

Planning and Adaptation of Virtual Human Character Motion

Cormac O'Brien

June 16, 2013

A thesis submitted to the University of Dublin, Trinity College in candidature
for the degree of Doctor of Philosophy in Computer Science.

Declaration

I declare that this thesis has not been submitted as an exercise for a degree at this or any other university and it is entirely my own work.

I agree to deposit this thesis in the University's open access institutional repository or allow the Library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

Cormac O'Brien

June 16, 2013

Abstract

This thesis covers some work on planning and adapting motions for a virtual 3d human character. The motivation is to use the physical features of the motion to drive both the planning of motion sequences and the subsequent editing of the motions to fit with necessary adaptations to the placement of contacts between the character and the geometric environment.

We present a scheme for planning sequences of motion in real time based on unrolling a graph of motions and testing availability of support for contacts. The contact model uses raycasts into the geometric scene to check for locations that approximately conform to pre-defined hand grip and foot contacts. Short-horizon planning (on the order of a few seconds of animation) can be done in real-time given a static geometry pre-process, though with a high memory cost.

Since the foot and hand placement for these motions is allowed to vary somewhat, we look at physically-based adaptation of animations to accommodate altered contact locations and mass-centre positioning. In the context of a vertex-based character representation, with constraints defining the character skeleton at each animation frame, we introduce some novel constraints to describe dynamic quantities by finite differences over several frames. The method gives visually acceptable results for smaller adaptations to external constraints, resulting in more realistic motion than with inverse kinematics (IK) constraints and acceleration smoothing alone. However, the solver is not guaranteed to converge and tends to become unstable for larger deformations.

We also explore an existing method for data-driven IK using principal geodesics analysis of an input set of motion capture data. We give full implementation details, and introduce an idea to improve the performance of the technique as a general full-body IK solver. The novel idea is to split the character skeleton into regions, extract the principal geodesics for each region and then recombine them in the IK solution. This gives more stable poses and allows a greater range of novel poses to be produced from the same input data set.

Relevant Publications:

1. **Real-Time Geometry Aware Motion Planning:**
Nithin Tharakan, Cormac O'Brien, Steven Collins;
Computer Graphics International 2010
2. **Spacetime Vertex Constraints for Dynamically-based Adaptation of Motion-Captured Animation:**
Cormac O'Brien, John Dingliana, Steven Collins;
Symposium on Computer Animation 2011

Acknowledgements

I would like to thank my supervisors, John Dingliana and Steven Collins, for being so generous of their time and for all their support over the past few years.

Thanks also go to my colleagues in GV2, especially Nithin Tharakan and Jan van der Kamp, with whom I collaborated on this project, and Alex Castells and my good friend Micheal Larkin who helped obtain the motion capture data used here.

I have my parents, Mary and Dermot, to thank for so much in life - including a strong sense of the value of a good education.

And thanks to my wife Akiko and daughter Momo who were not only endlessly patient with me during the evenings and weekends spent working on this thesis, but are the centre of my whole world.

Cormac

Contents

List of Figures	xi
Chapter 1 Introduction	1
1.1 Motivation	2
1.2 Scope	3
1.3 Contributions	3
1.4 Summary of chapters	5
1.5 Appendices	6
Chapter 2 Background and related work	8
2.1 Skeletons and rotations	8
2.2 Animation	9
2.2.1 Representation	9
2.2.2 Markup	9
2.3 Data-driven animation and motion synthesis	11
2.3.1 Interpolation	11
2.3.2 Motion graphs	13
2.3.3 Graph searching	16
2.3.4 Roadmaps	17
2.4 Procedural techniques in motion synthesis and editing	18
2.4.1 Inverse kinematics	18
2.4.2 Motion warping	21
2.4.3 Dynamics	21
2.4.4 Constrained optimisation	24
2.4.5 Spacetime optimisation	25
2.4.6 Controllers	29
2.4.7 Filters	32
2.4.8 Vertex-based motion	32
2.5 Evaluation of synthesised motion	33

2.5.1	Objective metrics	33
2.5.2	Dynamical correctness vs. plausibility	34
2.5.3	Visual evaluation	35
Chapter 3	Geometry-aware motion planning	36
3.1	Docking model	36
3.2	Support test	38
3.3	Planning a motion	41
3.4	Motion Atoms	42
3.5	Pre-processing the scene	43
3.6	Querying a dock against the pre-processed scene	45
3.7	Results	46
3.7.1	Query and probe	46
3.7.2	Static scene pre-process	48
3.8	Conclusions	50
Chapter 4	Vertex-based motion adaptation	52
4.1	Vertices and constraints	53
4.1.1	Constraint relaxation	54
4.1.2	Skeleton structure	54
4.1.3	Joint limits	56
4.1.4	Centre-of-mass and balance control	57
4.1.5	Kinetics constraints	58
4.1.6	Dynamics constraints	60
4.1.7	Geometric constraints on forces	63
4.2	Spacetime motion synthesis ab initio	64
4.2.1	Particle and single rigid body	65
4.2.2	Two-bone hopper	65
4.2.3	Simple two-legged character	66
4.3	Human character mocap adaptation	67
4.3.1	Convergence	71
4.3.2	Performance	74
4.3.3	Comparison with ground-truth mocap	76
4.4	Conclusions	81
Chapter 5	Data-driven inverse kinematics using principal geodesics	83
5.1	Principal geodesics	83
5.2	Geodesics Jacobian	84

5.3	End-effector Jacobian	85
5.4	Inverse kinematics solver	86
5.4.1	Computational efficiency	87
5.5	Skeleton decomposition	87
5.6	Results	88
5.7	Conclusions	98
Chapter 6	Conclusions	99
6.1	Summary	99
6.1.1	Geometry-aware motion planning	99
6.1.2	Vertex-based motion adaptation	100
6.1.3	Data-driven inverse kinematics using principal geodesics	101
6.2	Future directions	101
Appendix A	Generalised IK	104
A.1	Rigging and Cost functions	104
A.2	Solving	105
Appendix B	Simple motion fix-up using analytical IK	106
Appendix C	Optimization-based manipulation of momentum splines	108
C.1	Cubic b-spline curve notation	109
C.2	Cubic b-spline integral	109
C.3	Cubic b-spline derivative	109
C.4	Momentum curve adjustment	110
C.5	Conclusions	111
Appendix D	Glossary of terms and acronyms	113
Bibliography		114

List of Figures

1.1	A spacetime view of human motion: <i>Nude Descending a Staircase, No.2</i> [Marcel Duchamp 1912].	1
2.1	Locomotion parametric space. Source: Martin Prazak (with permission) . . .	12
2.2	Motion graphs: representation of the correspondence between the frames of two animations (left), and some example resulting synthesised motion (right). Source: [KGPo2]	14
2.3	A set of parametric motion spaces as hubs connected by fat edges. Source: [HGo7]	16
2.4	Behaviour planning by unrolling a motion state machine. Source: [LKo5] . . .	17
2.5	Analytical two-joint human limb IK. There is 1 degree of freedom in the solution, namely the swivel angle of the elbow.	19
2.6	Knee pop. Graph shows the adjustment in knee angle necessary to extend the leg by 1% of its maximum length versus the current length of the leg, expressed as a percentage of its maximum length. The bend in the graph is the point at which the knee ‘pops’ - something which is not apparent in a human knee due to compliance and tension from muscles and ligaments. Source: [KGSo2] . . .	20
2.7	Segmentation of the human body. Source: [Lab88]	23
2.8	Leaping motion synthesised by spacetime optimization from a simple initial motion. Source: [LPo2]	27
2.9	Momentum curve modelled after biomechanics data. Source: [LPo2]	28
2.10	Character simplification: levels of detail for lower-dimensional optimisations. Source: [PW99]	29
2.11	Controllers and keyframes in NaturalMotion’s Endorphin. The character on the left (in blue and grey) is being simulated, while the character on the right (grey) is keyframe-animated. Source: [Nata]	31
3.1	Illustration of hand docking showing point-and-normal pairs for each finger and the palm.	37

3.2	Example docking templates.	37
3.3	Local geometry test for each docking template.	39
3.4	Planning motions by unrolling predicted contact points into an environment.	41
3.5	An example of motion graph unrolling resulting in a pruned spacetime tree of actions.	42
3.6	Feasible spatial locations for each docking template. Allowed locations are marked in pink, and orientations marked with a short yellow line segment indicating direction after rotation about the world up-axis.	44
3.7	Test scene subdivided by a different octree for each docking template.	45
3.8	Directed graph of motion atoms.	47
3.9	The character skeleton's root path over some unrolled motion atoms.	47
3.10	The character used in the application, with propagated docks in the scene showing feasible (green) and infeasible (red).	48
4.1	Vertex-based character.	55
4.2	Centre-of-mass and static balance. Large purple orb marks COM vertex, smaller orbs mark COMs of body segments. COM is constrained to lie above foot contact.	57
4.3	Aggregate body dynamics. Particle positions over three frames of motion define the momentum derivative of the aggregated body.	61
4.4	Simple body structures.	64
4.5	Particle and body motion synthesis by spacetime vertex dynamics constraints.	65
4.6	Motion synthesis of simple articulated 'hopper'. Top: initial motion. Bottom: synthesised motion.	66
4.7	Motion synthesis of simple two-legged character. Top: initial motion. Bottom: synthesised motion.	67
4.8	Contact forces for two-legged character motion.	67
4.9	Editing motion: Leap. Original motion (blue). Edited motion (orange): mass-centre during leap (in purple) and landing foot plants (marked by red orbs) have been moved forward.	68
4.10	Editing motion: Long steps. Original motion (blue). Edited motion (orange): footsteps have been moved outward.	69
4.11	Editing motion: Tic-tac. Original motion (blue). Edited motion (orange): foot plant on wall (marked by red orb) has been moved upward.	70
4.12	Editing motion: Vault. Original motion (blue). Edited motion (orange): mass-centre during vault (in purple) has been moved upward.	70
4.13	Editing motion: Walk. Original motion (blue). Edited motion (orange): footsteps have been moved outward.	71

4.14	A failure case of motion adaptation. Moving the foot plant constraints too far from their original positions (top) results in divergence and unphysical motion.	72
4.15	Converging dynamics cost function during pre-solve stage.	73
4.16	Converging dynamics cost function during motion adaptation to altered external contact constraints.	74
4.17	Dynamics-based motion adaptation. Top row: original motions. Bottom row: adapted <i>Walk</i> (red) vs. ground-truth mocap <i>Walk wide one step</i> (green).	78
4.18	Dynamics-based motion adaptation. Top row: original motions. Bottom row: adapted <i>Walk</i> (red) vs. ground-truth mocap <i>Walk wide steps</i> (green). Note somewhat unnatural displacement of hip at bottom-left.	79
4.19	Dynamics-based motion adaptation. Top row: original motions. Bottom row: adapted <i>Step up</i> (red) vs. ground-truth mocap <i>Step up high</i> (green). Note the character does not straighten upright in the adapted motion but passes through a squatting posture.	81
5.1	Skeleton regions (from left): left arm, right arm, left leg, right leg, spine.	88
5.7	PGA: the first 30 full-body geodesics	91
5.2	Cumulative percentage variation captured by geodesics for different skeleton regions (single walk motion).	92
5.3	Cumulative percentage variation captured by geodesics for different skeleton regions (set of arm-dominant motions).	92
5.4	Cumulative percentage variation captured by geodesics for different skeleton regions (set of locomotion animations).	93
5.5	Cumulative percentage variation captured by geodesics for different skeleton regions (set of whole-body motions).	93
5.6	Cumulative percentage variation captured by geodesics for different skeleton regions (all motions in data set).	94
5.8	PGA: decomposed skeleton geodesics. In rows from top: left arm, right arm, left leg, right leg, spine	94
5.9	Decomposed PGA IK (top) shows better stability than full-body PGA IK (bottom) when poses are some distance from mean pose. Here only the right foot target is moving. Note the large resulting variation in the head pose when using full-body geodesics.	95
5.10	Comparing full-body geodesics (green) with decomposed geodesics (pink), given the same end-effector targets. Example poses close to the mean pose. The full-body geodesics tend to give better-looking results.	96

5.11	Comparing full-body geodesics (green) with decomposed geodesics (pink), given the same end-effector targets. Example poses at intermediate distance from the mean pose. Poses reconstructed from full-body geodesics start to break down, whereas poses from partial-skeleton geodesics remain stable. . .	97
5.12	Comparing full-body geodesics (green) with decomposed geodesics (pink), given the same end-effector targets. Extreme example poses. Partial-skeleton geodesics outperform full-body geodesics when far from the mean pose. . . .	98
B.1	Simple motion fix-up using analytical IK and spring-damper for skeleton root position. Original motion (blue) and fixed-up motion (orange).	106
C.1	Momentum spline adjustment. Respectively: original/adjusted COM path (blue/orange); interval of adjustment (cyan/magenta); original/adjusted momentum curve (green/red).	108
C.2	Schematic of mass displacement at t_i , with adjustment over the previous k knot intervals.	110

Chapter 1

Introduction

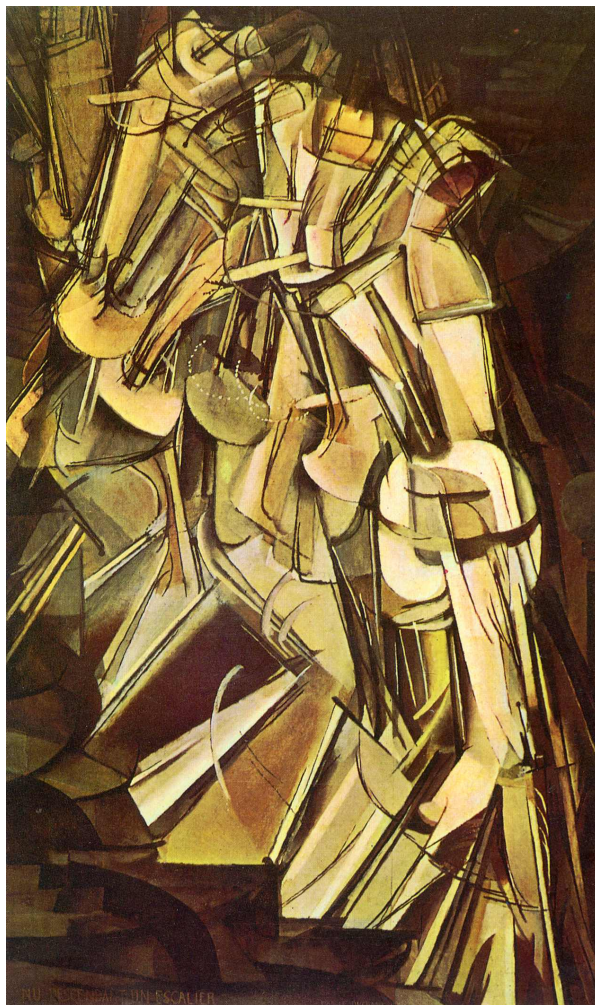


Figure 1.1: A spacetime view of human motion: *Nude Descending a Staircase, No.2* [Marcel Duchamp 1912].

1.1 Motivation

Animation of human-like characters is of intense interest in the fields of video games and animated movies, as well as generating a substantial body of literature in the computer graphics field.

Here we will focus on issues relating to video games, since the challenges of real-time animation are of particular interest. The requirements for video games are different to movies in that games must compose motion on the fly, interacting with player intent and nondeterministic circumstances. These problems have been studied a great deal in academic literature, but for the most part the video game industry relies on tried-and-tested animation blending techniques pioneered in the 1980s.

In latter years, real-time rendering techniques have advanced so far that many modern games have near photo-realistic graphics. Lighting and skin shading in particular give extremely realistic appearances to human characters in games such as *Battlefield 3*.

However, the quality of animation still lags behind that of graphics. The quality of individual animations is extremely good, whether it be sourced from a motion capture performance or artist-authored.

Tools allow artists to compose many animations to be blended into graphs and state-machines. And while the resulting sequences of animation are high-quality, the effect sometimes falls short. Common artefacts such as interpenetrations and skating feet ruin the illusion of a lifelike character engaging with its world. And with the composition of animations being prescribed in advance, there is little scope for improvisation or novelty in the character's performance as it responds to player input. Extra animations can be added to improve the situation, but the extra volume of data puts demands on artists, the asset pipeline and runtime efficiency.

The problem of dynamically generating plausible animation is not an easy one to solve, but there are notable efforts to improve this. Already, smaller productions such as *Overgrowth* are using mostly-procedural animation to relieve the content creation burden, and to deliver good-looking and responsive animation to the player. On the whole procedural motion synthesis is less popular in games since the use of procedural techniques tends to take some of the control of the final appearance of the animation out of the animators' hands.

But larger developers have also begun to adopt more procedural techniques. The *Assassin's Creed* series uses a full-body inverse kinematics solution [Aut] to animate its highly athletic, climbing characters. Several games, including *Grand theft Auto 4* use force-based controllers [Natb] to create reactive animation by simulation. In the *Uncharted* series, many additive animations are used to give detail and life to characters by layering breathing, stance variation and other features on top of basic animations [McI10]. And in the recent *Max Payne 3*, diving animations are generated procedurally by combining physics simulation with diving postures.

The challenge at hand is to find ways to synthesise novel and dynamic sequences of motion, while keeping the style and expressiveness needed to convey emotional content.

1.2 Scope

This thesis explores some methods for procedurally editing and synthesising animation.

We focus on planning sequences of motions and editing those motions to conform to external constraints in a static scene, with motion capture being our source of animation data.

We are interested in procedural methods for motion editing and synthesis; both physically-based methods and data-driven methods that can capture motion style from some input animations.

1.3 Contributions

The problem addressed in the thesis is to provide tools to synthesise sequences of animation in a static scene in a realistic fashion driven by the physical contacts between the character and the scene.

The contact model that provides the basis for planning sequences of motion in Chapter 3 also specifies the points at which forces can be applied by the character to effect motion through space. The output of the planning stage is a sequence of unedited animations whose contact points may be not quite where they need to be to register with the corresponding supporting geometry in the scene. So these animations must be adapted to conform to the contacts in a realistic fashion.

For physically plausible motion, the changes in momentum apparent in the animation must conform to the possible forces at these contacts; and the character's end-effectors - hands and feet - must make solid contact with the target positions.

This principle informs the physically-based *spacetime vertices* motion adaptation of Chapter 4, taking an input animation and deforming it so that the contacts are made with the motion remaining dynamically realistic.

Chapters 3 and 4 take a view of motion as a spacetime phenomenon (figure 1.1), where an event happening at a particular time and place constrains what must happen leading up to that.

The full-body *geodesics IK* method in Chapter 5 constructs poses from end-effector targets alone, using the animation data to reduce the possible pose space to a much lower-dimensional set of control parameters.

Whereas the spacetime vertices method is stable only for small deformations of the input animation, the geodesics IK method is robust enough to give plausible poses for a wide range of end-effector target positions. So the latter has the potential to act as an intermediate stage of

motion editing, plausibly handling large deformations to the animation before a final pass that improves the dynamical realism.

The chief novel contributions in this thesis are:

- A scheme for planning sequences of short motion clips based on a model of contact between the character and the geometric environment (Chapter 3).

The strength of the technique is in taking a set of animations and a few simple descriptions of contact and automatically determining in general what motions can be performed at what locations in an arbitrary environment.

We implemented an example application with simple models of hand and foot contact, and found that as we unrolled a spacetime graph of potential motions into our scene, the number of branches caused a combinatorial explosion in the number of geometrical queries needed to test for the potential contacts. Pre-processing the scene and storing the results in a octree for fast spatial querying resulted in a significant speed-up. This enabled us to test feasible potential animations around 2 seconds in advance, pruning away motions not supported by geometry and mapping user input to a selection among feasible animations.

- Some novel constraints extending a vertex-based IK solver into a spacetime motion editing method (Chapter 4).

An existing technique for cloth and soft-body simulation [MHHR07] also supports a type of IK solver. A character can be constructed from vertices connected by differentiable constraints, which are solved iteratively by projecting the vertices along the gradient of each constraint in turn.

We provide details of the constraints we used to construct a character skeleton with joint limits. Furthermore, we introduce a new type of constraint that relates consecutive animation frames and can affect kinetic and dynamical quantities of motion.

We applied this to physically-based editing of motion-captured animations. The method gives physically-plausible results for small adjustments to the motion, but the solver can become unstable and does not give predictable results.

- A novel idea for extending a data-driven statistical IK solution to more general full-body IK (Chapter 5).

We implemented the ‘principal geodesics’ IK method [TWC⁺09], and came up with a novel way to prepare the geodesics that capture motion variation from a dataset and form the basis of the IK technique. By preparing geodesics for subsets of the character skeleton, we can recombine these in the IK solver to reduce spurious coupling between

the motions of disparate body parts. This results in a more stable solver that can produce a wider range of plausible poses, with the drawback that it tends to take more solver iterations for end-effectors to reach their targets.

1.4 Summary of chapters

The remainder of this thesis is structured as follows:

Chapter 2 A review of some of the literature relevant to animation for human characters. Beginning with an overview of schemes for representing character skeletons and motions, we divide the field of animation techniques approximately into data-driven and procedural methods.

Data-driven methods are chiefly concerned with putting together sequences of animations from a database. We cover motion graphs, and some methods used to plan sequences of animation in a geometric environment.

Under the heading of procedural animation synthesis we cover various techniques for generating and editing motion, including inverse kinematics, kinematic motion editing (such as motion warping) and dynamics-based motion editing. The consideration of dynamics is of particular interest, using the physical equations of motion to add realism to procedurally-generated animation. We provide examples from the literature of several different approaches to this, in particular force-based controllers, dynamics filters and spacetime optimization.

Some notes on evaluation of the plausibility of edited motion are included. However, we have not covered the considerable body of work on perceptual evaluation of human character animation, as this was not a focus of this thesis.

Chapter 3 This chapter details our work on motion planning in a geometric environment.

We used a description of end-effector contacts to search the environment for suitable sites for interaction. This model of contact supported unrolling a graph of simple motions within the environment. The resulting tree structure described all the possible paths our character could take over the space within a certain time horizon. Using geometric queries to prune away infeasible branches of this tree we could plan possible motions and select between them based on user input in real-time.

Chapter 4 In this chapter we describe our work on vertex-based character motion.

Existing iterative constraint solving techniques allow a character skeleton to be constructed from vertices and constraints. We contributed several novel constraints for use in full-body inverse kinematics under this scheme.

We also extended the scheme to apply constraints not just between vertices in space but over each vertex's motion in time. These spacetime vertex dynamics constraints can be used in synthesising and editing motion of vertex-based characters.

Chapter 5 We briefly explored an existing technique for data-driven full-body inverse kinematics using principal geodesics analysis (PGA IK). Our contribution here is full implementation details for PGA IK and some novel work on generalized full-body IK using this method. We decompose the skeleton into bone subsets and perform PGA on each subset, recombining these to obtain more flexible IK that retains some plausible coupling of joints over the body.

Chapter 6 Summary of conclusions and some thoughts on possible future directions for research.

1.5 Appendices

Appendix A Following on from Chapter 5, we briefly consider how the method could be generalised beyond the scope of geodesics IK. The generalisation gives a possible framework for biomechanically-based IK.

Appendix B We describe a simple approach for adapting motions to external constraints that was implemented to work with the planning system in Chapter 3.

Appendix C This appendix describes an earlier idea for physically-based motion editing, using splines representing the aggregate momentum, force and COM trajectory of a character. We derive simple expressions for differentiating and integrating cubic b-splines, and formulate a Quadratic Problem for optimising the control points of a momentum spline so as to achieve a desired displacement of the COM at a given point in time.

Appendix D For ease of reference this appendix contains a glossary of acronyms and terms found throughout the text.

Chapter 2

Background and related work

2.1 Skeletons and rotations

Characters in 3d video games, movies and other applications typically use a skeleton to deform a mesh character model. A skeleton is a spatial hierarchy of *joints* (or *bones*, these terms tend to be used interchangeably). Each joint is displaced relative to its parent by a translation and rotation (and possibly a scale, though here we will only consider rigid transformations). Translation of a bone is represented by a 3-vector, being a point in the coordinate frame of its parent bone. Rotation can be represented by Euler angles, a 3x3 matrix or a quaternion [Lee08]. Quaternions have advantages over Euler angles and matrices, namely small memory footprint and avoiding gimbal-lock singularities [Sho85]. Alternatively, a rigid transformation (translation and rotation) can be represented by a homogeneous 4x4 matrix or a dual quaternion [KCvOo7]. The exponential map [Gra98] is another representation of rotations employing a vector in $\mathbb{R}^3$ denoting the axis and magnitude of a rotation. This is closely related to the logarithm of a quaternion. It is subject to singularities but is useful in representing differential rotations, or as a basis for the tangent space to the space of orientations [TWC⁺09]

In the motion editing literature, as in the area of articulated body dynamics [Fea07], it is usual to parameterise the character joints by reduced degrees of freedom. A hinge joint (knee or elbow) has a single angle parameter whereas a ball joint (hip, shoulder) would have three angle parameters. The root of the character is an unconstrained rigid transformation representing the character's position and orientation in space, parameterised by six degrees of freedom. A whole character typically has around 30-40 degrees of freedom using such a representation. The Human Animation working Group [Gro] provides a standard for representing virtual human characters.

2.2 Animation

2.2.1 Representation

An animation is a time-varying function $\mathbf{q}(t)$ over the *degrees of freedom* (DOF) $\{q_0, q_1, q_2, \dots, q_n\}$ of the character. This often takes the form of an interpolation over a set of *key frames* which hold the values of the DOF at particular key times. We can call a key frame sequence a *clip*, and a collection of clips a *motion database*.

Motion data is typically either obtained from motion capture (*mocap*) of a human actor or is keyframed by an animation artist. Interpolation of key frames can be done by linear interpolation if the key times are close together, or by other means such as spline interpolation [Sho85].

Considering motion as a signal can lead to some useful processing techniques [BW95]. The motion can be examined in the frequency domain, to pick out features at different scales in time and space. Splines can form the basis for a hierarchical representation [LS99] where successive layers of approximation [LWS97] capture features of the motion at different frequencies.

There exist many spline curve formulations but cubic b-splines are often favoured in animation because of their C^2 continuity and efficient evaluation due to locality; a point on a cubic b-spline curve is only affected by the nearest four control points. The curve stays within the hull of the control points and does not interpolate them. Another popular type of spline curve is the Kochanek-Bartels, or TCB, spline [KB84], which are a generalisation of cubic Hermite splines. A TCB spline interpolates its control points and offers three parameters to control the curve shape at each point, namely tension, continuity and bias.

Wavelets too allow a compact representation of motion features at different scales [LGC94, BPvdPo7]. The idea of motion as a signal also led to motion displacement maps [WP95, Gle97, GSKJo3] and additive motion blending as used to provide animation variety in modern video games [McI10].

Splines and wavelets are used in compressing motion data, allowing some loss of high-frequency modes in exchange for a more compact representation of the motion function. Other compression techniques make use of the statistical structure of the motion to reduce the number of DOF needed to describe it [GMHP04, TWC⁺09].

2.2.2 Markup

To capture interaction with a virtual environment it is necessary to add some metadata to the animation. Annotations to mark discrete events, such as making or breaking contact, provide timing information for blending as well as giving a hook for other systems such as sound or particle effects. Kinematic constraints enforcing environment contact with the character's end-

effectors provide a means to *dock* the character with the environment. This forms the basis of our method for planning sequences of motion in Chapter 3, wherein we determine the feasibility of performing a given motion at some position within a scene by testing whether the end-effector contacts are supported by the scene geometry. In Chapter 4 we use the end-effector contacts to determine the possible external forces on the character.

Foot sliding is one of the most noticeable artifacts of motion editing, and though common it has a negative effect on the plausibility of a character’s motion. It is important, then, to enforce static contact when intended. Much attention has been paid to this problem in the literature [BB98, LS99, KGS02].

For video game applications, kinematic constraints like footsteps are usually marked up manually by an artist. The human eyeball is a remarkably accurate instrument for the job, being well-adapted to perceiving the features of human motion, and is used as the benchmark against which constraint detection techniques are compared [PDo6]. However, there is interest in automating constraints mark-up, even for partial automation as an assist to the artist. Bindiganavale and Badler [BB98] used zero-crossings of the acceleration of a character’s end-effector relative to an item in their virtual scene to find contacts whose approximate location were known in advance. Liu and Popovic [LPo2] provided a method based on finding the set of points in space left invariant under the rotation-translation transform between successive animation frames. The method is susceptible to noise since for a small rotation component (as that between frames), small variation in the translation component results in large variation in the position of the invariant subspace. This makes the method unsuitable for mocap data which are inherently noisy. The method of Liu and Popovic was generalised to a moving frame of reference by Salvati et al. [SLCBo4] and LeCallenec and Boulic [LCBo6b] improved on this by examining the pattern of expected noise from a small manually marked-up training set. Statistical methods such as a k nearest neighbours classifier [IAFo6] allow training of an accurate oracle for constraints detection, with the downside that a training set is needed. Perhaps the simplest and most robust method of finding general static contacts is the speed threshold [LHo6]. For detecting footplants this can be combined with other metrics such as foot height [GBT06]. Stationarity with respect to the local scene provides a good predictor of contact with no need for a training set. Setting the speed threshold high gives good robustness to noise but many false positives, so this method is perhaps best as an automatic aid to manual mark-up [Natc]. Johansen [Joh09] uses velocity and height threshold for detecting footsteps, along with other motion features such as curvature of the graphs of heel and toe elevation.

2.3 Data-driven animation and motion synthesis

Purely data-driven techniques use a corpus of animation data to synthesise continuous sequences of motion. There are also procedural methods for generating motion automatically. Most often, motion synthesis for practical projects involves both data-driven and procedural techniques, and in classifying the various methods in the next sections there is no clear divide between the two. However we draw a broad distinction between methods such as animation blending and motion graphs, which are mostly about combining existing animation data, and procedural methods for creating animations ab initio or hybrid ‘procedurally data-driven’ methods for editing existing animations to make novel motions.

2.3.1 Interpolation

A basic tool for manipulating and editing animation data is blending, or in other words forming an animation from a linear combination of other animations. Characters driven by skeletal animation most commonly use blending in two ways, namely transitions and parametric motion spaces.

Transitions

A transition between two motion clips can be achieved smoothly by cross-fading. This is a convex blend

$$f(t) = (1 - t)A + (t)B$$

of the two clips A and B where the blend parameter t goes from 0 to 1, typically along a C^2 continuous *ease curve*. There are two caveats for constructing a good-looking transition. First, the motions should have reasonably similar poses in the time interval immediately surrounding the transition frame. Second, the animations should be timewarped during the transition so that discrete events during the transition, such as footsteps, occur simultaneously in both clips.

Parametric motion spaces

A parametric motion space is an n -way blend of motion clips where the n blend parameters are defined as a function of m input control parameters. The same caveats apply to blending clips in a parametric space. To support such structured blends, Kovar and Gleicher came up with ‘registration curves’ [KG03], which control timing and the orientation of the character over a blend.

For an illustrative example of a parametric motion space common in video games, suppose we have a set of locomotion clips - namely standing still, walking in 4 directions (forward,

backward, left, right) and running in the same 4 directions. Our input parameters are locomotion speed and direction, so $n = 9$ and $m = 2$. The blend parameter function calculates a blend parameter between 0 and 1 for each clip so that the resulting output motion has the desired properties. The simplest way to do this is to construct a Delaunay triangulation in the parameter space. Each clip is represented by a point in m -dimensional space, and the space is divided into simplices (triangles for 2-dimensional space) whose vertices are those points (Figure. 2.1). A control parameter vector that lies within one such simplex can be mapped to a blend parameter vector by using its barycentric coordinates within that simplex. Only the clips that correspond to the vertices of that simplex have nonzero blend parameters, meaning that spurious clips (such as those walking in a different direction entirely) are not included in the blend.

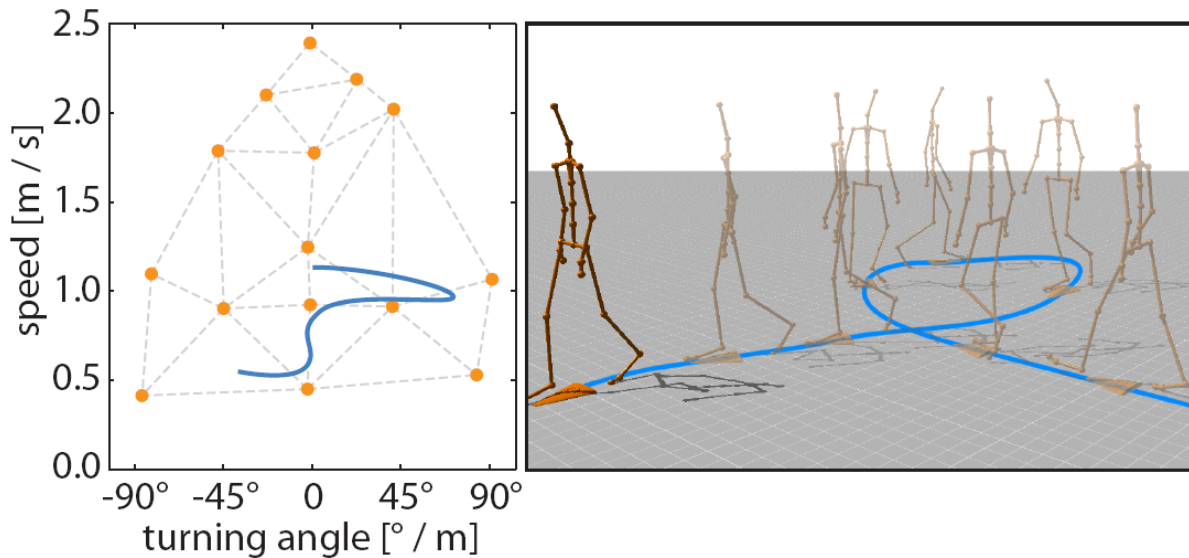


Figure 2.1: Locomotion parametric space. Source: Martin Prazak (with permission)

Something similar can be done using radial basis functions [RCB98], but these have undesirable properties in that they are not compact and hence can be nonzero in regions of the parameter space far from the region of interest [Her11].

Johansen [Joh09] uses a new method called Gradient Band Interpolation, which assigns a normalised influence function to each sample based on the proximity of other neighbouring samples. The influence function for a sample is unity at the sample, decreasing to zero along the boundary of a convex polygonal region demarcated by the nearest neighbour samples. This method has the advantages over radial basis function interpolation of boundedness and locality (or compactness, being nonzero only in the region near the sample).

Abe et al. use linear interpolation [ALPo6], but construct the clips by dynamics-based editing of a single input clip. This allows them to automatically construct a whole parametric

motion space from one clip.

The current state of the art in video game technology is to use authored animation state machines where each state may consist of layered blends (or *blend trees*) of custom-authored animation data. Synchronisation is handled by adding events, for instance denoting footsteps so that a walking animation may be synchronised with a run. Automatic markup of these events typically involves speed-thresholding methods for contact detection. Manual authoring is preferred over procedural generation as it allows artists to have fine control over the appearance of the resulting animation.

2.3.2 Motion graphs

The canonical data structure for automatically synthesising motion from an existing database is the motion graph. Motion graphs as such were introduced by Kovar and Gleicher [KGP02] wherein a metric for the difference between two frames of animation was established, taking into account pose and velocity. Using the similarity metric to find corresponding motion frames, transition points are identified throughout the motion database (Figure 2.2). This results in a directed graph whose nodes are motion clips and edges are transitions between those clips. Arikan and Forsyth [AF02] assigned a cost to each transition to penalise discontinuities in a sequence of motions constructed by traversing the graph.

Kovar and Gleicher's paper also presents a method for pruning the graph to remove sinks and dead ends so that arbitrarily long streams of data can be generated. Motion synthesis over a motion graph is a matter of selecting graph edges based on some cost function to build a graph walk of minimal cost. This paper has spawned a distinguished family of literature exploring various refinements in approach such as new distance metrics, modifications for graph connectivity and graph augmentations by motion interpolation.

An alternative but equivalent formulation models the transitions as a Markov process with a matrix of probabilities where each element P_{ij} describes the probability of transition from frame i to frame j [LCR⁺02].

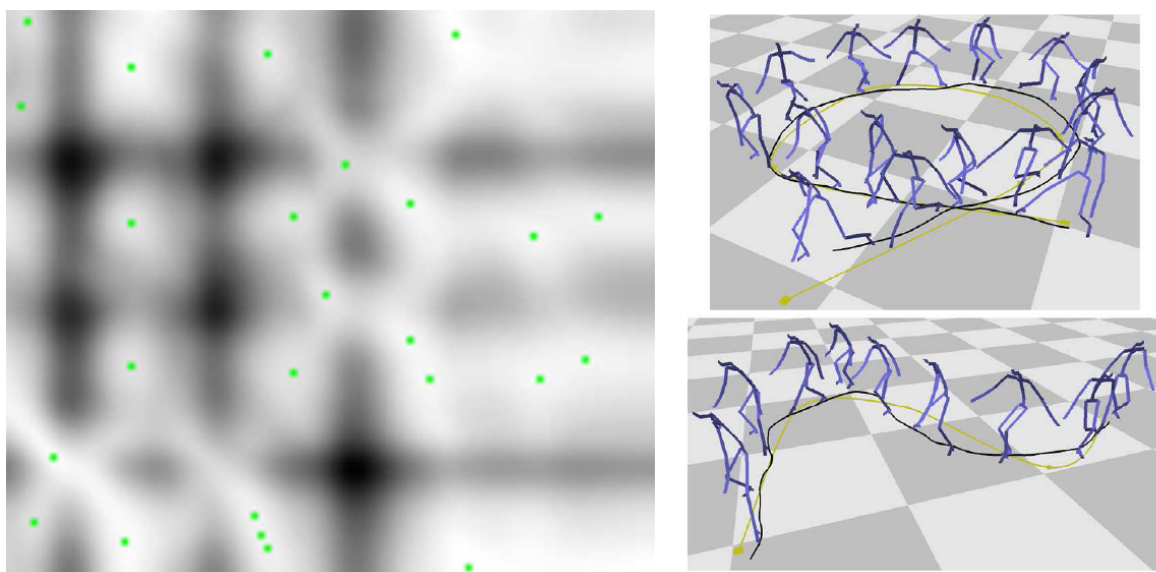


Figure 2.2: Motion graphs: representation of the correspondence between the frames of two animations (left), and some example resulting synthesised motion (right). Source: [KGPO2]

Motion similarity metric

The Kovar-Gleicher metric [KGPO2] is the de facto standard for measuring pose difference in animations. The metric takes character postures over a small window of animation frames about the target frame, considers the posture as a point cloud, and compares this to a similar point cloud obtained from another frame elsewhere in the motion database. The point clouds are aligned by a closed form solution to the minimal weighted sum of squared distances over rigid 2d transformations. Using a window of frames implicitly takes into account velocity and acceleration information and the point cloud representation is robust to noise.

Further work has been done into schemes for weighting the contributions of the various body parts to better describe similarity of movement. Matsunaga and Zordan [MZ07] considered mass displacement of each body segment plus its outboard segments, together with extra weighting for body parts in contact with the ground. They also came up with an automatable measure of how many good transitions a database has, based on ranking detected transitions by cost and graphing the ranked transitions against the cost. This chart has an elbow where the slope increases, and selecting the first N ranked transitions before this elbow results in a well-behaved motion graph. Furthermore, they found that transitioning during a period of dynamic motion is preferable as viewers tend to be more forgiving of flaws.

Mass displacement has also been noted elsewhere as a significant factor in perceptual similarity of poses in motion [PW99]. We used this metric in Chapter 4 as a measure of the similarity between an adapted motion and a ground-truth motion from mocap data.

Connectivity

The goal of motion synthesis is to form arbitrary sequences of motion on demand. It is undesirable for a graph walk to take too long to transition or to pass through unrelated motions, so connectivity is important.

Further work by Gleicher et al. [GSKJ03] focuses on selecting *hub* poses as multi-way transitions between many clips. The technique here is to find *match* frames common to many clips, compute an average pose for each matching set and form this into a hub by building displacement curves that smoothly transition each clip through the common pose. The hubs give flexibility, particularly for responsiveness to user control.

Zhao and Safonova [ZSo8] augment the motion capture database underlying the graph. They form new motions by interpolation over motions with matching kinematic constraint states using a physics-preserving interpolation scheme [SH05], then construct and prune a well-connected graph. To further improve connectivity, Ren et al. [RZS10] use constrained optimization to compute physically-realistic transitions between diverse parts of the graph.

Interpolation/parametric spaces

Shin and Oh's work [SO06] extends the hub type motion graph by grouping together related clips that share a hub into a *fat edge*. A graph walk passing through the fat edge can use a blended combination of those clips. In other words a fat edge is a parametric motion space.

Heck and Gleicher go further [HG07], representing continuously changing transition points as well as interpolated edges, so they are not limited to using hub nodes (Figure 2.3).

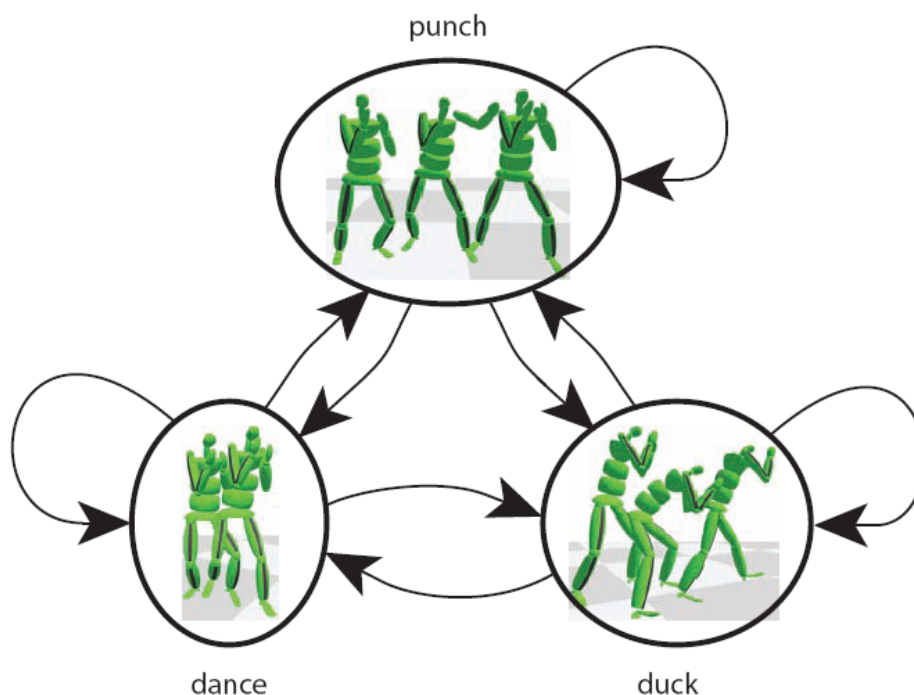


Figure 2.3: A set of parametric motion spaces as hubs connected by fat edges. Source: [HGo7]

Safonova and Hodgins [SHo7] represent a motion as an interpolation of two time-scaled paths through a motion graph, only allowing interpolation of clips with matching contact patterns. Their approach is focused on synthesising whole motions, and interpolation eliminates or reduces error in satisfying user-specified constraints on the path.

2.3.3 Graph searching

After a motion database is segmented into clips the problem becomes one of searching the database to find motion sequences that fit some user-specified criteria.

Creating a higher-level representation of a motion graph can reduce the computational cost of such a search. Examples include a randomised search over a hierarchical graph to extract motions that satisfy user-specified constraints [AFo2], and a two-level structure for planning a path where motion is embedded in the virtual environment [LCLo6].

Control policies that overlay an action-state model on the lower level motion graph [LCR⁺o2, LLo6] allow user interactivity in selecting what action comes next. Manually annotating motion clips with descriptors adds a semantic layer and another set of possible constraints for the choice of motion to include in a sequence [AFOo3]. Such annotations can also form the basis for a state machine layer on top of the motion graph [CZW07], giving further possibilities for user-responsive motion synthesis.

We can use a virtual environment to provide constraints on a motion graph search, making the problem one of planning a path through the environment using a sequence of graph edges.

2.3.4 Roadmaps

There exists a substantial body of research in the robotics field concerning planning collision-free paths through an environment. A roadmap represents the connectivity of paths through space, allowing construction of motion along those paths [VKSM05]. Data-driven motion planning has the added constraint that motion must be constructed from segments that originate in the motion database. Choi et al. [CLSo3] construct a roadmap from a mocap database as a directed graph of transitions between postures which satisfy foothold constraints in the environment. Another proposed approach [MKo8] is to use a path planning algorithm to inform the motion graph search while traversing a roadmap.

Motion graph unrolling

Each motion in a motion graph describes a spatial trajectory, so we can *unroll* these into a tree where each of the tree's branches denotes a choice of transitions leading from a graph node. The tree can be overlaid on a virtual environment and used to plan out a sequence of motion [LKo5].

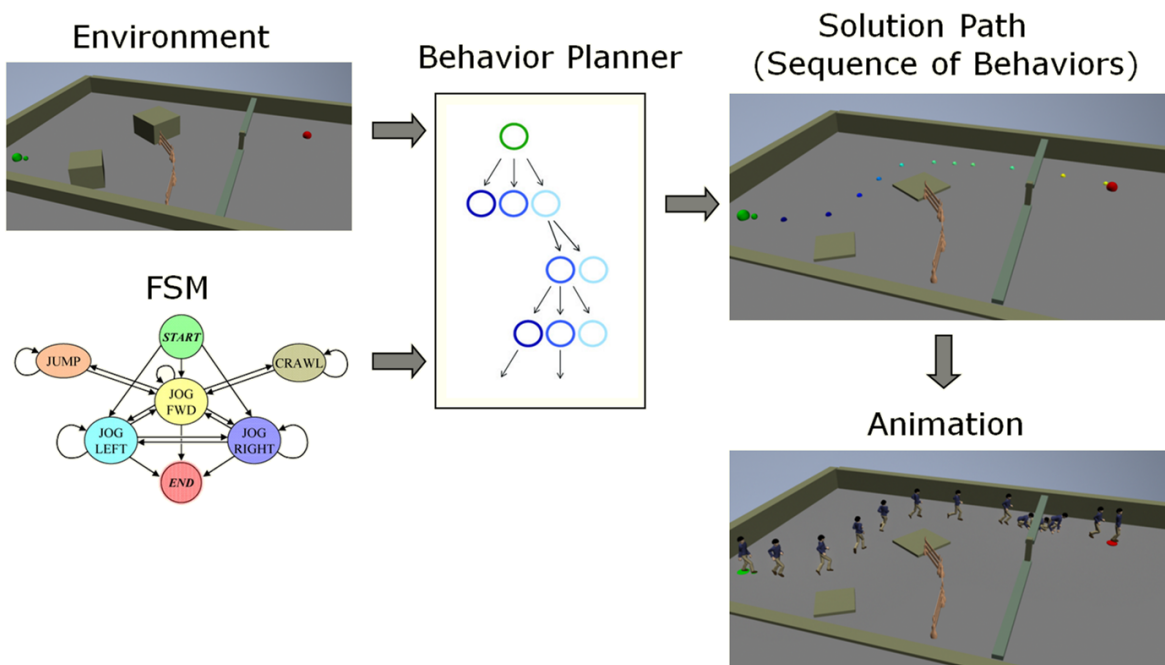


Figure 2.4: Behaviour planning by unrolling a motion state machine. Source: [LKo5]

Most approaches from the literature are global, solving for a whole motion whose requirements are determined in advance (Figure 2.4). Because of the geometric explosion in the number of branches on an unrolled graph, this type of planning is typically done offline. A pre-process can be used to mark grid locations in an environment as reachable by certain graph walks [LKo6]. This type of process captures the interaction of a motion graph with a virtual environment, and metrics can be defined such as environment coverage, path efficiency and local maneuverability [RPo7].

In Chapter 3 we perform a local version of this kind of planning. We unroll a hand-constructed motion graph in real-time, pruning away unfeasible branches that are blocked by the available scene geometry.

2.4 Procedural techniques in motion synthesis and editing

2.4.1 Inverse kinematics

The concatenation of joint transforms from parent to child along a hierarchical chain results in the world transform of the end joint. When one joint in the chain moves, all of its descendant joints must be traversed to find their updated world transforms. The determination of joint positions in this fashion is known as *forward kinematics* (FK). It can be difficult to control an animation by forward kinematics since small changes to the joints in a chain can displace the end joint by a large amount. *Inverse kinematics* (IK) allows the posing of the whole chain by positioning a single target object, or *IK goal* and minimising the distance of the end joint, or *end effector*, from that target. The IK goal typically lives in world space, allowing it to be parented to objects that exist outside the character, though it can be useful to be able to express its location relative to the root joint of the IK chain. IK is a large field originating in robotics literature. Welman [Wel93] surveys various methods for solving IK problems, particularly the Jacobian transpose method and Cyclic Coordinate Descent. Fedor [Fô3] also looks at several types of IK solver with reference to their use in real-time animation.

Real-time animation requires fast calculation of joint chain poses. The analytical 2-joint human limb type solver (Figure 2.5) proposed by Tolani and Badler [TGBoo] is fast and gives a repeatable solution, with no hysteresis. This solver has been incorporated into several analytical whole-body solutions [WMCGo4, KMAo5, KMo5, UPBSo8] where the skeleton is decomposed into separate chains (e.g. arms, legs, head) with an individual analytical solution for each chain.

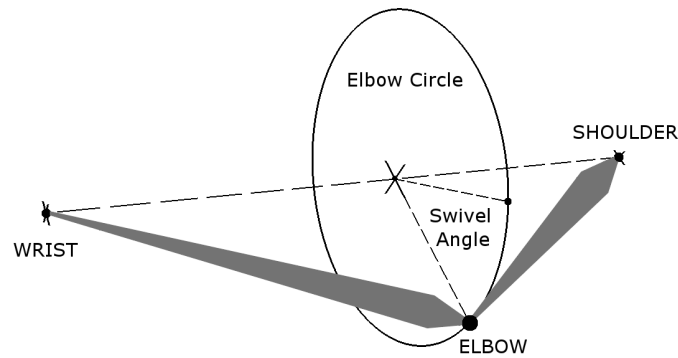


Figure 2.5: Analytical two-joint human limb IK. There is 1 degree of freedom in the solution, namely the swivel angle of the elbow.

There may be no analytical solution to posing joint chains with multiple end-effectors, so these general problems are best handled with numerical solutions [ZB94, BK05].

IK can also be used to position the centre of mass (COM) of a chain of joints, since the COM position is a linear combination of the mass centres of the links in the chain. Boulic et al. use an iterative solver to position the COM [BMT96], allowing control of balance.

In Chapter 4 we achieve a similar result, using composable constraints over a set of vertices that make up the character skeleton rather than a traditional joint hierarchy.

Style IK

It is well-known that IK has limitations in synthesising realistic human motion. Players of video games note a ‘robotic’ quality to motion constructed by IK, with one particular problem being ‘knee pop’ [KGS02] which is a loss of smoothness in movement that occurs when the IK leg is stretched toward its maximum possible length (Figure 2.6). This kind of artifact is a consequence of using relatively simple algorithms to try to reproduce the motion of a system as complex as the human body.

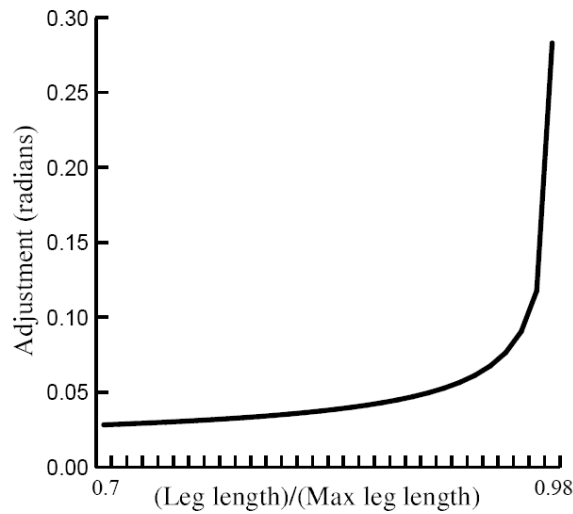


Figure 2.6: Knee pop. Graph shows the adjustment in knee angle necessary to extend the leg by 1% of its maximum length versus the current length of the leg, expressed as a percentage of its maximum length. The bend in the graph is the point at which the knee ‘pops’ - something which is not apparent in a human knee due to compliance and tension from muscles and ligaments. Source: [KGS02]

Work has also been done into incorporating motion style into IK positioning, using motion data from human actors to inform the IK solution so as to find more plausible poses.

Rose et al. formulate IK as a direct blend of animator-created poses [RPpSC01], while Grochow et al. use a training set of motions to build up a statistical model to reproduce similar motions from keyframes of the character’s trajectory [GMHP04]. Carvalho et al. construct a ‘motion pattern space’ [CBT07], parameterising the motion by its principal components and reconstructing motions by end-effector keyframing.

Principal components in the tangent space of orientations, or *principal geodesics* are used by Tournier et al. [TWC⁺09] to reconstruct a motion from its end-effector trajectories. We revisit this method in Chapter 5 and attempt to improve upon it by using a larger set of motion data as input; by constructing principal geodesics for subsets of the skeleton and recombining them in the solver, we can synthesise poses that are more robust over a greater variety of end-effector target locations.

Retargetting

Motion retargetting, where motions are mapped from the original character setup on which the motion was performed to another character setup, incorporates elements of both IK and *motion warping*, which will be treated in the next section. Gleicher [Gle98] retargets motion to a character with a different geometry from the original by preserving constraints that are char-

acteristic of the motion. The technique of Monzani et al. [MBBT00] can handle topologically different characters (where the number of bones is varied) as well as geometrically different by first mapping the motion to an intermediate skeleton whose features can be bound to both the performer skeleton and the end-user skeleton.

2.4.2 Motion warping

The purely kinematical approach to motion editing is known as *motion warping*. In the eponymous paper by Witkin and Popovic [WP95] a motion is edited by scaling and adding displacements to the motion curves. Motion curves in the paper are a set of 1-dimensional cardinal splines, each representing the value over time of a single degree of freedom of the character, also known as motion signals. Editing the motion in this way so that it passes through user-defined keyframes is conceptually simple and fits well into typical animation pipelines. However, there are some limitations, particularly the difficulty of enforcing geometric constraints between keyframes.

To address this limitation, Gleicher and Litwinowicz introduced a nonlinear constrained optimisation approach to motion editing [GL98, Gle97]. This numerical technique allowed kinematic constraints such as pose and foot placement to be enforced at certain times during the motion while maintaining minimal deformation from a reference animation by constructing an additive motion curve, or *displacement map* with the desired properties. Constrained optimisation will be dealt with in more detail in section 2.4.4.

Displacement maps can also be constructed hierarchically [LS99, CBSGo8]. This is a logical progression of the representation of motion at a spectrum of frequencies, and allows preservation of fine motion details when the larger-scale features of the motion are being altered. This approach has the same limitations as the original motion warping technique, so typically has to be augmented by an IK solution that enforces the kinematic constraints. For this reason, IK is an important component of most non-optimisation based motion warping techniques. Continuity of motion can be preserved by blending the IK solution in and out over a window of several frames around the constrained frames [KGS02, KKS04].

2.4.3 Dynamics

Articulated body dynamics

The human body is composed of semi-rigid segments linked by mobile joints. Its dynamic characteristics can be approximated by an *articulated body*, which is a system of perfectly rigid bodies connected by idealised joints. In such a reduced-coordinate representation the joints are parameterised by sequences of single-axis rotations (for revolute joints) or translations along

an axis (for prismatic joints, which are less common in human character representations than in robotics). The joints of the human body have complex movement on a small scale, exhibiting compliance and limited ranges of motion, but can be approximated to an acceptable degree of accuracy by idealised revolute joints of appropriate dimensionality, as mentioned in section 2.2.1. Simplified representations of the human body as a collection of linked rigid bodies have been used for years in video games, as in the now-accustomed *ragdoll* character motion.

Algorithms for acyclic linked multibodies due to Featherstone [Fea00, Fea07] have dominated the character dynamics literature [Vano4]. Featherstone’s three main algorithms are: the recursive Newton-Euler algorithm for a general tree-structure mechanism, which calculates the velocity and acceleration of the bodies from root to tip of the tree, and then joint torques from tip to root; the composite rigid body algorithm which calculates the joint-space inertia matrix (the inertia of the combined outboard bodies at each joint); and the articulated body algorithm which finds the accelerations of the bodies in the tree given the forces on each body.

Other formulations exist, including Baraff’s method for forward dynamics [Bar96] which is not based on the reduced-coordinate approach, but instead adds joint constraint forces and external contact forces in the form of Lagrange multipliers. Video game physics engines typically use a similar approach, simulating ragdoll characters as a system of rigid bodies with constraints. This approach also allows efficient and conceptually simple treatment of kinematic loops and is well-suited to general articulated bodies. However, it may be less suited to motion synthesis by spacetime optimisation due to the increased dimensionality (adding a variable for each Lagrange multiplier as well as the maximal variables for the rigid body states).

Anthropometry

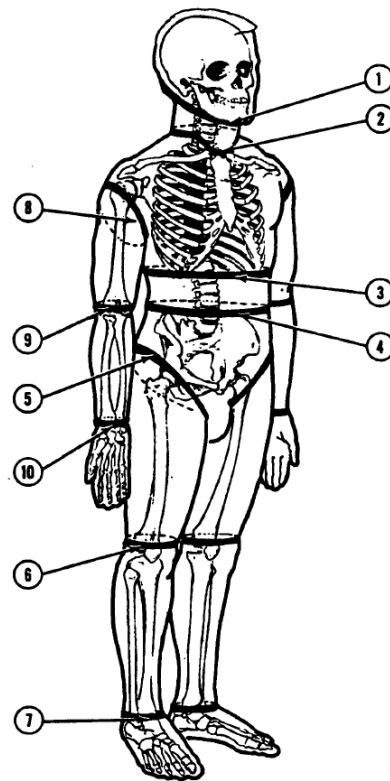


Figure 2.7: Segmentation of the human body. Source: [Lab88]

Approximating the dynamics of the human body requires a model of the mass parameters of the various body segments (Figure 2.7), including masses, centres of gravity and moments of inertia. There are several sources for such data, including measurements made on dissected cadavers [WTD67], estimations from live military personnel [Lab88] and the results of non-invasive scans on volunteers [dL96]. Bjornstrup [Bj095] provides a historical background for these measurements, and Zatsiorsky’s book [Zato2] contains regression equations for body segment mass parameters based on height for average builds. Alternatively and usefully, mass parameters can be inferred directly from the animation data [SKGo3]. Liu et al. go further, extracting a vector of physical parameters including joint compliance and elasticity [LHP05] by a complex ‘inverse optimisation’.

Reactive dynamics

Dynamics can be incorporated into animation by blending a physical reaction to external forces of perturbation with some reference motion, typically from mocap.

Zordan and Hodgins [ZHo2] use a proportional-derivative (PD) controller to track mocap data and a balance controller to keep their character upright while reacting to physical collisions. Later, Zordan et al. [ZMCF05] enhance this approach by allowing the physical simulation to take over (ragdoll style), and then searching for the best re-entry point into the motion library playback. This is an approach that is in evidence in many modern video games that blend ragdoll with keyframed animation. Komura et al. use a similar search approach [KHL05], but instead of allowing the ragdoll to take over they use the results of the dynamics simulation to guide a choice of reactive mocap clip. Mitake et al. also link the results of a physical simulation to a choice of keyframed animation output [MAA⁺09]. Their emphasis is different, with a considerably simpler character and a simulation consisting of only an inverted pendulum. The result is nevertheless effective and endearing. The inverse pendulum model has been used elsewhere for reactive dynamics, distilling as it does the dynamics of an upright stance into the simplest form. Tsai et al. [TLC⁺10] use the inverse pendulum as the basis for control of reactive bipedal locomotion, solving a linear complementarity problem (LCP) for forward simulation and tracking control and contact forces simultaneously. They use velocity-based tracking of reference motion to ameliorate the usual problem of setting gains for position-based PD controllers.

Ye and Liu use principal components analysis to determine the character's degrees of freedom with the least actuation [YLo8]. They then formulate a constrained optimisation to solve for poses that satisfy the equations of motion for perturbative forces in those degrees of freedom while leaving unchanged the main motion of the character.

2.4.4 Constrained optimisation

Synthesising human motion is a high-dimensional and highly underconstrained problem. Just as in IK, where there can be an infinite number of solutions for the set of joint angles resulting in a particular end-effector position, there may be an infinity of possible solutions for the joint forces that result in a certain root trajectory or sequence of footsteps. In addition to specifying kinematic constraints such as footsteps, key poses and trajectories [Gle97], we may wish the motion to have certain properties such as smoothness, similarity to a reference motion or minimisation of kinetic and potential energy [WK88]. Furthermore, some approaches to motion editing enforce dynamic realism by applying the Euler-Lagrange equations of motion as *physics constraints*, ensuring that the motion obeys physical laws from frame to frame.

The mathematical framework for solving such problems is *constrained optimisation*, where a problem is cast in the form:

$$\min_{\mathbf{x}} f(\mathbf{x})$$

subject to:

$g_i(\mathbf{x}) = c_i$ for $i = 1, \dots, n$ equality constraints

$h_j(\mathbf{x}) \leq d_j$ for $j = 1, \dots, m$ inequality constraints

where $\mathbf{x}$ is the vector of variables (usually the concatenation of the joint angles $\mathbf{q}_0, \mathbf{q}_1, \dots, \mathbf{q}_N$ for the N frames of motion) and $f(\mathbf{x})$ is a scalar function called the *objective*. The general case where $f(\mathbf{x})$ is a nonlinear function results in a *nonlinear program* or NLP. The most common and efficient classes of solvers for NLPs are *sequential quadratic programming* or SQP [GMMS97] and interior point methods [WBo6]. An object-oriented implementation of these different classes of solvers and some comparisons are provided by Blaszczyk et al. [BKMo7].

SQP methods break the solution of a general NLP into a sequence of subproblems by linearising the constraints and making a quadratic approximation of the objective at each step. These *quadratic programs* (QP) themselves comprise an important class of problem of the form:

$$\min_{\mathbf{x}} f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T Q \mathbf{x} + \mathbf{c}^T \mathbf{x}$$

subject to one or more constraints of the form:

$$A\mathbf{x} \leq \mathbf{b}$$

$$E\mathbf{x} = \mathbf{b}$$

where, again, $\mathbf{x}$ is the vector of variables and A , E and Q are matrices. If Q is symmetric positive-definite, the QP is said to be *convex* and can be solved in polynomial time. Convex QPs are equivalent to another class of optimisation problem, the linear complementarity problem (LCP) [Wrio1]. Among other places, LCPs crop up in collision resolution for rigid body animation. QPs also allow for versatile treatment of the constraints. A hard linear constraint $\mathbf{c} = 0$ can be turned into a soft objective $\mathbf{c}^T \mathbf{c}$ to be minimised [JYL09].

Nonlinear optimisation is computationally expensive, and often the most expensive part is the calculation of the Jacobian and Hessian matrices of the objective and constraints. This can also be extremely tricky to implement for coupled systems such as articulated bodies. To this end, Gleicher and Witkin devised ‘snap-together mathematics’ [GW93], where the dynamics of a system such as an articulated body is composed of blocks which can generate their function values and derivatives. This conceptually simplifies the construction of such systems, allowing for much faster prototyping.

2.4.5 Spacetime optimisation

Witkin and Kass [WK88] were the first to suggest optimisation as a general and automatic method for computing controls for a motion trajectory. They used an SQP solver to minimise the energy expended over a trajectory while enforcing certain contact constraints.

Energy minimisation seems to give good results for naturalistic motion, but has not been adequately demonstrated to be the principle on which human motion is based. Furthermore, it does not necessarily capture the style of a particular motion. This limitation was addressed by Liu et al. [LHP05] who used an iterated optimisation to estimate a set of parameters capturing relative preferences for using some muscles more than others; elastic mechanisms at joints due to the mechanical properties of tendons, ligaments and muscles; and variable stiffness at joints.

In the original Spacetime Optimisation paper [WK88], the character Luxo the jumping lamp demonstrated anticipation of impact by stretching out to minimise energy on impact. This is one of the strengths of using optimisation for a whole motion over simpler per-frame approaches. The trade-off is in computational complexity. Spacetime optimisation tends to take on the order of minutes even for short motion clips. Consequently, much of the effort in subsequent papers in this lineage is invested in simplifying the high-dimensional motion synthesis problem for efficiency. The main categories of simplification are outlined in the subsections below.

The recent work of Mordatch et al. [MTP12] is a spacetime optimization approach that sets the ‘weight’ of each potential contact as a free variable in the optimization. A motion is regarded as having ‘phases’ where contact configuration is constant. Using the ‘weight’ of each contact as an independent variable in the optimization allows the option to constrain or unconstrain contacts during any motion phase. The contact state for a phase then falls out of the spacetime optimization. The phases and possible contacts still have to be determined in advance, but this is a step toward automatically discovering complex motions.

Aggregate forces

It was discovered that, given that all the joints in a body are actuated, solving for the exact torques at each joint was unnecessary, and only the aggregate dynamics of the character need be considered when solving for its trajectory. Recursive equations for calculating the aggregate force and torque on an articulated body were introduced by Fang and Pollard [FP03]. The low-order dynamics constraints of Sulejmanpasic and Popovic [SP05] were equivalent to this formulation, enforcing the Euler-Lagrange equations of motion on the character’s root joint.

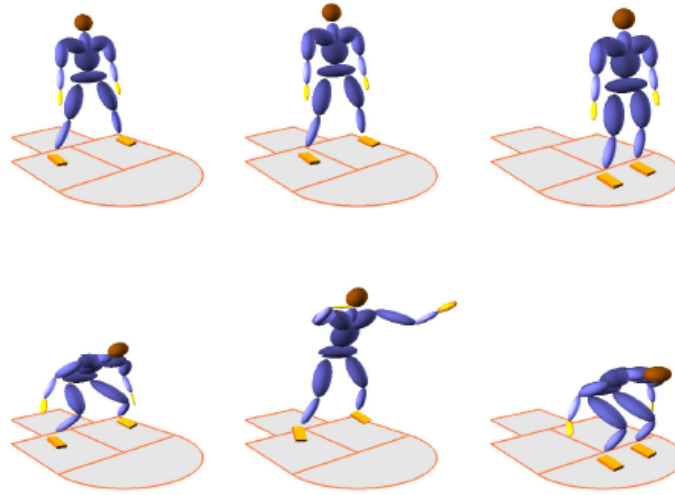


Figure 2.8: Leaping motion synthesised by spacetime optimization from a simple initial motion. Source: [LPo2]

This was also effectively the framework used by Liu and Popovic [LPo2] who only considered the momentum of the whole character for the initial trajectory synthesis (Figure 2.8). The per-frame, optimisation-based motion synthesis approach of Jain et al. [JYL09] used aggregate force constraints on the character root, with added enforcement of muscle force (i.e. joint torque) smoothness to obtain natural-seeming motion.

Heuristic momentum curves

Both velocity and force can be obtained from momentum, so the momentum trajectory over a motion includes all the dynamics. Liu and Popovic [LPo2] divided a motion into constrained stages, where the character was in contact with its environment, and unconstrained, or flight, stages. They controlled the momentum to remain constant (up to the force of gravity) in flight stages and to follow a heuristic pattern modelled after biomechanics data during constrained stages. The pattern mimics the transfer of momentum in biomechanical systems achieved by storing energy (decreasing the momentum) and releasing it in a burst causing small overshoot (Figure 2.9). Abe et al. [ALPo6] based their motion editing on a similar momentum curve scheme, but obtained their curves by fitting a cubic b-spline to the reference motion.

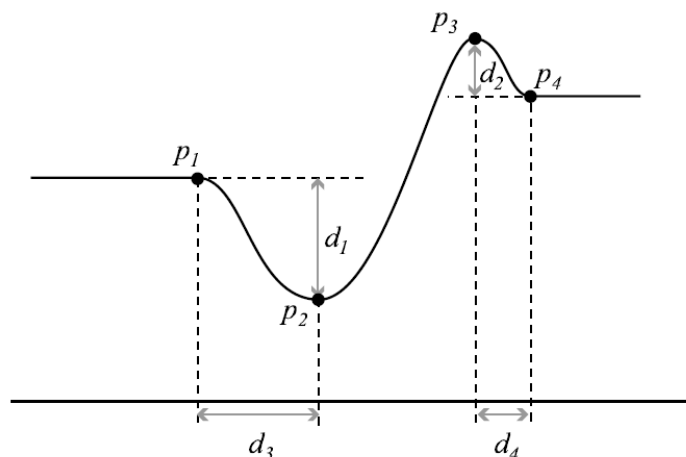


Figure 2.9: Momentum curve modelled after biomechanics data. Source: [LP02]

Levels of detail

Simplifying the computation of character dynamics by *levels of detail* (LOD) can be effective. Popovic and Witkin constructed simplified character representations tailored to particular motions [PW99]. The lower-DOF characters were specified manually to capture the essential properties of particular motions, for instance a jumping motion represented by a character with a single main prismatic joint representing leg extension and its upper body reduced to a single mass point (Figure 2.10). They then had to map the new motion created by spacetime optimisation back to the original full-detail character with a retargeting algorithm. A more general approach, aimed at forward dynamics simulation of large articulated body systems adaptively freezes subtrees of an articulated body based on their importance [RGL05]. Calculation of the inertia properties of the subtrees goes back to Featherstone’s composite body inertia algorithm [Fea00].

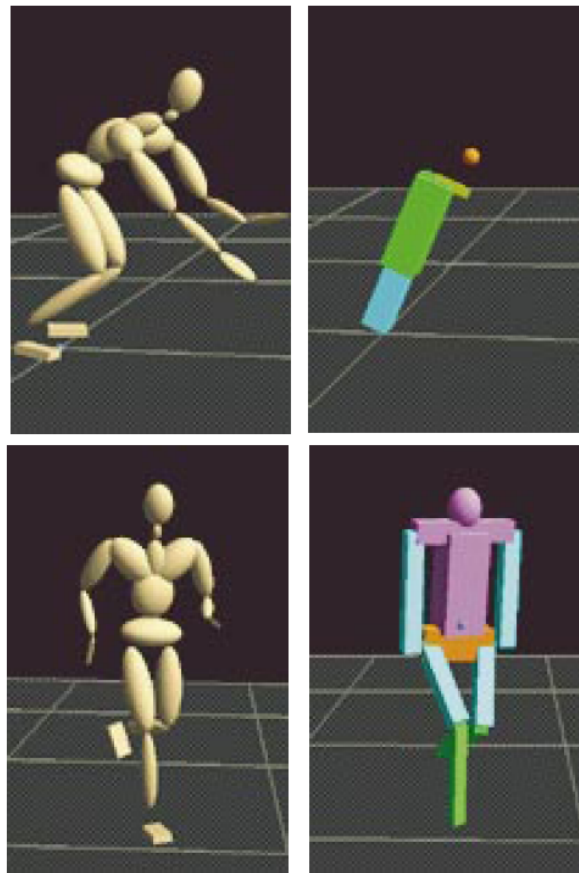


Figure 2.10: Character simplification: levels of detail for lower-dimensional optimisations. Source: [PW99]

Principal components

Another approach toward reducing the dimensionality of the motion synthesis problem is to invoke methods that capture the style or dynamics of the motion in a statistical model. Principal components analysis (PCA) was used by Safonova et al. [SHP04] to this end. They used a spacetime optimisation to synthesise the new motion based on a sketch, with the PCA providing a means to improve the convergence and efficiency of the optimisation.

2.4.6 Controllers

Controllers comprise a class of motion synthesis and editing based on determining control forces that result in the desired motion when integrated along with the forward dynamics of the system.

Constrained optimisation techniques have been applied to solve for control forces on rigid body systems, taking the systems through user-specified keyframes [BN88]. Discretising the

Lagrange-D'Alembert equations of motion for a rigid body system [JMOBo5] results in a high-dimensional *discrete optimal control problem*, equivalent to spacetime optimisation.

There are simpler and less computationally expensive ways to formulate control forces that result in flexible and efficient controllers. One such approach is *proportional-derivative* (PD) control for trajectory tracking. The control force is analogous to a damped spring, acting to reduce the error between the current system state and the desired state [Vano4].

It has been found that tracking the joint motion in model space (relative to the character's root joint) rather than local space (relative to the joint's parent) gives better momentum preservation [WJMo6]. This is due to the fact that the Hessian describing the system dynamics in world space coordinates varies less across poses than the corresponding Hessian describing the system dynamics in joint space [Wuo9].

Motion tracking can be formulated in terms of IK tasks [LCBo6a], with trajectory tracking for a manipulator being the primary task and posture preservation being the secondary task, projected into the null space of the primary task. Applying a similar methodology to joint-space inertia matrices rather than Jacobians results in control forces for forward dynamics [APo6].

Faloutsos et al. [FvdPT01] implemented a framework for composing individual controllers on a character. Their controller state machine is governed by a supervising controller that selects possible transitions based on a learned model of the acceptable pre-conditions necessary for each controller to execute. The composition method does not restrict the design of individual controllers. Transitions between keyframed animation and forward dynamics simulation with control forces [SPFo3] can leverage the utility of controllers and the quality of keyframed animation. NaturalMotion's Endorphin product [Nata] is (possibly uniquely) an example of games-industry middleware employing this type of framework (Figure 2.11). It is still something of a black art to engineer the controllers for such behaviours.

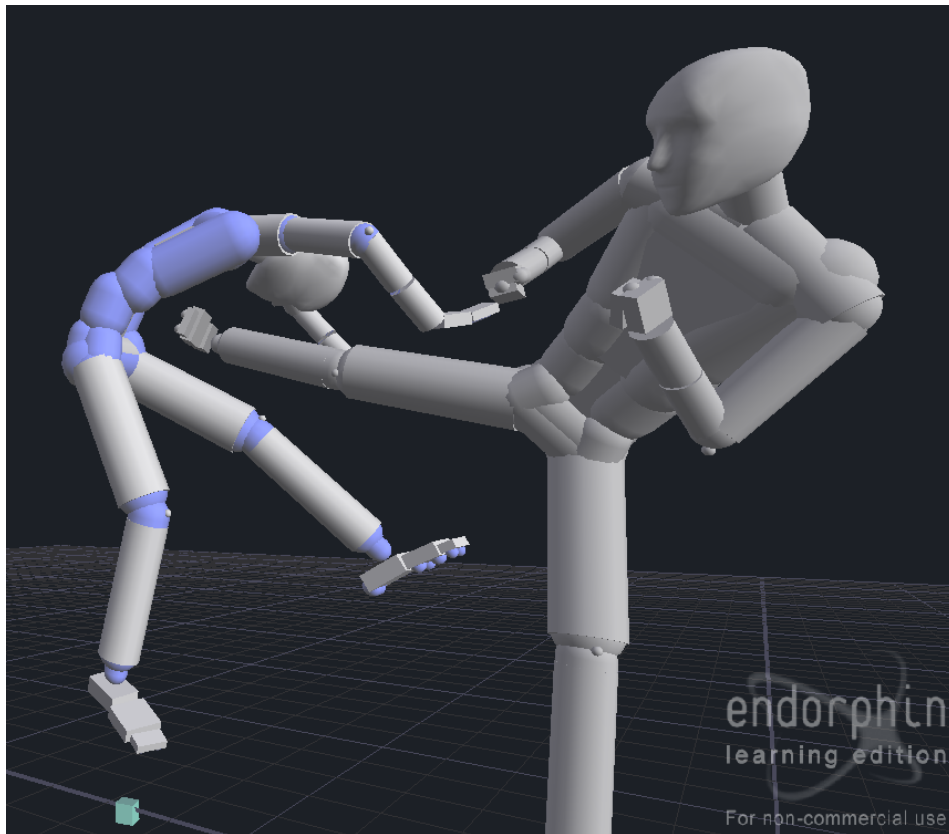


Figure 2.11: Controllers and keyframes in NaturalMotion’s Endorphin. The character on the left (in blue and grey) is being simulated, while the character on the right (grey) is keyframe-animated. Source: [Nata]

The work of Jain et al. [JYL09] also uses a controller state machine to synthesise motion. Here, constrained optimisation in the form of a quadratic problem is used to solve for joint angles per frame, enforcing the aggregate Euler-Lagrange equations of motion and kinematic constraints, and maximising joint torque smoothness for natural motion quality. This approach is similar to that used by Abe et al. [AdSP07], who use a quadratic program to compute actuation forces taking into account the limits imposed by the unilateral nature of contact constraints.

One limitation of per-frame controllers as compared to spacetime optimisation is the lack of anticipatory movement. Jain et al. mentioned this, suggesting that controlling the trajectory of an end-effector would provide a means to incorporate anticipation in their controller framework. This is a partial solution but has the drawback of leading to the usual problem with IK, namely a robotic-looking motion. Other attempts have been made to address this limitation [LNS05], but no automatic anticipatory motion that lives up to the standard set by constrained optimisation has yet been demonstrated by controllers.

2.4.7 Filters

Related to controllers is the class of motion filtering techniques. There is some overlap between the two classes but generally, filters take an input motion and edit it according to some principles (usually physically-based), as opposed to using forces and physical simulation to generate the motion.

Shin et al. [SKGo3] used standard IK together with constraints on momentum and balance to improve the realism of input motions. For the purposes of constraining momentum behaviour they partition the motion into ground frames and flight frames. There is no need to do this explicitly with our method. Instead, ground and flight phases emerge from the external kinematic constraints on each frame of the motion.

Tak and Ko worked on motion filtering to enforce dynamic balance constraints [TySKoo] and later more general physical equations of motion [TKo5]. At each step of their iterative solver constraints on momentum and balance are satisfied using a Kalman filter, followed by a least-squares solver that enforces kinematic constraints.

Pollard and Reitsma [PRo1] used a least-squares solver to enforce dynamical realism based on a model of frictional contact at the supporting end-effectors of the character.

2.4.8 Vertex-based motion

A body of relatively recent literature describes the use of systems of massive particles (or vertices) to represent the dynamics of deformable bodies, and the application of this scheme to character animation. The use of a vertex-based system for ragdoll characters was pioneered in [Jako1]. The ragdoll characters developed for the game *Hitman: Codename 47* had joints represented by points and connected by ‘sticks’ constraining the distance between them. Joint orientations were calculated such that, for instance, knees and elbows were not bending backward.

Kawachi et al. [KKSo4] used a similar character representation to [Jako1] for vertex-based animation editing. They adapted motion capture clips to have new kinematic constraints while ensuring continuity of position and velocity, but did not consider dynamics.

An efficient way to map from vertex positions to bone transforms was described in [SZT⁺07], where sets of four vertices forming ‘tetrabones’ encode general homogeneous transforms.

Hecker et al. [HRE⁺08] used the vertex-based character approach for their work on the game *Spore*, adding spline IK for the spine and simplified solutions for determining limb orientation.

Jakobsen’s technique was developed into a more general dynamical simulation scheme in [MHHRo7]. Any differentiable function over the vertex positions can serve as a constraint, allowing much greater expressive power in the construction of soft bodies. The authors described

constraints to control the bending stiffness, stretching stiffness and thickness of cloth.

A vertex-based view of characters allows for some novel thinking about interactions. Ho and Komura [HKT10] use a vertex-based approach to preserve the topology of an interaction between two characters. Continuously deforming the characters while maintaining the connectivity of an ‘interaction mesh’, they retarget complex intertwined motions, such as wrestling, to characters of widely-varying sizes.

We explore the vertex-based approach to IK and motion editing in Chapter 4. First we construct a character skeleton, complete with joint limits, from vertices and composable constraints developed from those in [MHHR07]. Later we extend the idea of these constraints to describe dynamical quantities over several frames of motion. We end up with a vertex-based spacetime motion editing technique, which we apply to adapting motion captured animations to altered contact constraints.

2.5 Evaluation of synthesised motion

Evaluation of edited motion must take into account not only dynamical realism but also a more difficult to define quality of ‘naturalness’ pertaining to human motion.

2.5.1 Objective metrics

Physically-based motion editing techniques enforce metrics of dynamical realism on the output motion. Optimisation techniques can enforce the appropriate equations of motion as constraints on the optimisation [AdSP07, JYLo9].

It is especially important that the motion of the centre of mass is correct, particularly as regards the conservation of angular momentum and acceleration due to gravity when unconstrained, and correct behaviour under aggregate external forces [FP03]. Another important measure of human motion realism is balance [SKGo3, AdSP07, JYLo9]. Typically balance is enforced by constraining the ZMP within the convex hull of support of the character [SBo4], as mentioned above in section 2.4.7.

Measures of dynamical realism can also be employed post-facto when validating edited motions. Pronost and Dumont create a biomechanical model and calculate joint and ground contact forces [PDo6], using the results in a comparison with direct measurements from force plates. Wei et al. [WXZo6] use an optimisation to correct unphysical in-flight motions such as jumps and flips. Their method is to minimise the aggregate forces on the whole body while constraining the difference between the original mocap and the resulting motion to be within certain bounds. Safonova and Hodgins consider linear and angular momentum, frictional foot contact and static balance in interpolated motions [SHo5], and devise a scheme for determining

interpolation parameters so as to maximise the preservation of physical correctness.

Measuring human motion style is difficult in general so another simple and common technique to get data-driven motion style is for the edited motion to track an input motion as closely as possible [PW99, PR01, LP02, WJMo6]. Measures of motion similarity have also been discussed previously in section 2.3.2.

Statistical methods can be used to evaluate human motion quality [RPE⁺05] and synthesise motions with a data-driven style [GMHP04]. But large training sets are needed, on the order of several hours' worth of motion clips.

2.5.2 Dynamical correctness vs. plausibility

To some extent dynamical correctness is subsumed by natural motion style, since an unphysical motion tends to seem unnatural. However there is some wiggle room, as the human eye is less sensitive to certain kinds of physical anomaly, small changes may not be noticed at all, and sensitivity seems to decrease for more highly dynamic motion. Perceptual studies of physically-based motion indicate the rough overall sensitivity of the human observer to errors in various properties of motion [Wu09]:

- Static Balance: High
- Dynamic Balance: High
- Linear Momentum: High
- Linear Velocity: Moderate
- Angular Velocity: Moderate
- Angular Momentum: Low
- Conservation of Energy: Low

The spacetime motion editing method we describe in Chapter 4 attempts to preserve these quantities directly by incorporating measures of static and dynamic balance, and measures of deviation from the Newton-Euler equations of motion as cost functions over the character's motion.

Some specific results from various papers are as follows: expansion of the angle between post-collision trajectories was preferred to contraction and increases in velocity were preferred to decreases [ODGKo3]; if gravity is between -9.0 and -12.7 the error is not visible to human observer [RP03]; small changes in angular momentum and take-off angle are not perceptible [MF07]; change in angular momentum will not be noticed if the inertia of the body is high since the corresponding change in angular velocity is small [SH05].

Further results in perceptual evaluation of blended transitions [WBo4] and motion graph similarity metrics [vBEo9] show that a uniform blend over all the character's degrees of freedom better preserves physical coherence of motions and reinforce the finding that foot-skating is particularly to be avoided as it is perhaps the single biggest factor in destroying the perception of motion plausibility.

2.5.3 Visual evaluation

If approximate dynamical correctness is a necessary condition for natural motion style, it is not a sufficient condition. Attempts at quantifiable measures for naturalness have met with some success. In optimisation approaches, appropriate choice of objective minimised can give natural-seeming motion. Early papers usually chose a metric based on energy expended in control forces [WK88], but latterly researchers have settled mostly upon smoothness of joint torques as an objective giving reasonable results [JYLo9, PW99].

Simple metrics cannot completely capture the complexity of human motion style so subjective evaluations are common in the motion editing literature. As noted before, the human eye is possibly the most accurate instrument for the perception of natural motion quality. As a result, many researchers rely on a subjective visual assessment of the naturalness of the edited motion [SHPo4, SPo5, PRo1, PW99]. Most motion editing papers include some sequences of animation frames presented as illustration [SHPo4, TKo5, PRo1, FvdPTo1, YNo3, PW99, LPo2], while some offer visual comparison with similar sequences taken from mocap to act as a ground truth [SKGo3].

Certain concrete observations can improve the objectivity of visual appraisal. Many authors note the presence or absence of certain expected realistic behaviour in their edited motions - namely motion features like swaying or leaning to preserve balance, or compensatory motion under changes in momentum [TKo5, SKGo3, PW99, LPo2]. Furthermore, it is straightforward to observe the presence or absence of certain anomalies known to destroy the perceived realism of a motion [LS99, PRo1]. The most obvious such anomalies are foot slide [KGSo2], discontinuities of motion [SPo5], and the violation of joint limits and self-collision [UPBSo8]. The absence of these anomalies lends strength to the motion evaluations in the cited papers.

Chapter 3

Geometry-aware motion planning

Here we present work done on a system for planning motions in a geometric scene. The aim was to define a simple physically-motivated model of the contacts that drive a character's interaction with its environment, and to use that model as a basis for planning motion in real-time with high fidelity of registration between character and environment. The work in this chapter was done in collaboration with Nithin Tharakan, with some portions of code implemented by Jan van der Kamp.

We describe a static environment as a list of points where a dynamic character can interact. Each point is a location in the scene associated with a certain type of allowed interaction - for instance a foot contact or a hand grip (Section 3.1). This gives a means to decide whether a particular action can be performed by the character at a certain position in the scene (Section 3.2).

From a given starting state with the character situated within the scene, we can construct a branching tree of all the possible action sequences the character might perform (Section 3.3). We mark certain branches as infeasible if the contacts required are not present at those locations. Then we can take into account user input to select among the feasible actions in real time.

We used a pre-processing step to optimise the feasibility query, with the result that the technique as implemented is only practical for static scenes.

3.1 Docking model

We chose to represent the local interaction between the character and environment using a set of point-normal pairs in a local coordinate space. In real-time physics simulations this is known as a contact manifold; being an approximation of a real-world contact manifold which is a set of continuous surfaces over which contact pressure is applied. This contact manifold model gives a basis for the space of all possible forces that a character can exert at a particular

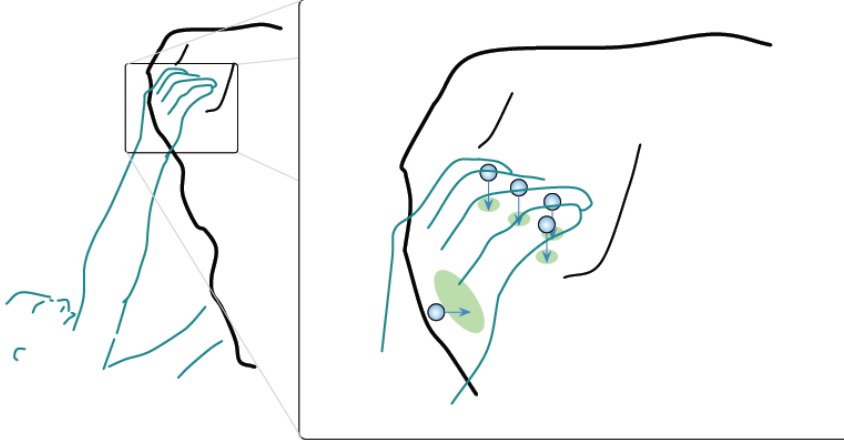


Figure 3.1: Illustration of hand docking showing point-and-normal pairs for each finger and the palm.

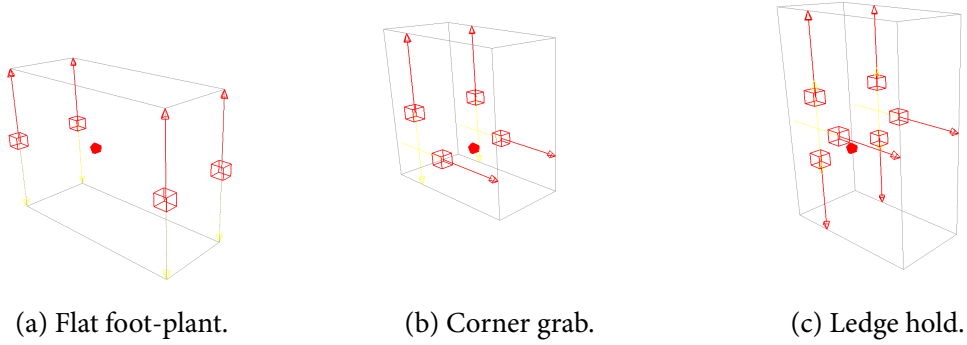


Figure 3.2: Example docking templates.

location.

A set of points-and-normals is sufficiently general to describe various kinds of interaction including footsteps, hand grips and so on. We call such interactions *docking* and this collection of points and normals a *docking template* (Figure 3.1). The local coordinate space of the docking template can be attached to an end-effector bone in the character’s skeleton.

To demonstrate the geometric query we used several different templates.

A docking model can describe various types of contact with different geometry (Figure 3.2).

This model can be generalised to more complex interactions such as full-body ground contacts or wrestling grapples. All that is required is a complete list of body sections in contact, together with a point-and-normal manifold for each. The typical *ragdoll* physical character model in use in many modern video games uses this model, since it is a collection of rigid bodies undergoing collision with its environment.

For the motion graph unrolling work in Section 3.3 we used a simple model of 2 contact points for each foot (heel and ball of foot) and a single point for each hand. For simplicity we

considered only animations where the end-effectors were the only parts of the body in contact with the environment.

Given a model of each contact, we need to determine when in the animation these contacts occur and add annotations to the animation describing the docking templates and relative time intervals of contact.

Speed thresholding is a simple and robust way to test for end-effector contacts. When the end-effector's world speed falls below some threshold for more than a certain number of frames we consider it fixed and assume it is in contact with the environment. Constrained intervals below a minimum duration are discarded and intervals broken by gaps of below a minimum duration are merged. The speed threshold and other parameters are adjusted until visually acceptable results are obtained.

Motion capture data can be noisy so we found that it is better to set the threshold high and manually prune false positives. This method has the advantages (over other more sophisticated approaches) of simplicity of implementation and robustness to noise.

Once we have determined these external constraints we have an 'action' that consists of

- A single animation:
 - ★ joint poses
 - ★ model-space trajectory of the character's root bone
- For each of the character's interactions with the world during that animation:
 - ★ a docking template
 - ★ the model-space coordinates of the dock
 - ★ the relative time interval of the dock

Here *model space* refers to a coordinate frame local to a given animation. An action can be situated in world space, by concatenating a world-space starting transform, and in time, by adding a world start time. Thus we can calculate where and when the character needs to come in contact with the environment. Next we must determine whether there is environmental geometry that can support those contacts.

3.2 Support test

For a given docking template at a given location in world space, we want to test whether the local scene geometry provides surfaces that can support the contact points in the model (Figure 3.3).

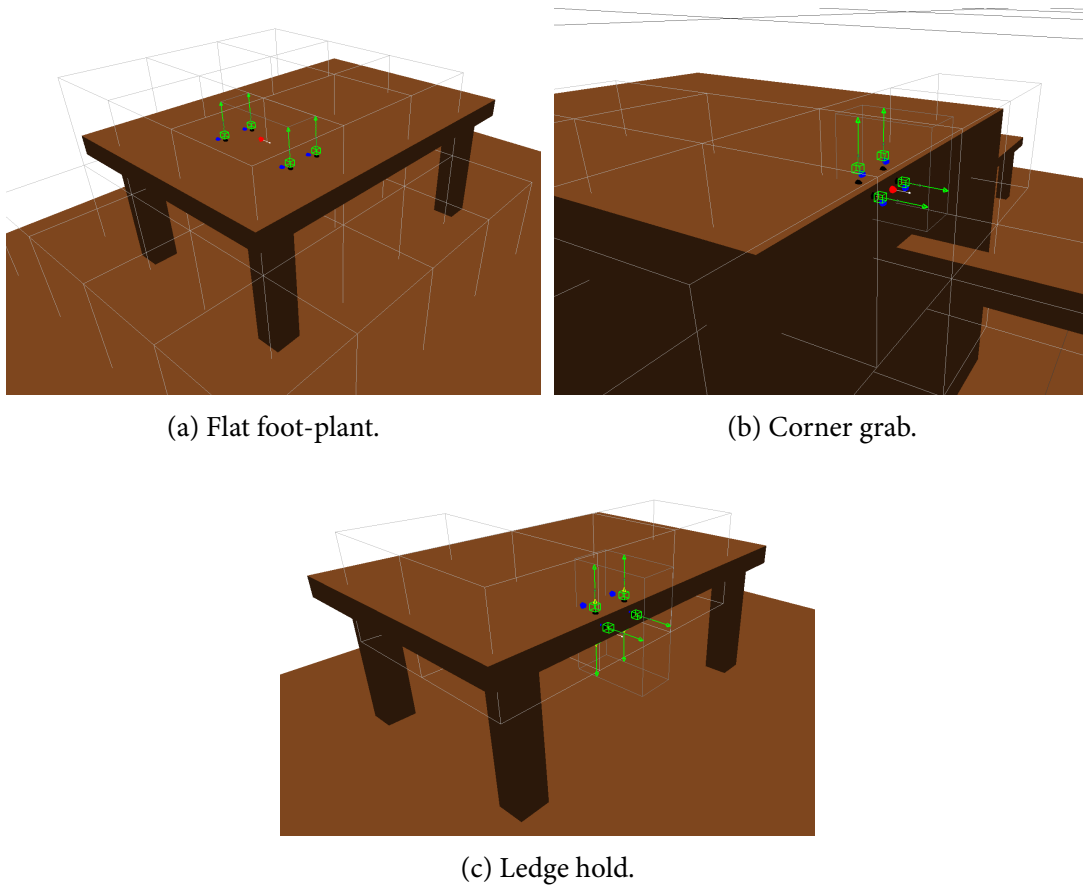


Figure 3.3: Local geometry test for each docking template.

Since the scene need not have been specifically designed to work with the animation, it is unlikely that the points of the dock will ever exactly coincide with the surfaces in the scene. The query for feasibility shown in Algorithm 1 allows some local variation in the position and surface normal of found contacts.

This algorithm is linear in the number of points in the contact manifold describing the dock, which is typically a small number. A naive approach would also be linear in the number of triangles in the scene geometry, so we used an octree bounding volume hierarchy to accelerate the query in large scenes. Each octree node that intersects with some triangles in the scene is subdivided, and the lists of triangles intersecting with its children are propagated. The root node encompasses the whole scene and is recursively subdivided until the nodes reach some minimum size. Each leaf node stores a list of triangles intersecting its bounding box (only the leaf nodes need store triangle lists in the final octree). The query then only needs to test a small number of triangles from a single leaf node, reducing the complexity to $O(\log n)$ in the number of triangles.

From the points of the contact manifold, we cast short rays in both directions along the lines of the contact normals. This is so as to pick up similar contact points on a surface that may have some irregularity, assuming the surface in the region of the query location is reasonably similar to the one described by the original manifold. If the local surface is too dissimilar, or if there is no surface nearby, the raycast test will fail and the sampling location will be deemed *infeasible*, or unable to support docking.

As long as the candidate contact point found by raycast is within some distance of the template's point, and the candidate surface normal is within a certain cone about the template's normal then we accept the contact. The distance tolerance can be set per point, say to take into account the relative mobility of different parts of a hand or foot. We used a tolerance of half the smallest extent of the docking template. The angular tolerance can be determined by the friction cones. The docking normal represents a direction along which force can be applied, which must lie within the friction cone of the surface. We used a 10 degree cone.

The test of support feasibility should also include some notion of spatial occupancy, so that the character does not end up intersecting with the scene.

We used a swept bounding box as a simple measure of the volume occupied by the character's torso during the motion. Intersections between this volume and the scene geometry were taken to indicate some obstruction that would prevent the character traversing the scene at that location.

More accurate solutions are possible, for instance using an existing collision-detection software development kit to represent the character as a collection of capsules and querying for collisions with the mesh representing the scene geometry.

Algorithm 1 Supporting geometry test (per dock)

```

for all dock's points do
  ray cast along normal in positive and negative directions
  if ray hit and hit distance is less than sampling increment and face normal is acceptable
  then
    record hit
  end if
end for
if number of hits equals number of constraint points then
  support exists at this location
else
  support does not exist at this location
end if

```

3.3 Planning a motion

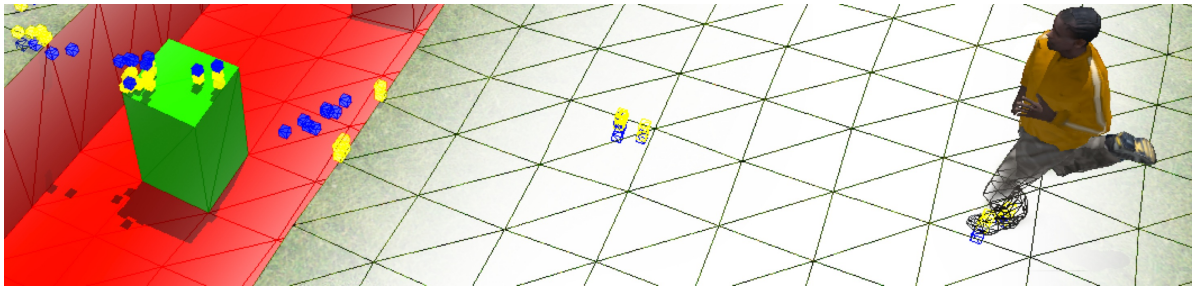


Figure 3.4: Planning motions by unrolling predicted contact points into an environment.

The above model of interaction between a character and static scene geometry can be used as the basis for planning sequences of motion (Figure 3.4). We assume we have a motion graph or other deterministic animation state machine, composed of actions as described above.

At a state transition (an edge in the graph) the next graph node can be located in world space by displacing it relative to the current node so that the root bone at the start of the action aligns with its ending location in the previous action. By traversing the graph, each successive action can be located in world space. We obtain a spacetime tree of potential actions in the scene, extending outward from our initial choice of location and action.

Each action in the tree must be tested for feasibility within the scene. We check the resultant world locations of the expected docks using our query to test that each dock is supported. The tree of potential actions is hierarchical since the reachability of a given node is dependent on all its ancestors being reachable so whole branches of the tree can be pruned away. There are two situations in which we can prune the planning tree at a node.

- if the node itself is infeasible

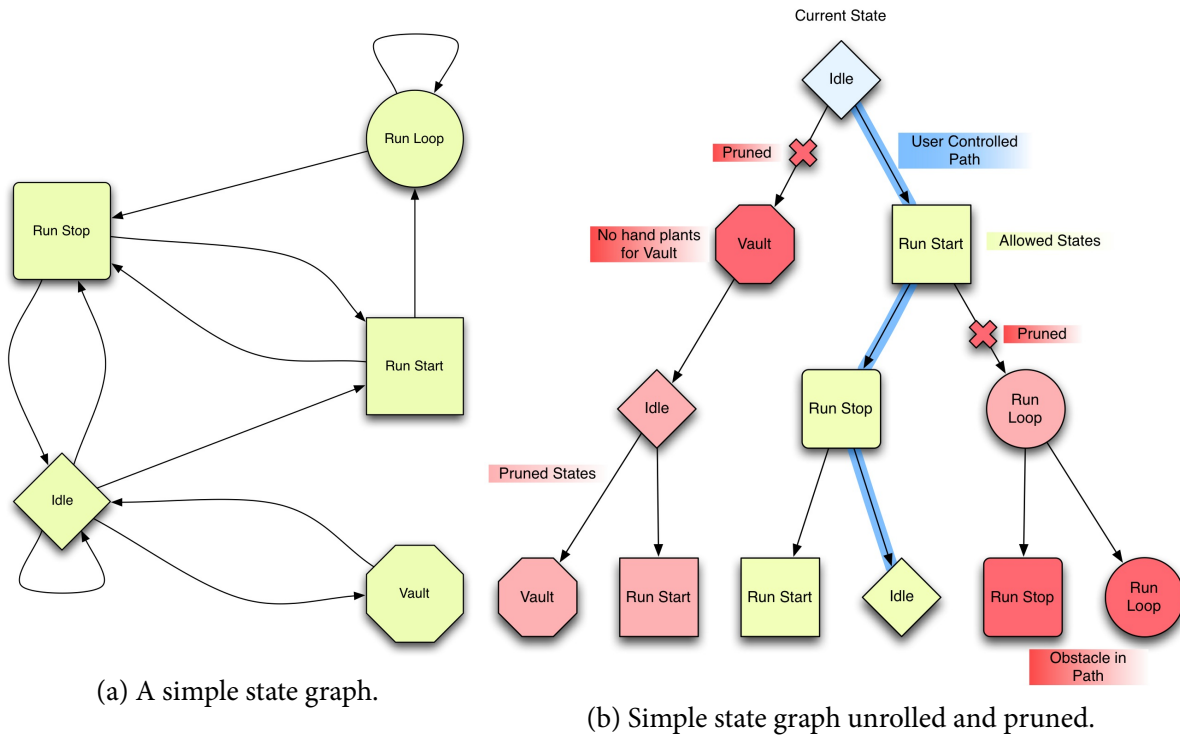


Figure 3.5: An example of motion graph unrolling resulting in a pruned spacetime tree of actions.

- if all the leaf nodes of the subtree below the node become infeasible

In both situations we can discard the pruned node and any descendant nodes (Figure 3.5).

The result is a tree of reachable actions. Planning a sequence of motions consists of selecting a particular path through the tree. As the depth of the tree increases, the number of nodes increases exponentially. The length in advance to which we can plan a motion sequence tends to be limited by the processing required to perform feasibility tests for all of the nodes.

3.4 Motion Atoms

One of the chief limitations of traditional motion graphs for real-time applications is their lack of responsiveness to user input. A desired transition from one action to another requires traversal of graph edges, introducing a lag between triggering the transition and executing the desired action. Shorter animations and denser connectivity would allow for quicker response to real-time user input.

Given some hypothetical method to adapt short sections of motion, we could increase the graph connectivity by splicing fairly dissimilar sections of motion. We propose that a minimal requirement for splicing motions in this manner is that the support states of the two motions be similar over the splice.

A definition of support state must include a set of body parts in contact with the environment (and therefore able to exert supporting force). It must also include some model of the forces that can be exerted, which varies by configuration. For instance, a hand grip while hanging from a bar differs from a hand plant while vaulting over an obstacle; and a transitory foot-push off a wall differs from a solid foot plant on flat ground.

Since we must test whether the next support state is feasible before committing to the next planned motion, a single change in support state represents the smallest interval over which it is possible to do motion planning. A motion can be broken down into atomic actions defined by a change in support state.

So in the context of geometry-aware planning of motion sequences we define a *motion atom* as being a short section of motion comprising a change in support state. In general, this could also be a null change as in the case of idling or gesturing.

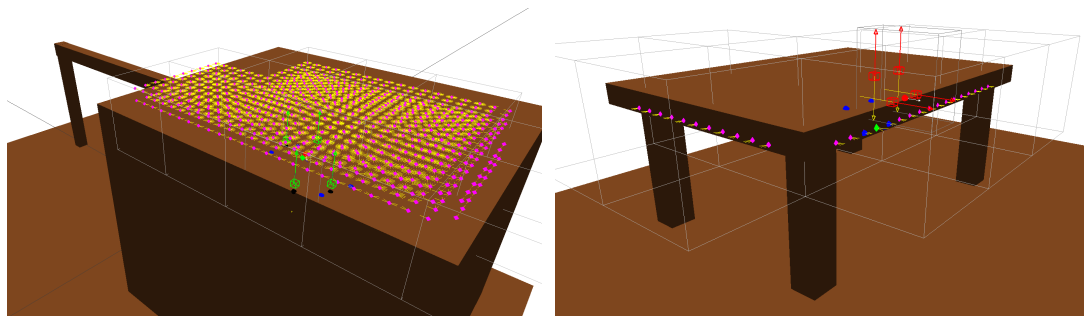
3.5 Pre-processing the scene

An action can in principle be performed anywhere in 3d space, so we wish to determine where over the whole volume of the scene each kind of docking is permitted. To make the problem tractable, we effectively discretize the scene volume by sampling at regular intervals.

We chose linear increments, the same on all three axes, roughly equivalent to the smallest extent of the docking template - typically 10cm. In sampling locations to place a foot for example, spacing your samples by the width of the foot will reasonably pick up all available footholds. At each sampling location we test the docking template at angular increments of 15 degrees about the up axis. It is necessary to rotate about the up-axis only as human motion is invariant under rotation about an axis along the direction of gravity but not other axes. A small amount of angular variation is taken into account by the docking feasibility query, but for instance a footstep on a significantly non-horizontal surface like a wall can be treated as a different kind of docking to the usual horizontal footstep.

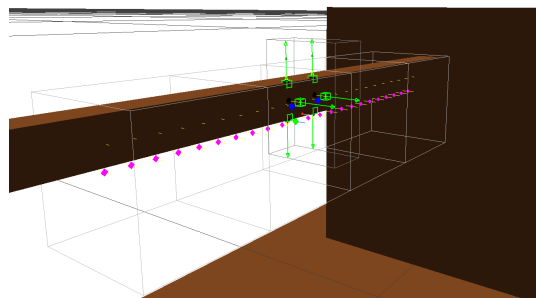
The raycasts performed need only be as long as the sampling distance, since scene features further away will be detected at other sampling locations. This allows some acceleration of the query by first performing a less costly bounding volume query against the scene to select candidate triangles for raycast.

For each docking template we now have a list of spatial locations $\{x, y, z, \text{angle}\}$ at which there exists supporting geometry, and hence at which the docking interaction can take place (Figure 3.6). We may now query this information to test whether a full animation can play out at any particular location in the scene.



(a) Flat foot-plant.

(b) Corner grab.



(c) Ledge hold.

Figure 3.6: Feasible spatial locations for each docking template. Allowed locations are marked in pink, and orientations marked with a short yellow line segment indicating direction after rotation about the world up-axis.

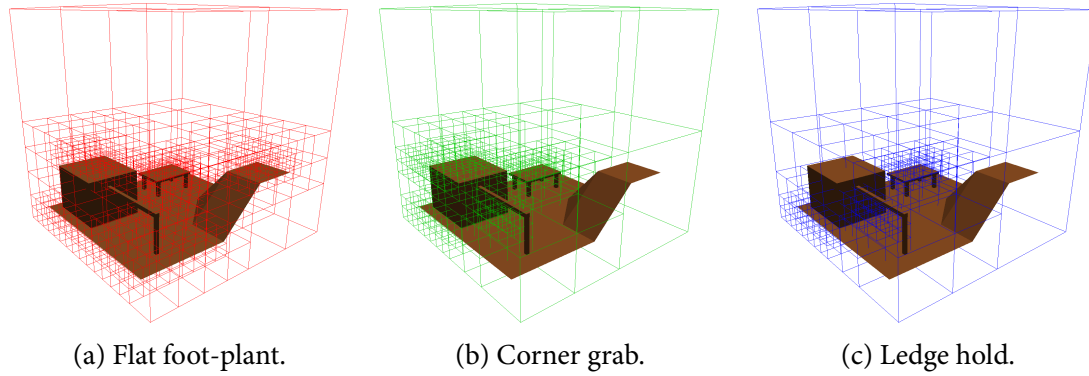


Figure 3.7: Test scene subdivided by a different octree for each docking template.

3.6 Querying a dock against the pre-processed scene

To accelerate querying for available docking sites we partition the world space using an octree. We have one octree per docking template, and the octree is built by subdividing nodes that contain feasible docking sites (Figure 3.7).

The octree is a bounding volume hierarchy that accelerates the search for docking sites near a given input position. We construct it by first placing an axis-aligned box around the scene, big enough to contain all the sampled locations. This box is regularly subdivided into 8 child boxes. The subdivision recurses for each child that contains docking sites, until some minimum box size is reached.

Searching for the nearest docking site to an input position is then reduced from a linear search of all the docking sites in the scene to a traversal of the octree hierarchy, finding the leaf node that contains the input position, followed by a linear search of a much smaller list of docking sites contained within the leaf’s bounding box.

Each octree leaf node has a list of docking site data $\{x, y, z, \text{angle}\}$. We can either store the actual contact points and normals with the docking site data, or reconstruct these at runtime by raycasting.

Let us take an example to illustrate the querying process. Suppose we wish to test whether a climbing animation can play at some world location. For a particular handgrip we have a docking template and a world position and rotation where the hand ought to grip something.

We wish to find nearby sites that support the dock so we query the relevant octree (corresponding to the docking template) to obtain a list of octree leaf nodes that intersect some bounding volume about the dock’s world location. The location of the chosen site should be as near as possible to the expected location of the dock so as to minimise the amount of motion deformation required to achieve a registration between the motion and the geometry. The bounding volume allows us to limit how much we allow the world position of the dock to vary. In our example, perhaps the climbing motion will seem implausible if the character reaches

some distance to the side for a handgrip, but still seems plausible if the grip is displaced some distance upward. This can be represented using a volume of the appropriate shape. For simplicity we used a sphere as a bounding volume.

Collecting the sampled docking sites from the leaf nodes, we find the closest fit to our desired dock by comparing the position and angle of the dock's world-space transform with the position and angle of each sample. If there were no nearby sites within the desired bounding volume we can reject the query, and say that the animation is unsupported at that world location.

3.7 Results

For our example application we constructed a motion graph as a state machine consisting of several locomotion and traversal motions.

Each action was subdivided into motion atoms consisting of a single change in docking state, with appropriate transitions between motions based on matching support states. For example, while standing on the left foot our character could transition from walking normally to walking with long leaping steps, stopping, turning, beginning a climb and so on.

Figure 3.8 is a visualisation of the potential transitions between the short motions in our state machine. When some of these motions are unrolled into the scene, the result looks like figure 3.9, showing the potential paths of the character's root bone as it traverses the motion graph.

We used simplified docking templates for the application, two points representing contacts at each foot and each hand (Figure 3.10). To adapt the motion to the displaced docks we used the simple fix-up method described in Appendix B.

At first we did not pre-process the scene. Instead we attempted a 'query and probe' approach where we probed the region around each expected dock by a similar sampling strategy to the one described above, performing the supporting geometry test query on the fly.

Later, by shifting the expensive probing operation into an offline pre-process, we made the feasibility queries more efficient. This did incur the memory overhead of having to store the octree structure with all the feasible samples contained in its leaf nodes.

3.7.1 Query and probe

Table 3.1 shows the total number of animations and the time taken to evaluate them (by probing for support locations) in an unrolled state graph at different unrolling depths. The results did not allow for real-time animation. Therefore we explored ways of embedding meta-data in the scene using a pre-process to achieve real-time performance.

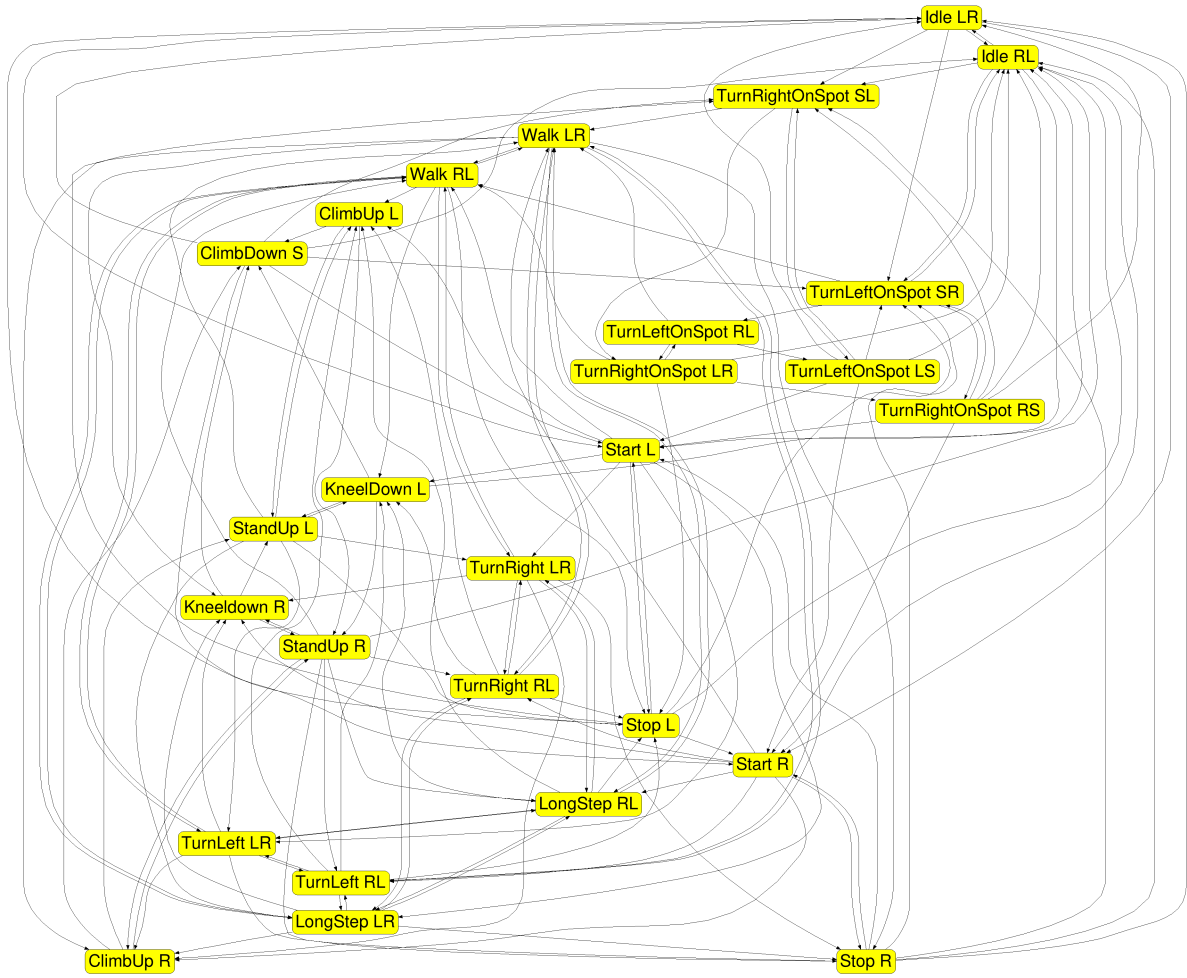


Figure 3.8: Directed graph of motion atoms.

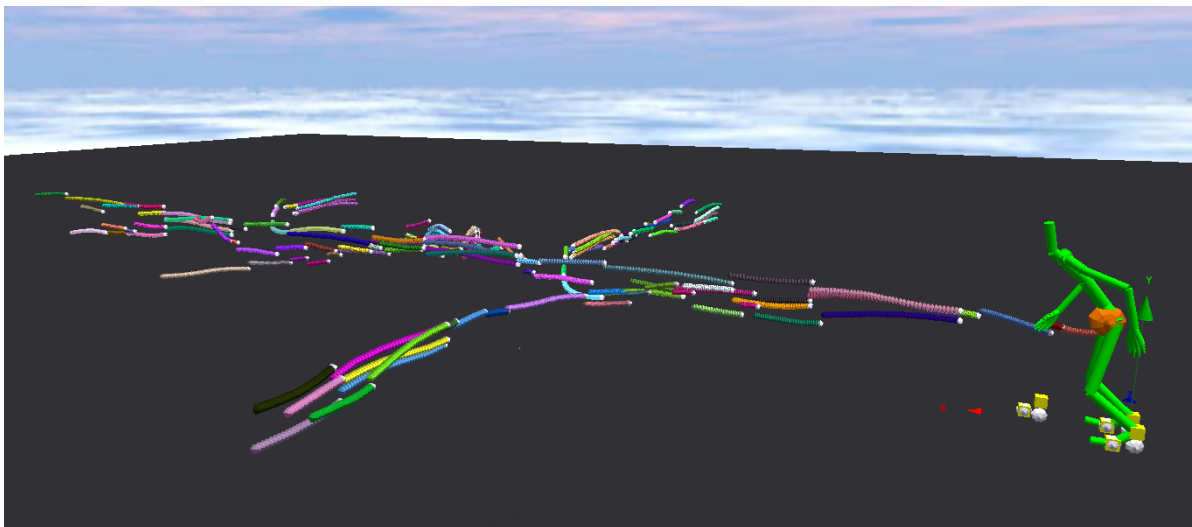


Figure 3.9: The character skeleton's root path over some unrolled motion atoms.

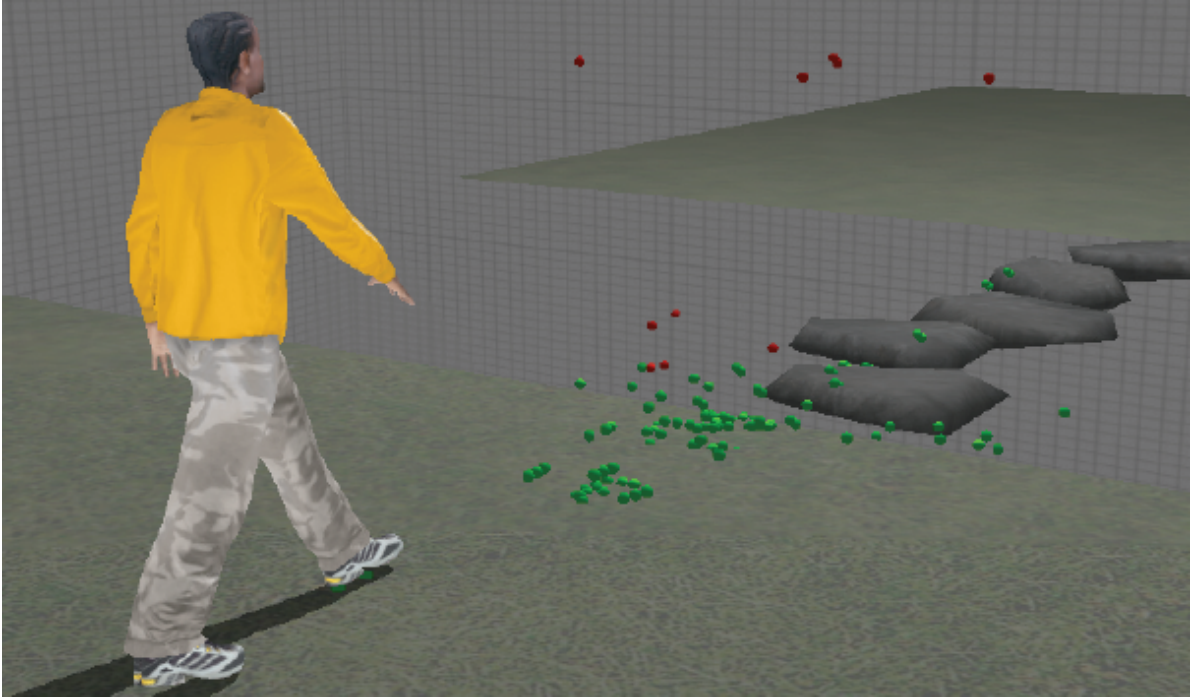


Figure 3.10: The character used in the application, with propagated docks in the scene showing feasible (green) and infeasible (red).

Tree Depth	Animations to Prune	Query & probe Total Prune Time (s)	Pre-processed scene Total Prune Time (s)
1	5	0.9754	0.0005
2	32	9.1953	0.0056
3	197	75.1461	0.0538
4	1236	588.1479	0.4969

Table 3.1: Animation tree depth vs. Pruning time.

3.7.2 Static scene pre-process

By shifting the expensive probing operation into a pre-process, we can make the feasibility queries more efficient but incur the overhead of needing to store the pre-processed data. The current algorithm also assumes static scene geometry.

Given the docking templates from our animation database we pre-process the environment to find regions where these docks are supported. We store this information in octree structures, one for each type of docking template, which we refer to as support-trees. Given an expected dock location we perform a nearest-neighbour search over the support-tree to find whether the dock is supported in or near that location. This search replaces the expensive runtime query-

and-probe support search with a fast octree traversal.

For a single dock the offline processing time and the number of ray-triangle intersection tests increases linearly with the number of dock points (Table 3.2). Detecting increasingly complex docks only incurs a cost in the offline processing time.

Dock Points	Ray-Triangle Tests	Processing Time (s)
2	29956064	5.8797
4	58514624	12.118
6	95037024	16.6224
8	123839360	23.6544

Table 3.2: Number of points per dock vs. processing time. The sampling was done at 0.1 m intervals in a scene with volume of 48 m³ and surface area of 213 m².

The memory required to store a single support-tree node was approximately 112 bytes. The number of support-tree nodes for a given docking template in a scene depends on the sampling density and the scene surface area (or more generally, the number of docking support locations). The memory required to store a full support-tree, in this case for a foot plant dock was measured for several scenes (Table 3.3). We choose a foot plant dock because it is a common requirement for all animations in our corpus.

Scene	Surface Area (m ²)	Volume (m ³)	Nodes	Memory (MB)
Stepping	213	48	13665	1.46 MB
Climbing	309	240	20009	2.14
Game Level	13240	127428	218017	23.28

Table 3.3: Memory usage for a support octree.

The game level geometry used here was the map ‘de_dust’ from the competitive multiplayer game Counter-Strike. The ‘Stepping’ and ‘Climbing’ scenes were simple geometry we created to test these specific actions with the example motion graph.

At runtime, the number of animations to evaluate and consequently the time taken increases exponentially with search depth as shown in Table 3.1. The average time to evaluate a single animation using feasibility geo-query however is approximately 0.1 ms, making real-time motion planning possible while remaining responsive to user input.

3.8 Conclusions

We have described a means to plan sequences of motion from a motion graph, based on a model of contact between the moving character and the geometric environment surrounding it. The stated requirement for real-time planning led to optimising the spatial search of potential contact support points by pre-processing the scene and storing the docking data in an octree structure. There are several limitations to this method:

- The method is very memory intensive, since we store a full list of contact locations at the leaf nodes of the octrees. For large scenes this can be a lot of data. However, it is not difficult to imagine ways to compress this data. For instance, contiguous regions denoting allowable contact over some surface in the scene could be represented much more efficiently as a set of polygons. This is similar to the standard *nav mesh* typically used for spatial navigation in modern video games. The octree acceleration structure can also be generalised to any type of bounding volume hierarchy, common in collision detection literature.
- Unrolling the motion graph into the scene quickly results in a huge number of animations whose contacts must be tested. This is a basic limitation of approaching motion planning this way, and means we could only plan about 4 motion atoms (equivalent to about 2 seconds) in advance.
- At the planning stage we can only handle deterministic and discrete motion graphs. That is to say blend trees, blended motion graphs or other procedural techniques producing a continuous space of output motions are currently incompatible with this method. Even with a discrete set of branching motion sequences, the set of contact locations that must be queried grows exponentially. It is not clear how to query for support over a continuous space of potential future motions since we do not have a compact representation of the corresponding loci of the resultant contacts.
- We are limited to a small finite number of pre-determined contact templates. Since each contact template's feasible support locations are stored in a separate octree, we have no provision for dynamically-determined contact shapes.
- The scene pre-process limits our method to static scenes, since the query acceleration structures must be built offline before user interaction takes place. It would be possible to extend the static scene to a partially-dynamic scene by adding moving regions that can be queried independently in addition to the static octrees.

We have allowed for a margin of error in the docking query, so the docking positions output by the planning stage may not align exactly with the contact locations marked up in the input

animations. The following chapters describe ways to deform or *adapt* the input animations so as to respect the docks found in the scene while maintaining the visual quality of the animation. We try some different approaches, each aimed at preserving a particular form of ‘visual quality’, whether that be dynamical realism or some measure of natural motion style.

The method in Chapter 4 is designed for small deformations, preserving dynamical realism and balance. We also propose the use of geodesics IK, in Chapter 5, as a potential means to achieve larger deformations while retaining plausible motion style. Appendix B describes a simple prototype system we implemented to adapt motions in order to demonstrate the motion planning method.

Chapter 4

Vertex-based motion adaptation

The problem that now presents itself is to take an input motion, with certain constraints such as external contacts, and adapt it in some way. The positions of those external contacts may have to be altered to conform to a new setting. We also wish to enhance or preserve the sense of physicality of the motion since this may be lacking from either of two causes: the input motion can come from IK-based motion synthesis that has no explicit metric of dynamical realism; or, altering the positions of contacts means altering the space of possible forces the character can exert to actuate its motion.

So we require a model of the dynamical equations of motion of the character to form an objective function for optimisation. This chapter explores the application of a *vertex-based* approach to character dynamics.

Rather than model the character as a reduced-coordinate articulated body, or even as a set of rigid bodies with joint constraints between them, the vertex-based approach uses a set of massive point particles (or vertices, the terms are almost interchangeable here) together with composable constraints. Each constraint is a differentiable function over the positions of some subset of the particles. Constraints describe spatial relationships between particles and as such can be composed to build up a character skeleton (see Section 4.1).

In a forward-dynamics simulation of such a system, the motion of the particles would be integrated forward in time and the constraints solved to preserve the structure of the character. Constraints are solved by visiting each constraint in sequence and projecting the positions of its particles along the gradient such that the constraint function becomes instantaneously zero. When every constraint has been visited we loop back over the whole list again, and iterate in this manner to gradually improve the overall solution. When solving multiple constraints in this cascading way, each individual constraint projection may be invalidated by the solution of the next. However, the whole system tends to converge toward a compromise state. This is not strictly an optimisation since there is technically no objective function being minimised, only a set of constraints. However, in practice the dynamics constraints act as objective functions to

minimise the deviation of the output motion from dynamical correctness.

We solve the vertex constraints as described, but rather than forward-integrating in time we instead represent a motion over time by considering multiple frames of motion for each particle. This allows us to introduce constraints with the same mathematical form that apply over multiple frames of motion, and in this way to represent the equations of motion as objective functions over the vertex positions.

4.1 Vertices and constraints

In this chapter we discuss our use of a class of iterative solver from soft-body simulation [THMGo4, MHHR07].

This type of solver describes deformable bodies like cloth or jelly using vertices, distributing the mass of the body among the vertices. The structure of the body is defined by constraints, each being a differentiable function of the positions of one or more vertices. The constraints are solved iteratively in a process analogous to Gauss-Seidel constraint solving, except that the constraint functions need not be linear.

This type of solver has previously been applied to IK for animated characters [Jako1, HRE⁺08]. Instead of a soft body the vertices and constraints make up an articulated skeleton. We extended this application by adding some novel constraints for skeleton structure and joint limits; and also for centre-of-mass and pose balance control.

We also applied the scheme to the problem of motion synthesis. We devised new constraints that apply not only to the positions of vertices but also to their velocities and accelerations, by considering finite differences in positions over multiple frames of animation.

The set-up for the vertex-based character skeleton consists of a set of vertices, together with a list of constraints. Each vertex has position $\mathbf{p}$, mass m_i and inverse mass $w_i = 1/m_i$ (we will use the term ‘particle’ to refer to a vertex with nonzero mass). A constraint is any differentiable scalar-valued function on any subset of the vertices.

Constraints are solved by iterative relaxation. In this solution scheme, there is no single overall objective function that is being minimised. Rather, each constraint function acts as a sub-objective, and is solved in isolation by projection of the vertex positions along the gradient, before moving along to the next constraint. This is the kind of cascading scheme employed by Shi et al. [SZT⁺07] and Stam [Stao9]. Since constraints visited later in the sequence may cause the effects of earlier constraints to be overwritten, higher priority constraints should be put last.

On each iteration, for each constraint in turn, the vertices are projected along a direction that minimises that constraint. Over several iterations, the vertex positions tend to converge to values that (approximately) satisfy all the constraints. For the full derivation of the iterative constraint projection scheme, and a proof that the scheme conserves linear and angular mo-

mentum, we refer to [MHHR07]. Here we will simply present in brief the equations needed for implementation.

4.1.1 Constraint relaxation

Consider a constraint with cardinality n , having scalar-valued function C over some subset $\mathbf{p}_1, \dots, \mathbf{p}_n$ of the vertices. We will denote it $C(\mathbf{p})$. An equality constraint is considered satisfied when $C(\mathbf{p}) = 0$, and inequality constraints when $C(\mathbf{p}) \geq 0$.

Given $\mathbf{p}$ we want to calculate $\Delta\mathbf{p}$ such that $C(\mathbf{p} + \Delta\mathbf{p}) = 0$ (we apply projection to inequality constraints only when their function value is less than zero). The vertex positions are projected along the gradient $\nabla_{\mathbf{p}}C$ as follows:

$$\Delta\mathbf{p}_i = -s w_i \nabla_{\mathbf{p}_i} C(\mathbf{p}_1, \dots, \mathbf{p}_n) \quad (4.1)$$

where the scaling factor s is given by:

$$s = \frac{C(\mathbf{p}_1, \dots, \mathbf{p}_n)}{\sum_j w_j |\nabla_{\mathbf{p}_j} C(\mathbf{p}_1, \dots, \mathbf{p}_n)|^2} \quad (4.2)$$

In general, the inverse mass w_i is an abstract quantity that defines the susceptibility of a vertex to projection by a constraint. It can be considered independently of any concept of vertex mass, and indeed in [HRE⁺08] a constraint holds its own value of susceptibility for each of its domain vertices. In the case where $w_i = 0$, the vertex is considered fixed relative to the world. This is useful for defining hard external kinematic constraints such as end-effector targets.

If we wish to ‘soften’ the constraint enforcement, we can choose to project each vertex by only some fraction of the calculated distance. This is useful for certain ‘smoothing’ constraints.

4.1.2 Skeleton structure

The basic structural element forming the rigid parts of the skeleton is the equality distance constraint with rest distance d :

$$C(\mathbf{p}_0, \mathbf{p}_1) = \|\mathbf{p}_0 - \mathbf{p}_1\|^2 - d^2 = 0 \quad (4.3)$$

We use the squared distance to avoid expensive square root calculations and to simplify calculation of the gradient:

$$\begin{aligned} \nabla_{\mathbf{p}_0} C &= 2(\mathbf{p}_0 - \mathbf{p}_1) \\ \nabla_{\mathbf{p}_1} C &= -2(\mathbf{p}_0 - \mathbf{p}_1) \end{aligned} \quad (4.4)$$

Constraining 4 vertices with 6 constraints to give a tetrahedron lets us define a bone transform (rotation and translation) using the *tetrabone* method of Shi et al. [SZT⁺07]. Here, the homogeneous coordinates of the 4 bone vertices form the columns of a 4×4 matrix $\bar{V}$. We can find the affine transform T of the bone in any deformed state relative to its original *bind pose* $\bar{V}$ by post-multiplying the matrix V formed using the deformed coordinates by the inverse of the bind matrix:

$$T = V \bar{V}^{-1} \quad (4.5)$$

This makes it straightforward to map the vertex skeleton pose to a traditional spatial hierarchy skeleton pose (figure 4.1).

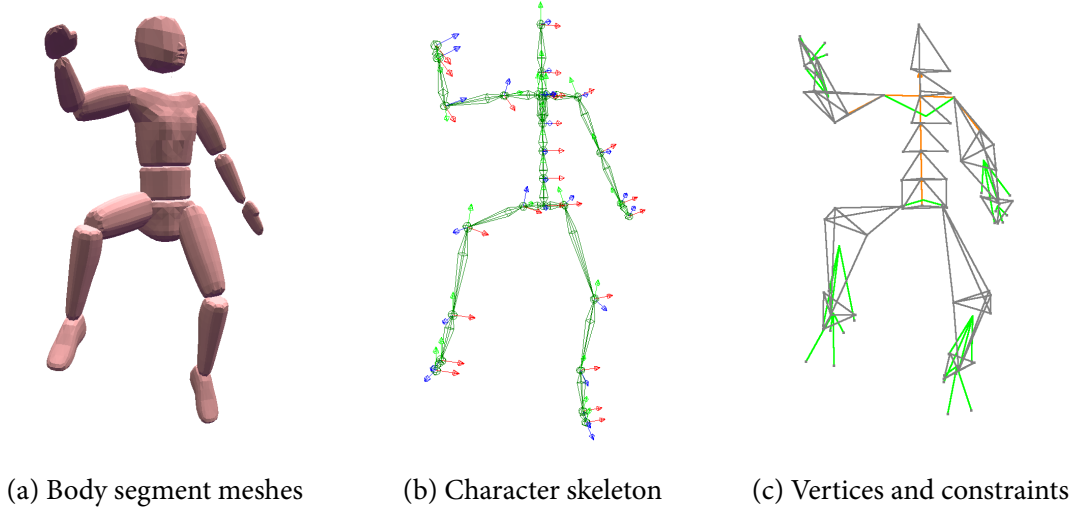


Figure 4.1: Vertex-based character.

A further *eversion* constraint prevents the bone tetrahedron from flipping inside out, as can happen under large deformations. The eversion constraint function

$$C(\mathbf{x}, \mathbf{o}, \mathbf{v}, \mathbf{w}) = ((\mathbf{v} - \mathbf{o}) \times (\mathbf{w} - \mathbf{o})) \cdot (\mathbf{x} - \mathbf{o}) = 0 \quad (4.6)$$

is similar to the tetrahedron volume preservation constraint from [THMGo4].

This constraint has gradient

$$\begin{aligned} \nabla_{\mathbf{x}} C &= (\mathbf{v} - \mathbf{o}) \times (\mathbf{w} - \mathbf{o}) \\ \nabla_{\mathbf{o}} C &= \nabla_{\mathbf{x}} C - \nabla_{\mathbf{v}} C + \nabla_{\mathbf{w}} C \\ \nabla_{\mathbf{v}} C &= (\mathbf{w} - \mathbf{o}) \times (\mathbf{x} - \mathbf{o}) \\ \nabla_{\mathbf{w}} C &= -(\mathbf{v} - \mathbf{o}) \times (\mathbf{x} - \mathbf{o}) \end{aligned} \quad (4.7)$$

Contiguous bones may share vertices, but we can also join bones together by constraining

vertices of one bone to have certain barycentric coordinates relative to another bone. A vertex $\mathbf{p}$ has barycentric coordinates (u, v, w) relative to a tetrahedron defined by vertices $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4$ if

$$C(\mathbf{p}, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4) = \|(1 - u - v - w)\mathbf{v}_1 + u\mathbf{v}_2 + v\mathbf{v}_3 + w\mathbf{v}_4 - \mathbf{p}\|^2 = 0 \quad (4.8)$$

Constraining a single vertex in this way gives a spherical joint for shoulder or hip. Constraining two vertices gives a hinge joint for knee or elbow. We also used this constraint to place end-effector vertices in the local coordinates of a bone.

We can pin a vertex to a position $\mathbf{d}$ in the world, or to another vertex position, using a zero-distance constraint:

$$C(\mathbf{p}) = \|\mathbf{p} - \mathbf{d}\|^2 = 0. \quad (4.9)$$

Thus, we can define external kinematic constraints on end-effector positions.

4.1.3 Joint limits

Limits on the motion of the joints can be achieved using inequality constraints.

To limit the distance between certain vertices we just use the unilateral version of the distance constraint:

$$C(\mathbf{p}_0, \mathbf{p}_1) \geq (\|\mathbf{p}_0 - \mathbf{p}_1\|^2 - d^2) * \text{sign}. \quad (4.10)$$

Positive sign gives a repelling constraint, negative sign an attracting constraint.

A constraint on the dot product between line segments allows us to limit the bending and twisting of certain joints. Given two line segments $\mathbf{a} = (\mathbf{v}_2 - \mathbf{v}_1)$ and $\mathbf{b} = (\mathbf{v}_4 - \mathbf{v}_3)$ with end-points defined by vertices $\mathbf{v}_1, \mathbf{v}_2$ and $\mathbf{v}_3, \mathbf{v}_4$ respectively, the function

$$C(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4) = \mathbf{a} \cdot \mathbf{b} - \|\mathbf{a}\| \|\mathbf{b}\| \cos(\theta) \geq 0 \quad (4.11)$$

limits the line segments to be within an angle θ of each other. When the vertices $\mathbf{v}_1$ and $\mathbf{v}_3$ coincide this produces a cone-shaped bending limit, as in the knees, elbows and spine joints. We used the more general case in the shoulder and hip to limit twisting with respect to the chest and pelvis, respectively.

At this stage we can construct a full character skeleton with joint limits. There is no actuation of the joints yet. Integrating the vertex positions over time, applying gravity force to each particle and solving the constraints at each timestep, gives us a ‘ragdoll’ simulation of character motion.

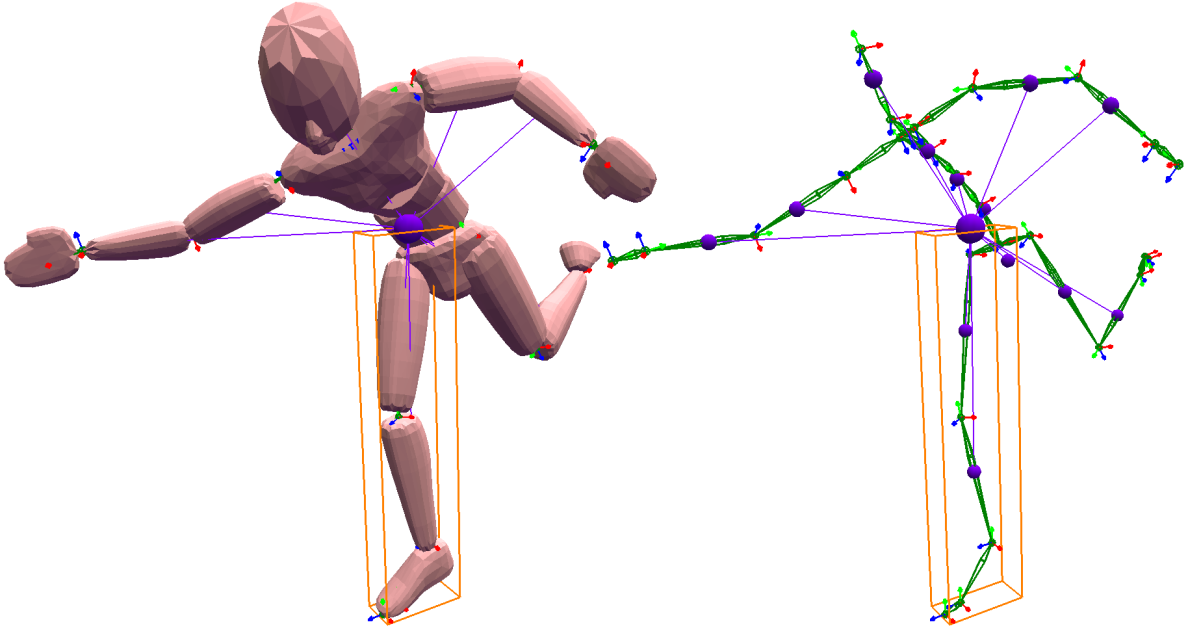


Figure 4.2: Centre-of-mass and static balance. Large purple orb marks COM vertex, smaller orbs mark COMs of body segments. COM is constrained to lie above foot contact.

4.1.4 Centre-of-mass and balance control

Adding a further constraint for the centre-of-mass of the whole character gives us the ability to control the character's balance.

Some of the skeleton vertices are assigned a mass for the purpose of representing the inertia of the character. We can attach additional particles using the tetrahedron barycentric constraint if necessary. Gathering the n massive particles $\{\mathbf{p}_1, \dots, \mathbf{p}_n\}$ into a total mass centre $\mathbf{c}$ for the whole character defines a constraint

$$C(\mathbf{c}, \mathbf{p}_1, \dots, \mathbf{p}_n) = \left\| \left(\sum_i m_i \right) \mathbf{c} - \left(\sum_i m_i \mathbf{p}_i \right) \right\|^2 = 0 \quad (4.12)$$

with gradient

$$\begin{aligned} \nabla_{\mathbf{c}} C &= 2 \left(\sum_i m_i \right) \left(\left(\sum_i m_i \right) \mathbf{c} - \left(\sum_i m_i \mathbf{p}_i \right) \right) \\ \nabla_{\mathbf{p}_i} C &= -2 \left(\left(\sum_i m_i \right) \mathbf{c} - \left(\sum_i m_i \mathbf{p}_i \right) \right) m_i \end{aligned} \quad (4.13)$$

The mass centre of the character can be controlled simply by setting the position of the extra vertex $\mathbf{c}$. The character is in static balance when the vertical projection of the centre-of-mass of the whole character is within a polygon of support that represents the convex hull of the ground contact area of the feet (figure 4.2).

We can obtain an equivalent result by constraining the centre-of-mass vertex to be within the intersection of a set of half-spaces each defined by a vertical plane using the kinematic plane

constraint:

$$C(\mathbf{p}) = \mathbf{p} \cdot \mathbf{n} - d \geq 0 \quad (4.14)$$

the gradient of this constraint is simply the plane normal

$$\nabla_{\mathbf{p}} C = \mathbf{n} \quad (4.15)$$

A plane constraint can be set to equality (so that the vertex is constrained to be *on* the plane) or inequality (with the vertex constrained to be *above* the plane).

4.1.5 Kinetics constraints

Now let's consider the motion of the character over several frames of animation. We can derive the velocity and acceleration of a given vertex by finite differences. Since a constraint is simply a function of several vertex positions, we can place constraints on those quantities by devising constraint functions over the successive positions of a particular vertex in time.

Constraining the velocity of a vertex at some frame to a value $\mathbf{v}$ is useful:

$$C = \left\| \frac{1}{\Delta t} (\mathbf{p}_i^{N+1} - \mathbf{p}_i^N) - \mathbf{v} \right\|^2 = 0 \quad (4.16)$$

This ensures, for instance, that the feet make and break contacts in a given direction.

Motion actuated by muscles must have smooth accelerations. To this end, we include a constraint that acts as a smoothing pass over the vertex accelerations.

Laplacian smoothing on the accelerations $\mathbf{a}$ adjacent to frame N gives a smoothed acceleration $\bar{\mathbf{a}}$

$$\bar{\mathbf{a}}^N = \frac{1}{3} (\mathbf{a}^{N-1} + \mathbf{a}^N + \mathbf{a}^{N+1}). \quad (4.17)$$

Projecting the following constraint:

$$C = \left\| \left(-\frac{1}{3} \mathbf{p}^{N-2} + \frac{4}{3} \mathbf{p}^{N-1} - 2\mathbf{p}^N + \frac{4}{3} \mathbf{p}^{N+1} - \frac{1}{3} \mathbf{p}^{N+2} \right) \right\|^2 = 0 \quad (4.18)$$

is equivalent to a Laplace smoothing pass over the acceleration of a vertex $\ddot{\mathbf{p}}_i$.

We use this to obtain a smooth path for the centre of mass and the end-effectors. Each iteration of the constraint projection gives one Laplacian smoothing pass over the accelerations. In practice, too many smoothing passes hamper convergence of the motion synthesis solution discussed below.

A similar constraint on the relative acceleration between a pair of vertices is an economical way to smooth joint motion, for example smoothing the motion of the knee joint by application to the hip and ankle joint centre vertices.

The static balance constraint described above can be extended to compute a dynamic balance constraint. The zero-moment-point (ZMP) is often used as the standard indicator for dynamic balance. Keeping this point within the polygon of support implies that the character is balanced while in motion.

The ZMP $\mathbf{z}$ for a vertex-based character has to satisfy the equation

$$\sum_i m_i \{(\mathbf{c}_i - \mathbf{z}) \times (\ddot{\mathbf{c}}_i \times \mathbf{g})\} = 0 \quad (4.19)$$

where the i th body segment has mass m_i and centre-of-mass position $\mathbf{c}_i$, and $\mathbf{g}$ is the acceleration due to gravity.

By finite differences the acceleration at frame N is

$$\ddot{\mathbf{c}}_i^N = \frac{1}{\Delta t^2} (\mathbf{c}_i^{N+1} - 2\mathbf{c}_i^N + \mathbf{c}_i^{N-1}) \quad (4.20)$$

so the constraint function to keep a vertex $\mathbf{z}$ at the ZMP of the character at frame N is

$$C = \|\Phi\|^2 \quad (4.21)$$

where

$$\Phi = \sum_i \frac{m_i}{\Delta t^2} [(\mathbf{c}_i^N \times \mathbf{c}_i^{N+1}) + (\mathbf{r}_i^N \times \mathbf{c}_i^{N-1}) - (\mathbf{z} \times \mathbf{c}_i^{N+1}) + 2(\mathbf{z} \times \mathbf{c}_i^N) - (\mathbf{z} \times \mathbf{c}_i^{N-1}) + \Delta t^2(\mathbf{z} \times \mathbf{g}) - \Delta t^2(\mathbf{c}_i^N \times \mathbf{g})]$$

The gradient of this is

$$\begin{aligned} \nabla_{\mathbf{c}_i^{N-1}} C &= 2 \frac{m_i}{\Delta t^2} (-\mathbf{c}_i^N + \mathbf{z}) \times \Phi \\ \nabla_{\mathbf{c}_i^N} C &= 2 \sum_i \frac{m_i}{\Delta t^2} (\mathbf{c}_i^{N+1} + \mathbf{c}_i^{N-1} - 2\mathbf{z} + \Delta t^2 \mathbf{g}) \times \Phi \\ \nabla_{\mathbf{c}_i^{N+1}} C &= 2 \frac{m_i}{\Delta t^2} (-\mathbf{c}_i^N + \mathbf{z}) \times \Phi \\ \nabla_{\mathbf{z}} C &= 2 \sum_i \frac{m_i}{\Delta t^2} (-\mathbf{c}_i^{N+1} + 2\mathbf{c}_i^N - \mathbf{c}_i^{N-1} + \Delta t^2 \mathbf{g}) \times \Phi \end{aligned} \quad (4.22)$$

As with static balance, ZMP is represented by an extra vertex constrained to be within a polygon of support by several inequality plane constraints. An additional equality plane constraint keeps the ZMP on the ground plane rather than simply above it as for static balance.

The COM and ZMP constraints are not mutually exclusive and can be used together in the same motion. For instance, later we show an example (figure 4.12) where the COM is constrained at a certain point during a vaulting motion, while the ZMP constraint applies during locomotion foot contacts.

4.1.6 Dynamics constraints

Lastly, we consider a system of vertices as an aggregate body and derive a constraint that enforces the Newton-Euler equations of motion. We must now introduce forces as extra entities into the constraint function.

A dynamics constraint over multiple frames is a differentiable function of a set of vertices *and* an optional set of forces and torques. Forces have a point of action and a force vector. The gradient of the dynamics constraint with respect to the forces is used to update the force vectors at each iteration, just as with vertex positions. The force vector takes the place of the vertex position in Equation 4.1.

We treat forces and vertices independently. Projecting the vertex positions while keeping the forces constant results in convergence to the solution of the forward dynamics problem, whereas projecting the force vectors while keeping the vertices fixed solves the inverse dynamics problem.

The simplest example of a dynamics constraint is one that enforces the dynamical equations of motion for a single particle. Consider the acceleration of a single particle with mass m and position $\mathbf{p}$ at motion frame N under a set of forces $\mathbf{f}_j$ and gravitational acceleration $\mathbf{g}$

$$C(\ddot{\mathbf{p}}^N, \{\mathbf{f}_j\}) = \|\mathbf{W}\|^2 \equiv \left\| \sum_j \mathbf{f}_j + m\mathbf{g} - m\ddot{\mathbf{p}} \right\|^2 = 0. \quad (4.23)$$

This is simply Newton's 2nd law for a particle.

The acceleration at frame N is calculated by finite differences over the adjacent three frames:

$$\ddot{\mathbf{p}}^N = \frac{1}{\Delta t^2} (\mathbf{p}^{N+1} - 2\mathbf{p}^N + \mathbf{p}^{N-1}). \quad (4.24)$$

The gradient of the particle dynamics constraint is

$$\nabla_{\mathbf{p}^{N-1}} C = \frac{-2m}{\Delta t^2} \mathbf{W} \quad (4.25)$$

$$\nabla_{\mathbf{p}^N} C = \frac{4m}{\Delta t^2} \mathbf{W} \quad (4.26)$$

$$\nabla_{\mathbf{p}^{N+1}} C = \frac{-2m}{\Delta t^2} \mathbf{W} \quad (4.27)$$

$$\nabla_{\mathbf{f}_j} C = 2\mathbf{W} \quad (4.28)$$

$$(4.29)$$

In practice, the gradient with respect to force is poorly scaled, so to improve convergence we use

$$\nabla_{\mathbf{f}_j} C = \frac{2}{\Delta t^2} \mathbf{W} \quad (4.30)$$

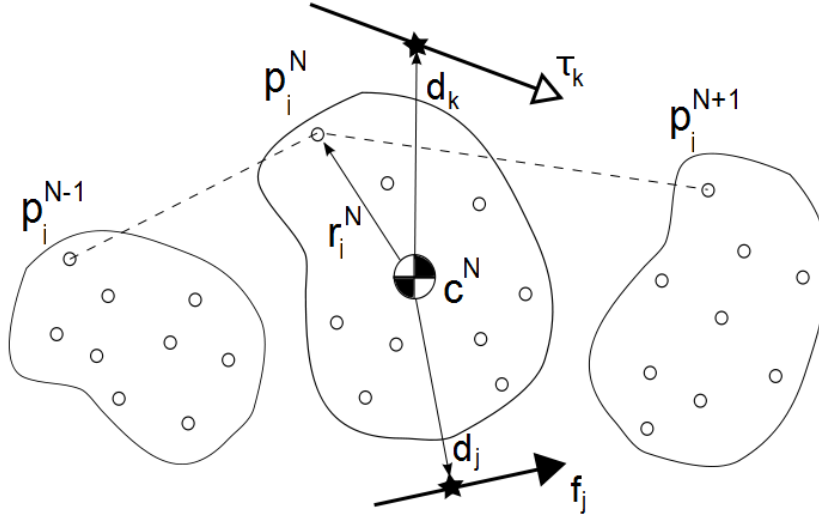


Figure 4.3: Aggregate body dynamics. Particle positions over three frames of motion define the momentum derivative of the aggregated body.

Newton-Euler EOM constraint

For a body described by a set of particles i with masses m_i , distributed in space with positions $\mathbf{p}_i$ we have simple expressions for linear momentum $\mathbf{P}$ and angular momentum $\mathbf{L}$

$$\mathbf{P} = \sum_i m_i \dot{\mathbf{p}}_i \quad (4.31)$$

$$\mathbf{L} = \sum_i m_i (\mathbf{r}_i \times \dot{\mathbf{p}}_i) \quad (4.32)$$

With discrete timesteps we can approximate these dynamical quantities with finite differences, expressing momentum as functions over the particle positions at successive frames of motion (figure 4.3).

The body dynamics constraint

$$C(\{\mathbf{p}_i^{N-1}, \mathbf{p}_i^N, \mathbf{p}_i^{N+1}\}, \{\mathbf{f}_j\}, \{\tau_k\}) = \|\mathbf{F} - \dot{\mathbf{P}}\|^2 + \|\mathbf{T} - \dot{\mathbf{L}}\|^2 = 0 \quad (4.33)$$

can be thought of as a cost function penalising deviation from correct behaviour as described by the Newton-Euler equations of motion.

Here, for external linear forces $\mathbf{f}_j$, external torques τ_k and gravitational acceleration $\mathbf{g}$:

$$\mathbf{F} = \sum_j \mathbf{f}_j + \sum_i m_i \mathbf{g} \quad (4.34)$$

is the total linear force at the centre of mass,

$$\mathbf{T} = \sum_k \tau_k + \sum_j \mathbf{d}_j \times \mathbf{f}_j \quad (4.35)$$

is the total torque at the centre of mass,

$$\dot{\mathbf{P}} = \sum_i m_i \ddot{\mathbf{p}}_i \quad (4.36)$$

is the time-derivative of the linear momentum of the aggregate body and

$$\dot{\mathbf{L}} = \sum_i m_i \{ (\dot{\mathbf{r}}_i \times \dot{\mathbf{p}}_i) + (\mathbf{r}_i \times \ddot{\mathbf{p}}_i) \} \quad (4.37)$$

is the time derivative of the angular momentum.

As before we use finite differences to approximate accelerations and velocities. The velocity of body vertex i is

$$\dot{\mathbf{p}}_i^N = \frac{1}{\Delta t} (\mathbf{p}_i^{N+1} - \mathbf{p}_i^N) \quad (4.38)$$

As for the single particle dynamics constraint, we scale the derivatives with respect to force by a factor of $\frac{1}{\Delta t^2}$ for faster convergence.

The body dynamics constraint applies to any body described as a distribution of massive particles, whether that body is rigid, articulated or deformable. We do not take internal forces or torques into account in these equations, except implicitly as changes in momentum due to deformation, so explicit force $\mathbf{f}_j$ and torque τ_k terms refer to external effects only.

The gradient of the aggregate body dynamics constraint is as follows:

$$\begin{aligned}
\nabla_{\mathbf{p}_i^{N-1}} C &= 2m_i \left\{ \frac{1}{\Delta t^2} (\mathbf{F} - \dot{\mathbf{p}}) \right. \\
&\quad \left. + (\mathbf{T} - \dot{\mathbf{L}}) \times \left(-\frac{1}{\Delta t} \dot{\mathbf{c}} + \frac{m_i}{M\Delta t} \dot{\mathbf{p}}_i - \frac{1}{\Delta t^2} \mathbf{p}_i^N + \frac{1}{\Delta t^2} \mathbf{c}^N \right) \right\} \\
\nabla_{\mathbf{p}_i^N} C &= 2m_i \left\{ \frac{1}{M} (\mathbf{F} - \dot{\mathbf{p}}) \times \left(\sum_k \tau_k \right) + \frac{2}{\Delta t^2} (\mathbf{F} - \dot{\mathbf{p}}) \right. \\
&\quad \left. + (\mathbf{T} - \dot{\mathbf{L}}) \times \left[\frac{1}{\Delta t} \dot{\mathbf{c}} - \frac{m_i}{M\Delta t} \dot{\mathbf{p}}_i + \frac{2}{\Delta t^2} \mathbf{p}_i^N \right. \right. \\
&\quad \left. \left. - \frac{2}{\Delta t^2} \mathbf{c}^N + \left(1 - \frac{m_i}{M}\right) \ddot{\mathbf{p}}_i + \frac{1}{M} \sum_j \mathbf{f}_j \right] \right\} \\
\nabla_{\mathbf{p}_i^{N+1}} C &= 2m_i \left\{ \frac{1}{\Delta t^2} (\mathbf{F} - \dot{\mathbf{p}}) \right. \\
&\quad \left. + (\mathbf{T} - \dot{\mathbf{L}}) \times \left(-\frac{1}{\Delta t^2} \mathbf{p}_i^N + \frac{1}{\Delta t^2} \mathbf{c}^N \right) \right\} \\
\nabla_{\mathbf{f}_j} C &= \frac{2}{\Delta t^2} \{ (\mathbf{F} - \dot{\mathbf{p}}) + (\mathbf{T} - \dot{\mathbf{L}}) \times \mathbf{d}_j \} \\
\nabla_{\tau_k} C &= \frac{2}{\Delta t^2} \{ (\mathbf{F} - \dot{\mathbf{p}}) \times \mathbf{d}_k + (\mathbf{T} - \dot{\mathbf{L}}) \}
\end{aligned} \tag{4.39}$$

where $M = \sum_i m_i$ is the total mass of the body, and $\mathbf{c}^N = \frac{1}{M} \sum_i m_i \mathbf{p}_i^N$ is the centre of mass of the body at frame N .

We have omitted some cross terms from the position derivatives. These come from taking the derivative of the mass-centre position and velocity. In practice the cross terms are small compared to the other terms; they contribute only a few percent of the value of the constraint function and their effect on the motion is not noticeable to the eye. However the summation over every vertex in the body makes these terms the most expensive part to compute. So neglecting these terms makes the dynamics calculation considerably faster.

4.1.7 Geometric constraints on forces

The data structure for a force contains a vector force value, a position value for the point of action, a susceptibility value for the constraint projection and an enumerated value indicating whether linear force or torque is signified.

We place geometric constraints on the forces. These are enforced during each solution step, separately to the dynamics constraint projection, by simply updating the force structure to ensure that the forces are legal.

We can constrain a force (or torque) to act at a certain point. Typically we choose either a vertex position or an external position as the point of action of each force.

We project the force to be within a cone described by a unit-length normal $\hat{\mathbf{n}}$ and an internal angle θ . This is useful for keeping contact forces within a friction cone in a Coulomb friction

model.

If

$$\frac{\mathbf{f}}{|\mathbf{f}|} \cdot \hat{\mathbf{n}} \geq \cos(\theta), \quad (4.40)$$

where $\mathbf{f}$ is the force vector, then the force is inside the cone and we do nothing.

Otherwise the force is outside the cone and we project the force to the cone surface:

$$\begin{aligned} \mathbf{x} &= \hat{\mathbf{n}} \cos(\theta) + \frac{(\hat{\mathbf{n}} \times (\mathbf{f} \times \hat{\mathbf{n}}))}{|\hat{\mathbf{n}} \times (\mathbf{f} \times \hat{\mathbf{n}})|} \sin(\theta) \\ \mathbf{f} &:= \frac{\mathbf{x}}{|\mathbf{x}|} |\mathbf{f}| \end{aligned} \quad (4.41)$$

4.2 Spacetime motion synthesis ab initio

As a proof of concept for the dynamics constraints, we first synthesised several example motions of simple structures from simple initial conditions. For each example (as well as for the motion adaptation section later) we model an object moving under forces to satisfy some external constraints. By using a model whose vertices and internal constraints do not change from frame to frame, we can represent rigid or articulated bodies that obey conservation of mass.

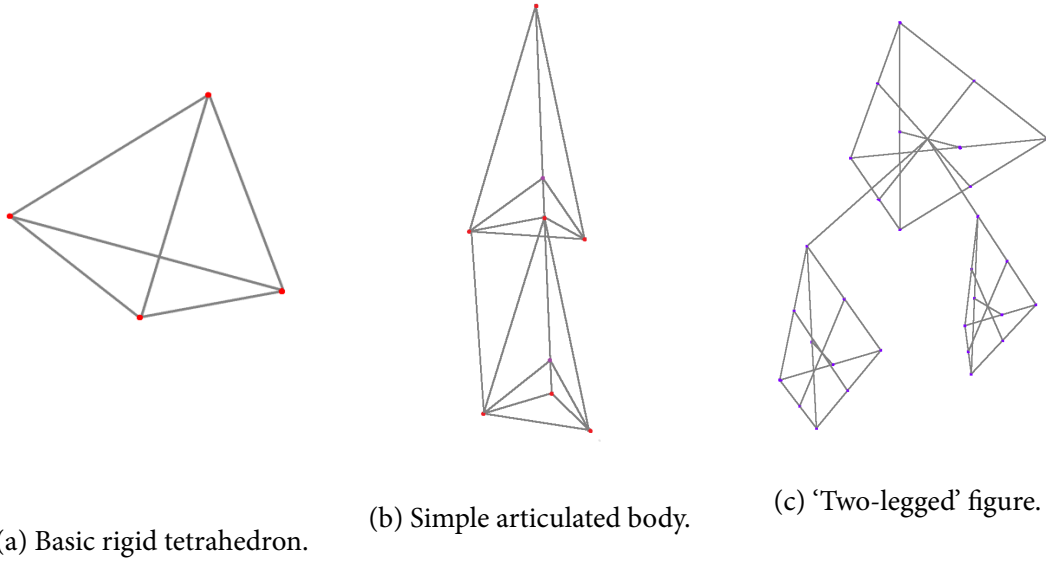


Figure 4.4: Simple body structures.

These structures were composed of vertices and distance constraints. Joints between ‘articulated’ sections were constructed by sharing vertices (see Figure 4.4) or, in the case of the two-legged figure, by using barycentric constraints to attach a vertex to an adjacent tetrahedron.

To initialize the spacetime system we then copied the model for each frame and created kinetics and dynamics constraints over multiple frames. External constraints such as a target position in world space at a given time were applied too. We also fixed the poses of the first and last frames by setting the inverse masses of the vertices to zero.

4.2.1 Particle and single rigid body

First we tested the dynamics constraints for a single particle and a rigid body consisting of four particles bound in a tetrahedron.

We set the initial motion to simply interpolate the beginning and end positions. Forces at the first few frames supply the impetus to reach a target defined by a positional constraint on the final frame.

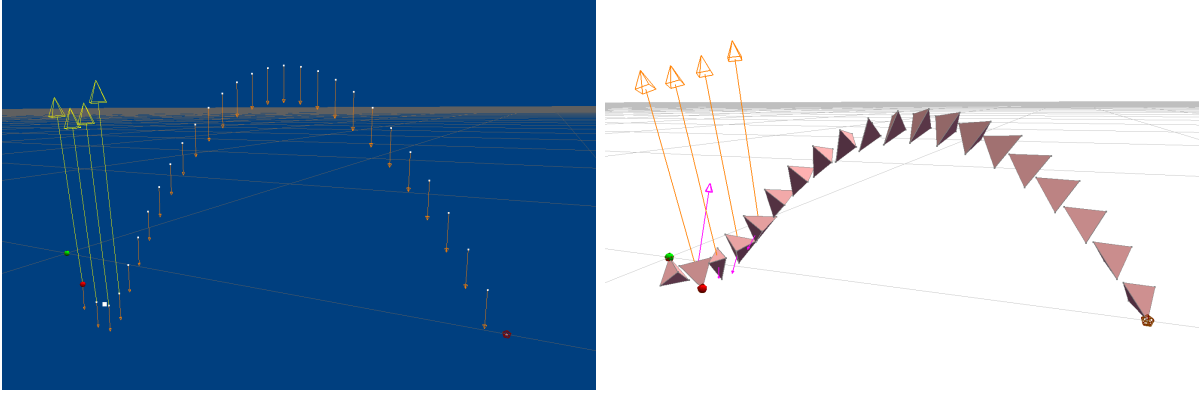


Figure 4.5: Particle and body motion synthesis by spacetime vertex dynamics constraints.

Both particle and simple body accelerate on the forced frames, and then move in a parabola to the target (figure 4.5). The body takes on a constant angular velocity over the unconstrained frames.

4.2.2 Two-bone hopper

Next we moved on to a simple articulated body (figure 4.6). The structure of the two rigid portions was defined by distance constraints among the particles. Two vertices shared between the rigid bodies defined a hinge articulation at the joint.

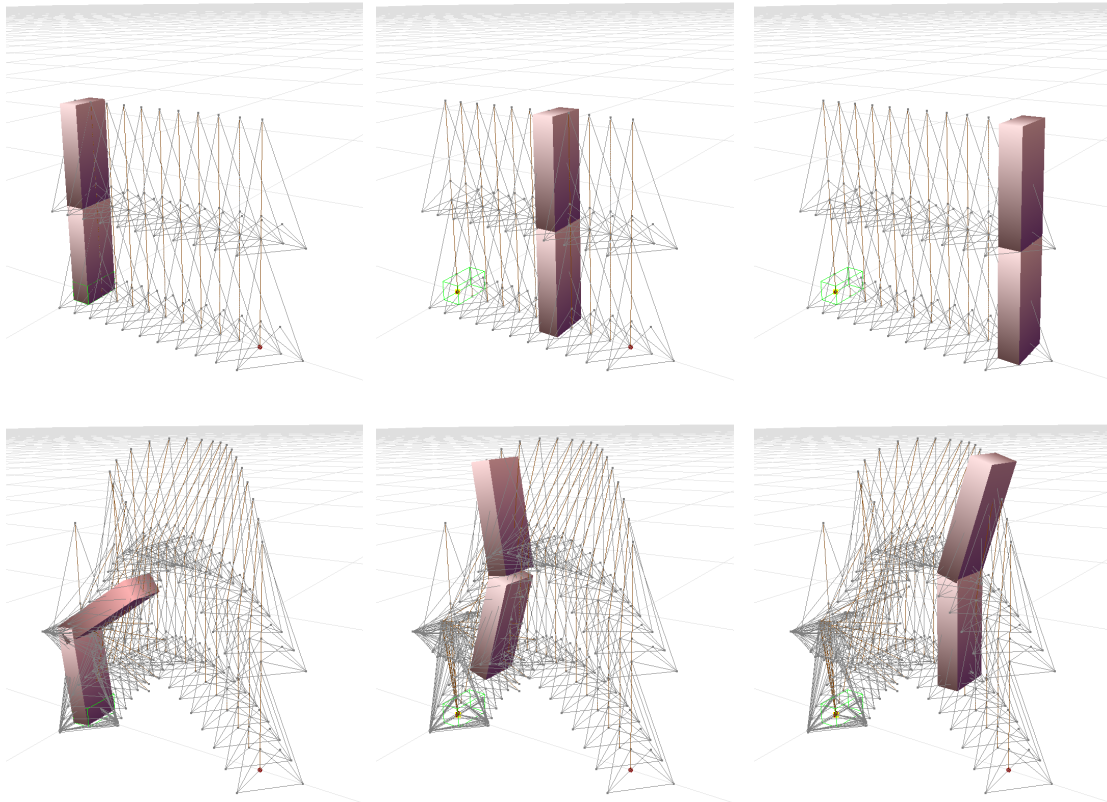


Figure 4.6: Motion synthesis of simple articulated ‘hopper’. Top: initial motion. Bottom: synthesised motion.

4.2.3 Simple two-legged character

With a slightly more complex articulated body (figure 4.7) we produced a primitive walking motion. This setup had a ‘ball-and-socket’ type joint, defined by a single shared vertex, at each ‘leg’. The ‘footsteps’ emerged from constraining the ‘foot’ vertices to the floor at certain frames (figure 4.8).

The character also displayed balance and joint reactions (the ‘head’ swaying in opposition to the ‘leg’ motion), emerging from the dynamics constraints.

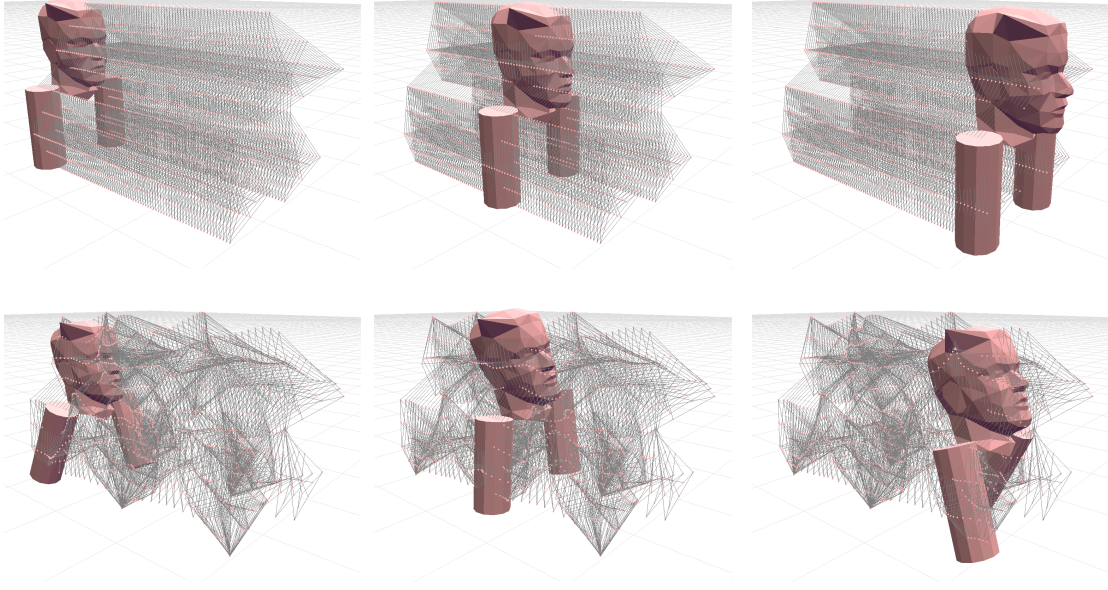


Figure 4.7: Motion synthesis of simple two-legged character. Top: initial motion. Bottom: synthesised motion.

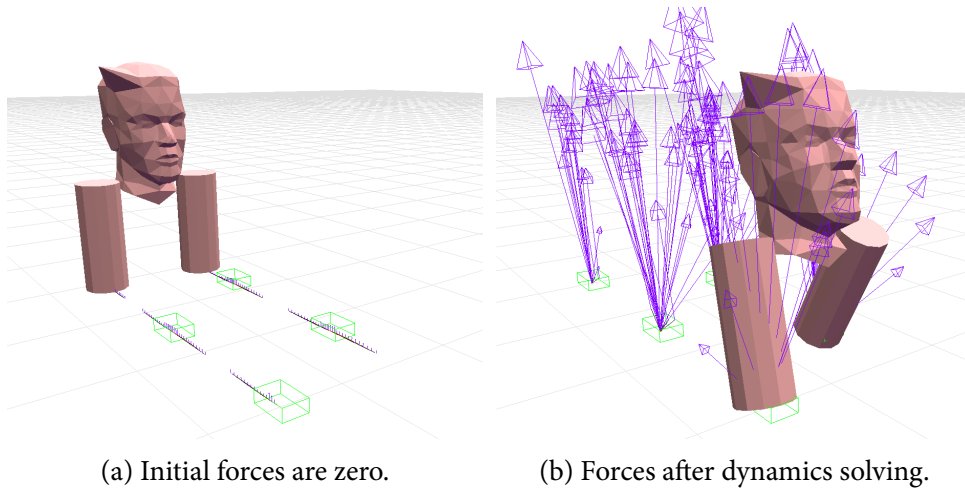


Figure 4.8: Contact forces for two-legged character motion.

4.3 Human character mocap adaptation

We begin with a sequence of poses, sampled uniformly at 15 frames per second from the mocap data and applied to the vertex skeleton at each frame. End-effector contacts were manually marked-up for each animation and contact points generated from these. The corresponding contact forces were initialized to zero.

We perform pre-solving in two stages. In the first stage forces are allowed to converge without moving the vertices (this is the inverse dynamics calculation as mentioned earlier).

After this we switch to projecting both forces and vertex positions. The purpose at both stages is to reduce the dynamics error in the vertex motion.

Given a fully pre-solved vertex motion, the adaptation phase consists of running the solver to adjust the motion to altered constraints, such as external contacts and mass-centre position.

Examples are shown in Figs 4.9 to 4.13.

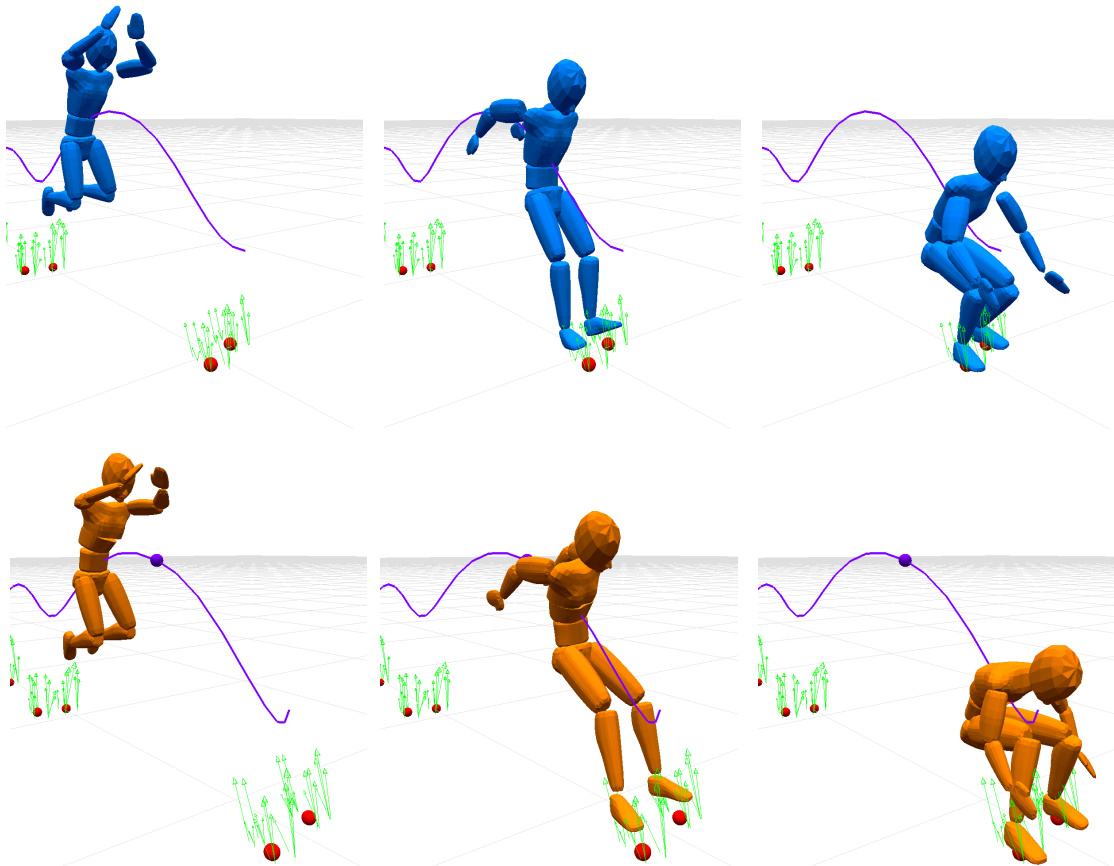


Figure 4.9: Editing motion: Leap. Original motion (blue). Edited motion (orange): mass-centre during leap (in purple) and landing foot plants (marked by red orbs) have been moved forward.

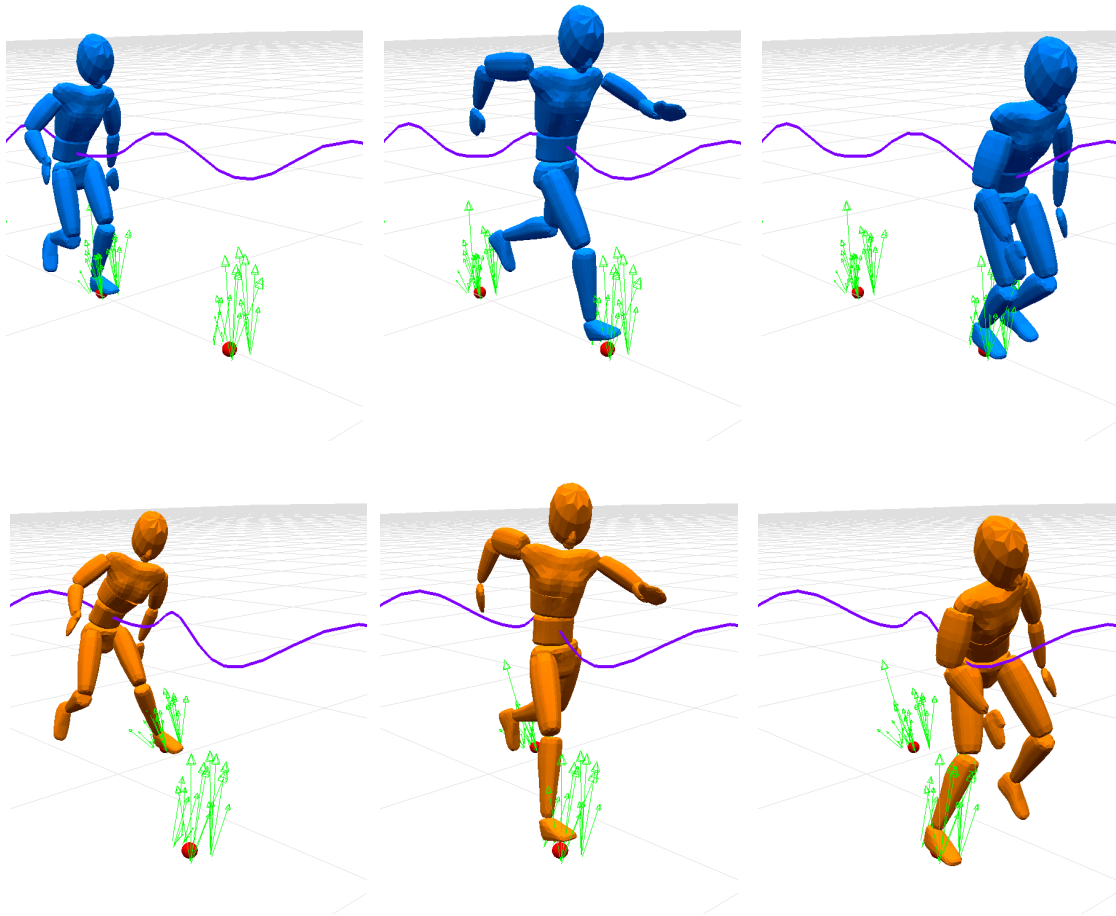


Figure 4.10: Editing motion: Long steps. Original motion (blue). Edited motion (orange): footsteps have been moved outward.

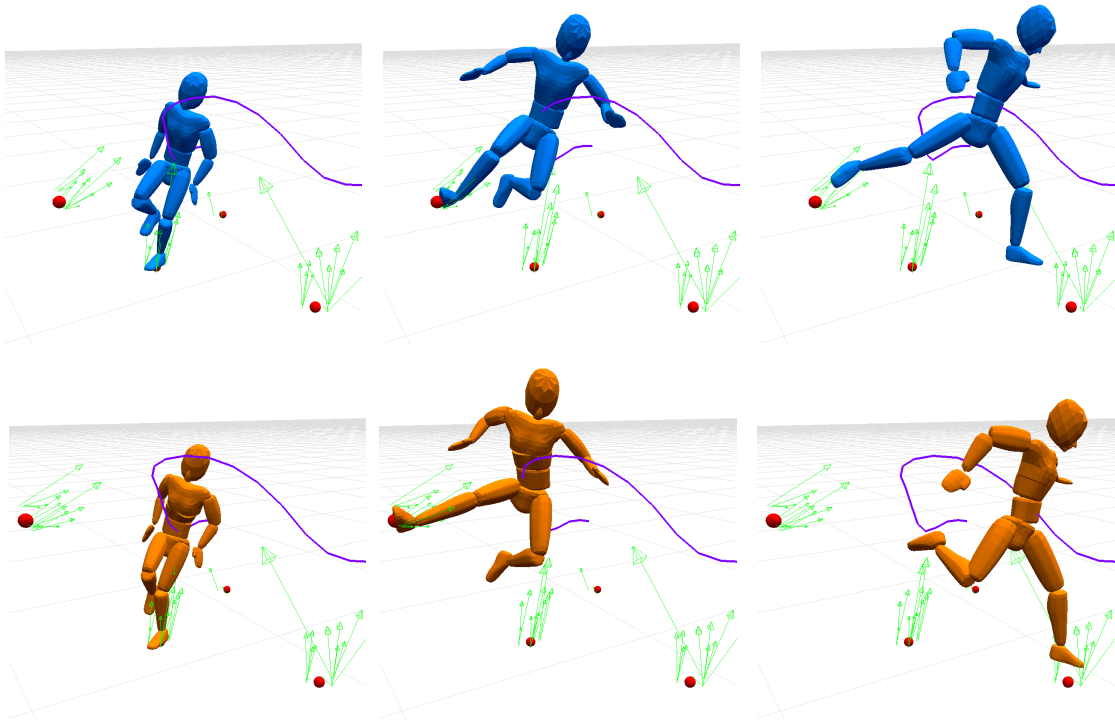


Figure 4.11: Editing motion: Tic-tac. Original motion (blue). Edited motion (orange): foot plant on wall (marked by red orb) has been moved upward.

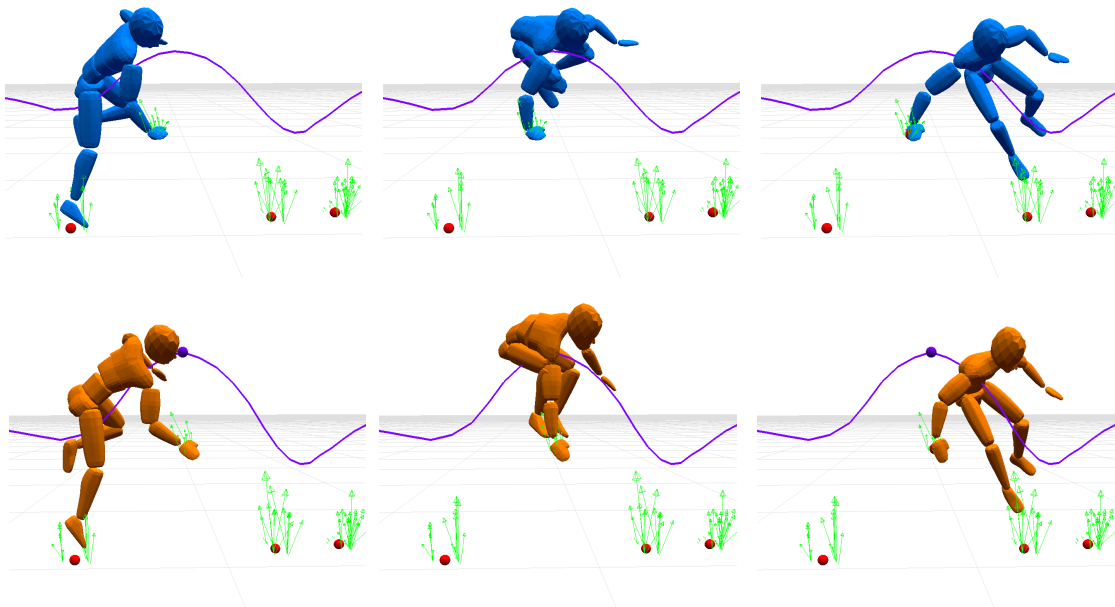


Figure 4.12: Editing motion: Vault. Original motion (blue). Edited motion (orange): mass-centre during vault (in purple) has been moved upward.

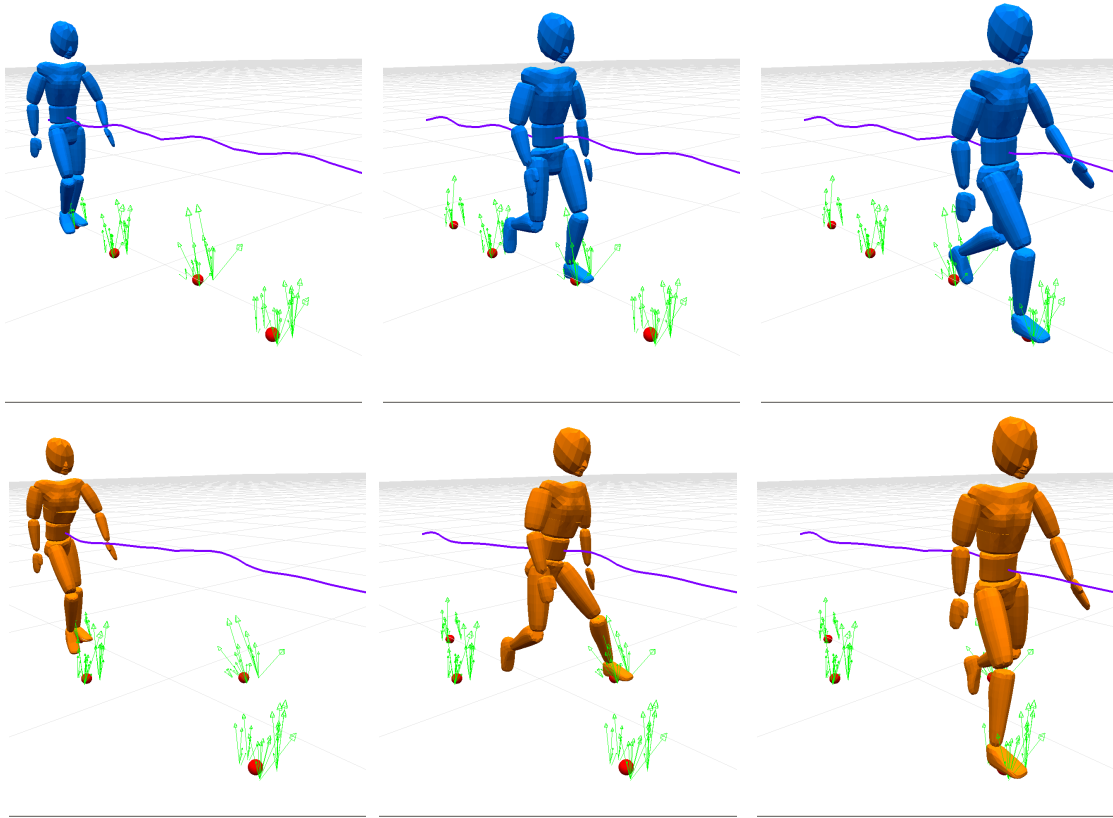


Figure 4.13: Editing motion: Walk. Original motion (blue). Edited motion (orange): footsteps have been moved outward.

When external constraints are sparse, trying to adapt to large changes in those constraints can lead to divergence and highly unrealistic motion. The tendency of the dynamics constraints is to treat the body as a particle cloud, allowing it to disperse under the velocity of the particles. This is opposed by the structural constraints making up the skeleton, and the external contact constraints.

This can happen for instance when moving the landing positions of the feet in the leap motion (figure 4.14).

To work around this, we can make small progressive adaptations or add extra constraints. For this example, an extra constraint on the mass-centre position as in figure 4.12 improves the stability and allows a somewhat larger adaptation of the landing position.

In any case, the spacetime vertex method is best suited to making relatively small adaptations to animations.

4.3.1 Convergence

To display the convergence behaviour of our solver we have used a measure of the error in dynamics, obtained by simply summing the values of all the dynamics constraints on the motion.

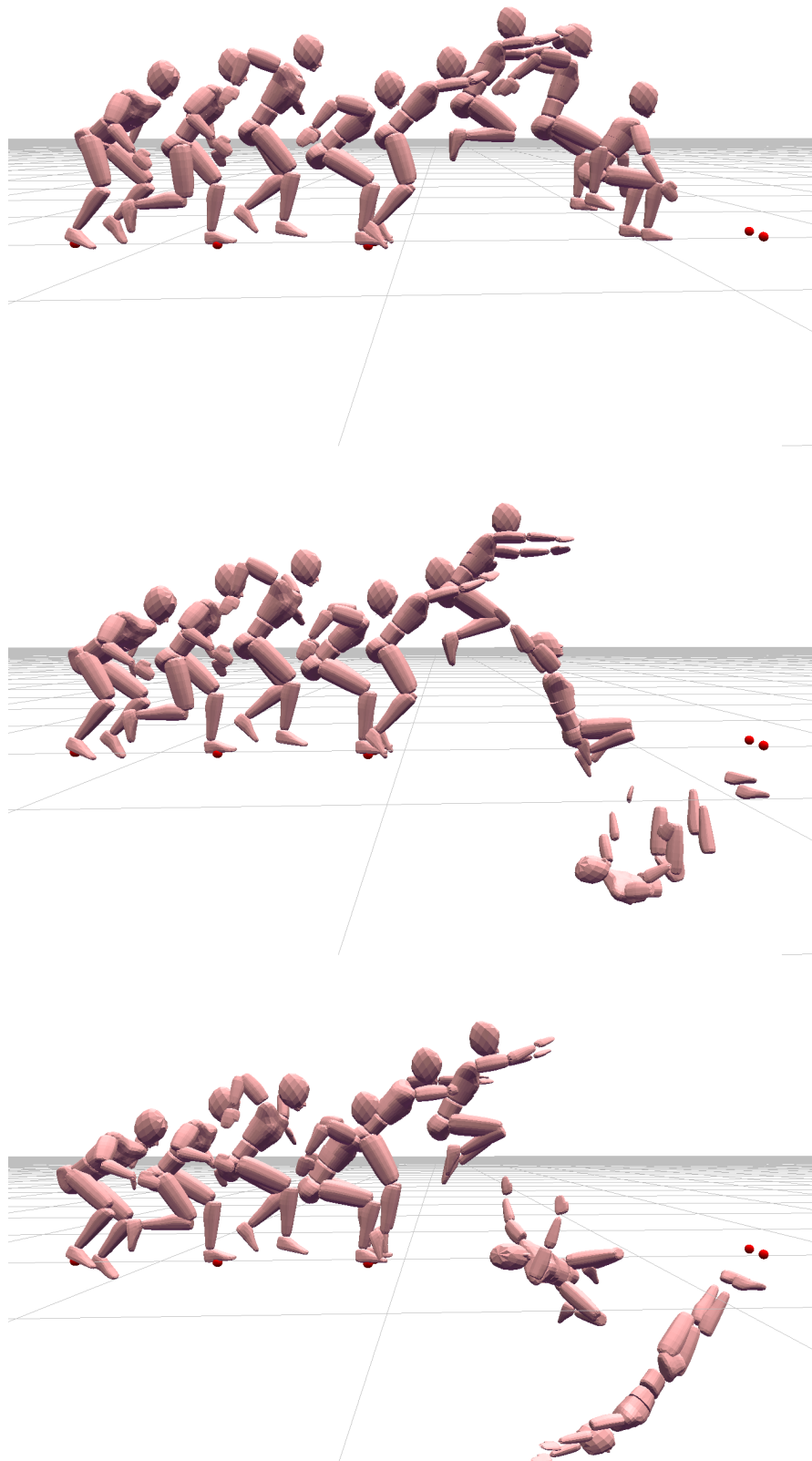


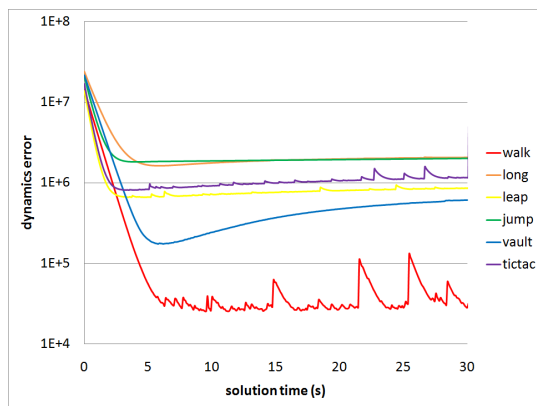
Figure 4.14: A failure case of motion adaptation. Moving the foot plant constraints too far from their original positions (top) results in divergence and unphysical motion.

Since each dynamics constraint is a cost function describing the local error in the dynamics over the motion frames forming the domain of the function, this summed value gives an aggregate estimate of the total dynamics error over the whole motion. This quantity does not have any formal units since the dynamics constraints mix the squared values of force and torque residuals.

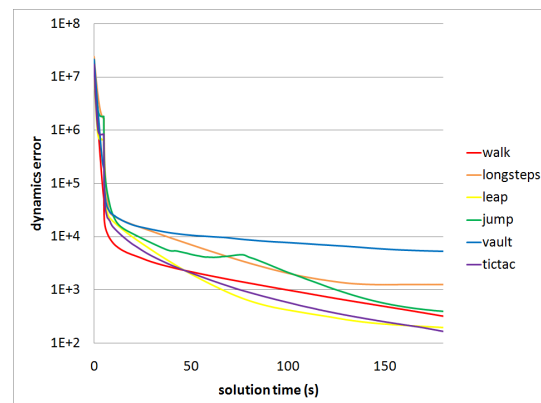
The relationship between our dynamics error metric and the perceived plausibility of an adapted motion is only approximate. Plausibility also depends on less measurable (and more difficult to capture) factors we refer to collectively as ‘natural motion style’. A concrete example is that jerky joint accelerations are very obviously unnatural. Indeed, we have found that a motion with a high value of dynamics error can still look plausible as long as the joint accelerations are smooth and the poses of the character look natural. Our technique works best on well-constrained but highly dynamic motions such as the ‘tic-tac’, wherein the performer kicks off a wall to change direction during a run.

Pre-solve convergence

Figure 4.15a shows that the inverse dynamics calculation tends not to keep converging after 5 seconds. The spikes in the graph are discontinuities in the forces caused by enforcing the geometric force constraints. We then allow adjustment of the vertex positions to that the dynamics converge further. The small ‘dogleg’ (at upper left) in the first 5 seconds of Figure 4.15b corresponds to the curves in Figure 4.15a. Omitting the second stage of the pre-solve results in poor results in the adaptation phase; the adapted solution then more resembles pure IK editing.



(a) Dynamics error for pre-solve 1st stage. Projecting the forces while vertices remain fixed.



(b) Dynamics error for 2-stage pre-solve.

Figure 4.15: Converging dynamics cost function during pre-solve stage.

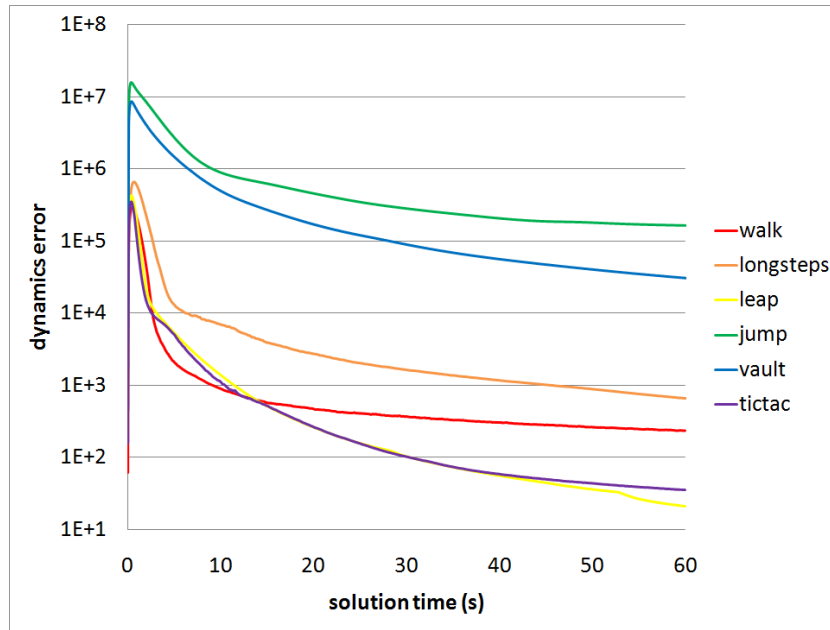


Figure 4.16: Converging dynamics cost function during motion adaptation to altered external contact constraints.

Adaptation Convergence

Figure 4.16 shows the dynamics cost function converging for some example adaptations. The key message of this graph is that the majority of the dynamics error is typically eliminated within the first 10 seconds.

4.3.2 Performance

We tested the performance of our method on a set of example animations. The timing results in this section were obtained using a single core of a 2.4GHz *Intel Core2 Quad* CPU with 2GB memory.

Table 4.1 lists the example animations along with estimates of the average time to project each kind of constraint.

For the examples we chose 25 iterations of each dynamics and kinematics constraint relaxation, and 1 iteration of smoothing constraint relaxation at each step of our solver algorithm. Fewer iterations leads to longer convergence time, but there are diminishing returns on larger numbers of iterations per step. Many iterations of dynamics constraints can lead to instability since these constraints do not respect the internal kinematic structure of the character, instead treating it simply as a particle cloud. Many iterations of kinematics constraints become redundant and will anyway be invalidated at the next step when the dynamics constraint projection is repeated. Too many smoothing iterations can interfere with the convergence of the dynamics.

Motion that is too smooth looks as unnatural as jerky motion, so a balance must be struck. A minimal amount of smoothing is best.

Animation	Num Frames	Kinematics	Dynamics
Walk	35	1.21	1.47
LongSteps	45	1.21	1.50
Jump	28	1.20	1.46
Vault	38	1.22	1.49
TicTac	22	1.20	1.41

Table 4.1: Example animations. Also shown: number of frames at 15Hz; average time (in milliseconds) per frame to project kinematics and dynamics constraints (25 iterations each) at each solver step.

Comparison of performance with other methods

Table 4.2 provides an approximate comparison between the performance of our method and that of two different motion editing methods.

The examples we chose for comparison are taken from the spacetime optimisation, efficient articulated body motion synthesis method of Fang and Pollard [FP03] and the least-squares spacetime mesh motion editing method of Xu et al. [XZY⁺07].

Editing Method	Processor Speed	Motion Length	Character Complexity	Precompute Time	Solve Time
Ours	2.4 GHz	2 s, 30 frames	15 joints	2 min	10 s
Fang & Pollard	750 MHz	2.3 s (spline)	9 joints	n/a	26 s
Xu et al.	3.2 GHz	154 frames	8 handles	3 min	28 s

Table 4.2: Approximate comparison of performance; spacetime vertices method vs. two different motion editing methods from literature.

Using a rough normalized measure from the formula:

$$\frac{\text{solve time} * \text{processor speed}}{\text{motion length} * \text{character complexity}} \quad (4.42)$$

we find that the three methods are comparable in computational cost (the number arrived at for each is about 0.06).

The method of Xu et al. produces visually pleasing mesh deformation sequences, but does not consider dynamics. The method of Fang and Pollard produces dynamically realistic and

compelling motion for an articulated rigid-body character, but we maintain that our approach offers the advantage in simplicity of implementation.

4.3.3 Comparison with ground-truth mocap

To evaluate the effect of using dynamics constraints we present a comparison between some adapted motions and corresponding ground-truth mocap.

We obtained some bespoke motion capture data:

- Base *walk* motion, together with corresponding elaborated
 - * *walk with wide steps*
 - * *walk with one wide step*
- Base *stepping-up over a box* motion, together with corresponding elaborated
 - * *stepping up over a higher box*

An attempt was made to capture a more diverse array of motions for comparison, but several limitations prevented this: the available space and props for mocap restricted the range of possible motions; some of the captured motions were improperly performed and could not be matched contact-for-contact at the markup/timewarping phase; and some of the captured motions were too noisy due to lost mocap markers.

To measure the difference between two motions we used the simple metric of summing the mass times squared displacement between corresponding particles over the vertex skeleton at each frame. This difference metric is similar to the mass displacement metric used in several spacetime optimisation papers [PW99, LP02, SP05], though for a particle-based body rather than a collection of rigid bodies. It also resembles a simpler version of the motion similarity metric used by Matsunaga and Zordan [MZ07].

We take a base motion, adapt it so that its foot plants match the elaborated motion and then compare the adapted motion with the original elaborated motion using the metric. To prepare the motions we manually marked up the foot plants and time-warped the base motion to be synchronised with the elaborated motion before sampling the frames for vertex motion. This was necessary since our vertex motion does not support time-warping itself, and for the purposes of the motion difference metric we need one-to-one correspondence between frames of motion.

As before, the dynamics solution went in two stages: inverse-dynamics pre-solve, where we keep vertex positions constant and only solve for the forces; and adaptation solve, where we solve both forward and inverse dynamics simultaneously, allowing the vertices to move about. The solver iteration settings (per round of solving) for the experiment were as in table 4.3:

kinematics constraints	20
dynamics constraints	5
ZMP constraints	5
acceleration smoothing	1

Table 4.3: Solver settings for dynamics-based adaptation.

We also ran the solver for the same amount of time without solving dynamics constraints, using the iteration settings in table 4.4:

kinematics constraints	20
dynamics constraints	0
ZMP constraints	5
acceleration smoothing	10

Table 4.4: Solver settings for adaptation without dynamics.

Walk vs. walk wide one step

Table 4.5 and figure 4.17 show an adaptation to a walking motion where the target deformation is a single large sideways footstep. The adapted motion shows dynamic balance being taken into account.

Solver Stage	Calculation Time (s)	Dynamics Error	Difference Metric
Original	0	1.3268×10^7	81.54
Pre-solved	12.58	655169	81.54
Adapted	25.40	3389.69	61.05
No dynamics	25.31	4.28×10^7	74.03

Table 4.5: Comparison of adapted *Walk* vs. ground-truth *Walk wide one step*.

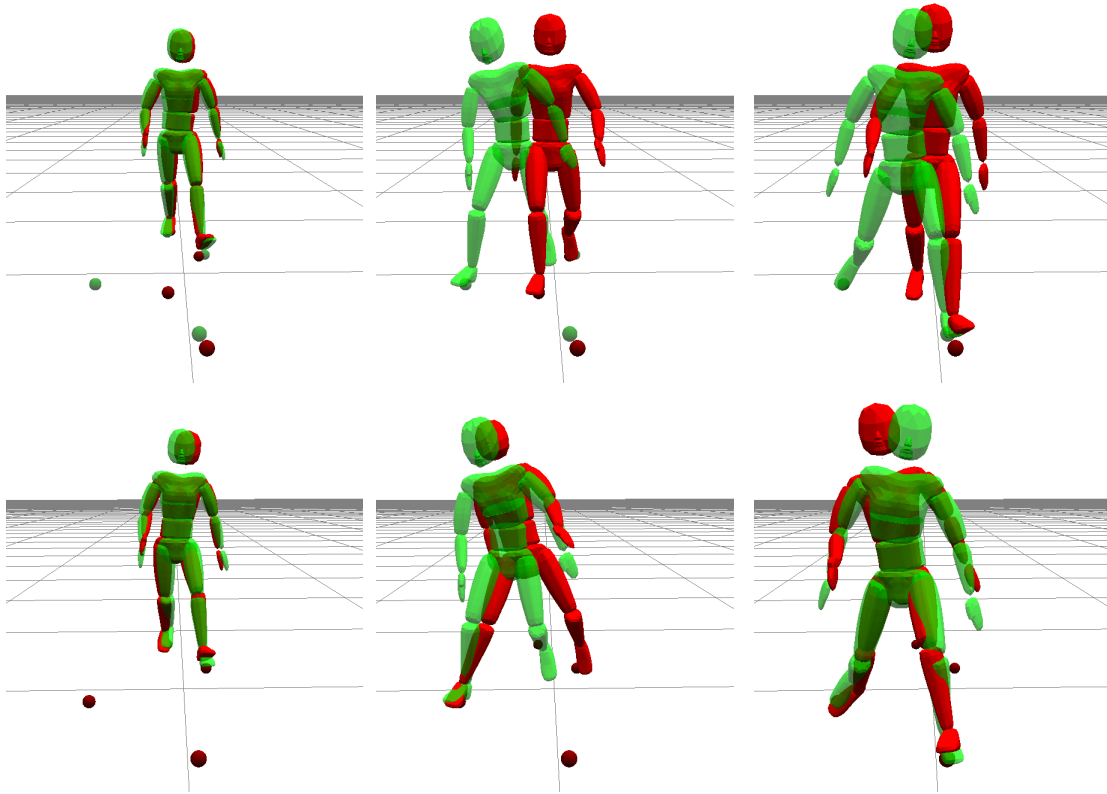


Figure 4.17: Dynamics-based motion adaptation. Top row: original motions. Bottom row: adapted *Walk* (red) vs. ground-truth mocap *Walk wide one step* (green).

Walk vs. walk wide steps

Next we adapt a straight walking motion to one where all of the footsteps are displaced sideways. Table 4.6 shows that the dynamics-based adaptation has a greater error than the adaptation using only IK and smoothing. In motion the dynamics-based adapted animation looks more lifelike because of a visual sense of momentum.

However, as can be seen from figure 4.18 there is a large displacement of the hip to preserve balance. The cause of this is twofold: first, the inertia of the character is based on the mass distribution of a set of massive particles and so does not accurately model the inertia of the motion capture actor; second, there is no modelling of biomechanics in our solution.

Without a cost function based on biomechanics, the results of the solver can include poses that, though correct in terms of forces and momentum, are implausible for a human to perform.

Solver Stage	Calculation Time (s)	Dynamics Error	Difference Metric
Original	0	1.4151×10^7	57.05
Pre-solved	9.40	749830	57.05
Adapted	31.385	1891.80	73.21
No dynamics	31.12	2.77×10^7	51.04

Table 4.6: Comparison of adapted *Walk* vs. ground-truth *Walk wide steps*.

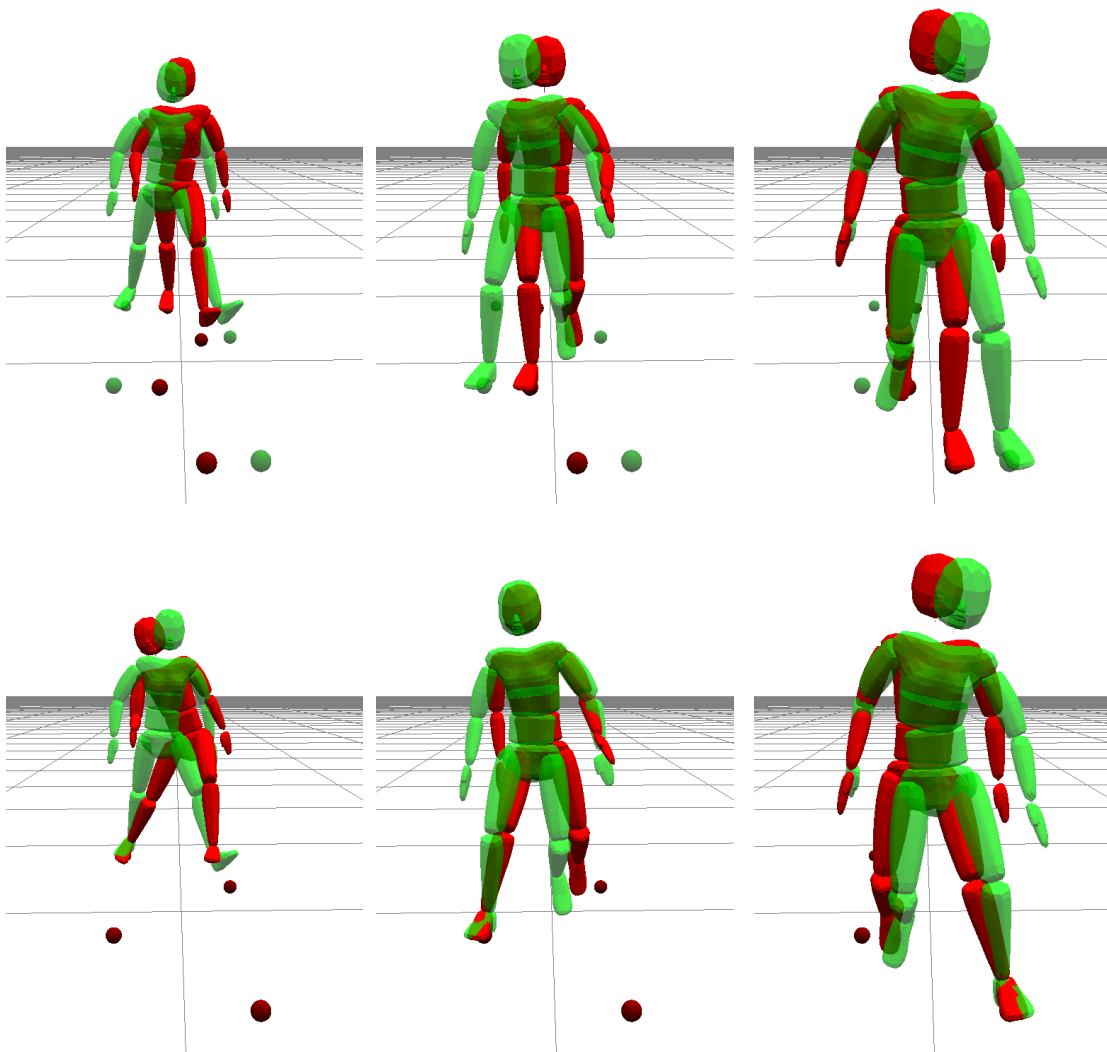


Figure 4.18: Dynamics-based motion adaptation. Top row: original motions. Bottom row: adapted *Walk* (red) vs. ground-truth mocap *Walk wide steps* (green). Note somewhat unnatural displacement of hip at bottom-left.

Step up vs. step up high

The third example takes a motion of stepping up onto and over an obstacle, and compares it with another motion using a higher obstacle.

Again, plausibility of momentum and dynamic balance are preserved in the adapted motion; but as in the previous example the character performs an ‘uncomfortable’ pose at the peak of the motion’s displacement. In this case, the character does not straighten up in the middle phase of the motion when stepping onto the obstacle (figure 4.19). Since there is no measure of muscle usage, the solver produces a squatting motion that would require a lot of quadriceps strength.

However, the inclusion of dynamics in the motion adaptation nevertheless reduces the difference between the adapted and ground-truth motions when compared with using IK and acceleration smoothing only (table 4.7).

Solver Stage	Calculation Time (s)	Dynamics Error	Difference Metric
Original	0	3.2173×10^7	310.40
Pre-solved	15.29	620714	310.40
Adapted	127.627	776.19	190.46
No dynamics	15.65	3.72×10^7	266.13
No dynamics	127.35	6.63×10^7	278.91

Table 4.7: Comparison of adapted *Step up* vs. ground-truth *Step up high*.

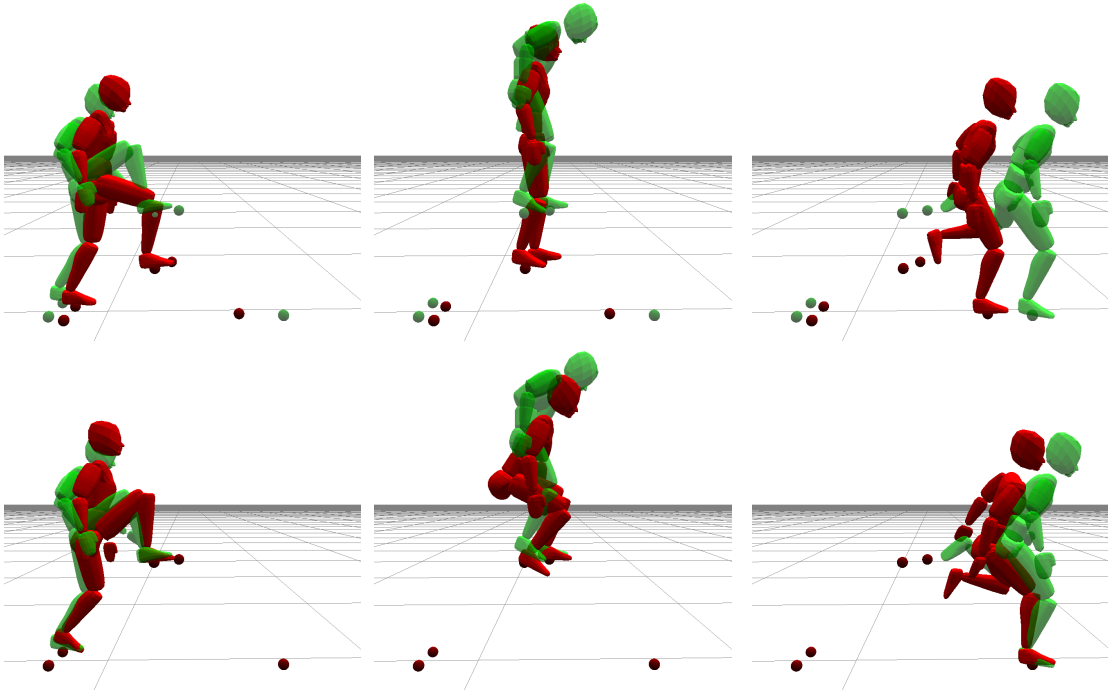


Figure 4.19: Dynamics-based motion adaptation. Top row: original motions. Bottom row: adapted *Step up* (red) vs. ground-truth mocap *Step up high* (green). Note the character does not straighten upright in the adapted motion but passes through a squatting posture.

4.4 Conclusions

In this chapter we have presented a physically-based motion editing technique whose novelty lies in applying an iterative solver to constraints over vertex coordinates in spacetime.

The vertex-based approach trades in the relatively low dimensionality of reduced-coordinate spacetime optimisation approaches for gains in flexibility and simplicity of formulation.

There are several disadvantages to this approach, the first being that the solver is not guaranteed to converge.

Attempts to adapt to large changes in the external constraints tend to cause the solver to diverge, especially when the external constraints are sparse. We can either limit the use of the method to small adaptations, or overcome the problem to some extent by breaking a large change into smaller increments; allowing the solver to come to a stable solution of each increment before applying the next. We could also add more external constraints, such as COM trajectory constraints, to improve stability.

When the system is overconstrained the solver may also not converge. This case tends to be less of a problem since the system usually settles into a compromise state, though we may be left with a motion that is obviously not physically realistic.

Another potential disadvantage is the path-dependence of the solver. Each iteration pro-

ceeds from the results of the previous iteration, so motion adaptations are not reversible. It should be noted that this is not entirely disadvantageous; adapted motion tends to retain a good deal of the style of the original mocap.

The advantages of the approach are its flexibility, versatility and simplicity of implementation. We require no nonlinear optimisation solvers. We can come up with a wide variety of constraints over the vertices since the only limitation is that a constraint be differentiable. We have freedom to add or remove constraints on the motion at any time, and the solver step can be configured in a variety of ways by recomposing its sub-steps and selecting different values for the parameters.

In addition we have all the advantages of vertex-based IK over traditional IK. In particular, we do not have to worry about singularities or joint chain topology since all cases are handled gracefully by the iterative solver. The keystone on which a ‘plausible’ automatic motion editor works is a metric of plausibility. A better metric for plausibility would help to formulate motion constraints for better adaptations.

Such a metric might incorporate an explicit model of muscle forces and storage of potential energy (as in tendons). We attempted to implement full torque control at every joint but the system was over-constrained and would not converge. Some representation of spring-like forces in the limbs and spine would probably improve the motion, allowing extra anticipatory and compensatory movement, or ‘squash-and-stretch’.

There is a non-trivial relationship between the timing and duration of external contacts (such as footsteps) and the characteristics of the motion. An interesting future development would be to investigate how timewarping, as in [MPS06], could extend the utility of our approach.

Finally, a more straightforward enhancement would be to support general dynamically-based contacts. Our current system has static contact points in the hands and feet. If we used a collision-detection system to generate contact points then our character could perform motions such as rolling and tumbling.

The technique presented in this chapter is suitable for making small deformations to an input motion, while preserving physical plausibility. As we have seen, it tends to perform less well with larger deformations; and once the desired deformation gets too large, the solver is likely to become unstable.

However, the motion planning technique in Chapter 3 includes a means to directly limit the amount of deformation required: we can simply decide to disallow a motion if the docking query returns a scene contact over a certain distance threshold away.

Furthermore, in Chapter 5 we describe a full-body IK technique that could potentially be used to make larger deformations to the input motion as a pre-processing step before applying the physically-based adaptation.

Chapter 5

Data-driven inverse kinematics using principal geodesics

The physically-based model described in Chapter 4 is suitable for adapting to small deviations from an original mocap animation. Here we describe a technique capable of dealing with more substantial deformations. This is a full-body data-driven IK algorithm based on pose reconstruction from Principal Geodesics Analysis (PGA) [TWC⁺09]. The authors of that paper extracted the *principal geodesics* from a single animation in order to reconstruct the whole motion later from only those geodesics and the end-effector trajectories.

Similar to Principal Components Analysis (PCA) often used to model variations in Euclidean spaces, PGA captures the variations over the non-Euclidean (spherical) space of orientations in which skeleton poses lie. Specifying the end-effector positions, we can infer over the K main Principal Geodesic components to recreate a plausible pose according to the input mocap data. This chapter proposes a refinement to the technique in [TWC⁺09], which permits reconstruction of more general poses from a set of motions rather than from a single animation.

We describe the details of the IK technique next. Sections 5.1 to 5.4 recap the work of Tournier et al. [TWC⁺09] but include some non-obvious implementation details such as an explicit description of how to calculate the Jacobians. Section 5.5 begins the discussion of our novel work with PGA IK.

5.1 Principal geodesics

Principal geodesics analysis (PGA) is similar to principal components analysis but works on rotation data; the poses of a character skeleton occupy the space $SO(3)^n$, which has a different geometry to $\mathbb{R}^{3n}$. To perform PGA on a collection of pose data, we first find the intrinsic mean [Moao2] and then perform Principal Components Analysis (PCA) on the tangent space

to $SO(3)^n$ at that mean. The tangent space is given by the exponential map [Gra98] taking rotations $q \in SO(3)$ to vectors $v \in \mathbb{R}^3$.

This gives us a set of principal geodesics, or *eigen-poses* (so called in [TWC⁺09]). The principal geodesics capture the statistical variation of poses in the input data set. A weighted combination of these can be used in an IK algorithm to reconstruct plausible poses based on the input pose data. Each principal geodesic is a direction in the tangent space to the pose at the mean pose. Concatenating the rotations given by a series of weighted geodesics with the mean pose gives a new pose. Since the geodesics are extracted from the original motion data, this means the poses synthesised are only those that are ‘reachable’ by the source data. So the output of the IK algorithm is a plausible pose.

A popular technique for data-driven IK is to use radial basis functions to reconstruct poses by interpolation [MK05]. Where this technique differs is in its ability to generate variation from fewer original animation samples. IK can be thought of as an optimization where the objective function is some function of the error between end-effector and target. Geodesics IK is a refinement that chooses the solver gradient based on knowledge from the input motion data set, rather than attempting to compute a pose by interpolation of that input data.

Suppose we have a character with N joints and D end-effectors, and we have chosen K geodesics for pose reconstruction. The function $f : \mathbb{R}^K \rightarrow \mathbb{R}^{3D}$, mapping geodesics coordinates $t \in \mathbb{R}^K$ to the model space positions of the D end-effectors, is a composition $f = g \circ h$ of the functions $h : \mathbb{R}^K \rightarrow SO(3)^N$, which maps geodesics coordinates to the pose of the character’s N joints, and the forward kinematics function $g : SO(3)^N \rightarrow \mathbb{R}^{3D}$, mapping the skeleton pose to the end-effector positions.

5.2 Geodesics Jacobian

The skeleton pose h is simply the array of N quaternion joint poses $q \in SO(3)$. Similarly each geodesic is an array of N rotation vectors v in the tangent space to $SO(3)$ at the mean pose μ of each joint. We have chosen the first K geodesics from the singular value decomposition that extracts the principal components of variation from the animation data. The pose of each joint is then reconstructed from the geodesics by starting from the mean pose $\mu \in \mathbb{Q}$ and accumulating (as additive rotations) the exponentiations of the weighted geodesics.

So for each joint pose $q \in SO(3)$ we have:

$$q(\mathbf{t}) = \mu \cdot \prod_{j=1}^K e^{t^j v^j}$$

This is a product of differentiable functions (i.e. exponentials) in a Lie group, and is therefore differentiable. The derivative with respect to the i th coordinate t_i is [MSZ94]

$$\frac{\partial q}{\partial t_i} = \mu \cdot \left(\prod_{j=1}^{i-1} e^{t^j v^j} \cdot \frac{\partial}{\partial t_i} (e^{t^i v^i}) \cdot \prod_{j=i+1}^K e^{t^j v^j} \right)$$

where

$$\frac{\partial}{\partial t} (e^{tv}) = \frac{1}{2} \tilde{v} e^{tv}$$

with $\tilde{v}$ being the pure-vector quaternion $[0, v_x, v_y, v_z]$

5.3 End-effector Jacobian

The derivative of g at the pose $h \in SO(3)^N$ is the Jacobian of the end-effectors with respect to the quaternion components $[q_0, q_1, q_2, q_3] \equiv [w, x, y, z]$ of each joint pose. This Jacobian is useful for general IK applications.

Suppose we have an end-effector p with coordinates p_{local} with respect to the m th joint of a chain (where the 0th joint is the skeleton root).

The model-space coordinates of the end-effector are:

$$p_{model} = \left(\prod_{j=0}^m T_j R_j \right) p_{local}$$

where T_j is the translation and R_j is the rotation of the j th joint in the chain with respect to its parent joint.

The derivative of the end-effector position with respect to the k th quaternion component of the i th joint in the chain is:

$$\frac{\partial p_{model}}{\partial q_i^s} = \left(\prod_{j=0}^{i-1} R_j \right) \left(\frac{\partial R_i}{\partial q_i^s} \right) \left(\prod_{j=i-1}^m T_j R_j \right) p_{local}$$

Formally, T and R can be thought of as 4×4 matrices operating on a homogeneous point p . The above formula holds because $\frac{\partial R}{\partial q} p$ comes out as a homogeneous vector x , and so $TRx = Rx$. So for convenience we can consider R to be a 3×3 rotation matrix and $TR : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ to be a general rigid transformation $TRp \mapsto p'$.

The rotation R of a joint is a function of the quaternion components:

$$R = \begin{bmatrix} 1 - 2(y^2 + z^2) & 2(xy - wz) & 2(xz + wy) \\ 2(xy + wz) & 1 - 2(x^2 + z^2) & 2(yz - wx) \\ 2(xz - wy) & 2(yz + wx) & 1 - 2(x^2 + y^2) \end{bmatrix}$$

Hence the derivatives with respect to each quaternion component are:

$$\frac{\partial R}{\partial w} = 2 \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}$$

$$\frac{\partial R}{\partial x} = 2 \begin{bmatrix} 0 & y & z \\ y & -2x & -w \\ z & w & -2x \end{bmatrix}$$

$$\frac{\partial R}{\partial y} = 2 \begin{bmatrix} -2y & x & w \\ x & 0 & z \\ -w & z & -2y \end{bmatrix}$$

$$\frac{\partial R}{\partial z} = 2 \begin{bmatrix} -2z & -w & x \\ w & -2z & y \\ x & y & 0 \end{bmatrix}$$

5.4 Inverse kinematics solver

The Jacobian of the end-effector positions with respect to the geodesics coordinates is the $3D \times K$ matrix $\frac{\partial f}{\partial t}$. By the chain rule, this is the product of the $3D \times 4N$ End-effector Jacobian $\frac{\partial g}{\partial q}$ and the $4N \times K$ Geodesics Jacobian $\frac{\partial h}{\partial t}$.

We use an iterative Levenberg-Marquardt method to solve for the pose. This method finds a solution t for the minimization problem:

$$\min_{\beta} S(\beta) = \sum_{i=1}^n [y_i - f(x_i, t)]^2$$

For the IK algorithm, t is the vector of geodesics coordinates (a weight for each geodesic's contribution to the final pose). Then f is the function mapping geodesics coordinates to end-effector positions, x_i the bone-local end-effector positions and y_i the end-effector targets.

At each iteration the parameter vector t is replaced with a new estimate $t + \delta$, calculated by solving:

$$(J^T J + \lambda \text{diag}(J^T J)) \delta = J^T (y - f(x_i, t))$$

where $J = \frac{\partial f(x_i, t)}{\partial t}$ is the combined IK Jacobian, recalculated at each step, and λ is a damping factor (Levenberg-Marquardt is also known as the *damped least squares* method).

The objective function for the optimisation is a linear combination of the individual end-effector error functions, each of which is the squared distance between the end-effector position

and its goal in world space. Consequently, with multiple end-effectors the goal positions may not be reached exactly; we obtain the least-squares end-effector error.

The pose is initialized to the intrinsic mean pose found during the Principal Geodesics Analysis. At each iteration the optimization updates the geodesics coordinates to find a new weighted combination of the principal geodesics that better approximates the desired end-effector position. This means that the IK pose is repeatable for a given end-effector target, and not history-dependent.

5.4.1 Computational efficiency

There are three stages to each iteration of the IK computation:

- calculate the Geodesics Jacobian
- calculate the End-effector Jacobian
- perform an iteration of Levenberg-Marquardt

The most expensive step is the third, requiring a matrix inversion of the combined IK Jacobian. Using geodesics projects the large parameter space of the whole skeleton pose down to a restricted space of plausible poses, similar to [SHPo4]. So the size of the matrix that must be inverted depends on the number of end-effectors in use and the number of geodesics chosen, fewer than the degrees of freedom of the character.

The computation of the other two steps is non-trivial, and the sizes of the Geodesics and End-effector Jacobian matrices are larger than the final combined IK Jacobian. The implementation could benefit from sparse matrix techniques, particularly when multiplying the Geodesics and End-effector Jacobians.

The End-effector Jacobian has a nonzero block (row i , column j) only when joint j is in the chain from the root joint to end-effector i . And because of the way we have constructed geodesics by decomposing the skeleton into parts, the Geodesics Jacobian has nonzero blocks (row j , column k) only when geodesic k includes joint j .

Each block can be computed independently, making the process highly parallelizable. However, it may also be possible to exploit partial results for common ancestors when recursively multiplying transforms in the joint chains leading to end-effectors.

5.5 Skeleton decomposition

The principal geodesics incorporate correlations between the movements of different body parts. This is the strength of the technique as applies to making plausible poses for data-driven

IK, since it implicitly captures the natural tendency of muscle forces to be distributed across the body and the coupling of disparate body parts by ligaments and tendons.

However, in a general IK application we want to allow disparate body parts to operate independently; for the movement of a hand to be isolated from the movement of the feet. Using full-body geodesics there will always be some correlation between each bone and all the others, limiting the ability to synthesise novel poses not present in the original motion data.

We came up with a simple way to keep the advantage of correlation of movement between nearby bones while allowing generalized IK without undesired interference from other parts of the body. We partition the skeleton into bone subsets (figure 5.1). Overlapping is allowed - some overlapping of the subsets allows desired coupling between actions of separate regions, for example the compliant bending of spine in extreme poses of arm or leg.

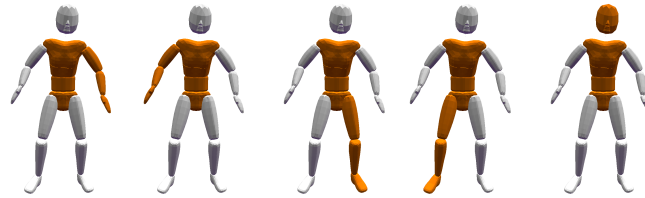


Figure 5.1: Skeleton regions (from left): left arm, right arm, left leg, right leg, spine.

For each region, we mask off other bones in the pose data set (by setting the poses of those bones to identity), and then perform PGA. The intrinsic mean pose of each bone will be the same as for the full data set, since we calculate the mean pose from full data before masking off the bone channels. However, the geodesics calculated for the masked-off region poses are different from the geodesics calculated for the full pose

The result is a set of geodesics for each body region.

The number of principal geodesics required to capture a given percentage of pose variation for each bone subset is typically fewer than for the whole skeleton. Recombining these to reconstruct full-body poses usually gives us a larger total number of geodesics, but a good deal more flexibility in the variety of poses available.

When recombining geodesics for the decomposed full-body IK, we took as many geodesics as required for 90% variation of each region. With the selected geodesics compiled into a list the IK problem is then solved as usual, by minimising a weighted sum of the end-effector error functions.

5.6 Results

Our motion-capture data set consisted of 49 motions, of which 16 were arm-dominant motions, 20 were locomotion type actions and 13 were motions involving both the upper and lower body.

In total, we used 8204 frames of animation. The breakdown can be seen in Tables 5.1, 5.2 and 5.3.

Motion	No. frames
Arm Motion	597
Celebration	74
Chopping Motions	246
Long Reach	477
One Hand Glasses Off	102
One Hand Glasses On	90
Pick Up Left	72
Pick Up Right	75
Punches a	229
Punches b	127
Reach Forward	78
Reach Left	77
Reach Right	59
Reach Up	83
Reaches	805
Taking Off Glasses	110
TOTAL	3301

Table 5.1: Mocap sample animations: arm-dominant motions.

Motion	No. frames
Cross Step Left a	113
Cross Step Left b	91
Cross Step Right	120
Running Bunny Hop High	124
Running Bunny Hop Short	113
Running Bunny Hop a	67
Running Bunny Hop b	53
Step Forward	94
Step Left a	119
Step Left b	113
Step Right a	95
Step Right b	91
Step Up High a	209
Step Up High b	177
Step Up a	220
Step Up b	173
Walk Long Wide Steps	58
Walk Wide One Step	86
Walk Wide Steps	78
Walk	58
TOTAL	2252

Table 5.2: Mocap sample animations: locomotion.

Motion	No. frames
Dodge Left	80
Dodge Right	71
Crab	619
Hitting	287
Leg Motions	404
Leg Over	162
Get Up	148
Flailing	471
Squat	103
Vault High a	119
Vault High b	64
Vault a	61
Vault b	62
TOTAL	2651

Table 5.3: Mocap sample animations: full-body motions.

Figures 5.2 to 5.6 show the percentage variation captured by the geodesics for each region, for several animation sets. Counting from the bottom of a column, it can be seen how many geodesics are required to capture a given percentage of motion variation for that region.

The geodesics extracted from the full data set are shown in Figures 5.7, for the basic full-body geodesics, and 5.8, for the decomposed skeleton geodesics.

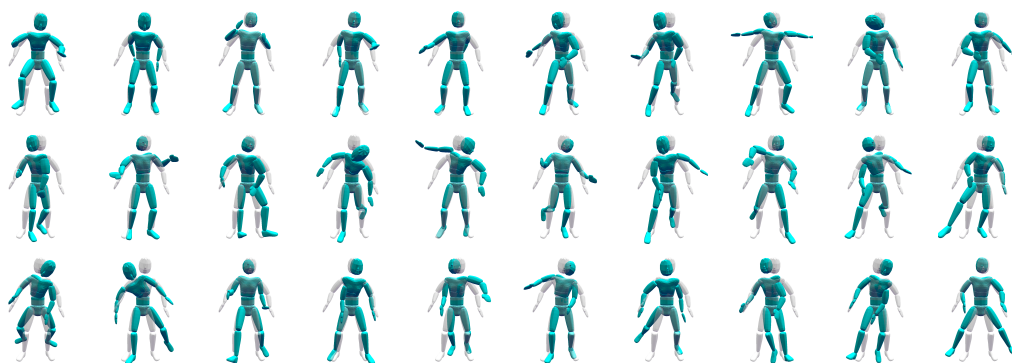


Figure 5.7: PGA: the first 30 full-body geodesics

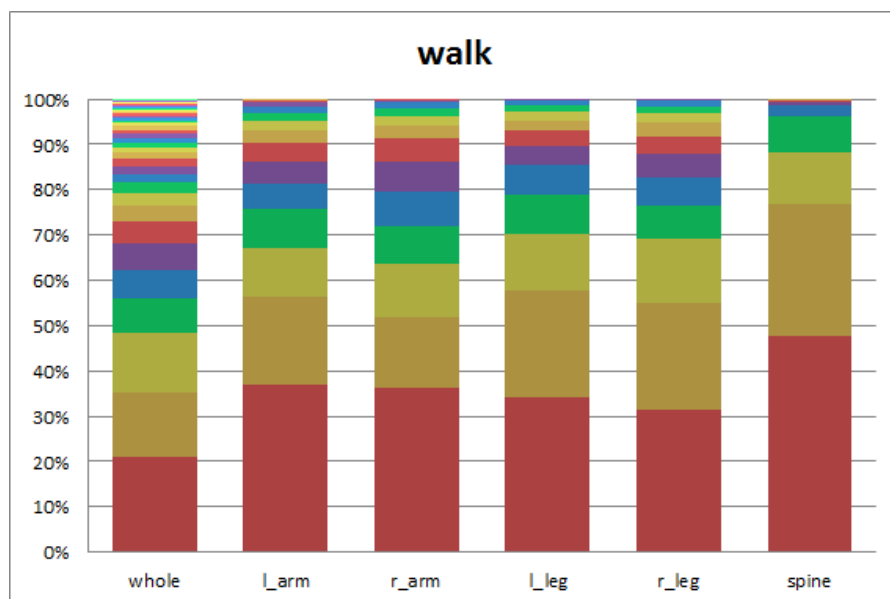


Figure 5.2: Cumulative percentage variation captured by geodesics for different skeleton regions (single walk motion).

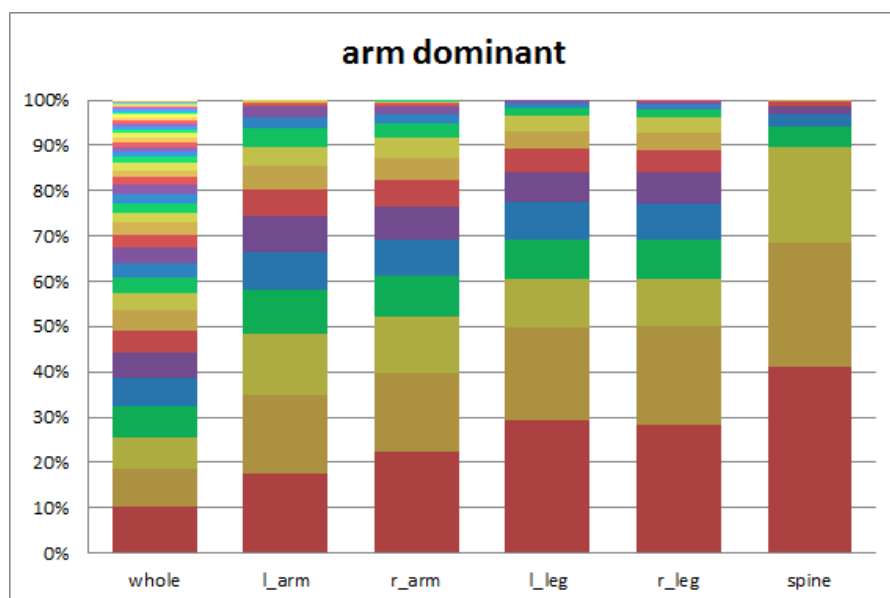


Figure 5.3: Cumulative percentage variation captured by geodesics for different skeleton regions (set of arm-dominant motions).

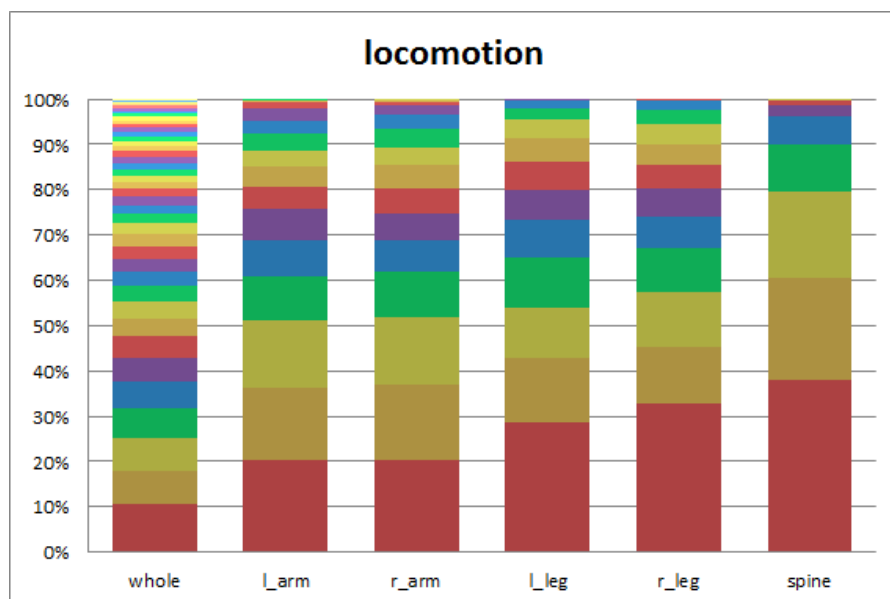


Figure 5.4: Cumulative percentage variation captured by geodesics for different skeleton regions (set of locomotion animations).

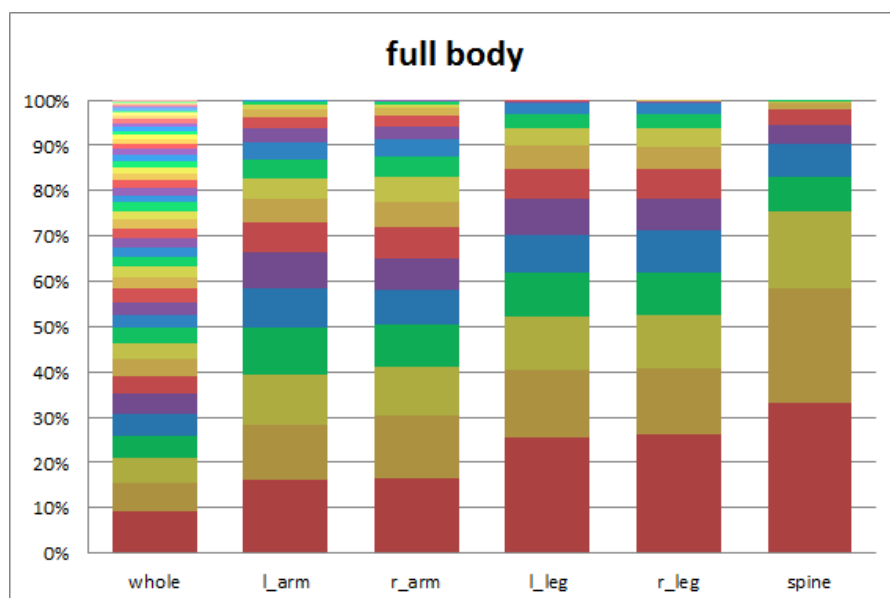


Figure 5.5: Cumulative percentage variation captured by geodesics for different skeleton regions (set of whole-body motions).

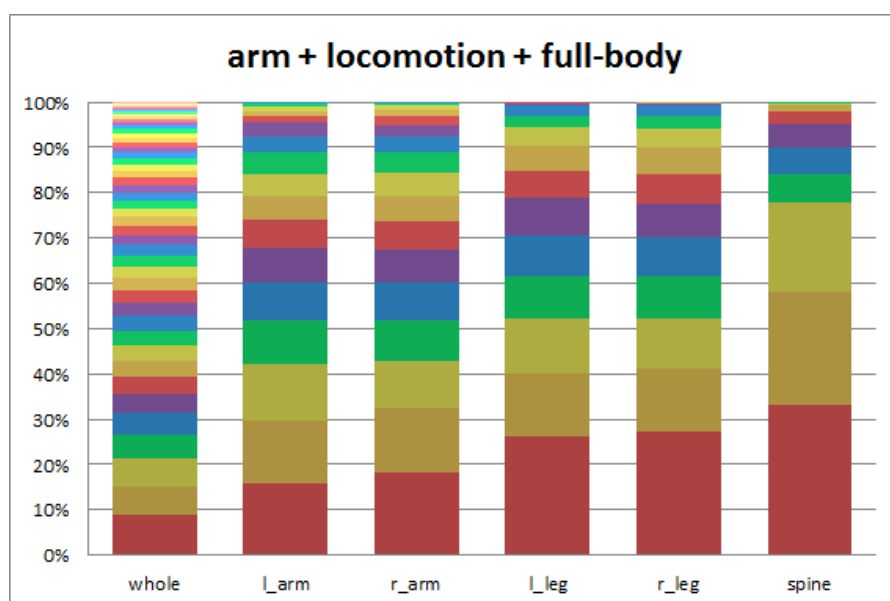


Figure 5.6: Cumulative percentage variation captured by geodesics for different skeleton regions (all motions in data set).



Figure 5.8: PGA: decomposed skeleton geodesics. In rows from top: left arm, right arm, left leg, right leg, spine

We found 10 iterations of the Levenberg-Marquardt solver were typically enough to give reasonable results. However, some target positions were less reachable than others and in those places, increasing iterations does little to help.

A damping (λ) value of around 10 was found to work well. Higher damping makes the IK solution more stable and less susceptible to poor behavior at extreme poses, but in turn requires more iterations to eliminate the end-effector error.

Combining partial-skeleton geodesics gives more stable IK, as can be seen in Figure 5.9. For poses close to the mean pose, the full-skeleton geodesics give visually better results (figure 5.10). However, the further we get from the mean, the more this breaks down; whereas the combined partial-skeleton geodesics remain viable (figures 5.11 and 5.12).

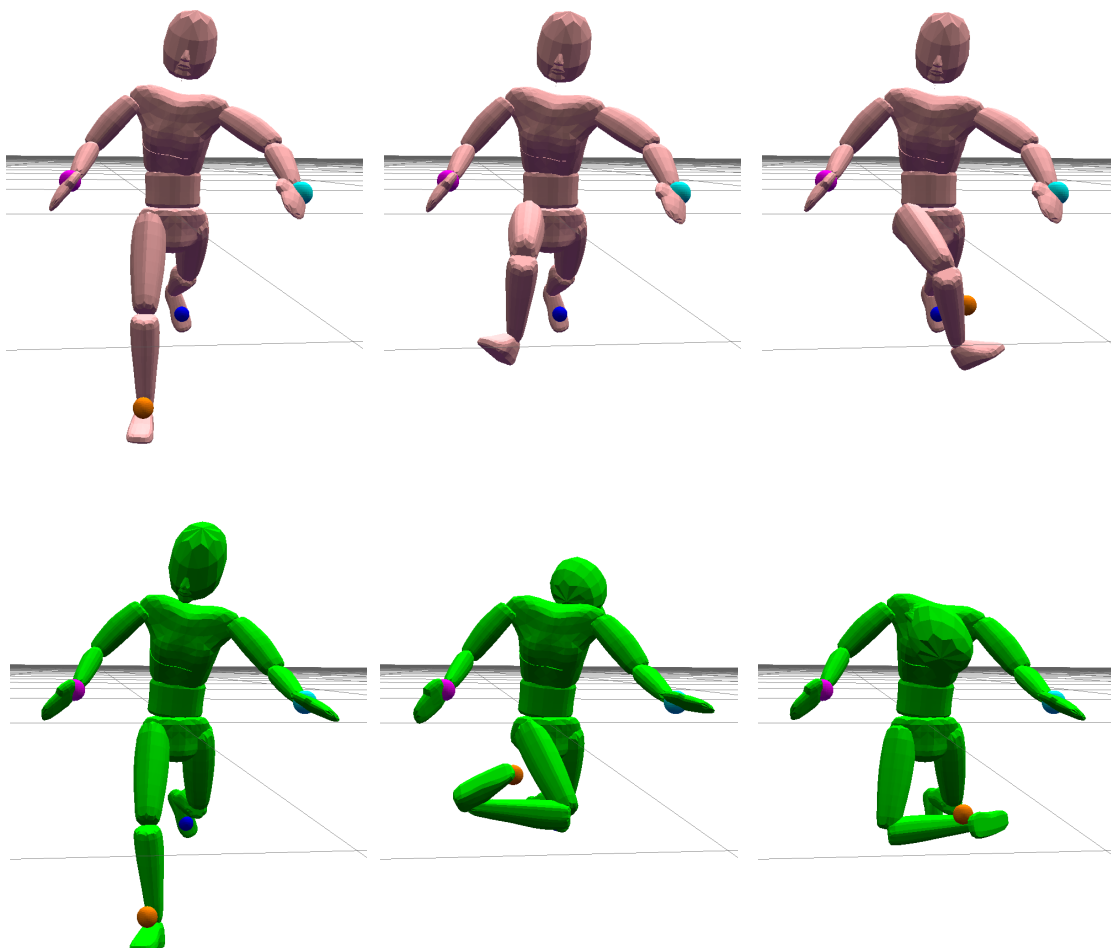


Figure 5.9: Decomposed PGA IK (top) shows better stability than full-body PGA IK (bottom) when poses are some distance from mean pose. Here only the right foot target is moving. Note the large resulting variation in the head pose when using full-body geodesics.

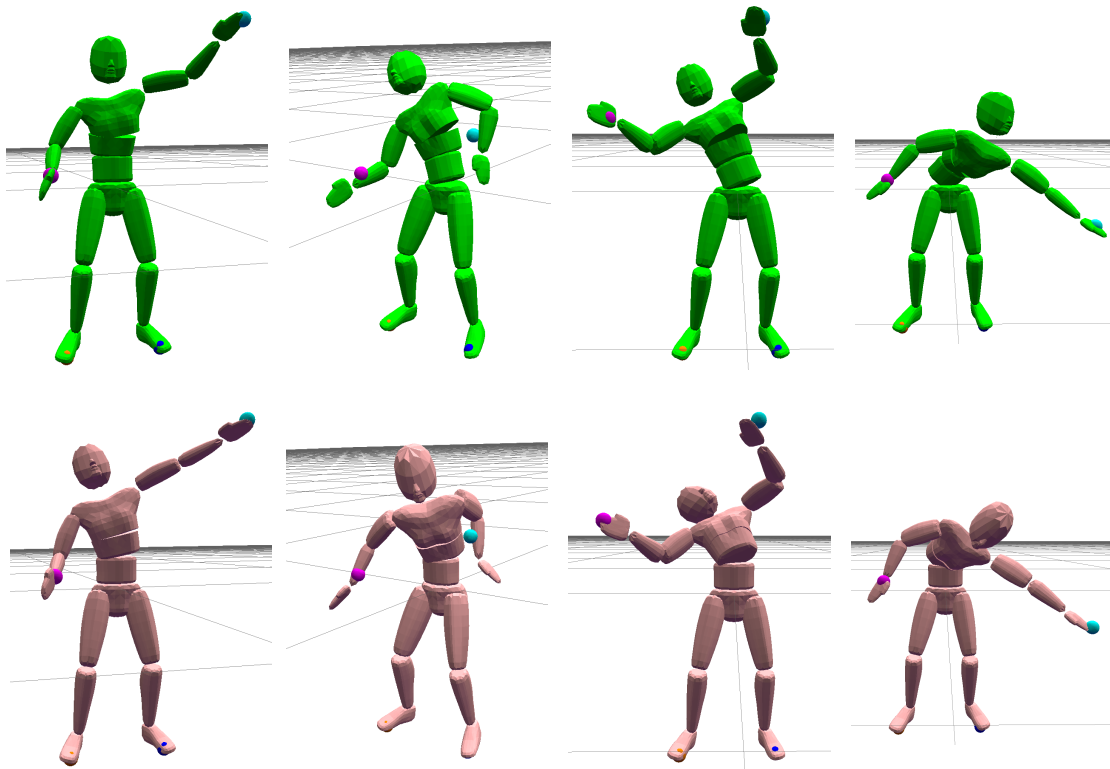


Figure 5.10: Comparing full-body geodesics (green) with decomposed geodesics (pink), given the same end-effector targets. Example poses close to the mean pose. The full-body geodesics tend to give better-looking results.

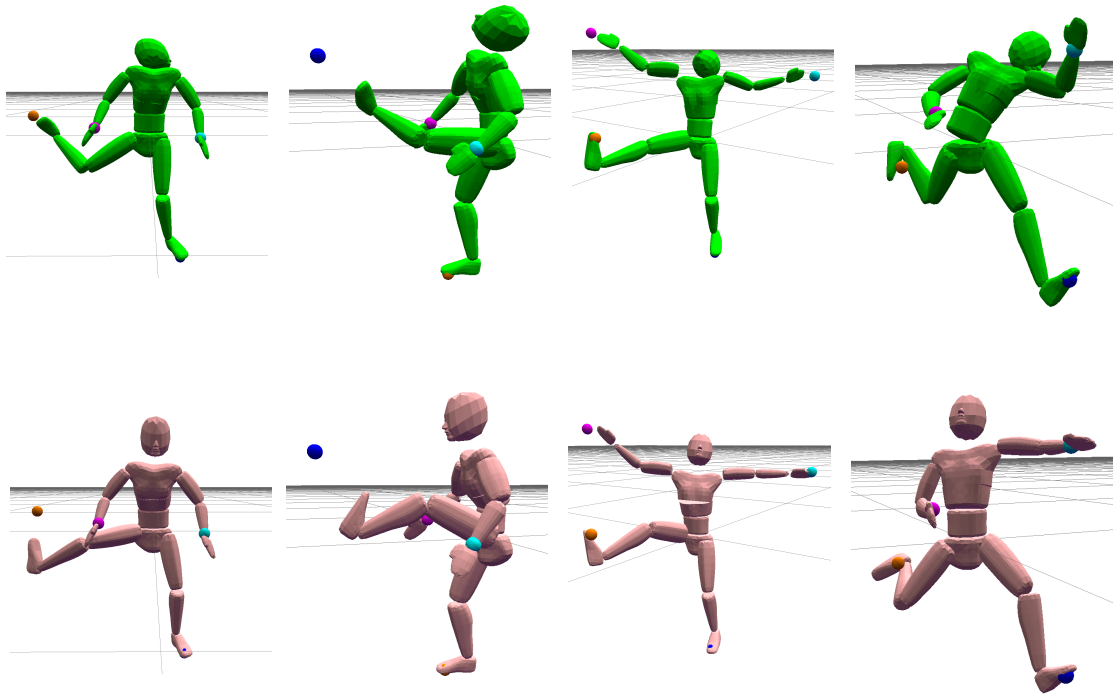


Figure 5.11: Comparing full-body geodesics (green) with decomposed geodesics (pink), given the same end-effector targets. Example poses at intermediate distance from the mean pose. Poses reconstructed from full-body geodesics start to break down, whereas poses from partial-skeleton geodesics remain stable.

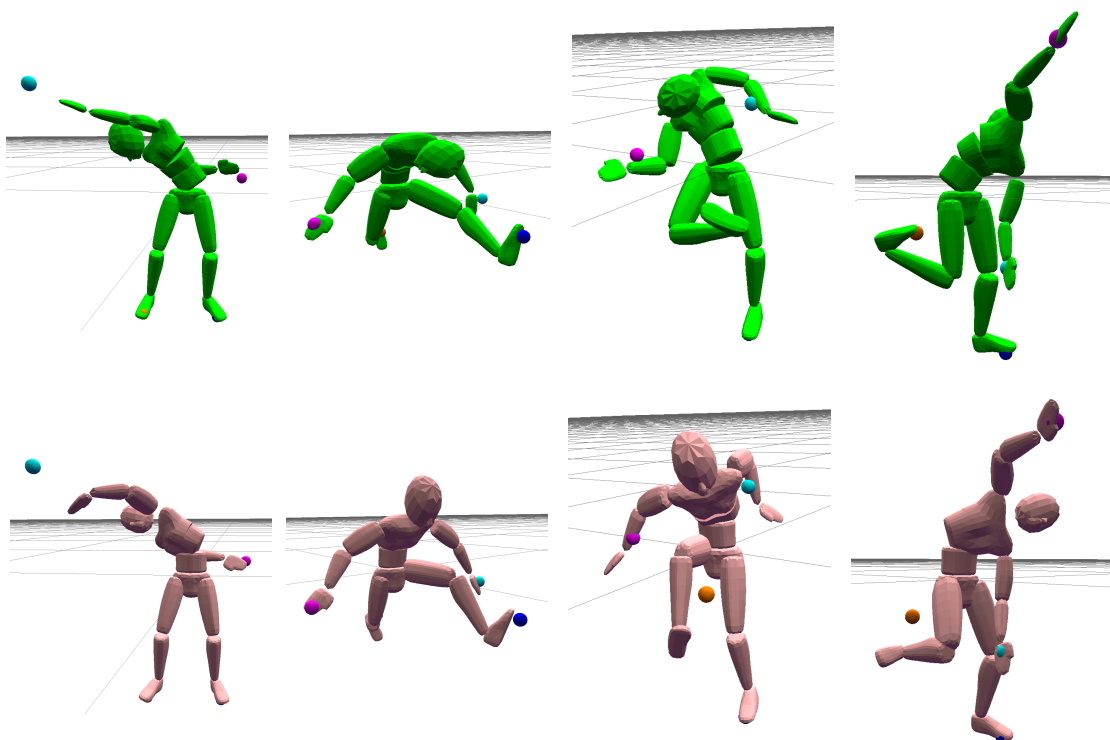


Figure 5.12: Comparing full-body geodesics (green) with decomposed geodesics (pink), given the same end-effector targets. Extreme example poses. Partial-skeleton geodesics outperform full-body geodesics when far from the mean pose.

5.7 Conclusions

We followed the method of Tournier et al. [TWC⁺09], giving full implementation details. Then we introduced the idea of decomposing the skeleton into regions before performing PGA. Re-combining the resulting partial-skeleton geodesics allows a more controllable amount of coupling between the motion of different body parts, by determining the overlaps between the chosen skeleton regions.

The novel aspect of this work is in the crafting of geodesics to extend the PGA IK solution to more flexible full-body poses. Using these geodesics for IK gives a more stable solution with more diverse poses possible, particularly when farther from the mean pose. However, the full-body geodesics typically give better-looking poses close to the mean.

In general, this type of IK solution may fail to reach end-effector targets, particularly when the original pose data has no poses with similar end-effector positioning. The solution may behave poorly when near the boundaries of the allowable pose space. These issues can be ameliorated by increasing the number of solver iterations, and the damping constant respectively. For end-effector targets within the volume reachable by the character, the method tends to give plausible poses.

Chapter 6

Conclusions

6.1 Summary

We began by providing some motivation for the research presented in this thesis, with reference to some of the challenges that arise in advanced video game animation technology. Academic literature provides many approaches to character animation synthesis. The background chapter surveyed some of the literature with the intention of providing a basis for the research presented in the subsequent chapters.

6.1.1 Geometry-aware motion planning

We presented a system for planning motions in an arbitrary geometric scene, based on searching the scene for locations that would support the hand and foot contacts in a set of motions. We constructed a graph of short motion clips annotated with contact information. Each contact, hand or foot, was described by a template shape made up from points and contact normals. We devised a query that could approximately match geometric features in the scene with contact templates and thereby determine whether a given motion could be performed at a given world location.

Doing the many such queries necessary to plan a branching tree of potential motions proved too slow for real-time motion planning. Pre-processing the scene and storing the query results in a spatial octree gave enough of a speed-up to plan feasible motions several seconds ahead. Our example application mapped user input to a choice between these feasible actions, ensuring that the character only performed motions with realistic environmental contacts.

Motion atoms

A short-horizon motion planning system such as the one presented here attempts to balance the conflicting objectives of looking ahead in time far enough to be able to plan out and adapt

a convincing-looking motion, and remaining reactive to the unpredictable inputs supplied by a user. The concept of a *motion atom* is useful here, since it is the smallest interval over which useful planning work can be done.

The support model contained in the definition of a motion atom is intended to provide a basis for physically-based motion editing. A list of contact points (and normals) and the body parts in contact gives a basis for the feasible forces the character can exert to affect its trajectory. For a character's motion to be convincing, there should ideally be some element of anticipation to upcoming contacts, as well as plausible physical characteristics to the motion.

For example: the anticipatory 'wind-up' before a jump is necessary so that the mass-centre has sufficient velocity when the foot leaves the ground - i.e. at the change in support state; while walking, the choice of the next foot plant informs the force and torque that must be applied at the current supporting foot.

This means we need to keep track of not only the current support state but also the next anticipated state, indicating that we must plan over at least one change in support state.

We suggest that any solution for responsive physically-based motion planning will be based on similar concepts.

6.1.2 Vertex-based motion adaptation

We applied the vertex-based soft-body simulation approach to editing character motion. A system of differentiable constraints over a set of vertices can be solved iteratively by a sort of Gauss-Seidel method, as described by Müller et al. [MHHR07]. Inspired by Jakobsen's earlier work [Jako1] we constructed a full character skeleton, including joint limits, from vertices and constraints.

Then we introduced a novel form of these constraints applied over multiple frames of motion. We demonstrated that this type of *spacetime vertex constraint* can be used to synthesise motion for simple systems, before scaling up the application to a full human character. The solver can easily diverge for complex setups like the full character, so we are limited to making small adjustments to existing motions.

We attempted to measure the realism of the adapted motion by comparing it to mocap ground-truth using a simple mass-displacement metric. The dynamics constraints gave better results than IK and smoothing alone for 2 out of 3 tests. For the third test, solving the dynamics constraints caused a balance-preserving displacement of the hips at the beginning of the motion in response to shifting the starting foot plant position. This displacement caused the resulting motion to differ significantly from the mocap ground-truth, while still appearing dynamically plausible.

The displacements sometimes did not match the natural motion of the mocap performer,

revealing one of the weaknesses of the scheme. Having no model of biomechanical quantities such as energy usage, muscle effort and tendon elasticity, our vertex-based technique tended to give results that looked physically possible but not natural, placing the character in poses that might be uncomfortable for a human being to perform.

6.1.3 Data-driven inverse kinematics using principal geodesics

We re-implemented the very interesting work of Tournier et al. [TWC⁺09] with a view to investigating its application to more general full-body pose synthesis by inverse kinematics. Since certain details of the implementation were non-trivial to track down from the original publication and citations, we shared those details here.

Then we split the character skeleton into regions and calculated geodesics that capture the motion variation in the animation data set separately for each region. Since the ‘partial’ geodesics are calculated from a common mean pose, we can trivially recombine them in the IK solution. This gives independence between the motion of disparate body parts - so for instance the motion of the foot need not affect the head as it might using only full-body geodesics.

The result is a more stable IK solver that can give a wider variety of plausible poses. The downside is that the recombined partial-pose solution tends to take more solver iterations to reach end-effector targets.

Future work

Though we have not implemented it, it would be possible to control the centre-of-mass position of the character using this method. We could define the mass-centre of each body part, fixed relative to a given bone, as an end-effector. Then the mass-centre Jacobian is trivial to calculate as a function of the mass-centres of each body part. Composing this with the Geodesics and End-effector Jacobians would give the Jacobian of the character’s centre-of-mass with respect to the geodesics coordinates. This could work alongside the existing end-effector IK.

It would also be interesting to investigate augmenting or bypassing the PGA portion of the method by hand-authoring geodesics. The behavior of the IK solution could be tailored with a tool allowing an animator to design partial- and full-skeleton geodesics, making this into a fully authored data-driven IK solution.

6.2 Future directions

In the future we would like to explore some extensions to work done here, and some other approaches to the problem of real-time motion adaptation and synthesis. For example:

- Improvements to the vertex-based approach.

Multiple levels of detail might make larger adaptations more stable and practical. A low-detail aggregate motion pass would determine the gross motion of the character by simulating the character as a simpler collection of vertices. Later higher-detail passes would then fix up the skeleton pose appropriately.

It is not clear whether biomechanical models could be captured as vertex constraints. Such models require an underlying model of body structure, in this case being provided by multiple vertices and constraints representing the skeleton and body mass. Adding further constraints risks the system becoming over-constrained and unlikely to converge to a solution.

- Combined approach

To some extent, the vertex-based motion adaptation and the principal geodesics IK methods are complementary and might be combined to give a better motion adaptation scheme. Geodesics IK gives quite robust results that preserve the human style of the mocap animation, but may not be exact in reaching end-effector goals. In contrast, the physically-based spacetime vertex approach is better at satisfying kinematic constraints and dynamical realism, but poor at making large deformations to a motion and has no metric of plausible style other than dynamics.

After planning a motion sequence we have the problem of adapting that motion to satisfy external contacts that in general do not line up with where they were in the original animations. We could have a two-stage motion adaptation step: first geodesics IK deforms the animations to approximately the correct positions - giving plausible human-looking poses but with no concern for dynamical realism; and finally we run the physically-based method to touch up the deformed motion, improving the compliance with the equations of motion and enforcing solid end-effector contacts.

- A more general support model for representing the possible forces a character can use to actuate its mass-centre motion.

The Jacobian of the contact points with respect to joint movement would give an estimate of how a character could push and pull against its surroundings to move its body about. We could calculate a model that included not only simple hand and foot contacts as in this thesis, but also general contacts with all body parts. Off-the-shelf collision detection software would help here.

- Further investigation into geodesics-based IK.

We would like to implement centre-of-mass control as described in Chapter 5.

Also, there may be synergy to explore between this approach and articulated-body dynamics approaches. Geodesics indicate preferred modes of actuation of a character's joints, in effect a type of 'muscle'. It would be interesting to see whether geodesics could be incorporated into inverse dynamics calculations, resulting in more realistic ragdoll actuation forces.

We can infer the joint torques driving a motion using the Recursive Newton-Euler Algorithm of Featherstone; applying PCA to the torques from a set of motion data, we could extract a low-dimensional representation of character actuation. Optimising over this relatively small set of control parameters, we could find joint torques to drive a physically simulated character with realistic motion style.

Appendix A

Generalised IK

The IK method of Chapter 5 finds a skeleton pose by optimisation, where we find the optimal value of a set of control parameters producing the pose such that a given cost function is minimised.

For geodesics IK, the control parameters are the weights on the geodesics, and the cost function is the combined error between the end-effector positions and their targets.

We can generalise this by allowing general *rigging* functions that map from control parameters to pose, and general *cost* functions that provide some metric of fitness of the output pose. As long as these functions are differentiable, we can apply the same optimisation to solve for the pose.

A.1 Rigging and Cost functions

Suppose we have:

- Differentiable *Rigging functions*

$$R(p) \mapsto q \tag{A.1}$$

mapping some vector of parameters p to a pose $q \in SO(3)^n$

- *Cost functions*

$$C(q, x) \mapsto k \in \mathbb{R} \tag{A.2}$$

where x is some other vector of parameters (such as end-effector target positions)

Here the rigging functions can be thought of as the ‘motors’ driving a pose. A rigging function could be as simple as a single parameter to drive elbow bending; or it could be a more

complex expression modelling the effect of a muscle contraction on the pose, or even a set of muscles such as along the spine.

Typically animators use a set of variable sliders and positional inputs to control the pose of a character, rather than tweaking the bone poses directly. Rigging functions could be used to model an animator's rig directly, reducing the pose parameter space down to a lower-dimensional control parameter space.

End-effector error is a simple function of the difference between the world-space target and the FK position of the end-effector, with a weighted sum for multiple end-effectors.

The cost functions' next most obvious purpose is to model joint limits. Using a barrier function, we could model hard joint limits. However, human joints have some compliance at their limits which could be modelled using a cost function with a steep but continuous gradient near the boundary of the allowable pose space.

Cost functions could also model biomechanical properties such as a tendon stretching over some joint chain; or muscle effort required to hold a given pose. Combining a set of cost functions modelling these factors would give a metric of 'comfort' for a given pose, potentially producing biomechanically plausible animations.

A.2 Solving

To solve the IK problem we minimise for the rigging control parameters p using the cost C

$$C(R(p), x) \tag{A.3}$$

Using the Levenberg-Marquardt method we iterate solving for $\delta = \Delta p$

$$(J^T J + \lambda \text{diag}(J^T J)) \delta = J^T (C(R(p), x)) \tag{A.4}$$

where

$$J = \frac{\partial C}{\partial q} \frac{\partial R}{\partial p} \tag{A.5}$$

Appendix B

Simple motion fix-up using analytical IK

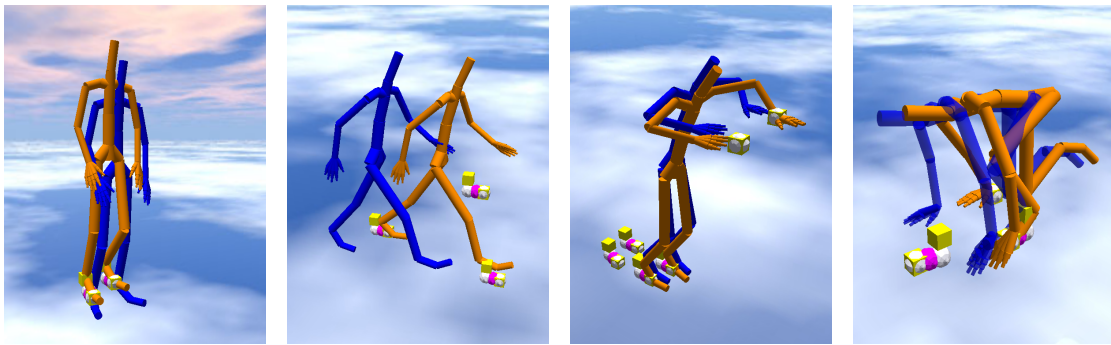


Figure B.1: Simple motion fix-up using analytical IK and spring-damper for skeleton root position. Original motion (blue) and fixed-up motion (orange).

To test the early prototype for motion planning we made a simple system to adapt the ‘unrolled’ motion (see Chapter 3) to the planned foot- and hand-contacts.

The adaptation consisted of several stages:

1. Adjust the origin of the action’s local space in the scene to give a best fit between between the original expected docks and the adapted docks. We considered the docking contact locations as a set of points and computed a rigid transform to minimize the sum of squared residual displacements between corresponding points [Ume91].
2. Find the desired root position during the constrained intervals, as follows:
 - Pose the character at the constrained frames of the animation.
 - Perform a gradient descent minimisation of an error function based on the weighted sum of the squared lengths of the displacements between the nominal end-effector positions and the support positions found by the geometry query.

- Ensure that each constrained limb can reach its target position by reducing the error weight of a limb to zero when the target is within the reach of the limb; the error function then depends only on the limbs that cannot reach their goals.
3. Drive the character's root bone toward this target by integrating a model of a damped spring between the root's current location and the desired position.
 4. Pose each constrained limb by analytical inverse kinematics [TGBoo]. This solver has the advantage of repeatability, producing the same limb pose for the same inputs; this ensures coherence between independently posed frames. The solutions to the limb inverse kinematics were blended in and out over short periods before and after the constraints' existence to prevent discontinuity artefacts ('popping').

This solution was fast to compute but contained no measure of balance or physical realism. The resulting unrealistic accelerations of the character's centre of mass could possibly be reduced in step 3 by modelling balance and the motion of the character's centre of mass under support forces as in [SKGo3].

Appendix C

Optimization-based manipulation of momentum splines

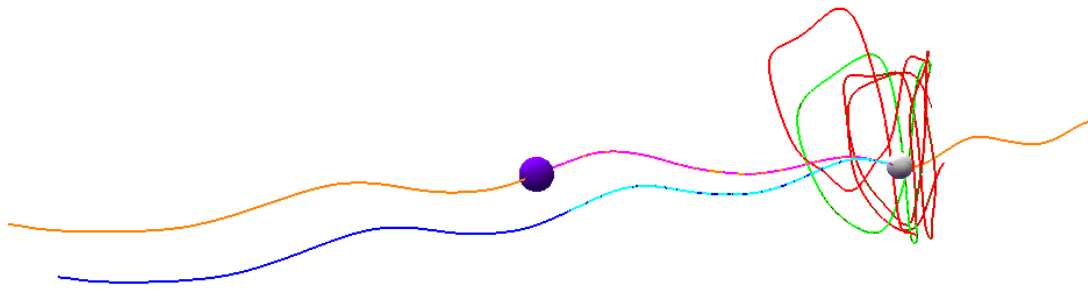


Figure C.1: Momentum spline adjustment. Respectively: original/adjusted COM path (blue/orange); interval of adjustment (cyan/magenta); original/adjusted momentum curve (green/red).

Suppose we wish to displace the centre of mass of the character by some amount at a desired time during the animation. By representing the momentum as a spline we can continuously adjust the mass displacement over some time window.

The key observation is that we can develop efficient expressions for the integral and derivative of a b-spline curve, allowing us to link some model of allowable forces (momentum derivative) to mass displacement over time (momentum integral). We consider only linear momentum since this method only takes into account the mass centre position and not the angular inertia.

It was intended that this momentum adjustment would be the first stage of animation synthesis, with pose adjustment happening later by some IK solution.

C.1 Cubic b-spline curve notation

A cubic b-spline with n segments has $n + 3$ control points $P_i, i \in \{0, n + 2\}$ and $n + 7$ knots $t_i, i \in \{-3, n + 3\}$. The value of the spline function $p(t)$ at parameter $t \in [t_i, t_{i+1})$ is

$$p(t) = P_i B_{i+0}(t) + P_{i+1} B_{i+1}(t) + P_{i+2} B_{i+2}(t) + P_{i+3} B_{i+3}(t) \quad (\text{C.1})$$

C.2 Cubic b-spline integral

By expanding the expressions for the basis functions given above we can express each basis function as a cubic polynomial in t

$$B(t) = \alpha t^3 + \beta t^2 + \gamma t + \delta \quad (\text{C.2})$$

The value of the spline curve over a knot interval $i, t \in [t_i, t_{i+1})$ depends only on the 4 local control points $P_i, P_{i+1}, P_{i+2}, P_{i+3}$. Hence, the value of the integral over that knot interval also depends only on the local control points.

$$\begin{aligned} \int p(t) dt = & P_i \left(\int B_{i+0}(t) dt \right) + P_{i+1} \left(\int B_{i+1}(t) dt \right) + \\ & P_{i+2} \left(\int B_{i+2}(t) dt \right) + P_{i+3} \left(\int B_{i+3}(t) dt \right) \end{aligned}$$

The integral of a basis function over the knot interval is simply the integral of the cubic polynomial expression

$$\begin{aligned} \int_{t_i}^{t_{i+1}} B(t) dt &= \int_{t_i}^{t_{i+1}} (\alpha t^3 + \beta t^2 + \gamma t + \delta) dt \\ &= \left\{ \frac{1}{4} \alpha t^4 + \frac{1}{3} \beta t^3 + \frac{1}{2} \gamma t^2 + \delta t \right\}_{t_i}^{t_{i+1}} \end{aligned}$$

C.3 Cubic b-spline derivative

Similarly, we can calculate the time derivative of the spline curve using

$$\frac{d}{dt} B(t) = \{3\alpha t^2 + 2\beta t + \gamma\}$$

and

$$\begin{aligned} \frac{d}{dt}p(t) = & P_i \frac{d}{dt}B_{i+0}(t) + P_{i+1} \frac{d}{dt}B_{i+1}(t) + \\ & P_{i+2} \frac{d}{dt}B_{i+2}(t) + P_{i+3} \frac{d}{dt}B_{i+3}(t) \end{aligned}$$

C.4 Momentum curve adjustment

Suppose we wish to displace the centre of mass by a vector $\mathbf{d}$. We choose the time t at which the displacement occurs to be knot value t_i , and we will have the adjustment happen over k knot intervals from knot values t_{i-k} to t_i .

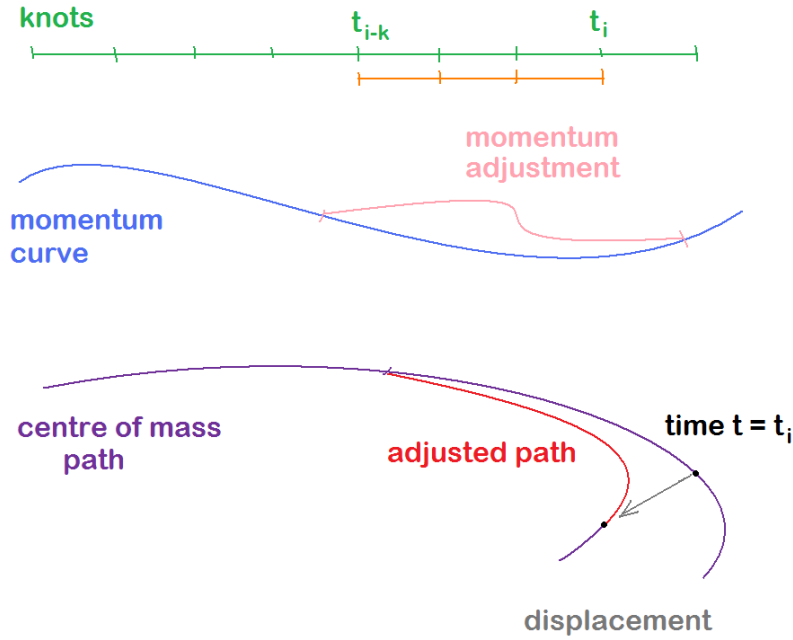


Figure C.2: Schematic of mass displacement at t_i , with adjustment over the previous k knot intervals.

Here we restrict our adjustments to occur only at discretised times equal to the knot values. This considerably simplifies the expressions and makes the calculation of constraints somewhat more computationally efficient too.

The Quadratic Optimisation Problem (QP)

$$\begin{aligned} \min_{\mathbf{x}} \quad & g(\mathbf{x}) = \left(\frac{1}{2} \mathbf{x}^T Q \mathbf{x} + \mathbf{q}^T \mathbf{x} \right) \\ \text{subject to} \quad & \\ & A_e \mathbf{x} = \mathbf{b}_e \text{ (equality constraints)} \\ & A_i \mathbf{x} \geq \mathbf{b}_i \text{ (inequality constraints)} \end{aligned}$$

is a constrained optimisation that will give us the adjusted control points of a new momentum curve $p'(t)$ whose integral results in the desired mass displacement from the original path of the centre of mass, and whose derivative at each knot lies within a chosen space of forces.

The mass displacement requirement gives us equality constraints

$$\sum_{j=i-k}^{j=i} \int_{t_j}^{t_{j+1}} p'(t) dt = \left(\sum_{j=i-k}^{j=i} \int_{t_j}^{t_{j+1}} p(t) dt \right) + M \mathbf{d}.$$

We can precalculate the value of the integral on the right hand side as it is a property of the original animation. We also constrain the starting and ending control points to be the same as in the original spline, so that the adjustment transitions smoothly. Constraining the derivative value at each knot $t_j \in [t_{i-k}, \dots, t_i]$ gives us the equality constraints

$$\begin{aligned} M \frac{d}{dt} p'(t) \Big|_{t=t_j} &= F(t) \Big|_{t=t_j} \\ &= Mg + \sum_{l=1}^r f_{j_l} \lambda_{j_l}, \end{aligned}$$

thus the derivative of the spline at a knot is equal to the combination of forces at that time.

The inequality constraints

$$\lambda_j \geq 0,$$

enforce the unilateral, non-sticking, nature of the contact forces being represented. Gripping contacts can be represented either with bilateral force multipliers, or by adding more basis forces to extend the space of possible forces in all directions.

C.5 Conclusions

We used the free solver QuadProg++ [DG], which employs a Goldfarb-Idnani active-set dual method to solve positive-definite Quadratic Problems in polynomial time.

It would be possible to extend the b-spline calculations to allow for adjustments at arbitrary times, not just at knot times. The principles are the same, but the expressions become a bit more complex.

It would also be possible to use the same scheme to displace end-effector positions by adjusting the velocity of the effectors over a window of animation. This is in fact a simpler calculation than for the centre of mass, requiring only the equality constraints on the spline integral. However, to make this useful we would have to incorporate a measure of the dynamic balance of the character, such as the Zero-Moment Point (ZMP) employed in the control of upright walking robots [SBo4], so that the hand/foot placement could interact with the centre of mass position.

While linear momentum seems to be dominant in the human eye's perception of dynamical realism [Wu09], angular momentum also plays a part. It is not clear how to extend the scheme to include consideration of angular momentum since the character's inertia tensor depends on pose.

The objective function in the optimization was simply sum of squared deviation from the input momentum. This is too simplistic to capture any interesting features of human motion.

To extend this technique to controlling several mass-centres of subsets of the character we would need some objectives that capture force transfer between limbs. The restriction to linear momentum makes this technique poor from that viewpoint.

Appendix D

Glossary of terms and acronyms

COM: Centre of mass

End-effector: A point defined in the local coordinate space of one bone of a skeleton. Typically this represents an important location at the end of a bone chain, such as the palm of the hand.

FK: Forward kinematics. Calculation of the resultant position of an end-effector given the values of the joint variables.

IK: Inverse kinematics. Calculation of joint variables such that an end-effector reaches a given goal (typically a position in world space).

Mocap: Motion-captured animation.

PCA: Principal components analysis. A way to reduce the dimensionality of a set of data without losing much information. A large set of input data comprising many samples of possibly correlated variables is reduced to a set of uncorrelated *principal components*, spanning as much of the data as possible. The first principal component represents the direction of largest variance in the input data, and each succeeding component has the largest variance possible while being orthogonal to the preceding components.

PGA: Principal geodesics analysis - principal components analysis performed on rotation data. Rotation data lives in a nonlinear space, so to capture its variance PCA is performed in the linear tangent space to the rotation space at a particular point. Performing PCA in the tangent space at the mean pose of a set of pose data gives a set of *principal geodesics*, or directions in rotation space away from the mean pose. The input pose data can be reconstructed with minimal loss by a weighted combination of these geodesics.

Solver: An algorithm used in optimization, to solve for the desired state of a system such that a set of constraints is satisfied.

ZMP: Zero-moment point. An indicator of dynamic balance.

Bibliography

- [AdSP07] Yeuhi Abe, Marco da Silva, and Jovan Popović. Multiobjective control with frictional contacts. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 249–258, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.
- [AF02] Okan Arikan and D. A. Forsyth. Interactive motion generation from examples. *ACM Trans. Graph.*, 21(3):483–490, 2002.
- [AFO03] Okan Arikan, David A. Forsyth, and James F. O’Brien. Motion synthesis from annotations. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 402–408, New York, NY, USA, 2003. ACM.
- [ALP06] Yeuhi Abe, C. Karen Liu, and Zoran Popović. Momentum-based parameterization of dynamic character motion. *Graph. Models*, 68(2):194–211, 2006.
- [AP06] Yeuhi Abe and Jovan Popović. Interactive animation of dynamic manipulation. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 195–204, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.
- [Aut] Autodesk. Humanik <http://www.http://gameware.autodesk.com/humanik>.
- [Bar96] David Baraff. Linear-time dynamics using lagrange multipliers. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 137–146, New York, NY, USA, 1996. ACM.
- [BB98] Rama Bindiganavale and Norman I. Badler. Motion abstraction and mapping with spatial constraints. In *CAPTECH '98: Proceedings of the International Workshop on Modelling and Motion Capture Techniques for Virtual Environments*, pages 70–82, London, UK, 1998. Springer-Verlag.

- [Bj095] Jørgen Bjørnstrup. Estimation of human body segment parameters - historical background. Technical report, Laboratory of Image Analysis, Institute of Electronic Systems, Aalborg University, 1995.
- [BK05] Samuel R. Buss and Jin-Su Kim. Selectively damped least squares for inverse kinematics. *J. Graphics Tools*, 10(3):37–49, 2005.
- [BKM07] Jacek Blaszczyk, Andrzej Karbowski, and Krzysztof Malinowski. Object library of algorithms for dynamic optimization problems: Benchmarking sqp and nonlinear interior point methods. *Applied Mathematics and Computer Science*, 17(4):515–537, 2007.
- [BMT96] Ronan Boulic, Ramon Mas, and Daniel Thalmann. Robust position control of the center of mass with second order inverse kinetics. In *In Computers Graphics Journal*, 1996.
- [BN88] Lynne Shapiro Brotman and Arun N. Netravali. Motion interpolation by optimal control. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 309–315, New York, NY, USA, 1988. ACM.
- [BPvdP07] Philippe Beaudoin, Pierre Poulin, and Michiel van de Panne. Adapting wavelet compression to human motion capture clips. In *Proceedings of Graphics Interface 2007, GI '07*, pages 313–318, New York, NY, USA, 2007. ACM.
- [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.
- [CBSGo8] Patrick Coleman, Jacobo Bibliowicz, Karan Singh, and Michael Gleicher. Staggered poses: a character motion representation for detail-preserving editing of pose and coordinated timing. In *SCA '08: Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 137–146, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.
- [CBTo7] Schubert Carvalho, Ronan Boulic, and Daniel Thalmann. Motion Pattern Preserving IK Operating in the Motion Principal Coefficients Space. In *Proceedings of WSCG 2007*, pages 97–104, 2007.
- [CLS03] Min Gyu Choi, Jehee Lee, and Sung Yong Shin. Planning biped locomotion using motion capture data and probabilistic roadmaps. *ACM Trans. Graph.*, 22(2):182–203, 2003.

- [CZW07] Bill Chiu, Victor Zordan, and Chun-Chih Wu. State-annotated motion graphs. In *VRST '07: Proceedings of the 2007 ACM symposium on Virtual reality software and technology*, pages 73–76, New York, NY, USA, 2007. ACM.
- [DG] Luca Di Gaspero. Quadprog++.
- [dL96] Paolo de Leva. Adjustments to zatsiorsky-seluyanov's segment inertia parameters. *Journal of Biomechanics*, 29(9):1223 – 1230, 1996.
- [Fô3] Martin Fêdor. Application of inverse kinematics for skeleton manipulation in real-time. In *SCCG '03: Proceedings of the 19th spring conference on Computer graphics*, pages 203–212, New York, NY, USA, 2003. ACM.
- [Fea00] Roy Featherstone. Robot dynamics: Equations and algorithms. In *IEEE Int. Conf. Robotics and Automation*, pages 826–834, 2000.
- [Fea07] Roy Featherstone. *Rigid Body Dynamics Algorithms*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.
- [FP03] Anthony C. Fang and Nancy S. Pollard. Efficient synthesis of physically valid human motion. *ACM Trans. Graph.*, 22(3):417–426, 2003.
- [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.
- [GBT06] Pascal Glardon, Ronan Boulic, and Daniel Thalmann. Robust on-line adaptive footplant detection and enforcement for locomotion. *Vis. Comput.*, 22(3):194–209, 2006.
- [GL98] Michael Gleicher and Peter Litwinowicz. Constraint-based motion adaptation. *Journal of Visualization and Computer Animation*, 9(2):65–94, 1998.
- [Gle97] Michael Gleicher. Motion editing with spacetime constraints. In *SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 139–ff., 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.

- [GMHPo4] Keith Grochow, Steven L. Martin, Aaron Hertzmann, and Zoran Popović. Style-based inverse kinematics. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 522–531, New York, NY, USA, 2004. ACM.
- [GMMS97] Philip E. Gill, Walter Murray, Michael, and Michael A. Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12:979–1006, 1997.
- [Gra98] F. Sebastian Grassia. Practical parameterization of rotations using the exponential map. *J. Graph. Tools*, 3(3):29–48, March 1998.
- [Gro] Human Animation Working Group. Iso/iec fcd 19774 humanoid animation (h-anim).
- [GSKJ03] Michael Gleicher, Hyun Joon Shin, Lucas Kovar, and Andrew Jepsen. Snap-together motion: assembling run-time animations. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 702–702, New York, NY, USA, 2003. ACM.
- [GW93] Michael Gleicher and Andrew Witkin. Supporting numerical computations in interactive contexts. In *In Graphics Interface*, pages 138–145, 1993.
- [Her11] Ivo Herzeg. Crysis 2: Getting more interactivity out of animation data. Lecture, Game Developers Conference, March 2011.
- [HGo7] Rachel Heck and Michael Gleicher. Parametric motion graphs. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pages 129–136, New York, NY, USA, 2007. ACM.
- [HKT10] Edmond S. L. Ho, Taku Komura, and Chiew-Lan Tai. Spatial relationship preserving character motion adaptation. In *ACM SIGGRAPH 2010 papers*, SIGGRAPH '10, pages 33:1–33:8, New York, NY, USA, 2010. ACM.
- [HRE⁺08] Chris Hecker, Bernd Raabe, Ryan W. Enslow, John DeWeese, Jordan Maynard, and Kees van Prooijen. Real-time motion retargeting to highly varied user-created morphologies. *ACM Trans. Graph.*, 27:27:1–27:11, August 2008.
- [IAFo6] Leslie Ikemoto, Okan Arikan, and David Forsyth. Knowing when to put your foot down. In *I3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 49–53, New York, NY, USA, 2006. ACM.
- [Jak01] Thomas Jakobsen. Advanced character physics. Proceedings, Game Developers Conference, 2001.

- [JMOB05] Oliver Junge, Jerrold E. Marsden, and Sina Ober-Blobaum. Discrete mechanics and optimal control. In *IN PROCEEDINGS OF THE 16TH IFAC WORLD CONGRESS*, page 6, 2005.
- [Joh09] Rune Skovbo Johansen. Automated semi-procedural animation for character locomotion. Master's thesis, Aarhus University, May 2009.
- [JYL09] Sumit Jain, Yuting Ye, and C. Karen Liu. Optimization-based interactive motion synthesis. *ACM Trans. Graph.*, 28(1):1–12, 2009.
- [KB84] Doris H. U. Kochanek and Richard H. Bartels. Interpolating splines with local tension, continuity, and bias control. *SIGGRAPH Comput. Graph.*, 18(3):33–41, 1984.
- [KCvO07] Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. Skinning with dual quaternions. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pages 39–46, New York, NY, USA, 2007. ACM.
- [KG03] Lucas Kovar and Michael Gleicher. Flexible automatic motion blending with registration curves. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 214–224, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.
- [KGPO2] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 473–482, New York, NY, USA, 2002. ACM.
- [KGS02] Lucas Kovar, Michael Gleicher, and John Schreiner. Footskate cleanup for motion capture editing. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 97–104, 2002.
- [KHL05] Taku Komura, Edmond S. L. Ho, and Rynson W. H. Lau. Animating reactive motion using momentum-based inverse kinematics: Motion capture and retrieval. *Comput. Animat. Virtual Worlds*, 16(3-4):213–223, 2005.
- [KKS04] Katsuaki Kawachi, Takeo Kanade, and Hiromasa Suzuki. A lightweight algorithm for real-time motion synthesis. In *VRST '04: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 49–56, New York, NY, USA, 2004. ACM.
- [KM05] R. Kulpa and F. Multon. Fast inverse kinematics and kinetics solver for human-like figures. In *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, pages 38–43, Dec. 2005.

- [KMA05] R. Kulpa, F. Multon, and B. Arnaldi. Morphology-independent representation of motions for interactive human-like animation. *Computer Graphics Forum, Eurographics 2005 special issue*, 24:343–352, 2005.
- [Lab88] Harry G Armstrong Aerospace Medical Research Lab. Anthropometry and mass distribution for human analogues. volume 1. military male aviators. Technical report, Wright-Patterson AFB OH, Mar 1988.
- [LCBo6a] Benoît Le Callennec and Ronan Boulic. Interactive motion deformation with prioritized constraints. *Graph. Models*, 68(2):175–193, 2006.
- [LCBo6b] Benoît Le Callennec and Ronan Boulic. Robust kinematic constraint detection for motion data. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 281–290, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.
- [LCL06] Kang Hoon Lee, Myung Geol Choi, and Jehee Lee. Motion patches: building blocks for virtual environments annotated with motion data. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 898–906, New York, NY, USA, 2006. ACM.
- [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. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 491–500, New York, NY, USA, 2002. ACM.
- [Lee08] Jehee Lee. Representing rotations and orientations in geometric computing. *IEEE Comput. Graph. Appl.*, 28(2):75–83, 2008.
- [LGC94] Zicheng Liu, Steven J. Gortler, and Michael F. Cohen. Hierarchical spacetime control. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 35–42, New York, NY, USA, 1994. ACM.
- [LHo6] Jun Li and Pengwei Hao. Easy foot plant. Technical Report RR-06-10, Department of Computer Science, Queen Mary, University of London, Mile End Road, London E1 4NS, UK, November 2006.
- [LHP05] C. Karen Liu, Aaron Hertzmann, and Zoran Popović. Learning physics-based motion style with nonlinear inverse optimization. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 1071–1081, New York, NY, USA, 2005. ACM.
- [LK05] Manfred Lau and James J. Kuffner. Behavior planning for character animation. In *2005 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 271–280, August 2005.

- [LKo6] Manfred Lau and James J. Kuffner. Precomputed search trees: Planning for interactive goal-driven animation. In *2006 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 299–308, September 2006.
- [LLo6] Jehee Lee and Kang Hoon Lee. Precomputing avatar behavior from human motion data. *Graph. Models*, 68(2):158–174, 2006.
- [LNSo5] Joe Laszlo, Michael Neff, and Karan Singh. Predictive feedback for interactive control of physics-based characters. *Comput. Graph. Forum*, 24(3):257–265, 2005.
- [LPo2] C. Karen Liu and Zoran Popović. Synthesis of complex dynamic character motion from simple animations. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 408–416, New York, NY, USA, 2002. ACM.
- [LS99] Jehee Lee and Sung Yong Shin. A hierarchical approach to interactive motion editing for human-like figures. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 39–48, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.
- [LWS97] Seungyong Lee, George Wolberg, and Sung Yong Shin. Scattered data interpolation with multilevel b-splines. *IEEE Transactions on Visualization and Computer Graphics*, 3(3):228–244, 1997.
- [MAA⁺09] Hironori Mitake, Kazuyuki Asano, Takafumi Aoki, Marc Salvati, Makoto Sato, and Shoichi Hasegawa. Physics-driven multi dimensional keyframe animation for artist-directable interactive character. *Comput. Graph. Forum*, 28(2):279–287, 2009.
- [MBBT00] Jean-Sebastien Monzani, Paolo Baerlocher, Ronan Boulic, and Daniel Thalmann. Using an intermediate skeleton and inverse kinematics for motion retargeting. In *Proc. Eurographics*. Blackwell Publishers, 2000.
- [McI10] Travis McIntosh. Animation and player control in uncharted: Drake's fortune and unchartedii: Among thieves. Lecture, Game Developers Conference, March 2010.
- [MFo7] Anna Majkowska and Petros Faloutsos. Flipping with physics: motion editing for acrobatics. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 35–44, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

- [MHHR07] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *J. Vis. Comun. Image Represent.*, 18(2):109–118, 2007.
- [MK05] Tomohiko Mukai and Shigeru Kuriyama. Geostatistical motion interpolation. *ACM Transactions on Graphics*, 24:1062–1070, 2005.
- [MK08] Mentar Mahmudi and Marcelo Kallmann. Fast path planning using motion graphs. In *I3D '08: Proceedings of the 2008 symposium on Interactive 3D graphics and games*, pages 1–1, New York, NY, USA, 2008. ACM.
- [Moao2] Maher Moakher. Means and averaging in the group of rotations. *SIAM J. Matrix Anal. Appl.*, 24(1):1–16, January 2002.
- [MPS06] J. McCann, N. S. Pollard, and S. Srinivasa. Physics-based motion retiming. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation, SCA '06*, pages 205–214, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.
- [MSZ94] Richard M. Murray, S. Shankar Sastry, and Li Zexiang. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1994.
- [MTP12] Igor Mordatch, Emanuel Todorov, and Zoran Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Trans. Graph.*, 31(4):43:1–43:8, July 2012.
- [MZ07] M. Matsunaga and V. B. Zordan. A dynamics-based comparison metric for motion graphs. In *Computer Graphics International (CGI)*, 2007.
- [Nata] NaturalMotion. endorphin <http://www.naturalmotion.com/endorphin.htm>.
- [Natb] NaturalMotion. euphoria <http://www.naturalmotion.com/euphoria.htm>.
- [Natc] NaturalMotion. morpheme <http://www.naturalmotion.com/morpheme.htm>.
- [ODGK03] Carol O’Sullivan, John Dingliana, Thanh Giang, and Mary K. Kaiser. Evaluating the visual fidelity of physically based animations. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 527–536, New York, NY, USA, 2003. ACM.
- [PD06] Nicolas Pronost and Georges Dumont. Validating retargeted and interpolated locomotions by dynamics-based analysis. In *GRAPHITE '06: Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pages 65–74, New York, NY, USA, 2006. ACM.

- [PR01] Nancy S. Pollard and Paul S. A. Reitsma. Animation of humanlike characters: Dynamic motion filtering with a physically plausible contact model, 2001.
- [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.
- [RGL05] Stephane Redon, Nico Galoppo, and Ming C. Lin. Adaptive dynamics of articulated bodies. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 936–945, New York, NY, USA, 2005. ACM.
- [RP03] Paul S. A. Reitsma and Nancy S. Pollard. Perceptual metrics for character animation: Sensitivity to errors in ballistic motion, 2003.
- [RP07] Paul S. A. Reitsma and Nancy S. Pollard. Evaluating motion graphs for character animation. *ACM Trans. Graph.*, 26(4):18, 2007.
- [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 Transactions on Graphics (SIGGRAPH 2005)*, 24(3):1090–1097, August 2005.
- [RPpSC01] Charles F. Rose, Iii Peter-pike, J. Sloan, and Michael F. Cohen. Artist-directed inverse-kinematics using radial basis function interpolation. *Computer Graphics Forum*, 20:239–250, 2001.
- [RZS10] Cheng Ren, Liming Zhao, and Alla Safonova. Human motion synthesis with optimization-based graphs. *Computer Graphics Forum*, 29(2):545–554, 2010.
- [SB04] P. Sardain and G. Bessonnet. Forces acting on a biped robot. center of pressure-zero moment point. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 34(5):630 – 637, sept. 2004.
- [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. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 106, New York, NY, USA, 2007. ACM.
- [Sho85] Ken Shoemake. Animating rotation with quaternion curves. *SIGGRAPH Comput. Graph.*, 19(3):245–254, 1985.
- [SHPo4] Alla Safonova, Jessica K. Hodgins, and Nancy S. Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Trans. Graph.*, 23(3):514–521, 2004.
- [SKGo3] Hyun Joon Shin, Lucas Kovar, and Michael Gleicher. Physical touch-up of human motions. In *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, page 194, Washington, DC, USA, 2003. IEEE Computer Society.
- [SLCB04] M. Salvati, B. Le Callennec, and R. Boulic. A generic method for geometric constraints detection. In *Proceedings of Eurographics04, Short presentation session*, 2004.
- [SO06] Hyun Joon Shin and Hyun Seok Oh. Fat graphs: constructing an interactive character with continuous controls. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 291–298, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.
- [SP05] Adnan Sulejmanpašić and Jovan Popović. Adaptation of performed ballistic motion. *ACM Trans. Graph.*, 24(1):165–179, 2005.
- [SPFo3] Ari Shapiro, Fred Pighin, and Petros Faloutsos. Hybrid control for interactive character animation. In *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, page 455, Washington, DC, USA, 2003. IEEE Computer Society.
- [Stao9] Jos Stam. Nucleus: Towards a unified dynamics solver for computer graphics. In *CAD/Graphics*, pages 1–11, 2009.
- [SZT⁺07] Xiaohan Shi, Kun Zhou, Yiyang Tong, Mathieu Desbrun, Hujun Bao, and Baining Guo. Mesh puppetry: cascading optimization of mesh deformation with inverse kinematics. *ACM Trans. Graph.*, 26, July 2007.
- [TGB00] Deepak Tolani, Ambarish Goswami, and Norman I. Badler. Real-time inverse kinematics techniques for anthropomorphic limbs. *Graph. Models Image Process.*, 62(5):353–388, 2000.

- [THMGo4] Matthias Teschner, Bruno Heidelberger, Matthias Muller, and Markus Gross. A versatile and robust model for geometrically complex deformable solids. In *Proceedings of the Computer Graphics International*, pages 312–319, Washington, DC, USA, 2004. IEEE Computer Society.
- [TKo5] Seyoon Tak and Hyeong-Seok Ko. A physically-based motion retargeting filter. *ACM Trans. Graph.*, 24(1):98–117, 2005.
- [TLC⁺10] Yao-Yang Tsai, Wen-Chieh Lin, Kuangyou B. Cheng, Jehee Lee, and Tong-Yee Lee. Real-time physics-based 3d biped character animation using an inverted pendulum model. *IEEE Transactions on Visualization and Computer Graphics*, 16(2):325–337, 2010.
- [TWC⁺09] Maxime Tournier, Xioma Wu, Nicolas Courty, Elise Arnaud, and Lionel Reveret. Motion compression using principal geodesics analysis. In *Eurographics*, april 2009.
- [TySKoo] Seyoon Tak, Oh young Song, and Hyeong-Seok Ko. Motion balance filtering, 2000.
- [Ume91] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(4):376–380, April 1991.
- [UPBS08] Luis Unzueta, Manuel Peinado, Ronan Boulic, and Ángel Suescun. Full-body performance animation with sequential inverse kinematics. *Graph. Models*, 70(5):87–104, 2008.
- [Vano4] Evangelos Kokkevis Vangelis. Practical physics for articulated characters. In *Proc. of GDC*, 2004.
- [vBE09] B. J. H. van Basten and A. Egges. Evaluating distance metrics for animation blending. In *FDG '09: Proceedings of the 4th International Conference on Foundations of Digital Games*, pages 199–206, New York, NY, USA, 2009. ACM.
- [VKSM05] Gokul Varadhan, Shankar Krishnan, T. V. N. Sriram, and Dinesh Manocha. A simple algorithm for complete motion planning of translating polyhedral robots. *THE INTERNATIONAL JOURNAL OF ROBOTICS RESEARCH*, 24(11):983–995, 2005.

- [WBo4] Jing Wang and Bobby Bodenheimer. Computing the duration of motion transitions: an empirical approach. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 335–344, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.
- [WBo6] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.*, 106(1):25–57, 2006.
- [Wel93] Chris Welman. “inverse kinematics and geometric constraints for articulated figure manipulation”. Master’s thesis, Simon Fraser University, 1993.
- [WJM06] Pawel Wrotek, Odest Chadwicke Jenkins, and Morgan McGuire. Dynamo: dynamic, data-driven character control with adjustable balance. In *Sandbox '06: Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames*, pages 61–70, New York, NY, USA, 2006. ACM.
- [WK88] Andrew Witkin and Michael Kass. Spacetime constraints. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 159–168, New York, NY, USA, 1988. ACM.
- [WMCG04] Xiaomao Wu, Lizhuang Ma, Zhihua Chen, and Yan Gao. A 12-dof analytic inverse kinematics solver for human motion control. *Journal of Information and Computational Science*, 1:137–141, 2004.
- [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.
- [Wri01] Stephen J. Wright. On reduced convex qp formulations of monotone lcp. *Mathematical Programming*, 90(3):458–473, 2001.
- [WTD67] George R. L. Gaughran Wilfrid Taylor Dempster. Properties of body segments based on size and weight. *American Journal of Anatomy*, 120(1):33–54, 1967.
- [Wu09] David Wu. Momentum vs. character animation. Lecture, Game Developers Conference, March 2009.
- [WXZ06] Yi Wei, Shihong Xia, and Dengming Zhu. A robust method for analyzing the physical correctness of motion capture data. In *VRST '06: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 338–341, New York, NY, USA, 2006. ACM.

- [XZY⁺07] Weiwei Xu, Kun Zhou, Yizhou Yu, Qifeng Tan, Qunsheng Peng, and Baining Guo. Gradient domain editing of deforming mesh sequences. In *ACM SIGGRAPH 2007 papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.
- [YLo8] Yuting Ye and C. Karen Liu. Animating responsive characters with dynamic constraints in near-unactuated coordinates. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers*, pages 1–5, New York, NY, USA, 2008. ACM.
- [YN03] K. Yamane and Y. Nakamura. Dynamics filter - concept and implementation of on-line motion generator for human figures. *IEEE Transactions on Robotics and Automation*, 19(3):421–432, 2003.
- [Zato2] Vladimir M. Zatsiorsky. *Kinetics of Human Motion*. Human Kinetics, 2002.
- [ZB94] Jianmin Zhao and Norman I. Badler. Inverse kinematics positioning using non-linear programming for highly articulated figures. *ACM Trans. Graph.*, 13(4):313–336, 1994.
- [ZH02] Victor Brian Zordan and Jessica K. Hodgins. Motion capture-driven simulations that hit and react. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 89–96, New York, NY, USA, 2002. ACM.
- [ZMCF05] Victor Brian Zordan, Anna Majkowska, Bill Chiu, and Matthew Fast. Dynamic response for motion capture animation. *ACM Trans. Graph.*, 24(3):697–701, 2005.
- [ZSo8] Liming Zhao and Alla Safonova. Achieving good connectivity in motion graphs. In *SCA '08: Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 127–136, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.

