane orthogonal to view, normalized:

$$e_2 \sim \pi_{w^\perp}(u)$$

- The third basis vector e_3 is obtained by taking the cross-product of the two first vectors, finalizing the basis:

$$e_3 = e_1 \times e_2$$

Such construction is obviously ill-defined when the view and up vectors coincide. Should this situation happen, we simply skip this feature in the final control form. The above construction results in a target rotation $g^\star \in SO(3)$. The head orientation control feature is thus:

$$\gamma_{\text{look}}(v) = J_f^b(g).v \; - \; \text{pd}\left(g^\star, \dot{f}_{\text{curr}}\right)$$

Figure 15.2 shows an example use of this feature control.

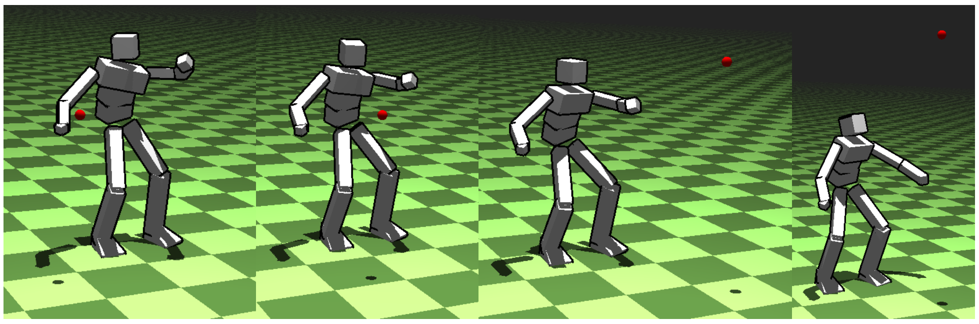


Figure 15.2: An example of head orientation control: the character is looking at the red target.

Though it is not strictly needed for controlling the head in our framework, we now mention how to obtain the derivative of the mapping relating the view vector y and corresponding orientation $g^\star$. This mapping can be used in the case of a moving target. $g^\star$ satisfies the following equations:

$$\pi_{e_y,e_z}\left(g^{-1}.w\right) = 0 \in \mathbb{R}^2 \tag{15.1}$$

$$e_z^T\left(g^{-1}.u\right) = 0 \in \mathbb{R} \tag{15.2}$$

where $e_x = (1,0,0)^T$, $e_y = (0,1,0)^T$, $e_z = (0,0,1)^T$ are the basis vectors expressed in the local frame, and $\pi_{e_y,e_z} = (e_y, e_z)^T$ is the orthogonal projection on e_y, e_z. Since $g^\star$ is a stationary point of the above equations, we obtain a condition on $d^b g^\star$ in terms of dw by differentiation of these equations:

$$\pi_{e_y,e_z}\left(-d^b g.g^{-1}.w + g^{-1}.dw\right) = 0 \in \mathbb{R}^2 \tag{15.3}$$

$$e_z^T\left(-d^b g.g^{-1}.u\right) = 0 \in \mathbb{R} \tag{15.4}$$

Keeping in mind that $d^b g$ represents an antisymmetric matrix, and that $\widehat{d^b g}.w = d^b g \times w$ (*cf.* 6.1.1), we may rewrite these equations using cross-products and the coordinates of $d^b g$ in the canonical basis of $\mathfrak{so}(3)$ as:

$$\pi_{e_y,e_z}\left(-e_x \times d^b g + g^{-1}.dw\right) = 0 \in \mathbb{R}^2 \tag{15.5}$$

$$-e_z^T\left(\left(g^{-1}.u\right) \times d^b g\right) = 0 \tag{15.6}$$

In the above equations, $d^b g$ is now a 3-vector. This linear system of equations relates the body-fixed velocity of the desired orientation, to the velocity of the view vector. We rewrite it as:

$$A.d^b g = B.dw \in \mathbb{R}^3$$

where A, B are 3×3 matrices, with A invertible unless the view and up vectors coincide. Given the view vector velocity dw, we thus obtain the desired orientation velocity by:

$$d^b g^\star = A^{-1}.b(dw)$$

15.2 Balance

Let us now turn to balance-related feature control. As shown in [MZS09], the linear and angular momentum are two high-level quantities of particular interest in this context.

15.2.1 Center of Mass

The Center of Mass (CoM) is given by CoM : $Q \to \mathbb{R}^3$:

$$\text{CoM}(q) = \sum_{i=1}^{n} m_i.\text{CoM}_i(q)$$

where $n \in \mathbb{N}$ is the number of rigid bodies, and $\text{CoM}_i : Q \to \mathbb{R}^3$ gives the position of the CoM for the ith body, and m_i is the mass for the ith body. Its time-derivative is called the *linear momentum*, noted L. The corresponding feature for CoM control follows immediately from the above definition.

15.2.2 Angular Momentum

Angular Momentum (AM) is related to the control of the Center of Pressure (CoP), as shown in [MZS09]. However, the connection between them relies on certain hypothesis (*e.g.* planar ground). In a feature-based control framework, it is possible to *directly* control the AM [LMH10], which is more general and better adapted than CoP control. The AM with respect to the CoM $c \in \mathbb{R}^3$ of n rigid bodies is defined as:

$$a = \sum_{i=1}^{n} R_i.\mathcal{I}_i.\omega_i^b \; + \; \sum_{i=1}^{n} m_i(c_i - c) \times v_i \quad \in \mathbb{R}^3$$

where $R_i \in SO(3)$ is the absolute orientation of the ith body, $\mathcal{I}_i \in \mathbb{S}_+^3$ is the body-fixed inertia tensor of the ith bone, $c_i \in \mathbb{R}^3$ is the position of the ith CoM, ω_i^b is the ith body-fixed angular velocity, and $v_i \in \mathbb{R}^3$ is the absolute linear velocity of the ith CoM in the world frame. The AM is linear in the body velocities, thus we may rewrite it as:

$$a = H.v$$

where H is a $3 \times 6n$ matrix depending on the current configuration, and v is the aggregation of the n rigid body velocities coordinates. If we now consider the forward kinematics for the PGA-reduced pose model (described in 8.2.2) as a function:

$$f : Q \to SE(3)^n$$

The AM feature matrix is given by $H(f(q)).J_f^b(q)$. We set the desired AM value to 0, corresponding to a motion with the least AM. The final feature is thus simply:

$$\gamma_{\text{AM}}(v) = H(f(q)).J_f^b(q).v$$

Figures 15.3 and 15.4 compare the resulting behaviors when the AM control is turned off and on.

15.2.3 Support Center

Instead of fully controlling the CoM position, it is usually sufficient to keep its projection on the ground *inside* the support polygon [JYL09, LMH10]. To do so, we record the position of the character contacts with the ground at each time step. We define the center of support $c \in \mathbb{R}^3$ as the mean of these contact points (*cf.* figure 15.5).

A control feature is added so that the CoM projections on the ground match the center of support.

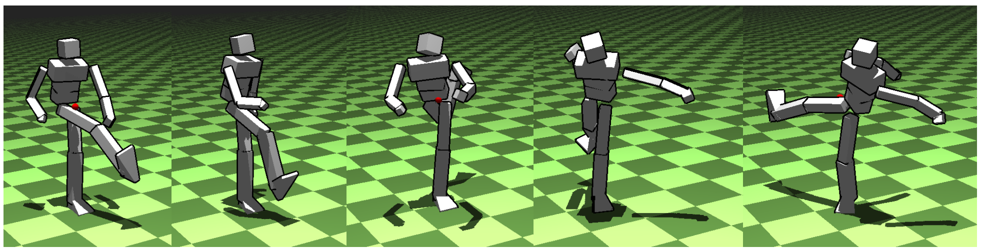


Figure 15.3: One foot balance, without AM control: the character makes broad moves and quickly falls down. The red marker shows the desired CoM position.

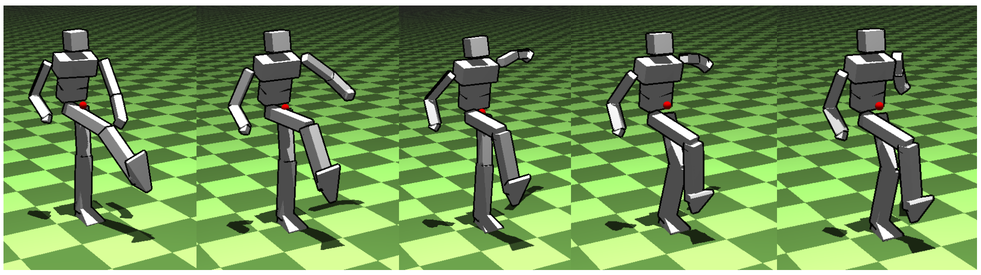


Figure 15.4: One foot balance, with AM control: the character adjusts his arms and moves in a balanced way. The red marker shows the desired CoM position.

15.2.4 Balance Controller

Our balance controller used the following, previously described feature set:

- Head and CoM projections should be at the support center
- Feet should be on the ground
- Angular momentum should be minimal
- CoM height

Figure 15.6 shows an example of one-foot balance control.

15.3 Actuation

We now give two control features on the *actuation*, whereas previous features were defined for velocity. These actuation objectives are used to control how

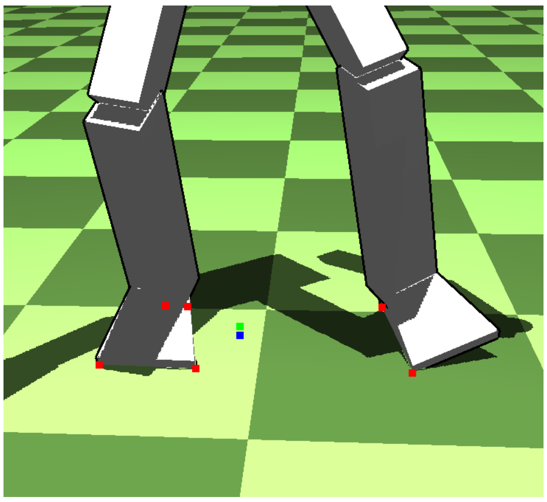


Figure 15.5: The contact points *(red)* are used to define the support center *(blue)* as their mean. The CoM projection *(green)* is controlled by setting its desired value to the blue point.

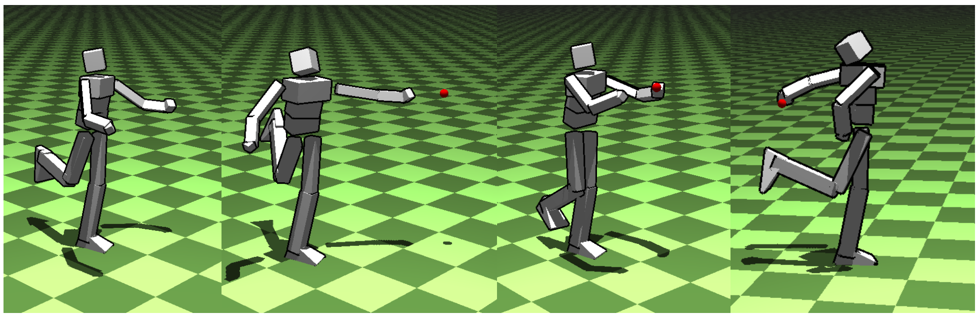


Figure 15.6: One foot balance, while tracking an IK target.

the character should use its actuator forces to move. The dissipation term described for the relaxation term (14.9) can also be used.

15.3.1 Strength

Though we placed convex limits on the actuation through the constraint $\mu \in \mathcal{A}$ in (14.7) and (14.8), we place an additional objective penalizing strong actuator forces. Instead of using the canonical metric on the actuators, by adding an optimization term $||A^T \mu||^2$ to the QP, we found that the associated kinetic energy resulted in a better measure of the energy associated to perform one motion. This metric is defined as:

$$||M^{-1}A^T\mu||_M^2 = ||A^T\mu||_{M^{-1}}^2 = \mu^T AM^{-1}A^T\mu$$

Adding this term also prevents the character from changing actuation too abruptly.

15.3.2 Laziness

In addition to controlling the amount of force exerted by the character, we also propose to favor forces that produce the least instantaneous work, so that the character exploits both its structure and the motion to minimize the energy expenditure.

Unfortunately, the objective term $\left(v^T.A^T\mu\right)^2$ is no longer quadratic and can therefore not be used in this framework. However, if we use the previous velocity instead, we obtain a quadratic form in μ:

$$\left(v_t^T.A^T\mu\right)^2 = \mu^T Av_t.v_t^T A^T\mu$$

Intuitively, this amounts to considering that the current actuation plan is based on the previous velocity to minimize actuation work. Unfortunately, as this objective grows as the *square* of the actuation forces work, we have found that it often becomes predominant in the optimization, producing large smooth limb motions that eventually cause the character to fall.

Conclusion

In this part, we presented two Quadratic Programming control frameworks exploiting our reduced pose parametrization:

- A simple pseudo-control approach that linearly blends the dynamics and control forms
- A more advanced control framework making use of the character actuators only

The first approach is conceptually simpler, but uses external forces to control the character. It allows to easily add small dynamic effects to the kinematic manipulation of a virtual character, but is somewhat limited to physically-based puppetry.

On the contrary, the second approach only exploits the generalized forces arising from the PGA-reduced pose model. We proposed a complementarity relaxation strategy producing smoother motions, together with an optional dissipation term. We presented simple balance controllers with good stability results.

Efficiency As expected, the dimension reduction offered by the PGA allows to improve efficiency, since all the examples we proposed run in full real-time, at around 100Hz on a quad-core machine. We believe these performances to be improvable by engineering a better optimized code.

Dimensionality Compared to the compression algorithm presented in chapter 9, we found that a higher number of principal geodesics are usually needed to obtain satisfying animations, which seems logical since the situations encountered are no longer strictly found in the input motion capture data. In our experience, between 15 and 20 geodesics generally produce good results, at least for balance control.

Actuator Bounds While setting actuator bounds for an unreduced pose parametrization can be done based on bio-mechanic studies, setting reasonable actuation bounds for the PGA-reduced actuators is much more difficult. In our experiments, we simply applied box constraints using empirically chosen values, obtained by successively increasing these bounds until the character can remain balanced. We also did not investigate the preference of certain actuators based on the corresponding eigenvalue in the PGA decomposition.

Robustness Though the balance controllers we presented performed generally well under user interaction, summing control terms in an un-prioritized way will necessarily be subject to robustness problem in the case of a user insisting too much.

Unfortunately, we did not implement a prioritized optimization strategy as found in [LMH10], though it could have largely benefited from our reduced dimension strategy. This kind of framework produces much more robust controllers than simply summing objective terms, as the core balance objectives can not be perturbed by user IK requests. This could also improve the robustness of our controllers to impulsive perturbations.

Friction We used a crude approximation of ground contacts by using only unilateral constraints, which corresponds to an infinite Coulomb friction coefficient. However, even under this simplification, the complete control problem is already non-convex and thus difficult to solve. It would be interesting to see if a relaxed, frictional control problem can be formulated, using an alternate projection algorithm similar to [KSJP08].

Complementarity The biggest remaining issue is that incorporating non-convex complementarity constraints in this framework is impossible. We tried several relaxation strategies, but none of them produced entirely satisfying results: adding our relaxation term attenuates the problem and produces smoother animations, but at the expense of satisfying the control objectives. Without hard complementarity constraints, there is simply no way of preventing contact forces to participate *directly* in the control objective minimization, contrary to how true reaction forces should behave.

In practice, this results in spurious contact forces that alter the quality of the control actuation forces. The most effective strategy to tackle this issue is to add a control objective penalizing normal velocities. This amounts to treating hard unilateral constraints as soft bilateral constraints, but it can lead to "sticky" angular limits. Generally, using this technique imposes that the contact scenarios must be known in advance.

Part VI

Conclusion

Chapter 16

Conclusion

In this work, we presented a new representation for human pose data, enabling dimension reduction by learning on existing motion capture data. When looking for a statistical analysis method suitable for rotational data, we were led to consider the use of the Principal Geodesic Analysis algorithm instead of other complex non-linear algorithm, as it both provides a sound geometric foundation, and remains tractable in practice. This choice turned out to be interesting at many levels:

- The PGA is a natural, coordinate-invariant extension of the well-known linear PCA to certain non-linear manifolds.
- The learning data are remarkably compact compared to other non-linear statistic analysis tools, by taking the geometry of the data manifold into account.
- The resulting sub-manifold parametrization comes as a smooth function with simple derivatives, allowing its use within a large class of character animation techniques.

We evaluated the relevance of this dimension reduction technique through a motion compression algorithm, which enables high compression ratios despite its simplicity. The experience acquired with the geometry of rotations led us to propose a simple geometric method to approximate the joint angular limits for a motion capture sequence. Combined with the PGA-reduced pose model, the resulting constraints effectively prevent unnatural poses from being reached. Experimenting with the resulting PGA-based Inverse Kinematics naturally made us wonder how to incorporate dynamics into this model, in order to improve the quality of resulting animations.

⋆ ⋆ ⋆

Switching to the physically-based animation was not an easy change: the depth and the complexity of the subject are sometimes frightening. Kinematic constraints and contacting system are an especially difficult topic, thus we had to acquire knowledge in convex optimization and quadratic programming.

After discovering the Lie group theory through the PGA algorithm, we became interested in geometric integration strategies, since they provide sound theoretic basis for physically-based animation with Lie groups. While the first contact is a little difficult, it turned out that the resulting first-order equations simplified matters considerably, by eliminating the need for second derivatives computations or approximations.

Keeping the algorithms suitable for real-time use was a driving goal from the beginning, therefore we proposed an explicit time integration scheme, based on a reasonable approximation of a geometric integrator. Together with our dedicated damping model, it allows one to use quite large time-steps, while keeping the simulation stable.

At this point,we realized that despite their repulsive first impression, Lie group formulations were actually a gain of speed regarding software development, as they allow to factor most of the treatments under the same conceptual framework. In fact, all the algorithms we programmed, from spline interpolation to time-integration, are expressed in this framework, which favors modular software development.

⋆ ⋆ ⋆

Once the dynamics were incorporated into the model, the next step was motion control. Task-space control seemed the most promising and flexible approach, especially regarding the potential benefits offered by our reduced model. Here again, the use of a velocity-level formulation for dynamics allowed to greatly simplify the computations involved in control features.

By considering the analogy between classical IK optimization schemes and kinematic constraints resolution, we proposed a simple pseudo-control framework that interpolates between these two behaviors, allowing for much higher-level control than hard kinematic constraints only. However, using external forces for character control reduces to a form of virtual puppetry, and does not provide much insight about the control strategies happening in the real human body. Therefore, we turned ourselves to the complete control problem, using only the character actuators to achieve motion objectives.

At this point, things became increasingly complicated, mainly due to convexity issues arising in the presence of unilateral constraints, not even considering friction. While we proposed a convex relaxation for the feature-based motion control, the solution is not entirely satisfactory since the complementarity relaxation and the motion objective are competing against each other.

In the end, this theoretical limitation forces the user to carefully describe control objectives for contacts, which is an undesirable limitation, especially concerning the applicability of these algorithms for interactive robotic motion planning.

Future Work

Of course, a lot more remains to be done, on multiple issues. Here are the main topics we would like to investigate in the future.

Kinetic Metric in PGA

As we have seen, the PGA depends on a Riemannian metric to compute the principal geodesics. Instead of choosing the canonical, bi-invariant metric on $SO(3)^n$, it would be interesting to try the kinetic energy metric instead:

- The mean pose corresponds to the pose that is the closest, energetically-speaking, to all the data samples
- Choosing a set of geodesic directions that are orthogonal, in the sense of the kinetic metric, would provide mechanically independent modes (instead of only statistically)
- The resulting model would provide a mechanically orthogonal basis (at the mean) spanning the sub-manifold of motion samples, combining the advantages of both statistical analysis and modal reduction.

Of course, the kinetic geodesics can no-longer be computed using the Lie exponential. Instead, geometric integrators would have to be used, since the kinetic geodesics correspond exactly to the time-integration of a pure kinetic-energy Lagrangian. The good energy behavior of geometric integrators would of course be helpful in this purpose.

Projective Constraints for Computer Vision

It would be interesting to test our pseudo-control framework as a pose and dynamics prior in a marker-less motion capture system. Silhouette tracking could be implemented as several control objectives. In the absence of tracking information (*i.e.* zero silhouette gradient/control force), the momentum of the character together with the reduced pose space would automatically provide a relevant pose candidate.

Modal Proportional-Derivative Control

While the task-space control framework allows high-level control strategies, PD motion controllers have proven very robust in the past, despite the relative lack of expressiveness of the resulting animations. Adapting these controllers to our reduced pose model should not pose any major problem, and could result in very fast, data-driven motion controllers.

Control and Bilateral Constraints

As mentioned above, the true task-space control problem under unilateral constraints is non-convex, and thus much harder to solve. Some algorithms have been proposed for solving such problems, as a Sequential Quadratic Program. Since these iterative algorithms are computationally-expensive, our reduced pose model may provide a low enough dimension reduction to make their real-time use possible.

⋆ ⋆ ⋆

In retrospect, this thesis allowed us to discover a lot of interesting mathematical structures, that we feel are not as widespread as they ought to be among the graphics community. Therefore, we put a lot of effort in presenting these abstract concepts in a (hopefully) more intuitive way.

Our approach stresses the importance of geometry in modeling: once this geometry is understood, interesting approximations can be derived, allowing to preserve relevant structures while improving computational efficiency.

To conclude, we hope that the present work will be able to serve as a basis for improving the various stages of the character animation pipeline: capture, storage, processing, simulation and control.

Bibliography

[AH04] Mihai Anitescu and Gary D. Hart. A fixed-point iteration approach for multibody dynamics with contact and small friction. *Mathematical Programming*, 101(1):3–32, 2004. 10.1007/s10107-004-0535-6.

[AMR$^+$07] Jérémie Allard, Clément Menier, Bruno Raffin, Edmond Boyer, and François Faure. Grimage: markerless 3d interactions. In *ACM SIGGRAPH 2007 emerging technologies*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[Ani05] Mihai Anitescu. On using the elastic mode in nonlinear programming approaches to mathematical programs with complementarity constraints. *SIAM J. on Optimization*, 15(4):1203–1236, April 2005.

[AP97] M. Anitescu and F. A. Potra. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics*, 14(3):231–247, 1997. 10.1023/A:1008292328909.

[AP99] Franck C. Anderson and Marcus G. Pandy. A dynamic optimization solution for vertical jumping in three dimensions. *Comput Methods Biomech Biomed Engin*, 2(3):201–231, 1999.

[Ari06] Okan Arikan. Compression of motion capture databases. *ACM Trans. Graph.*, 25(3):890–897, 2006.

[Bar92] David Baraff. Rigid body simulation. In *SIGGRAPH 95 Course Note 34. ACM SIGGRAPH*, 1992.

[Bar94] David Baraff. Fast contact force computation for nonpenetrating rigid bodies. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, SIGGRAPH '94, pages 23–34, New York, NY, USA, 1994. ACM.

[Bar96] David Baraff. Linear-time dynamics using lagrange multipliers. In *Proceedings of the 23rd annual conference on Computer graphics* *and interactive techniques*, SIGGRAPH '96, pages 137–146, New York, NY, USA, 1996. ACM.

[BB01] Paolo Baerlocher and Ronan Boulic. Parametrization and range of motion of the ball-and-socket joint. In *Proceedings of the IFIP TC5/WG5.10 DEFORM'2000 Workshop and AVATARS'2000 Workshop on Deformable Avatars*, DEFORM '00/AVATARS '00, pages 180–190, Deventer, The Netherlands, The Netherlands, 2001. Kluwer, B.V.

[BDCDA11] Florence Bertails-Descoubes, Florent Cadoux, Gilles Daviet, and Vincent Acary. A nonsmooth newton solver for capturing exact coulomb friction in fiber assemblies. *ACM Trans. Graph.*, 30(1):6–1, February 2011.

[BF01] Samuel R. Buss and Jay P. Fillmore. Spherical averages and applications to spherical splines and interpolation. *ACM Transactions on Graphics*, 20(2):95, 2001.

[BH00] Matthew Brand and Aaron Hertzmann. Style machines. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 183–192, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[BM95] F. Bullo and R. M. Murray. Proportional derivative (pd) control on the euclidean group. In *In European Control Conference*, pages 1091–1097, 1995.

[BM05] Stephen Braun and John E. Mitchell. A semidefinite programming heuristic for quadratic programming problems with complementarity constraints. *Comput. Optim. Appl.*, 31(1):5–29, May 2005.

[BPP07] Philippe Beaudoin, Pierre Poulin, and Michiel van de Panne. Adapting wavelet compression to human motion capture clips. In *GI '07: Proceedings of Graphics Interface 2007*, pages 313–318, New York, NY, USA, 2007. ACM.

[BRM09] Nawaf Bou-Rabee and Jerrold E. Marsden. Hamilton-pontryagin integrators on lie groups part i: Introduction and structure-preserving properties. *Found. Comput. Math.*, 9(2):197–219, March 2009.

[Bus04] Samuel R. Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. Technical report, 2004.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[BW95] Armin Bruderlin and Lance Williams. 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, 1995. ACM.

[CH05] Jinxiang Chai and Jessica K. Hodgins. Performance animation from low-dimensional control signals. *ACM Trans. Graph.*, 24(3):686–696, 2005.

[CH07] Jinxiang Chai and Jessica K. Hodgins. Constraint-based motion optimization using a statistical dynamic model. In *ACM SIGGRAPH 2007 papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[Cot09] Richard W. Cottle. Linear complementarity problem. In *Encyclopedia of Optimization*, pages 1873–1878. 2009.

[CP03] Michael B. Cline and Dinesh K. Pai. Post-stabilization for rigid body simulation with contact and constraints. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation, ICRA 2003, September 14-19, 2003, Taipei, Taiwan*, Proceedings of the 2003 IEEE International Conference on Robotics and Automation, ICRA 2003, September 14-19, 2003, Taipei, Taiwan, pages 3744–3751. IEEE, 2003.

[DK00] J. J. Duistermaat and J. A. C. Kolk. *Lie Groups*. Universitext. Springer-Verlag, New York, 2000.

[DRBR09] Julien Diener, Mathieu Rodriguez, Lionel Baboud, and Lionel Reveret. Wind projection basis for real-time animation of trees. *Computer Graphics Forum (Proceedings of Eurographics 2009)*, 28(2), mar 2009. to appear.

[Erl07] Kenny Erleben. Velocity-based shock propagation for multibody dynamics animation. *ACM Trans. Graph.*, 26(2), June 2007.

[Eto10] Jean Bosco Etoa. Solving convex quadratic bilevel programming problems using an enumeration sequential quadratic programming algorithm. *J. of Global Optimization*, 47(4):615–637, August 2010.

[Fea87] Roy Featherstone. *Robot Dynamics Algorithm*. Kluwer Academic Publishers, Norwell, MA, USA, 1987.

[FLJ03] P. Thomas Fletcher, Conglin Lu, and Sarang C. Joshi. Statistics of shape via principal geodesic analysis on lie groups. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2003 Proceedings CVPR-03*, pages –95, 2003.

[FLPJ04] P. Thomas Fletcher, Conglin Lu, Stephen M. Pizer, and Sarang C. Joshi. Principal geodesic analysis for the study of nonlinear statistics of shape. *IEEE Transactions on Medical Imaging*, 23(8):995, 2004.

[FLRS06] Roger Fletcher, Sven Leyffer, Danny Ralph, and Stefan Scholtes. Local convergence of sqp methods for mathematical programs with equilibrium constraints. *SIAM J. on Optimization*, 17(1):259–286, January 2006.

[FP03] Anthony C. Fang and Nancy S. Pollard. Efficient synthesis of physically valid human motion. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 417–426, New York, NY, USA, 2003. ACM.

[Fuk90] Keinosuke Fukunaga. *Introduction to statistical pattern recognition (2nd ed.)*. Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[FvdPT01] Petros Faloutsos, Michiel van de Panne, and Demetri Terzopoulos. Composable controllers for physics-based character animation. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 251–260, New York, NY, USA, 2001. ACM.

[Gle97] Michael Gleicher. Motion editing with spacetime constraints. In *Proceedings of the 1997 symposium on Interactive 3D graphics*, I3D '97, page 139, New York, NY, USA, 1997. ACM.

[Gle98] Michael Gleicher. 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, 1998. ACM.

[GM85] Michael Girard and A. A. Maciejewski. Computational modeling for the computer animation of legged figures. *SIGGRAPH Comput. Graph.*, 19(3):263–270, 1985.

[GMHP04] Keith Grochow, Steven L. Martin, Aaron Hertzmann, and Zoran Popović. Style-based inverse kinematics. *ACM Trans. Graph.*, 23(3):522–531, 2004.

[Gra98] F. Sebastin Grassia. Practical parameterization of rotations using the exponential map. *J. Graph. Tools*, 3(3):29–48, March 1998.

[HUH02] Lorna Herda, Raquel Urtasun, and Andrew Hanson. Automatic determination of shoulder joint limits using quaternion field boundaries. *In Proceedings of the 5th International Conference on Automatic Face and Gesture Recognition*, 2002:95–100, 2002.

[HWBO95] Jessica K. Hodgins, Wayne L. Wooten, David C. Brogan, and James F. O'Brien. Animating human athletics. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 71–78, New York, NY, USA, 1995. ACM.

[JBP06] Doug L. James, Jernej Barbič, and Dinesh K. Pai. Precomputed acoustic transfer: output-sensitive, accurate sound generation for geometrically complex vibration sources. *ACM Trans. Graph.*, 25(3):987–995, July 2006.

[Joh73] G. Johansson. Visual perception of biological motion and a model for its analysis. *PandP*, 14(2 1973):201–211, 1973.

[JP02] Doug L. James and Dinesh K. Pai. Dyrt: dynamic response textures for real time deformation simulation with graphics hardware. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '02, pages 582–585, New York, NY, USA, 2002. ACM.

[JYL09] Sumit Jain, Yuting Ye, and C. Karen Liu. Optimization-based interactive motion synthesis. *ACM Trans. Graph.*, 28(1):10–1, February 2009.

[KCD09] Marin Kobilarov, Keenan Crane, and Mathieu Desbrun. Lie group integrators for animation and control of vehicles. *ACM Trans. Graph.*, 28(2):16–1, May 2009.

[KCZO08] Ladislav Kavan, Steven Collins, JiŖí Zára, and Carol O'Sullivan. Geometric skinning with approximate dual quaternion blending. *ACM Trans. Graph.*, 27(4):105–1, November 2008.

[KG04a] Z. Karni and C. Gotsman. Compression of soft-body animation sequences, 2004.

[KG04b] Lucas Kovar and Michael Gleicher. Automated extraction and parameterization of motions in large data sets. *ACM Trans. Graph.*, 23(3):559–568, 2004.

[KGP02] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. *ACM Trans. Graph.*, 21(3):473–482, 2002.

[KJP02] Paul G. Kry, Doug L. James, and Dinesh K. Pai. Eigenskin: real time large deformation character skinning in hardware. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '02, pages 153–159, New York, NY, USA, 2002. ACM.

[KKS95] Myoung-Jun Kim, Myung-Soo Kim, and Sung Yong Shin. A general construction scheme for unit quaternion curves with simple high order derivatives. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '95, pages 369–376, New York, NY, USA, 1995. ACM.

[KMY03] Piyush Kumar, Joseph S. B. Mitchell, and E. Alper Yildirim. Approximate minimum enclosing balls in high dimensions using core-sets. *J. Exp. Algorithmics*, 8, December 2003.

[KPZ$^+$04] Eamonn Keogh, Themistoklis Palpanas, Victor B. Zordan, Dimitrios Gunopulos, and Marc Cardle. Indexing large human-motion databases. In *In Proc. 30th VLDB Conf*, pages 780–791, 2004.

[KRFC09] Paul Kry, Lionel Revéret, François Faure, and Marie-Paule Cani. Modal locomotion: animating virtual characters with natural vibrations. *Comput. Graph. Forum*, 28(2):289–298, avr 2009. Special Issue: Eurographics 2009.

[KSJP08] Danny M. Kaufman, Shinjiro Sueda, Doug L. James, and Dinesh K. Pai. Staggered projections for frictional contact in multibody systems. In *ACM SIGGRAPH Asia 2008 papers*, SIGGRAPH Asia '08, pages 164–1, New York, NY, USA, 2008. ACM.

[KYT$^+$06] L. Kharevych, Weiwei Yang, Y. Tong, E. Kanso, J. E. Marsden, P. Schröder, and M. Desbrun. Geometric, variational integrators for computer animation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '06, pages 43–51, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[LCF00] J. P. Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 165–172, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[LCR$^+$02] Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard. Interactive control of avatars animated with human motion data. *ACM Trans. Graph.*, 21(3):491–500, 2002.

[Lei81] G. Leitmann. *The calculus of variations and optimal control: an introduction*. Mathematical concepts and methods in science and engineering. Plenum Press, 1981.

[Len99] Jerome Edward Lengyel. Compression of time-dependent geometry. In *I3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 89–95, New York, NY, USA, 1999. ACM.

[LHP05] C. Karen Liu, Aaron Hertzmann, and Zoran Popović. Learning physics-based motion style with nonlinear inverse optimization. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 1071–1081, New York, NY, USA, 2005. ACM.

[LM06] Guodong Liu and Leonard McMillan. Segment-based human motion compression. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '06, pages 127–135, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[LMH10] Martin de Lasa, Igor Mordatch, and Aaron Hertzmann. Feature-based locomotion controllers. *ACM Transactions on Graphics*, 29number, 2010.

[LP02] C. Karen Liu and Zoran Popović. Synthesis of complex dynamic character motion from simple animations. *ACM Trans. Graph.*, 21(3):408–416, July 2002.

[LS99] Jehee Lee and Sung Yong Shin. A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 39–48, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[LS01] Jehee Lee and Sung Yong Shin. A coordinate-invariant approach to multiresolution motion analysis. *Graph. Models*, 63(2):87–105, 2001.

[LT08] Sung-Hee Lee and Demetri Terzopoulos. Spline joints for multibody dynamics. In *ACM SIGGRAPH 2008 papers*, SIGGRAPH '08, pages 22–1, New York, NY, USA, 2008. ACM.

[LWS02] Yan Li, Tianshu Wang, and Heung-Yeung Shum. Motion texture: a two-level statistical model for character motion synthesis. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '02, pages 465–472, New York, NY, USA, 2002. ACM.

[LZ09] Bruno Lévy and Hao (Richard) Zhang. Spectral mesh processing. In *ACM SIGGRAPH ASIA 2009 Courses*, SIGGRAPH ASIA '09, pages 17–1, New York, NY, USA, 2009. ACM.

[MCC09] Jianyuan Min, Yen-Lin Chen, and Jinxiang Chai. Interactive generation of human animation with deformable motion models. *ACM Trans. Graph.*, 29(1):1–12, 2009.

[MG01] Thomas B. Moeslund and Erik Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81(3):231–268, 2001.

[MK05] Tomohiko Mukai and Shigeru Kuriyama. Geostatistical motion interpolation. *ACM Trans. Graph.*, 24(3):1062–1070, July 2005.

[MLH10] Igor Mordatch, Martin de Lasa, and Aaron Hertzmann. Robust physics-based locomotion using low-dimensional planning. *ACM Transactions on Graphics*, 29(3), 2010.

[Moa02] Maher Moakher. Means and averaging in the group of rotations. *SIAM J. Matrix Anal. Appl.*, 24(1):1–16, 2002.

[MSZ94] Richard M. Murray, S. Shankar Sastry, and Li Zexiang. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., Boca Raton, FL, USA, 1994.

[MZS09] Adriano Macchietto, Victor Zordan, and Christian R. Shelton. Momentum control for balance. *ACM Trans. Graph.*, 28(3):1–8, 2009.

[NCM$^+$08] Jun Nakanishi, Rick Cory, Michael Mistry, Jan Peters, and Stefan Schaal. Operational space control: A theoretical and empirical comparison. *Int. J. Rob. Res.*, 27(6):737–757, June 2008.

[Pen06] Xavier Pennec. Intrinsic statistics on riemannian manifolds: Basic tools for geometric measurements. *Journal of Mathematical Imaging and Vision*, 25(1):127, 2006.

[PFA06] Xavier Pennec, Pierre Fillard, and Nicholas Ayache. A riemannian framework for tensor computing. *Int. J. Comput. Vision*, 66(1):41–66, January 2006.

[PTVF92] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical recipes in C (2nd ed.): the art of scientific computing*. Cambridge University Press, New York, NY, USA, 1992.

[PW89] A. Pentland and J. Williams. Good vibrations: modal dynamics for graphics and animation. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '89, pages 215–222, New York, NY, USA, 1989. ACM.

[PW99] Zoran Popović and Andrew Witkin. 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, 1999. ACM Press/Addison-Wesley Publishing Co.

[RCB98] Charles Rose, Michael F. Cohen, and Bobby Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Comput. Graph. Appl.*, 18(5):32–40, 1998.

[RDS+05] Inam Ur Rahman, Iddo Drori, Victoria C. Stodden, David L. Donoho, and Peter Schroder. Multiscale representations for manifold-valued data. *Multiscale Modeling & Simulation*, 4(4):1201–1232, 2005.

[RGBC96] Charles Rose, Brian Guenter, Bobby Bodenheimer, and Michael F. Cohen. Efficient generation of motion transitions using spacetime constraints. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 147–154, New York, NY, USA, 1996. ACM.

[RH91] Marc H. Raibert and Jessica K. Hodgins. Animation of dynamic legged locomotion. *SIGGRAPH Comput. Graph.*, 25(4):349–358, 1991.

[Rob88] B. Robertson. Mike the talking head. *Computer Graphics World*, 11(7):57, 1988.

[RP03] Paul S. A. Reitsma and Nancy S. Pollard. Perceptual metrics for character animation: sensitivity to errors in ballistic motion. *ACM Trans. Graph.*, 22(3):537–542, 2003.

[RPE+05] Liu Ren, Alton Patrick, Alexei A. Efros, Jessica K. Hodgins, and James M. Rehg. A data-driven approach to quantifying natural human motion. *ACM Trans. Graph.*, 24(3):1090–1097, 2005.

[SCLS07] Salem Said, Nicolas Courty, Nicolas Lebihan, and Stephen J. Sangwine. Exact principal geodesic analysis for data on so(3). In *Proceedings of the 15th European Signal Processing Conference, EUSIPCO-2007 15th European Signal Processing Conference, EUSIPCO-2007*, pages 1700–1705, Poznan Poland, 2007. EURASIP. Département Images et Signal.

[SH05] Alla Safonova and Jessica K. Hodgins. 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, 2005. ACM.

[SH07] Alla Safonova and Jessica K. Hodgins. Construction and optimal search of interpolated motion graphs. *ACM Trans. Graph.*, 26(3), July 2007.

[Sho85] Ken Shoemake. Animating rotation with quaternion curves. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '85, pages 245–254, New York, NY, USA, 1985. ACM.

[SHP04] Alla Safonova, Jessica K. Hodgins, and Nancy S. Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 514–521, New York, NY, USA, 2004. ACM.

[SKL07] Kwang Won Sok, Manmyung Kim, and Jehee Lee. Simulating biped behaviors from human motion data. In *ACM SIGGRAPH 2007 papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[SLHN10] Stefan Sommer, François Lauze, Søren Hauberg, and Mads Nielsen. Manifold valued statistics, exact principal, geodesic analysis and the effect of linear, approximations. In *Proceedings of the 11th European conference on Computer vision: Part VI*, ECCV'10, pages 43–56, Berlin, Heidelberg, 2010. Springer-Verlag.

[SLN10] Stefan Sommer, François Lauze, and Mads Nielsen. The differential of the exponential map, jacobi fields and exact principal geodesic analysis. *CoRR*, abs/1008.1902, 2010.

[SS00] Holger Scheel and Stefan Scholtes. Mathematical programs with complementarity constraints: Stationarity, optimality, and sensitivity. *Math. Oper. Res.*, 25(1):1–22, February 2000.

[SSK05] Mirko Sattler, Ralf Sarlette, and Reinhard Klein. Simple and efficient compression of animation sequences. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 209–217, New York, NY, USA, 2005. ACM.

[Swe98] Wim Sweldens. The lifting scheme: a construction of second generation wavelets. *SIAM J. Math. Anal.*, 29(2):511–546, 1998.

[TA77] A. N. Tikhonov and V. Y. Arsenin. Solutions of ill-posed problems. 1977.

[TJ09] Matthew C. Tresch and Anthony Jarc. The case for and against muscle synergies. *Curr. Opin. Neurobiol.*, 19(6):601–7, 2009.

[TLP07] Adrien Treuille, Yongjoon Lee, and Zoran Popović. Near-optimal character animation with continuous control. *ACM Trans. Graph.*, 26(3):7, 2007.

[UAT95] Munetoshi Unuma, Ken Anjyo, and Ryozo Takeuchi. 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, 1995. ACM.

[UFF06] Raquel Urtasun, David J. Fleet, and Pascal Fua. Temporal motion models for monocular and multiview 3d human body tracking. *Comput. Vis. Image Underst.*, 104(2):157–177, November 2006.

[WB08] Jing Wang and Bobby Bodenheimer. Synthesis and evaluation of linear motion transitions. *ACM Trans. Graph.*, 27(1):1–1, March 2008.

[WB10] Peter G. Weyand and Matthew W. Bundle. Point: Artificial limbs do make artificially fast running speeds possible. *Journal of Applied Physiology*, 108(4):1011–1012, 2010.

[WK88] Andrew Witkin and Michael Kass. Spacetime constraints. *SIGGRAPH Comput. Graph.*, 22(4):159–168, June 1988.

[WP95] Andrew Witkin and Zoran Popovic. Motion warping. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 105–108, New York, NY, USA, 1995. ACM.

[YL08] Yuting Ye and C. Karen Liu. Animating responsive characters with dynamic constraints in near-unactuated coordinates. In *ACM SIGGRAPH Asia 2008 papers*, SIGGRAPH Asia '08, pages 112–1, New York, NY, USA, 2008. ACM.

[YLvdP07] KangKang Yin, Kevin Loken, and Michiel van de Panne. Simbicon: simple biped locomotion control. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 105, New York, NY, USA, 2007. ACM.

[ZB94] Jianmin Zhao and Norman I. Badler. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Trans. Graph.*, 13(4):313–336, 1994.

[ZS09] Liming Zhao and Alla Safonova. Achieving good connectivity in motion graphs. *Graphical Models*, 71(4):139, 2009.

Acronyms

AM Angular Momentum. 134, 175

ccfk Canonical Coordinates of the First Kind. 80

ccsk Canonical Coordinates of the Second Kind. 80, 82

CoM Center of Mass. 174

DOF Degree of Freedom. 23, 33, 56, 57, 70, 83, 90, 91, 93, 113, 114, 121, 138, 166

FSM Finite State Machine. 156

GPU Graphical Processing Unit. 121

HMM Hidden Markov Model. 68

HP Hamilton-Pontryagin. 117

IK Inverse Kinematics. 27, 51, 58, 59, 63–65, 67, 68, 71, 75, 86, 89, 91, 92, 95, 96, 99

KKT Karush-Kuhn-Tucker. 119

LCP Linear Complementarity Problem. 119

LDS Linear Dynamic System. 69

LOD Level of Details. 100, 101

MRI Magnetic Resonance Imaging. 73

MVEE Minimum Volume Enclosing Ellipsoid. 143, 144

PCA Principal Component Analysis. 68–71, 73, 75, 76, 78, 81–83, 95–97

PD Proportional-Derivative. 155

PDF Probability Density Function. 68, 69

PGA Principal Geodesic Analysis. 27, 73, 75, 76, 78, 81–85, 91, 95, 102

PSD Positive Semi-Definite. 117, 121, 128

QP Quadratic Program. 118

RBF Radial Basis Function. 66, 68, 69

SLERP Spherical Linear Interpolation. 48, 72, 97, 101

SQP Sequential Quadratic Programming. 64

SVD Singular Value Decomposition. 94

Full Contents