Locomotion Synthesis Methods for Humanoid Characters

by

Jack Meng-Chieh Wang

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Abstract

Locomotion Synthesis Methods for Humanoid Characters

Jack Meng-Chieh Wang
Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto
2010

This thesis introduces locomotion synthesis methods for humanoid characters. Motion synthesis is an under-constrained problem that requires additional constraints beyond user inputs. Two main approaches to introducing additional constraints are physics-based and data-driven. Despite significant progress in the past 20 years, major difficulties still exist for both approaches. In general, building animation systems that are flexible to user requirements while keeping the synthesized motions plausible remain a challenging task. The methods introduced in this thesis, presented in two-parts, aim to allow animation systems to be more flexible to user demands without radically violating constraints that are important for maintaining plausibility.

In the first part of the thesis, we address an important subproblem in physics-based animation — controller synthesis for humanoid characters. We describe a method for optimizing the parameters of a physics-based controller for full-body, 3D walking. The objective function includes terms for power minimization, angular momentum minimization, and minimal head motion, among others. Together these terms produce a number of important features of natural walking, including active toe-off, near-passive knee swing, and leg extension during swing. We then extend the algorithm to optimize for robustness to uncertainty. Many unknown factors, such as external forces, control torques, and user control inputs, cannot be known in advance and must be treated as uncertain. Controller optimization entails optimizing the expected value of the objective function,

which is computed by Monte Carlo methods. We demonstrate examples with a variety of sources of uncertainty and task constraints.

The second part of this thesis deals with the data-driven approach and the problem of motion modeling. Defining suitable models for human motion data is non-trivial. Simple linear models are not expressive enough, while more complex models require setting many parameters and are difficult to learn with limited data. Using Bayesian methods, we demonstrate how the Gaussian process prior can be used to derive a kernelized version of multilinear models. The result is a locomotion model that takes advantage of training data addressed by multiple indices to improve generalization to unseen motions.

Acknowledgements

I will be forever grateful to my advisors Aaron Hertzmann and David Fleet for their guidance and support over the past six years. I learned from Aaron to be ambitious, be ready to cross field boundaries, and that "it's not research if you already know how to do it." David showed me how it is actually possible to be a great researcher, teacher, and family man all at the same time. Most importantly, the results presented in this thesis would not have been possible without both of their deep involvement and crucial contributions.

Many thanks go to the rest of my committee: Eugene Fiume, Karan Singh, and external examiner Jessica Hodgins for their feedbacks and comments to improve this document, as well as their availability to meet on short notice for numerous checkpoints.

Thanks to John Hancock for all the technical support over the years. I'm indebted to Peter O'Donovan for his tireless video production efforts for the SIGGRAPH Asia submission, and Martin de Lasa and Igor Mordatch for inspiring technical discussions. Thanks to Nikolaus Troje for providing the ground truth motion capture data used in Chapters 3 and 4, Michiel van de Panne and KangKang Yin for providing detailed information regarding SIMBICON, and Nikolaus Hansen for his publicly available CMA implementation. Early discussions between Zoran Popović and Aaron Hertzmann help inspired Chapter 4. The motion capture data used in Chapters 5 and 6 were obtained from the CMU motion capture database.

I have personally been funded by the Natural Sciences and Engineering Research Council of Canada (NSERC), in the form of a Postgraduate Scholarship and a Canada Graduate Scholarship. Additionally, this research has been supported in part by the Alfred P. Sloan Foundation, Canadian Foundation for Innovation, Canadian Institute for Advanced Research, Microsoft Research, NSERC, and the Ontario Ministry of Research and Innovation.

I want to acknowledge many more faculties, postdocs, visitors, and students I had the pleasure to meet during graduate school, including Anand Agarawala, Ravin Balakrishnan, Xiaojun Bi, Simon Breslav, Marcus Brubaker, Gerry Chu, Fanny Chevalier, Patrick Coleman, Mike Daum, Pierre Dragicevic, Irene Fung, Tovi Grossman, Sam Hasinoff,

Allan Jepson, Alex Kolliopoulos, Kyros Kutulakos, Joe Laszlo, Christian Lessig, Noah Lockwood, Shahzad Malik, Nigel Morris, Tomer Moscovich, Matthew O'Toole, Faisal Qureshi, Abhishek Ranjan, Ryan Schmidt, Leon Sigal, Patricio Simari, Eron Steger, Huixuan Tang, Khai Truong, Raquel Urtasun, Koji Yatani, and Jian Zhao. Their influences on me through lectures or countless discussions over (lunch/dinner/coffee/beer) have been immeasurable. Thanks in particular to Koji for sharing my habit of having lunch on Spadina and dinner on Bloor pretty much every day for the longest time. Abhishek, Alex, and Eron were fellow co-founders of DAG, which will always have a special place in my heart.

Thanks to my friends from undergrad (it's been almost 10 years!) who have settled down in Toronto: Vince Chan, Jimmy Chen, Rachel Lin, and Richard Lin for their lasting friendships. Thanks to Bryan Chan, who has always been a source of inspiration. Thanks also to Kevin Chang and Jonathan Lee, who have become fellow travelers in academia, for sharing their experiences and frustrations from elsewhere.

Raymond Gardener, Emidio Iacobucci, Shawn Oh, Bala Sachithananthan, and Thiru Siva have been my close friends since high school (15 years!!). I owe my adaptation to the Canadian life style and whatever success I may have had afterwards to their company.

Finally, thanks as always to my family — my sister Jenny, for always being a friend and having to share an apartment with me for many years, her husband Stéphane and his brother Jean-Nicolas. My deepest thanks go to Mom, Dad and Grandma, without whom this degree would never have been finished, for their full support and confidence in me.

Contents

Chapter 1

Introduction

The creation of character animation, or motion synthesis, is central to applications ranging from the production of feature films to interactive computer games. In particular, a long-standing goal is to create interactive characters that can walk, jump, play soccer, and perform other everyday motor skills that humans take for granted. For this to be accomplished, the creation of character animation must be as effortless and intuitive for users as possible, which places a limitation on the amount of information a good system can expect as input. For example, it is desirable to minimize the amount of keyframes required to generate a given animation. The actual range of user inputs differs depending on the specific application. In the most extreme case, autonomous characters with behavior models are expected to function without inputs. Players of interactive games give high-level commands such as "walk forward", "stop", or "turn left". On the other hand, animators working on feature productions specify highly specific constraints on the motion; such as "place the character's hand at coordinate (5.5, 3.2, 10.0) at frame 25."

A character's motion is typically described by a sequence of poses, which is a time-series of high-dimensional vectors. Even in the most specific case, user constraints are not enough to uniquely specify the motion. Though the specific requirements of an animation system vary based on the application, fundamentally, it needs to solve the under-constrained problem of mapping user requirements to full-body motion. Hence, the animation system needs to select between many possible solutions, some of which are stylistically more appealing and physically more plausible than others. The key to recovering good solutions is to design algorithms that account for our prior knowledge

about the motion being recovered.

Two promising sources of prior knowledge are from the laws of physics combined with biomechanics, and empirical observations of human motion. Correspondingly, they have been referred to as *physics-based* and *data-driven* approaches, and have been the two main paradigms of research in character animation. Physics-based animation relies on limiting the solutions to ones that, among other things, satisfy equations of motion given a biomechanically plausible simplified character model, whereas data-driven approaches typically restrict solutions to ones that are near observed motion capture (mocap) data.

The two approaches are by no means mutually exclusive. For example, everyday activities that are difficult to simulate can often be easily performed in a mocap studio, whereas the dangerous motion of a character falling down stairs can be generated through simulation. A more specific point is that, if one wants to play an animation clip without the need for interaction, and clips containing very similar poses are available in the form of motion capture data or previously created animation, then a data-driven approach is sufficient. On the other hand, if environmental interaction or other forms of generalization to unseen poses and motions are desired, then additional physics-based constraints are likely needed to ensure the plausibility of the generalizations. It is natural to argue that an effective animation system should make use of both approaches. Indeed, hybrid systems, proposing different forms of integration, have been receiving increasing attention in the research literature [18, 60, 71, 94, 112, 139]. We take the point of view that active research in both physics-based and data-driven methods of motion synthesis are important, as future systems are likely to depend on elements of both approaches.

In general, a trade-off that needs to be considered by animation systems is between flexibility and plausibility — being flexible to user requirements while keeping the synthesized motions plausible. The incorporation of physics-based and data-driven constraints is particularly important for addressing the issue of plausibility. For example, by adopting a physics-based forward simulation and control approach, the synthesized motion is guaranteed to satisfy equations of motion. Similarly, from the data-driven point of view, one can also restrict solutions to be only from available mocap data. However, despite impressive demonstrations such as autonomous creatures learning to move in physically consistent ways [28,29,76,99,117,118] and interactive systems built upon motion graphs [5,44,52,93], generating motions to satisfy specific user constraints remain difficult.

The primary issue is that, by maintaining plausibility with hard constraints, the animation system becomes highly inflexible in the motions it is able to synthesize. Indeed, it could be argued that an ideal system for motion synthesis is one that is able to satisfy any reasonable user demands with only motions that satisfy plausibility constraints. This issue manifests itself in different ways depending on the approach taken. For physics-based simulation and control, the hard constraints are in theory not overly restrictive, but motion synthesis for even mundane desired actions is difficult. For example, a controller intended for walking could easily make the character fall to the ground, or generate gaits that appear robotic and unnatural. On the other hand, while it is not too difficult to obtain mocap data of human walking, we are highly constrained by amount of available data relative to the range of possible human motion. Namely, it is difficult to modify or interact with the mocap data in a reasonable way. We introduce methods to address problems in both domains in a two-part thesis. Though the techniques we introduce in the parts are disjoint, the problems we address are all aimed to allow the animation system to be more flexible to user demands, without radically violating constraints that are important for maintaining plausibility.

The first part of this thesis deals with the physics-based approach. In particular, we will define controllers for 3D human-like characters. Controllers are mappings from the current state of the character to force outputs by virtual muscles, which are joints in our case, such that desired activities can be achieved in a forward physical simulation. This approach to motion synthesis has the potential to generate virtual characters that can react to environmental or interactive disturbances without relying on prerecorded motion data. However, controllers are extremely difficult to specify. For example, a reliable way to define robust and stylistically appealing walking controllers still remains elusive, and is the main problem addressed in this thesis.

A fundamental problem in manually designing locomotion controllers is that while there exists principles such as the zero-moment point criterion [124] and other heuristics from humanoid robotics for increasing stability and robustness, their implementation typically require high-gain, precise joint trajectory tracking that is beyond the capability of human muscles and require much higher energy [15]. As a result, the motion styles resulting from these controllers are typically robotic and unnatural. Compared to mainstream humanoid robots, people move in ways that are inherently more passive and less cautious, yet we do not easily trip or fall. On the other hand, except for highly simplified low-dimensional

models, automatic controller synthesis methods have not shown promise in 3D humanoid locomotion so far.

In Chapters 2 to 4, we provide evidence to support the main thesis that, without relying on existing motion data, an automatic optimization technique can be used to synthesize walking controllers for 3D humanoid virtual characters that are both robust to environmental disturbances and captures important features of human walking. We first demonstrate the viability of this idea by generating straight walking controllers for characters of varying body shapes and different speeds and step lengths. We then incorporate environmental and other disturbances in the optimization process to increase the robustness of the solutions to specific types of disturbances.

The second part of this thesis deals with the data-driven approach and the problem of motion modeling. Fitting a model to data is a principled method to generalize from available motion data, and has the added benefit of being able to evaluate the likelihood of new motions, as oppose to just synthesis. The latter feature makes motion modeling useful to motion analysis applications beyond computer animation. However, finding suitable models for human motion data is non-trivial. Simple models such as linear dynamical systems are not expressive enough, while more complex models require setting many parameters and are difficult to learn with limited data. In Chapters 5 and 6, we present a Bayesian approach to modeling existing motion data. We demonstrate how the Gaussian process prior can be used to derive a kernelized version of multilinear models. The result is a locomotion model that takes advantage of training data addressed by multiple indices to improve generalization to unseen motions.

1.1 Contributions

- We describe a method for optimizing the parameters of a physics-based controller for full-body, 3D walking. The objective function includes biomechanically motivated terms for power, angular momentum, and head movement minimization. These terms produce a number of important features of natural walking, such as active toe-off, near-passive knee swing, and leg extension (Chapter 3).

- Many factors such as external forces, control torques, and user control inputs cannot

be known in advance and must be treated as uncertain when designing controllers. We show that controllers can be optimized with respect to an expected return function approximated with Monte Carlo methods. The optimization under uncertainty increases robustness, produces interesting variations in style, and can be applied to build transition controllers for the problem of controller composition (Chapter 4).

- Chapters 3 and 4 together demonstrate the viability of automatic optimization for 3D walking controller synthesis from a set of simple biomechanical principles. Previous work in control optimization have dealt with much lower dimensional creatures, required reference motion data, or assumed the availability of a stable initial controller.

- We propose an extension of multilinear models using Gaussian processes. By taking advantage of data addressed by multiple indices (e.g., identity, style, gender), the proposed multifactor model improves the generalization ability of existing Gaussian process models of human motion. We demonstrate the approach using time-series prediction, and by synthesizing novel animation from the model (Chapter 6).

Contributions in the aforementioned chapters have been previously published in the ACM Transactions on Graphics (Proceedings of SIGGRAPH and SIGGRAPH Asia) [128,129], and at the International Conference on Machine Learning [126]. We refer to accompanying videos containing important illustrations of results throughout the thesis. Specifically, they can be found in the following websites:
`http://www.dgp.toronto.edu/~jmwang/optwalk/`
`http://www.dgp.toronto.edu/~jmwang/optuie/`
`http://www.dgp.toronto.edu/~jmwang/gpsc/` .

Part I

Optimizing Physics-based Walking Controllers

Chapter 2

Physics-based Motion Synthesis and Control

In this chapter, we provide background for Chapters 3 and 4. We first give an example of how motions satisfying the equations of motion can be formulated and solved as an initial value problem and how controllers can influence the solutions. Related work on methods for controller synthesis, and automatic methods of trajectory optimization are then reviewed. Finally, we describe covariance matrix adaptation (CMA) [31], the derivative-free optimization algorithm used extensively in subsequent chapters.

2.1 Forward simulation and control

A character's pose can be described by a high-dimensional vector, and a sequence of such vectors describes the character's motion. It is not difficult to see that out of all possible motions, only a very small portion is physically plausible. Therefore, simply restricting the motion to satisfy the laws of physics removes a significant portion of unlikely motions. Additional hard/soft constraints can be placed on the forces used to drive the simulation, either in the form of a limited control parameterization or a preference for small forces. We model a humanoid character as a collection of rigid bodies connected by joints. The motion satisfies physical constraints if the trajectories of all of the rigid bodies follow Newton's equations of motion, subject to contact and joint constraint forces.

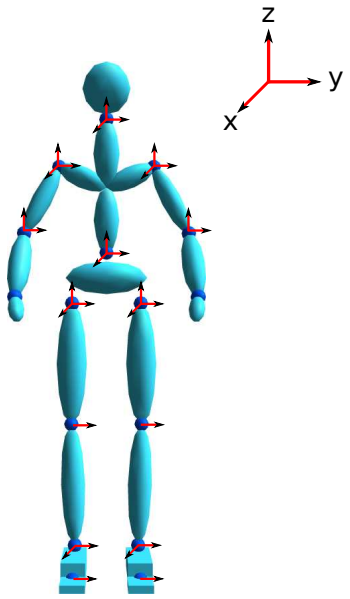


Figure 2.1: The humanoid model and joint degrees of freedom (DOF). The values and derivatives of 30 joint DOFs plus six global DOFs form a 72 dimensional state vector.

Let $\mathbf{y}$ denote the state vector describing the position and velocities of rigid bodies in the simulation at a given instant. The equations of motion can be written as a first order ordinary differential equation in the form of

$$\dot{\mathbf{y}} = f(\mathbf{y}, t) \ . \tag{2.1}$$

Given $\mathbf{y}_0$, the simulation state at $t = 0$, finding future values of $\mathbf{y}$ is a standard initial value problem, and can be solved numerically. For example, Euler's method entails recursively computing

$$\mathbf{y}(t + h) = \mathbf{y}(t) + hf(\mathbf{y}(t), t) \ , \tag{2.2}$$

where h is the simulation step size, and $\mathbf{y}(t)$ is the state at time t. The resulting state trajectory, corresponding to a character's motion, satisfies physical constraints up to the accuracy of the numerical solution.

The evolution of the state over time is dictated by f, as shown by (2.1). To see how control forces influence f, we will consider a simple example. For a single rigid body, the state vector $\mathbf{y}$ must describe both its position and orientation, and the translational and

angular velocities. One standard choice is to let

$$\mathbf{y} = \begin{pmatrix} \mathbf{x} \\ \mathbf{R} \\ \mathbf{p} \\ \mathbf{L} \end{pmatrix} , \tag{2.3}$$

where $\mathbf{x} \in \mathbb{R}^3$ is the position of the body's center of mass, $\mathbf{R}$ is a 3 by 3 rotation matrix[1] about $\mathbf{x}$. $\mathbf{p} \in \mathbb{R}^3$ is the body's linear momentum, and $\mathbf{L} \in \mathbb{R}^3$ is the angular momentum.

The momenta variables encode the velocity information of the rigid body. In order to compute them, we need to specify two additional quantities: the total mass of the body M and the object space inertia tensor $\mathbf{I}_{obj}$. These quantities relate the linear and angular momenta of the body to its velocities, and remain constant throughout the simulation:

$$\mathbf{p} = M\mathbf{v} \tag{2.4}$$

$$\mathbf{L} = \mathbf{I}\boldsymbol{\omega} , \tag{2.5}$$

where $\mathbf{v}$ is the velocity of the center of mass, $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^T$ is the angular velocity. The direction of $\boldsymbol{\omega}$ indicates the axis of rotation, while its magnitude gives the angular speed. The body's angular velocity and its angular momentum are related by the inertia tensor, defined as

$$\mathbf{I} = \int_x \int_y \int_z \rho(x, y, z) \begin{pmatrix} y^2 + z^2 & -xy & -xz \\ -yx & x^2 + z^2 & -yz \\ -zx & -zy & x^2 + y^2 \end{pmatrix} dxdydz , \tag{2.6}$$

where $\rho(x, y, z)$ is the mass density of the object at a point (x, y, z), expressed relative to $\mathbf{x}$, and δ_{ij} is the Kronecker delta function. The specific form of $\mathbf{I}$ depends on the choice of coordinate frame, which changes as the body rotates during the simulation. In fact, if we define $\mathbf{I}_{obj}$ as what is given by (2.6) at the start of the simulation, it can be shown that $\mathbf{I} = \mathbf{R}\mathbf{I}_{obj}\mathbf{R}^T$, which means the inertia tensors during simulation can always be computed from the initial object space inertia tensor.

In particular, since $\mathbf{I}$ is real and symmetric, there exists a coordinate frame such that $\mathbf{I}$

[1] We abuse the notation and assume $\mathbf{R}$ is flattened to a column vector when placed in $\mathbf{y}$.

is diagonal. For example, a solid cylinder centered at the origin, aligned with the z-axis with radius r, height h, and mass m has inertia tensor:

$$\mathbf{I} = \begin{pmatrix} \frac{1}{12}m(3r^2+h^2) & 0 & 0 \\ 0 & \frac{1}{12}m(3r^2+h^2) & 0 \\ 0 & 0 & \frac{1}{2}mr^2 \end{pmatrix}. \tag{2.7}$$

The diagonal elements are called moments of inertia, and are rotational analogs of mass about the x, y, and z-axis, respectively. Notice that increasing the height of the cylinder does not change the cylinder's tendency to rotate about the z-axis. In practice, the object space coordinate frame is often chosen such that the inertia tensor $\mathbf{I}_{obj}$ is diagonal.

To form the differential equation, the time derivatives of the state variables need to be computed. The center of mass velocity can simply be computed from the linear momentum $\dot{\mathbf{x}} = \mathbf{v} = \frac{\mathbf{p}}{M}$. The angular velocity can be computed from angular momentum $\boldsymbol{\omega} = \mathbf{I}^{-1}\mathbf{L}$, and can then be used to compute the derivative of the orientation matrix. Specifically, let

$$\boldsymbol{\Omega} = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}, \tag{2.8}$$

then it can be shown that $\dot{\mathbf{R}} = \boldsymbol{\Omega}\mathbf{R}$. Finally, derivatives of linear and angular momenta are the external force ($\mathbf{f}$) and torque ($\boldsymbol{\tau}$) applied to the center of mass, as can be derived from the equations of motion. The differential equation (2.1) can be written as

$$\dot{\mathbf{y}} = \begin{pmatrix} \mathbf{v} \\ \boldsymbol{\Omega}\mathbf{R} \\ \mathbf{f} \\ \boldsymbol{\tau} \end{pmatrix}. \tag{2.9}$$

It should be clear that other than changing the initial conditions, the only way to influence the resulting motion (while keeping the rest of the simulation parameters fixed) is through changing $\hat{\mathbf{f}} = [\mathbf{f}, \boldsymbol{\tau}]$ over time, which we can further decompose as $\hat{\mathbf{f}} = \hat{\mathbf{f}}_e + \hat{\mathbf{f}}_i$. The $\hat{\mathbf{f}}_e$ component of force results from environmental interactions such as gravity, ground contact, and forces due to other constraints. Setting $\hat{\mathbf{f}} = \hat{\mathbf{f}}_e$ is sufficient if we are simulating a purely passive scenario, such as a brick falling under gravity. On the other hand, if the

brick has a motor attached to it, then finding a $\hat{\mathbf{f}}_i$ trajectory (subject to the limitations of the motor) such that the brick reaches a particular location is an example of a control problem. A controller is an algorithm that determines the motor outputs over time, which in turn is mapped to $\hat{\mathbf{f}}_i$, and is the only way to influence the rigid body's motion during the simulation.

The rigid body example serves as an illustration of how forces influence the solution to the initial value problem, which is in part based on Witkin and Baraff's course notes [130]. We refer the interested readers to these notes for a more extensive and general tutorial on physics-based modeling. The problem of simulating and controlling a humanoid character is more complex, but similar in spirit. Since the character is modeled as rigid bodies connected by joints, the equations of motion need to also ensure the connections do not break apart during the simulation. Conceptually, this can be done by expanding (2.9) to include all the body parts and apply the correct $\hat{\mathbf{f}}_e$'s to maintain the joint constraints. In practice, simultaneously solving for all correct constraint forces is non-trivial, and a re-derivation of the equations of motion based on for example the *principle of virtual work* is necessary [25].

Alternatively, the state vector of the system can be re-parameterized in terms of *generalized coordinates*, which implicitly define the constraints. For example, the character can be parameterized in terms of its joint angles instead of the position and orientation of each of the parts. Methods of Lagrangian mechanics [108] can be used to derive the equations of motion in this case. The use of generalized coordinates is mathematically more elegant and guarantees that constraints are satisfied exactly, but a new set of equations must be derived whenever the kinematic structure of the character changes.

Regardless of the simulation methods, a controller for a typical humanoid character is only allowed to generate torques at the joints as an abstraction to more realistic muscle models.[2] Since arbitrary forces can not be applied directly to any body part, the system is *underactuated*. In particular, the global position and orientation of the character can only be controlled indirectly. For example, taking a step forward requires the correct joint torques to generate the correct ground reaction forces that moves the torso forward.

It should be clear that, unlike data-driven animation methods based on motion capture

[2] Allowing for forces to be directly applied to the torso would be akin to letting the character carry a jetpack.

(mocap) data, where it is challenging to generalize to new motions, the main difficulty with physical constraints is to control the generated motions. Fundamentally, determining effective control algorithms to satisfy user constraints, often defined in pose space, is extremely challenging for human motion.

2.2 Controller synthesis

Animation researchers have tried to design controllers for specific activities such as locomotion for virtual characters [87], human athletics [39], and swimming [134]. Somewhat surprisingly, the most basic form of human locomotion – walking, has proven to be one of the most challenging control algorithms to design. Full-body walking controllers in graphics have been mostly based on hand-tuned state machine models [22, 48, 137]. Perhaps the most important such model is the recent SIMBICON controller by Yin et al. [137]. While remarkably robust, and capable of producing several styles, it produces motions that differ in several key ways from human walking. For example, it relies significantly on the hip to generate forward momentum, while normal human walking relies heavily on the ankle, with a near-passive knee [78]. As a consequence, SIMBICON produces a marching-like gait.

While most efforts in controller synthesis have been devoted to basic motor skills (e.g., walking, running, reaching, etc.), complex motions can be executed if the basic skills can be composed and transitioned to each other in a sensible way. Impressive complex animations can be generated by constructing a graph structure connecting basic hand tuned controllers [22, 137]. Successful transitioning is ensured by requiring basic controllers to specific pre and post conditions on the state space. Coros et al. [16] learn optimal control policies to transition between low-level controllers to accomplish different tasks. These methods assume that the low-level controllers, and hence the motion style, are specified in advance and fixed. Our approach in the following chapters complements these, as we focus on learning low-level controllers, including those capable of making transitions.

Controller synthesis is a fundamental problem in robotics. While the real world introduces additional complexities, many of the design goals for control algorithms are shared by animation researchers. The control strategy should be robust, reliable, and ideally look natural. Indeed, Raibert and Hodgins used their work on legged robots [38, 86] as

a starting point, in some of the most important early work on applying controllers to animation [87]. Burridge et al. [11] demonstrated a funneling approach for controller composition on a paddle robot. The recently developed humanoid robots such as Honda's ASIMO [36], as well as robots at Waseda University [79] and KAIST [40], have comparable physical complexity and degrees of freedom to virtual humanoids of interest in animation. They require sophisticated control algorithms that could potentially be useful for animation as well. However, the walking gaits produced by these humanoid robots to-date are still highly distinct from humans. One glaring difference is the constant bending of the knee during walking, which is still being addressed by ongoing research [41, 79].

It has been observed in the early robotics literature that simple bipeds without motors, called passive dynamic walkers, can be designed to walk downhill indefinitely [68]. Controllers based on passive dynamics have been used to predict observed properties of human walking [46], and building powered bipedal robots that can walk on level ground [15]. They have also been used as basis for 3D human tracking [10]. These models share an energy consumption pattern similar to human beings [15], but are highly specific to walking. This is in contrast with the mainstream humanoid robots mentioned previously, where the control is largely based on precise tracking of joint trajectories generated to say, satisfy the zero-moment point criterion [124], which require much more energy. Incorporating elements of passive dynamics into controlling more complex motions, passive or not, remains an interesting open problem.

The design of controllers is time consuming, and is usually limited to highly specific tasks. Therefore the potential of using learning techniques to automatically generate controllers is extremely attractive. Reinforcement learning problems deal with decision making in an uncertain environment [104]. In this context, the character being simulated is an agent operating in a physical environment, attempting to make decisions that will maximize a scalar reward function. The reward function could correspond to for example, a function of distance moved in and energy consumed. The decisions are related to the amount of torque applied to the joints at each timestep (i.e., the controller). We demonstrate an application of reinforcement learning to synthesize walking controllers with respect to an uncertain environment in Chapter 4. In particular, we optimize the parameters of the controller with respect to the expected reward function.

The original control optimization methods in animation were applied to low-dimensional

virtual characters [76, 99, 117, 118], where the search space is much smaller than full-body locomotion. Grzeszczuk et al. [28, 29] demonstrate control optimization for high-dimensional marine animals, where smooth dynamics without ground contacts lead to smoother objective functions than for ground-based locomotion. Sharon and van de Panne [96] optimize planar humanoid characters to match the style of reference motion, but their results suggest that this approach has advantages and disadvantages similar to Yin et al. [137].

For 3D humanoid models, Hodgins and Pollard [37] adapt stable controllers from one character to different characters by searching over a small number of high-level control parameters. More recently, optimization-based methods are used to adapt walking controllers for more difficult tasks [136], or to create solutions for constrained terrain navigation [17]. In both cases, a hand-tuned robust walker is available as initialization, only a relatively small subset of task-relevant parameters are optimized, and the basic style of motion is not modified from the initial controller. In Chapters 3 and 4, we show how to significantly improve walking style and robustness from a manually initialized rough, unstable controller.

All previous work in controller optimization for character animation assumes completely deterministic dynamics. As robotic controllers must account for uncertain conditions, optimization under uncertainty has been used for several low-dimensional robotic controllers. For example, Tedrake et al. [106] optimize the control of a low-dimensional passive-based humanoid walker. This walker is designed to be very stable even in the absence of control, thereby reducing greatly the role of uncertainty. Abbeel et al. [1] demonstrate impressive controllers for helicopter stunts, another system with low-dimensional dynamics but no ground-contact discontinuities. Peters and Schaal [82] apply policy gradient methods to train a robot arm (holding a bat) to hit a stationary baseball. Morimoto and Atkeson [70] improve the stepping and walking performances of a humanoid robot with reinforcement learning.

Another approach is to employ mocap data to define the desired controller [18, 71, 101, 112]. This frees the designer from specifying the style of the motion, only requiring the maintenance of balance. The use of mocap data allows for high-quality human-like results to be simulated, as recently demonstrated by Muico et al. [71] on an impressive array of locomotion skills. However, such methods are limited in their ability to generalize to

situations where no mocap data are available. Furthermore, since the resulting motion style is defined by the mocap data, such techniques cannot be used to investigate effects of the environment or body mass distribution on style.

2.3 Trajectory optimization

Trajectory optimization aims to optimize some measure of energy or error from user constraints with respect to a trajectory (of pose, torque, or muscle excitation) over time. These techniques can often be viewed as a form of inverse dynamics, and are of interest to researchers in both animation and biomechanics. While animation researchers employ relatively simplified human models (i.e., model muscles using joint actuators), researchers in biomechanics build models to analyze individual muscle usage under hypothesized objective functions. Perhaps most well-known to animation researchers, Anderson and Pandy [4] recover muscle excitation trajectories during a walk cycle by minimizing metabolic energy over distance traveled.

The main difference between trajectory optimization and optimization approaches discussed in the previous section is that optimal control in our sense aims to recover controllers that work beyond the duration for which they were optimized, possibly subject to minor environmental disturbances. In contrast, the recovered forces from trajectory optimization are not generally robust to any change in environmental interactions if used to control new simulations.

Another difference is that approaches discussed in the previous section enforce physical constraints through forward simulation, and the only way to control the character is by adjusting muscle/joint outputs. This makes satisfying specific user constraints (e.g., keyframes) extremely difficult. The *spacetime constraints* formulation [131] minimizes physical quantities such as power and torque, and can be formulated to make the satisfaction of user constraints straightforward. Instead of aiming to recovering torque trajectories, which ensures the equations of motion are followed; the motion itself is recovered, which ensures user constraints are satisfied. Impressive results were achieved on a simple Luxo lamp model, and an interactive animation system with similar model complexity has been built [14].

A large part of subsequent efforts have centered on extending the technique to complex articulated figures: by using a different state space [63], focusing on high-energy motions [62], or identifying computationally efficient objective functions [23]. Another route has been to employ trajectory optimization in the simpler, but important problem of motion editing. Given mocap clips, physics can be used to transition between clips [90], or to modify constraints such as footsteps [83].

By directly controlling the motion trajectory, specific user constraints can be met in the spacetime constraints framework. However, the optimization process might not converge to a physically realistic motion in practice. Moreover, most demonstrations of spacetime constraints are on high-energy motions, where power or torque minimization could lead to good solutions. For low-energy motions such as walking, the optimization problem is more difficult. The reason is that while high-energy motions are largely constrained by the combination of physics and animator constraints, low-energy motions contain more stylistic elements. For example, there are many physically valid ways to walk through a series of footstep constraints.

One way of addressing the problem of applying spacetime constraints to low-energy motions is to employ more complex objective functions that contain local optima at the desired style. However, it is unclear how one could approach specifying such objectives. Liu et al.'s [60] work on inverse optimization from motion capture data begins to address this issue, but the learning process is expensive, and requires more detailed physical modeling of characters. Another problem within the spacetime constraints framework that is beginning to receive attention is the interaction of multiple characters [61]. More recently, Wampler and Popović [125] combine traditional spacetime constraints with CMA [31], to generate plausible locomotion gaits for a variety of virtual creatures. CMA is also used in this thesis, and is described in the next section.

2.4 Covariance matrix adaptation

As discussed above, we take an optimization approach to controller synthesis. We represent the control algorithm with a number of real-valued parameters, and then optimize these parameters with respect to an objective function. Naturally, selecting a suitable optimization algorithm is crucial to solving the problem. Most numerical optimization

algorithms rely on gradient information to iteratively approach a local minimum. Specific algorithms [77] range from steepest descent methods, which directly follow the gradient, to Newton's method, which requires the Hessian. In between the two extremes are quasi-Newton methods such as L-BFGS [138], which approximates the inverse Hessian using gradient information.

Since evaluating our objective function requires running a forward simulation involving collision detection, joint limits, and other non-smooth constraints, we cannot compute the gradient of the objective analytically. Finite difference methods can be used to approximate the gradient, but require at least as many evaluations of the objective function as the number of variables being optimized, which is intolerable given the dimensionality of our problem. Moreover, most gradient-based methods are only concerned with approaching a nearby local minimum quickly, often leading to unacceptable solutions for non-smooth objective functions with many local minima.

There are well-known optimization algorithms that are *derivative-free*, such as the Nelder-Mead downhill simplex method [74], used to implement the `fminsearch` function in MATLAB, and simulated annealing [42], which has seen a number of applications in graphics [2, 28, 133]. In this thesis, we adopt CMA [31], a more recently developed method from the evolutionary computation literature. The algorithm is selected both due to its empirical effectiveness, and that it is easily parallelizable.

Like gradient-based methods, CMA is an iterative algorithm that attempts to improve the objective function based on its local geometry. However, instead of iteratively improving a single candidate solution, it estimates a Gaussian distribution over the parameter space, so that samples from each Gaussian tend to have better objective function values as the number of iterations increases. These samples are also used to "improve" the Gaussian distribution for the next iteration. In particular, consider M samples drawn from the Gaussian at the i-th iteration:

$$\mathbf{x}_k^{(i+1)} \sim \mathcal{N}\left(\mathbf{m}^{(i)}, \mathbf{C}^{(i)}\right), \tag{2.10}$$

where $k = 1 \ldots M$. The core of the CMA algorithm is in defining the Gaussian distribution for the next iteration by specifying $\mathbf{m}^{(i+1)}, \mathbf{C}^{(i+1)}$ from the samples $\mathbf{x}_k^{(i+1)}$. In our problem, each sample represents running a forward simulation of the type described in

Section 2.1, and sample evaluation is the bottleneck of the algorithm. Fortunately, CMA does not require a large number of samples to be effective. Indeed, we will see below that the use of a small cluster of computers to evaluate the samples in parallel is critical to arriving at a solution to our problem within a reasonable time.

We will denote $\mathbf{y}_1 \dots \mathbf{y}_N$ as a permutation of $\mathbf{x}_1^{(i+1)} \dots \mathbf{x}_N^{(i+1)}$, such that $E(\mathbf{y}_1) < \dots < E(\mathbf{y}_N)$ and E is the objective function being minimized. The update rule for the mean is straightforward:

$$\mathbf{m}^{(i+1)} = \frac{1}{M} \sum_{j=1}^{M} w_j \mathbf{y}_j \ , \tag{2.11}$$

where $1 \leq M \leq N$, $w_1 \geq w_2 \geq \dots \geq w_M > 0, \sum_{j=1}^{M} w_j = 1$. In other words, the mean of the next Gaussian is a weighted average of the M best samples from the current Gaussian. Typically, M is set to be roughly half of N, and w_j is set to be proportional to $\log\left(\frac{M}{j}\right)$.

More care must be taken to update the covariance matrix. A natural choice is the sample covariance of $\mathbf{y}_1 \dots \mathbf{y}_M$, i.e.,

$$\mathbf{C}^{(i+1)} = \frac{1}{M} \sum_{j=1}^{M} \left(\mathbf{y}_j - \mathbf{m}^{(i+1)}\right) \left(\mathbf{y}_j - \mathbf{m}^{(i+1)}\right)^T . \tag{2.12}$$

However, in typical settings (i.e., Figure 2.2), these samples only cover a local neighbourhood that does not include the minimum. Furthermore, the better samples tend to occupy a much smaller neighbourhood than the full set of samples. Consequently, (2.12) tends to decrease the variance of the distribution as the algorithm progresses. The overly aggressive, often exponential, shrinking of the distribution leads to premature convergence (i.e., the variance of the Gaussian becomes too small to make progress, before reaching a local minimum).

A key observation in the CMA algorithm is to replace (2.12) with the following:

$$\mathbf{C}^{(i+1)} = \frac{1}{M} \sum_{j=1}^{M} \left(\mathbf{y}_j - \mathbf{m}^{(i)}\right) \left(\mathbf{y}_j - \mathbf{m}^{(i)}\right)^T , \tag{2.13}$$

where $\mathbf{m}^{(i)}$ is the mean of the Gaussian in the current iteration. We can see that the difference between (2.13) and (2.12) is minimized when the mean of the best M samples

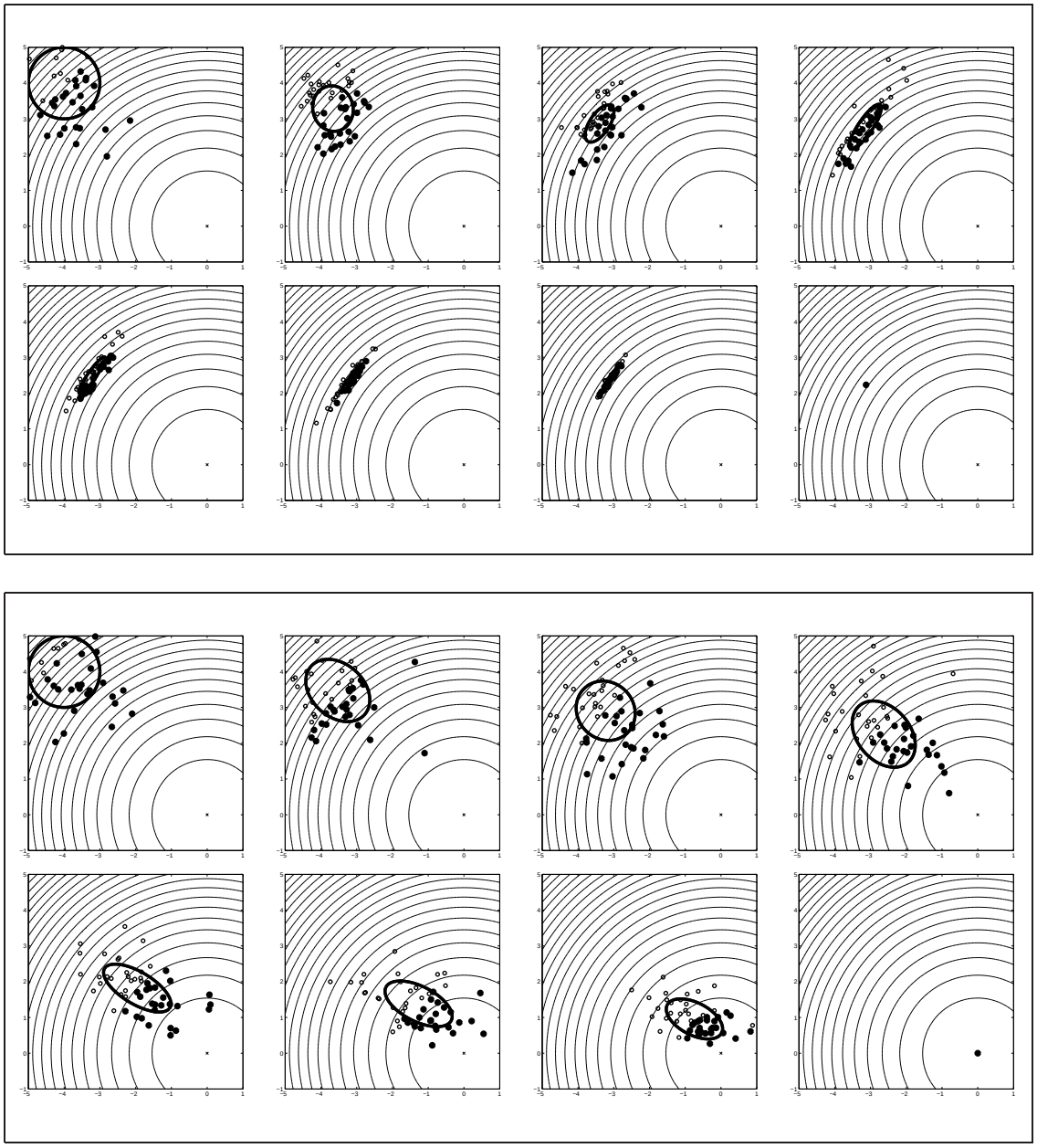


Figure 2.2: Optimizing the objective function $E = -\mathbf{x}^T\mathbf{x}$. The ellipse represent one standard deviation of the Gaussian, solid points are the M "better" samples used to improve the distribution. The true minimum at $(0, 0)$ is labeled by the cross. *Top:* The first seven iterations (left to right, top to bottom) and the final solution when the covariance matrix is updated by (2.12). *Bottom:* The same information for the CMA strategy (2.13). Note that the naive update strategy (top) quickly converges to an incorrect solution due to the fast decrease in variance. On the other hand, CMA (bottom) increases variance in the gradient direction and locates the correct solution.

stops moving between successive iterations, which should happen upon convergence. In particular, $\mathbf{m}^{(i+1)} - \mathbf{m}^{(i)}$ is analogous to the gradient of the objective function. Therefore during a typical iteration, (2.13) tends to stretch the Gaussian in the general gradient direction, which allows more samples to be generated to explore that direction. This is not the case for (2.12), in fact, it often decreases the variance of the Gaussian along the gradient direction.

Figure 2.2 demonstrate the difference between the two update rules on a simple 2D quadratic function ($E = -\mathbf{x}^T\mathbf{x}$). We see that following (2.12) quickly converges to an incorrect solution due to the fast decrease in variance. On the other hand, (2.13) makes progress in the gradient direction and converges to the correct solution.

We have focused on giving a high-level intuition for CMA in this section and omitted many details. In particular, M is typically too small to give a reliable estimate of the covariance matrix, and matrices from previous iterations are used to aid in the estimation. Moreover, a separate scalar parameter, which scales $\mathbf{C}^{(i+1)}$ is also adapted to explicitly control the "stepsize" of the optimization algorithm. For details of the exact update rules, as well as how various internal parameters are chosen, we refer interested readers to Hansen's tutorial [32].

Chapter 3

Optimizing for a Human-like Gait

This chapter describes a method for optimizing the parameters of a physics-based controller for full-body, 3D walking. A modified version of the SIMBICON controller [137] is optimized for characters of varying body shape, walking speed and step length. The objective function includes terms for power minimization, angular momentum minimization, and minimal head motion, among others. Together these terms produce a number of important features of natural walking, including active toe-off, near-passive knee swing, and leg extension during swing. We explain the specific form of our objective criteria, and show the importance of each term to walking style. We demonstrate optimized controllers for walking with different speeds, variation in body shape, and in ground slope.

3.1 Introduction

Locomotion is an essential skill for simulated characters, but one for which designing controllers is particularly difficult. The control space is high-dimensional, the dynamics are nonlinear and discontinuous due to contact, and infeasible controllers (i.e., that trip and fall) are all too common. While robust walking controllers have recently been designed [40, 137], they produce gaits that appear unnatural. Indeed, it remains extremely challenging for either humans or computers to define robust controllers for natural-looking walking.

This chapter introduces a parameter optimization procedure for full-body, 3D walking

controller synthesis. Controllers may be optimized for characters of varying body shape, and, optionally, to achieve user-specified walking speeds and/or step lengths. The resulting gaits exhibit key properties of natural walking, including, for example, energy efficiency, a strong *toe-off* as support is transferred from one foot to the other, a nearly passive knee during leg swing, leg extension prior to *heel-strike*, torso lean, and the anti-symmetric phase of arm swing. We advance the state-of-the-art by demonstrating results that both capture these important features and are robust enough to tolerate minor environmental disturbances. Unlike many previous methods, the system does not require any mocap data or reference trajectories.

The control parameterization is a version of SIMBICON [137], with modifications to allow more realistic motion. The objective function is a weighted sum of several terms, many of which are inspired by biomechanical properties of human walking. We demonstrate how each of these terms contributes to creating human-like qualities of motion. The optimization is initialized with a walker that is unstable, but roughly captures observed features of human walking. Because the approach does not rely on mocap data, there is no restriction that the styles of walking follow any particular mocap database.

3.2 Character model and controllers

The character model has 30 internal degrees-of-freedom (DOFs), seen in Figure 2.1, 28 of which are identical to Yin et al. [137]. We add toe blocks, which are connected to the feet by hinge joints. Toes provide more flexibility during landing and ankle push-off (also called toe-off). The link locations and mass distributions are based on Wooten and Hodgins [132]. We scale the links and masses according to the dimensions of the character skeleton, and apply reasonable joint limits.

Single-state controller. As in SIMBICON, the walking controller is a finite-state machine. Each state contains proportional-derivative (PD) controllers for each joint and a balance feedback controller. The controller for each joint DOF includes gain and damping coefficients (k_p, k_d), as well as a target angle (θ_d). At each simulation time-step,

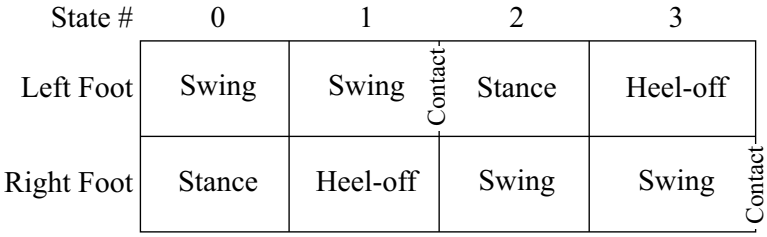


Figure 3.1: The state machine for the walking controller consists of four states, corresponding to ground contact and swing phases.

a torque (τ) for each joint DOF is generated according to

$$\tau = k_p\left(\theta_d - \theta\right) - k_d\dot{\theta}\ , \tag{3.1}$$

where θ and $\dot{\theta}$ are the current joint angle and angular velocity.

The input angles (θ) to the PD controllers are expressed in local coordinate systems, except for the hip and ankle. The hip angles operate in the world frame and are mapped to control the torso orientation, adjusted by balance feedback parameters (c_d, c_v) [137]. Since the stance hip_z DOF serves to rotate the body towards the desired facing angle, we only enable it when the stance foot is firmly planted (more than three points in contact on the foot or toe), otherwise it is servoed to zero. The world frame orientation of a body part is defined by measuring angles of their up and forward vectors projected in the coronal, sagittal, and transverse planes. Unlike SIMBICON, our controllers servo the ankles in the world frame, since world orientation is crucial for ensuring ground clearance after toe-off, as well as for landing at a good angle to allow weight transfer.

We couple the target angle of the arm DOF in the sagittal plane to the hip angles in the sagittal plane, so that the target angle is

$$\theta_{larm} = \alpha_{arm}(\theta_{rhip} - \theta_{lhip}) \tag{3.2}$$

$$\theta_{rarm} = -\theta_{larm}\ , \tag{3.3}$$

where α_{arm} is a scale factor. This allows the model to synchronize arm swing with the legs, but good arm swing still depends on a good selection of α_{arm} and arm spring-damper constants.

State machine and transitions. The finite state machine contains four states, corresponding to basic phases of walking (Figure 3.1). State 0 begins at foot strike, and

Start State		
DOF	q_0	$\dot{q}_0$
$globpos_x$	free	1.3
$lhip_y$	-0.4	0.3
$lknee_y$	1.35	0.1
$lankle_y$	0.35	-15
$rhip_y$	-0.4	1.0
$rknee_y$	0.6	2.0
$rankle_y$	-0.2	-9

Controller				
State	0		1	
DOF	k_p	θ_d	k_p	θ_d
$neck_{xyz}$	100	0	100	0
$larm_{xz}$	30	0	30	0
$larm_y$	30	n/a	30	n/a
$lelblow_{yz}$	30	0	30	0
$rarm_{xz}$	30	0	30	0
$rarm_y$	30	n/a	30	n/a
$relblow_{yz}$	30	0	30	0
$back_{xyz}$	300	0	300	0
$rhip_{xz}$	1000	0	1000	0
$rhip_y$	1000	-1	1000	-0.65
$lknee_y$	300	-1.3	50	-0.55
$lankle_x$	30	0	50	0
$lankle_y$	300	3	300	-0.35
$torso_{xyz}$	1000	0	1000	0
$rknee_y$	150	0.4	500	-0.5
$rankle_x$	100	0	100	0
$rankle_y$	300	-0.2	300	0.75
$ltoe_y$	20	0	20	0
$rtoe_y$	20	0	20	0.6

Table 3.1: *Left:* Initialization of position/angle, velocity in the start state. Unspecified DOFs are initialized to zero. *Right:* Initialization of PD control parameters. Dampers k_d are initialized to $0.1k_p$ in all cases.

continues as the swing leg lifts off and swings forward. The transition to State 1 occurs when the signed horizontal distance (in the sagittal plane) between the center-of-mass (COM) of the body and the ankle of the stance foot exceeds a threshold c_{trans}. This is motivated by our observations of when stance ankle push-off appears to occur. Notably, this differs from SIMBICON, which uses a time-based transition.

During State 1, the swing leg prepares for landing, and the stance ankle push-off begins. The transition to State 2 occurs when the swing foot makes contact with the ground, provided that the swing ankle global orientation (in the sagittal plane) exceeds 0.1 radians. Without this condition, the controller must lift the swing leg to an artificial height to ensure ground clearance. State 0 may also transition directly to State 2 if contact occurs (although this usually indicates a poor controller). States 2 and 3 are left/right reflected versions of States 0 and 1, with mirrored parameters.

Start state. The start state of the physical simulation is specified by six global DOFs, 30 joint DOFs, and their generalized velocities (72 DOFs in total). The start state is optimized along with the controller, and is manually initialized (see Table 3.1) to when the left leg is in the middle of its swing phase, prior to the transition from State 0 to State 1.

In principle, the start state and the controller parameters total over 200 DOFs in the system. However, since we focus on straight walking in the positive x direction in this chapter, we fix DOFs and other parameters that are unlikely to contribute to the task. More specifically, joint DOFs (including PD control target angles) that rotate with respect to the local x and z axis are fixed to zero, to discourage unnecessary motion in the coronal and transverse planes. The back joint is an exception, where rotation with respect to z-axis is free to be optimized, which is necessary for trunk rotation in the gait. The target angles for the toe joints are fixed to zero except for the stance foot during States 1 and 3. For the global DOFs in the start state, x and y positions can naturally be set to any value. Since the controller would not be walking straight if it deviates away from the y-z plane, we fix the start state global rotation (and angular velocity) with respect to x and z axis, as well as velocity in the y-direction to zero. With these constraints, the start state and the control parameters comprise a 184 dimensional search space.

3.3 Controller optimization

Optimizing a controller involves searching for control parameters and a start state that together produce good character simulations. The quality of the generated motion is measured with an objective function that evaluates simulations of duration 10 seconds (T simulation time-steps). The duration is selected to balance between computational costs and the need to ensure a basic level of robustness. For example, if we set the duration to only two seconds, the resulting controllers often fail shortly beyond the two-second mark even without external disturbances. The objective function comprises a weighted combination of terms motivated by task constraints and biomechanical features of human walking.

3.3.1 Objective function

An obvious way to define an objective function is with a weighted sum of quadratic penalty terms on quantities such as total power consumption and deviation from a target speed. In practice, finding suitable weights for such terms is extremely difficult. For example, if chosen poorly, then even when the target speed is nearly achieved, optimization might continue to ignore the energy term in favor of imperceptible speed refinements. Instead, we employ a weighted combination of objectives that do not penalize small differences from targets. Specifically, we define a thresholded quadratic as

$$Q(d;\epsilon) = \begin{cases} d^2, & \text{if } |d| > \epsilon \\ 0, & \text{otherwise .} \end{cases} \tag{3.4}$$

This objective assigns a penalty of zero when the distance d is below a threshold ϵ, but applies steep penalties beyond this threshold. This formulation is akin to specifying hard constraints on the motion to optimize power and stability related terms, subject to constraints that speed and other quantities fall within ϵ of target values. Unlike hard constraints, however, including the soft penalty component allows us to avoid the difficult problem of finding a feasible initialization (and ensuring that it exists).

User gait constraints. A user may specify high-level requirements on the average forward speed (v_x) or step length (s) of the simulation. This is done by penalizing the differences between these values and user-specified targets ($\hat{v}_x, \hat{s}$, respectively):

$$E_{user} = Q(v_x - \hat{v}_x; \epsilon_{vel}) + Q(s - \hat{s}; \epsilon_{step}) \ . \tag{3.5}$$

Both of these terms are optional.

Required gait constraints. There are several properties of gait that we find essential to the style and stability of the simulated walking motions. First, because we optimize for walking in the positive x direction, significant deviations in the y and z directions of motion are undesirable. Accordingly, they are penalized with the following objective:

$$E_{vel} = Q(v_y; \epsilon_{vel}) + \lambda_{vel} \left[Q(v_{0x} - v_x; \epsilon_{vel}) + Q(v_z; 2\epsilon_{vel}) \right] \ , \tag{3.6}$$

where v_y and v_z are the average simulation velocities (per second) of the COM in the y and z directions, respectively, and v_{0x} is the velocity of the start state in the $+x$ direction. This constraint encourages v_y and v_z to be small, and the start velocity v_{0x} to be similar to the average x velocity of the simulated motion.[1] The constant λ_{vel} reflects our preference that deviations in the y direction are penalized more heavily.

Since we only optimize the controller for simulations of a fixed duration (10 seconds), we need to encourage solutions that are more likely to lead to stable walk cycles when simulated beyond that duration. Two major sources of instability are swing foot toe-stubbing and toe-off before the stance foot is firmly planted on the ground. We discourage these situations with the following objective:

$$E_{land} = \frac{1}{T}\sum_{t=1}^{T}(stance_t + stubbed_t) \ , \tag{3.7}$$

where the summation is over all simulation steps t. If, during States 1 or 3 (toe-off), neither the stance toe nor the stance foot have 3 or more points of contact with the ground, then $stance_t = 1$; it is set to zero otherwise. We set $stubbed_t = 1$ at any time when the top of the swing toe is in contact with the ground.

Similarly, we define

$$E_{fail} = \frac{1}{T}\sum_{t=1}^{T} failed_t \ , \tag{3.8}$$

where for all states, if the COM falls below 0.7 m (i.e., the simulated character has fallen down) then $failed_t = 1$. With the exception of character "long arm" (see Figure 3.2), which fails if the COM falls below 0.4 m.

Our fourth gait objective encourages left-right symmetric timing of the controller, i.e.,

$$E_{sym} = Q(\Delta t_0 - \Delta t_2; \epsilon_t) + Q(\Delta t_1 - \Delta t_3; \epsilon_t) \ , \tag{3.9}$$

where Δt_i is the average duration of State i during the walk. This term, E_{sym}, requires left and right strides to have approximately the same duration. This helps to avoid spurious local minima producing asymmetric gaits.

[1] We replace v_x with $\hat{v}_x$ if the latter is available, since we optimize v_x to be equal to $\hat{v}_x$ in (3.5).

Head and body constraints. The angular momentum of the body about its COM is typically very small in human walking [35], and has been shown to assist with balancing [45, 65]. In particular, a mechanical effect of arm swing is to reduce torso torques about the vertical axis induced by the lower body [59]. Motivated by this, we include the following objective:

$$avgL = \frac{1}{T}\sum_{t=1}^{T} \dot{L}_t^2 \tag{3.10}$$

$$E_{ang} = Q(\sqrt{avgL}; \sqrt{\epsilon_{ang}}) \ , \tag{3.11}$$

where $\dot{L}_t$ is the normalized torque (time-derivative of the normalized angular momentum [35]) about the COM in the vertical direction at time step t, computed using finite differences. We find that this term helps to prevent unnatural arm swing, where the arms and legs are badly out of phase.

In natural human walking, the lateral and vertical motions of the head are typically smooth with small amplitudes. This helps to stabilize the visual and vestibular systems [84]. We include an objective that encourages the head to exhibit a fixed orientation and a constant forward velocity. Let $\mathbf{\Theta}_i = [\theta_{cor}, \theta_{sag}, \theta_{trans}]$ represent the head world frame orientation at simulation step i. The objective to stabilize the head motion is then

$$E_{head} = Q(\sqrt{\sigma_{head}}; \sqrt{\epsilon_{head}}) + \frac{\lambda_{head}}{T}\sum_{t=1}^{T} orient_t \ , \tag{3.12}$$

where σ_{head} is the standard deviation of the head velocity in the x direction during the walk, $orient_t$ is a binary variable which is set to 1 when $\|\mathbf{\Theta}_t\|_2 > 0.1$.

Efficiency and power terms. One notable biomechanical property of human walking is its efficiency [3, 15]. Indeed, the nearly passive nature of leg and arm swing is characteristic of a natural walking gait. To encourage an efficient gait, we penalize the sum of the squared torques over the duration of the simulation:

$$E_{power} = \frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{m} \tau_{tj}^2 \ , \tag{3.13}$$

where τ_{tj} is the torque output at time step t for joint DOF j. If the target velocity is unspecified, we replace E_{power} with $\frac{E_{power}}{v_x}$, to approximate the cost of transport [15].

It is also well known that, unlike human running, human walking is powered more by the ankle than the hip [78]. To encourage a natural distribution of torques in the hip, knee and ankle, we encourage the ratio of power outputs from these joints to match those observed in humans. Specifically, let $\vec{r}$ be the relative magnitudes of the power output from the hip, knee, and ankle, produced by the controller; i.e.,

$$\vec{r} = \frac{[P_{hips}, P_{knees}, P_{ankles}]}{P_{hips} + P_{knees} + P_{ankles}} , \tag{3.14}$$

where

$$P_{DOF} = \frac{1}{T} \sum_{t=1}^{T} \sum_{j \in DOF} \tau_{tj}^2 . \tag{3.15}$$

We penalize the deviation of $\vec{r}$ from the empirical power ratio (hip to knee to ankle), $\vec{r}_{walk} = [0.43, 0.04, 0.53]$ observed in humans [78]:

$$E_{ratio} = \|\vec{r} - \vec{r}_{walk}\|_2 . \tag{3.16}$$

Note that this term encourages a nearly passive knee.

Complete objective. The complete objective function for walking is given by

$$E = \sum_s w_s E_s , \tag{3.17}$$

where $s \in \{user, vel, land, fail, sym, ang, head, power, ratio\}$. We use the following parameters for all experiments: $w_{user} = 100$, $w_{vel} = 100$, $w_{land} = 1.2$, $w_{fail} = 120000$, $w_{sym} = 100$, $w_{ang} = 10$, $w_{head} = 100$, $w_{power} = 10^{-5}(70/mass)$, $w_{ratio} = 1$, $\lambda_{vel} = 0.01$, $\lambda_{head} = 0.012$, $\epsilon_{vel} = 0.025$ m/s, $\epsilon_{step} = 0.025$ m, $\epsilon_t = 0.025$ s, $\epsilon_{ang} = 0.05$ /s^2, $\epsilon_{head} = 0.1$ m/s.

3.3.2 Simulation

Simulation is performed using the Open Dynamics Engine (ODE, version 0.9)[2]; with a simulation frequency of 2400Hz. Ground contact is modeled using the default collision detector with a maximum of four points on the toe and four points on the foot. Both the toe and foot are modeled as boxes, and the ground is modeled as a plane. The contact parameters are adjusted to simulate a spring-damper system with $k_p = 75000, k_d = 2000$. We set the coefficient of friction to $\mu = 10$, which is higher than physically realistic, but necessary to prevent lateral slipping during toe-off.

3.3.3 Optimization algorithm

The optimization problem is high-dimensional, discontinuous, and subject to many local minima. Moreover, each function evaluation involves a simulation in ODE, which runs in approximately real-time. It is important for the method to use as few function evaluations as possible, without the need to evaluate gradients. We tested several different optimization algorithms, and found CMA [31] to work best.

As discussed in Section 2.4, CMA is an iterative algorithm that maintains a Gaussian distribution over parameter vectors. The Gaussian is initialized with a mean and a spherical covariance matrix with diagonal elements equal to σ^2, where σ is a problem-dependent step size parameter. M random samples are drawn from the Gaussian, and the objective function is evaluated for each. A new Gaussian is constructed using the best N samples, based on a function of these samples and the old mean. The process is repeated, as the Gaussian converges to a low-objective region. It is assumed that the parameters are scaled to lie roughly between 0 and 1.

We chose $\sigma = 0.0025$ experimentally. For the number of samples, we use default parameters that are automatically chosen by CMA based on the dimension of our optimization problem. In particular, $M = 19$, $N = 9$. The 19 simulations are computed in parallel using a cluster of 19 Intel Xeon 3.8GHz CPUs. In total, 1000 CMA iterations take approximately 3 hours. Heuristically, we stop the optimization either when no better values are discovered for 300 iterations, or when the total number of iterations exceeds 3000.

[2] `http://www.ode.org`

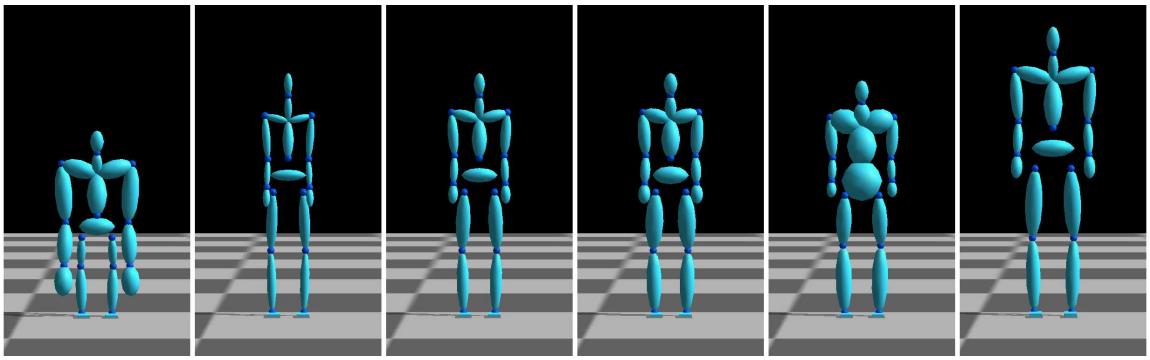


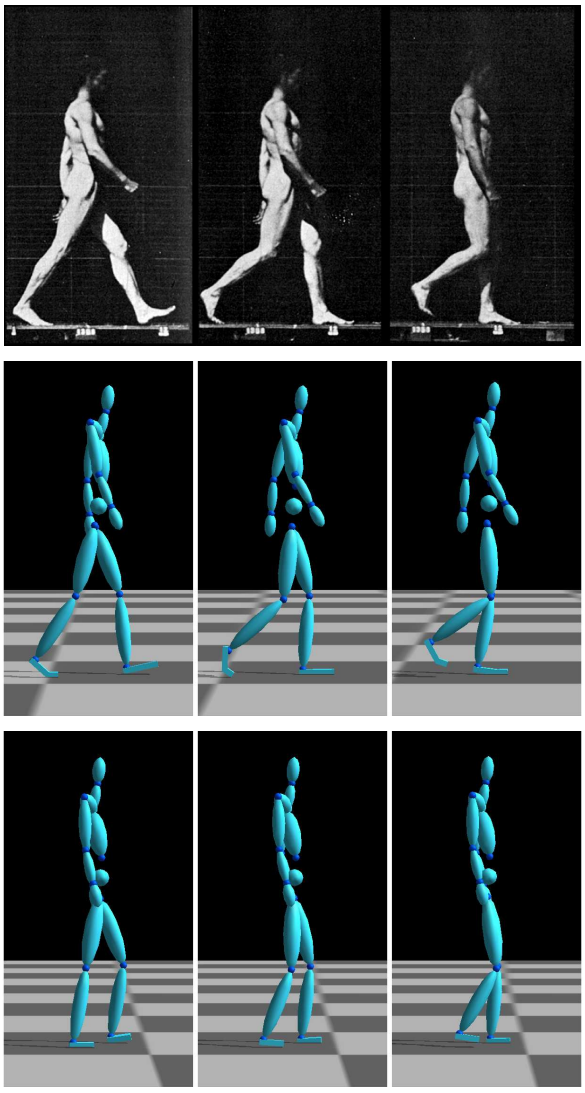
Figure 3.2: Characters used in the chapter. From left to right: "long arm" (127 cm, 65 kg) , "thin" (165 cm, 41 kg), "short" (165 cm, 57 kg), "stocky" (165 cm, 70 kg), "overweight" (161 cm, 86 kg), "tall" (196 cm, 79 kg).

We initialize the CMA optimization with a hand-tuned controller that generates significant ankle push-off in the stance leg during States 1 and 3. The parameters for the initial simulation state (start state) and the PD controllers are included in Table 3.1. Additionally, the coronal balance feedback parameters are initialized to $c_d = 0.2, c_v = 0.2$, the sagittal ones are initialized to zero. The arm swing and state transition parameters are initialized to $\alpha_{arm} = 1.0, c_{trans} = 0.01$. Note that this controller is only tuned to take a few steps in one of the character skeletons ("stocky" in Figure 3.2). It is not stable and does not walk in a straight line, but is sufficient for initialization. The initial values of k_p and k_d are scaled for very heavy and light characters, in proportion to the character's weight ratio relative to the default character.

3.4 Experiments

We now describe experiments that demonstrate the approach and test the effects of its various elements; results can be seen more clearly in the accompanying video. Characters used are depicted in Figure 3.2.

Features of human walking. Our model captures a number of features of normal human walking. Figure 3.3 shows images of a walking human, frames from a walker generated with our system, and frames from our implementation of SIMBICON [137], which represents the state-of-the-art in full-body controllers without mocap. The SIMBICON model employs a hip push-off strategy, whereas our model uses an ankle push-off more



Left Heel-Strike Right Toe-Off

Figure 3.3: Comparison of our model to human and SIMBICON from the left heel-strike to right toe-off stages of walking. *Top:* Man walking, taken from Muybridge [72]. *Middle:* One of our optimized controllers. *Bottom:* Our implementation of SIMBICON. Note the more conservative style in the bottom row, which keeps the right foot relatively flat, whereas our result and the human show significant rotation. Also note that the left leg in the left column does not reach full extension in the bottom row.

similar to human walking. The body synchronization approximates that of the human as well: stance foot heel-off occurs just before swing foot heel-strike. Furthermore, our model captures the passive knee and leg extension of the swing leg, as well as the foot rotation.

Effects of individual terms. Next we compare optimizing our walker with and without various energy terms. For example, we optimize the "short" character (Figure 3.2) to walk in 1.0 m/s without a specified step length. Figure 3.4 compares the model optimized with and without the power ratio term (E_{ratio}). Without this term, the power ratio between hip, knee, and ankle in the resulting motion is $[0.38, 0.21, 0.41]$; the knee

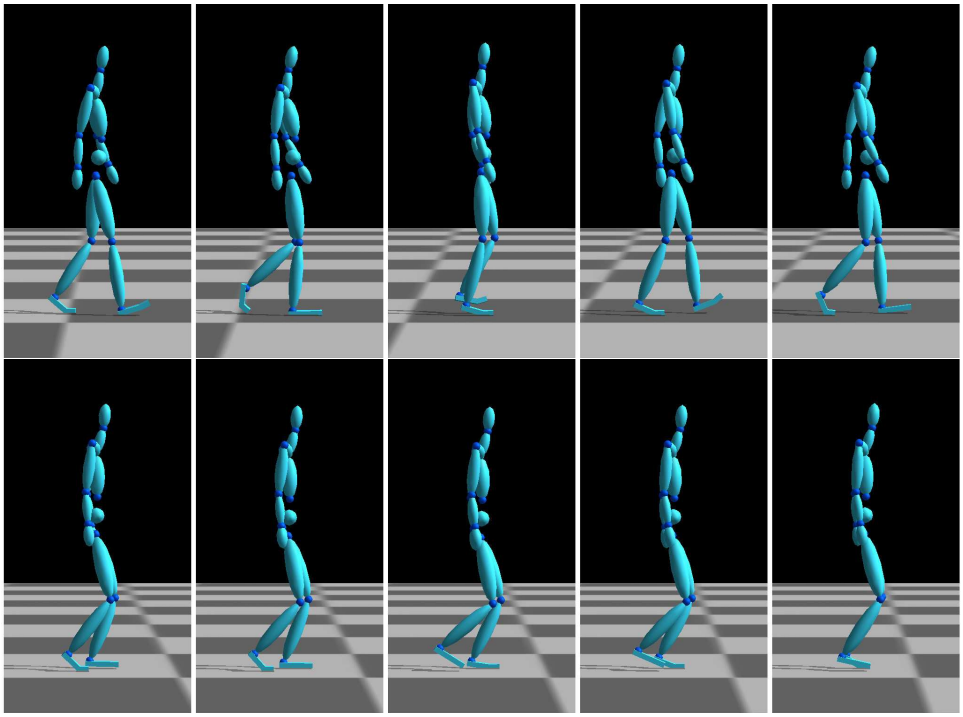


Figure 3.4: *Top:* Optimization of "short" (Figure 3.2) walking in 1.0 m/s. *Bottom:* Optimization without E_{ratio}. The lack of the power ratio term leads to a semi-crouching style.

takes a much heavier load than in human walking. This effect can be observed in the animation as a semi-crouching style, which appears unnatural and tiring. In contrast, our model discovers a more relaxed gait, with a power ratio of $[0.43, 0.05, 0.52]$, close to our optimization target.

Figure 3.5 shows our model optimized with and without the angular momentum objective (E_{ang}). In this example, we optimized the "stocky" character (Figure 3.2) to walk at 1.8 m/s with a step length of 0.7 m. The resulting motion correctly exhibits in-phase arm-swing, with the arms counter-oscillating with respect to the legs. Without the angular momentum term, the arms are in-phase with the wrong legs, and the walker is unstable. Indeed, the controller does not walk successfully much beyond the optimized duration of 10 seconds, whereas the controller learned with E_{ang} walks for at least 100 seconds.

We find that optimizing without the head stabilization term (E_{head}) leads to jerky upper-body motion (see video). We test optimizing without the $E_{landing}$ term, which prefers the stance foot to be stably planted before push-off. Without this term, foot landing and roll is jerky and abrupt, whereas including the $E_{landing}$ term makes the foot land and

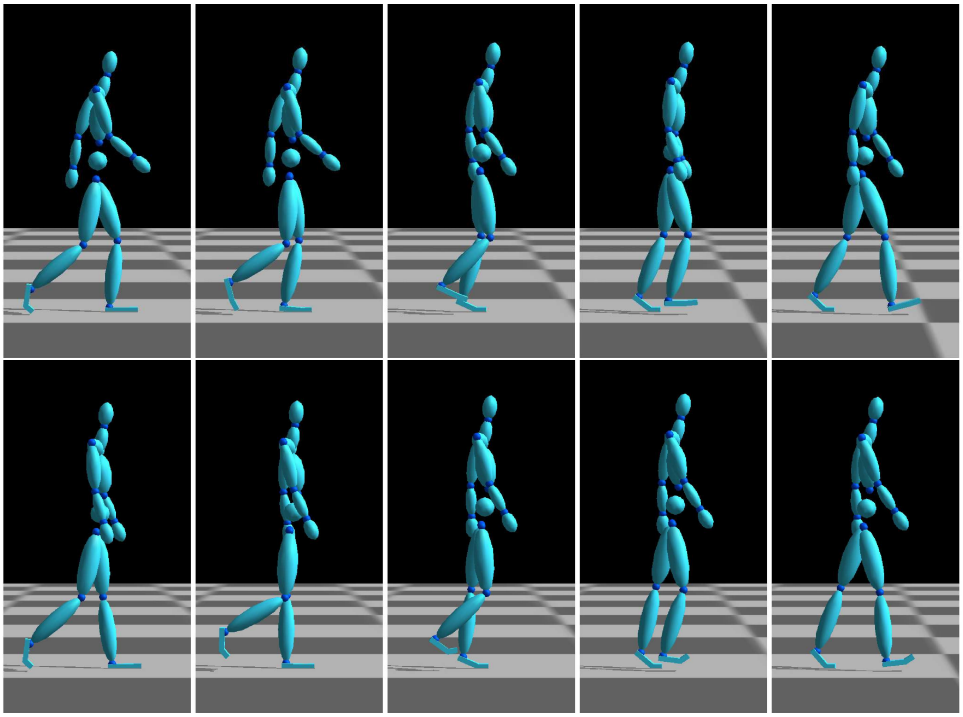


Figure 3.5: *Top:* Optimization of "stocky" (Figure 3.2) walking in 1.8 m/s, step length 0.7 m. *Bottom:* Optimization without E_{ang}. Note the difference in arm swing.

roll forward smoothly. The E_{power} term serves to constrain DOFs that are not directly influenced by the other terms to behave in a relaxed fashion. This is particularly relevant to the upper body; as shown in the video, omitting this term leads to jerky arm motion.

Comparison with motion capture data. We further evaluate our results by comparing our optimized controller output with mocap data[3]. The data comprise a manually segmented walk cycle from each of 115 subjects, all asked to walk at a comfortable speed. Figure 3.6 compares the thigh orientation, knee flexion/extension, and foot orientation for seven of our optimized controllers and the mean curve from the mocap dataset. The mean curves (solid lines) were computed by uniformly sampling 100 locations on the walk cycle, then taking the mean over all subjects on each location. The standard deviation curves (orange regions) were computed in the same fashion. Dashed blue and red curves represent seven optimized walkers and SIMBICON, respectively. We see that the minimum hip orientation (Figure 3.6a) in the mocap data is smaller than both of the physically simulated gaits. A similar, though less pronounced, effect occurs in the

[3]The capturing method is as described in Troje [111].

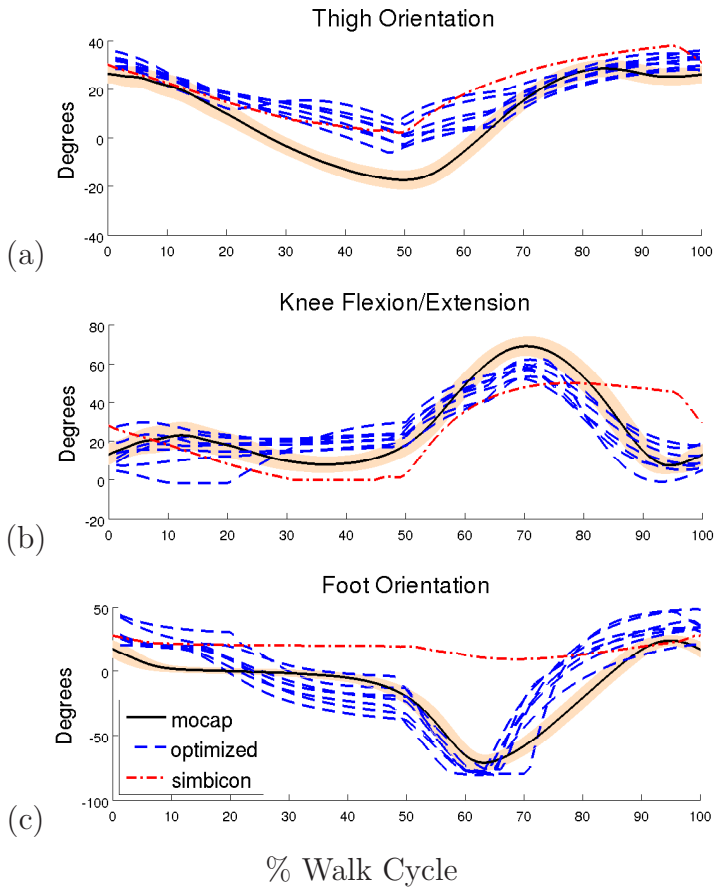


Figure 3.6: Sagittal plane angle versus percentage walk cycle plots. (a) Thigh orientation w.r.t. down vector. (b) Angle between thigh and shin. (c) Ankle orientation w.r.t. front vector. Thanks to Nikolaus Troje for providing the mocap data.

knee flexion/extension plot (Figure 3.6b) as well, where the optimized controllers tend to both flex and extend less than the mocap average. Figure 3.7 visually compares one of the optimized controllers with a similar mocap walk cycle. The smaller range of hip and knee motions seem to lead to a smaller step length. Two major differences between our optimized controllers and SIMBICON are knee extension at heel-strike (i.e., Figure 3.6b at 0 and 100%), and the foot trajectories plotted in Figure 3.6c. In both cases, motions generated by the optimized controllers are much closer to mocap. However, the foot orientation plot shows our optimized controllers tend to push-off with the ankle earlier than the mocap.

Optimizing SIMBICON. In order to separate the effects of our objective function from those of our body model, we apply our optimization to the SIMBICON body model and parameterization. The set of control parameters that generated the bottom row of Figure 3.3 is used as initialization. We do not specify the target speed. Instead we penalize the approximate cost of transport ($\frac{E_{power}}{v_x}$) in the objective. We find that our

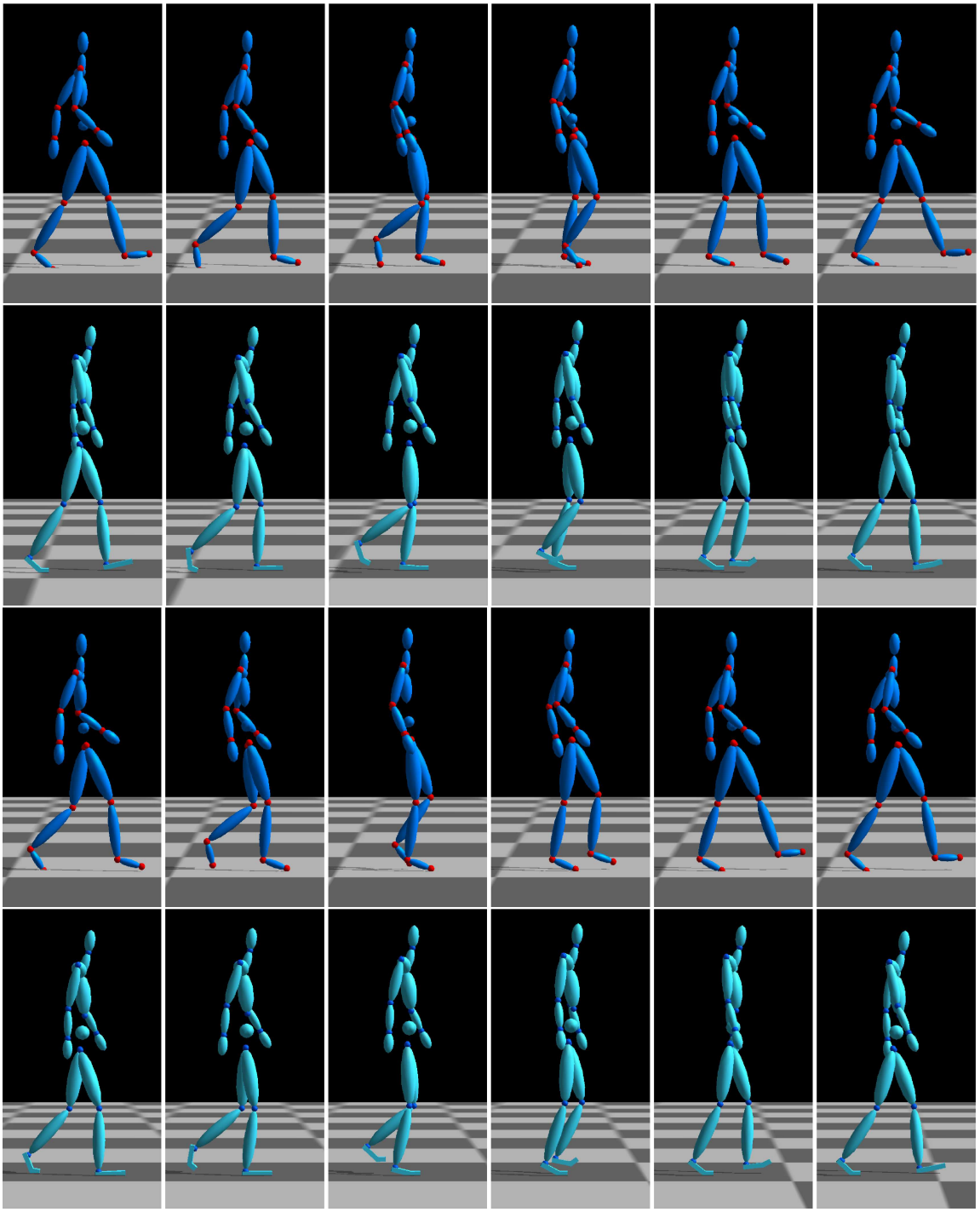


Figure 3.7: Visual comparison between mocap data (rows 1,3) and motion generated by character "tall" (Figure 3.2) with no user gait constraints (rows 2,4) for a full walk cycle.

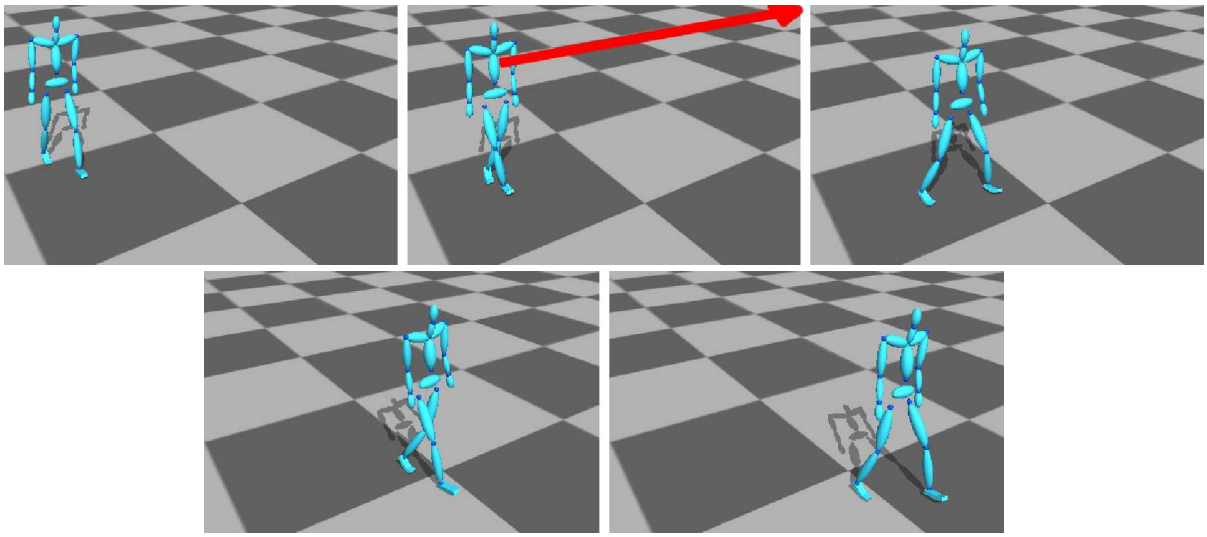


Figure 3.8: Controller tall (Table 3.2) reacting to a 200 N, 0.4 s push to the torso in the (0,1) direction. Our optimized controllers tend to be more robust to pushes from the side, than to pushes from front and back.

optimization process improves the motion in two ways. First, the optimization discovers anti-symmetric arm swing, even though this control parameterization does not explicitly couple arm swing to leg motion as our model does. Second, the character makes some use of ankle push-off. However, compared to our results, the swing leg does not extend as far, and the ankle push-off is still lacking. Despite the low percentage of knee torque output in the lower body, the knee still appears active.

We note that with the same optimization specification (same character, objective) using our model and initialization, we are able to discover a gait with $E_{power} = 12059$, much lower than the SIMBICON solution's $E_{power} = 33392$. Moreover, our solution walks with a faster speed of 1.48 m/s versus 1.36 m/s.

Variation in body shape. Our optimization can be used to generate controllers for a wide rage of body types. We minimize the approximate cost of transport ($\frac{E_{power}}{v_x}$) to generate walks for all characters in Figure 3.2, without specifying target speed and step length. This allows the walker to identify its preferred speed and step length. As shown in the accompanying video, the results retain the qualities of human walking discussed above. A natural relationship between body shape, speed and gait emerges; that is, the tall subject's preferred gait is much faster (1.79 m/s) than those of the shorter subjects. The overweight subject appears to walk with a more lumbering gait, and the long-armed

direction	short	overweight	tall	short2	stocky	simbicon
(1,0)	75	1	20	10	50	225
(1,1)	75	1	20	10	65	150
(0,1)	100	25	200	50	125	250
(-1,1)	50	50	70	40	75	100
(-1,0)	60	75	120	75	50	120
(-1,-1)	40	40	70	40	50	100
(0,-1)	100	20	200	75	125	275
(1,-1)	75	1	20	10	50	125

Table 3.2: Maximum disturbance force components (in newtons) tolerance for some of our controllers and our implementation of SIMBICON, where (1,0) is the forward direction. Controller overweight is optimized for a step length of 0.8 m, short2 for a step length of 0.65 m, and stocky for a step length of 0.7 m and a speed of 1.8 m/s. The others are optimized without user gait constraints.

humanoid keeps its heavy arms still in order to maintain balance.

Robustness. We quantify the robustness of some of our controllers via a pushing experiment (see e.g., Figure 3.8). For each controller, we simulate forward for 40 seconds, and apply a push force (F_x, F_y) for 0.4 seconds to the torso COM once every four seconds (to allow for recovery time). The controller passes the experiment if it is still walking forward (direction $(1, 0)$) after 40 seconds. The maximum amounts of tolerable push from 8 directions are summarized in Table 3.2. We observe that the robustness of our controllers varies greatly. The overweight controller is unstable when pushed from the back, but is fairly robust to pushes from the front and sides. All of the optimized controllers tend to be more robust to pushes from the side, than to pushes from the front and back. The experiment also shows that these controllers are less robust than our implementation of SIMBICON, which can stand more than twice as much force than the optimized controllers in some cases. Note that SIMBICON has been reported to withstand force components between 190 N and 340 N [137] in a related experiment. We suspect fine-tuning our implementation of it can lead to results approaching those numbers as well.

Changing terrain. In the spirit of Yin et al. [136], we obtained uphill walkers for slopes of up to 12 degrees (see Figure 3.9) by optimizing for a sequence of walkers. This began with optimization for a 3-degree slope, initialized with a walker on flat ground,

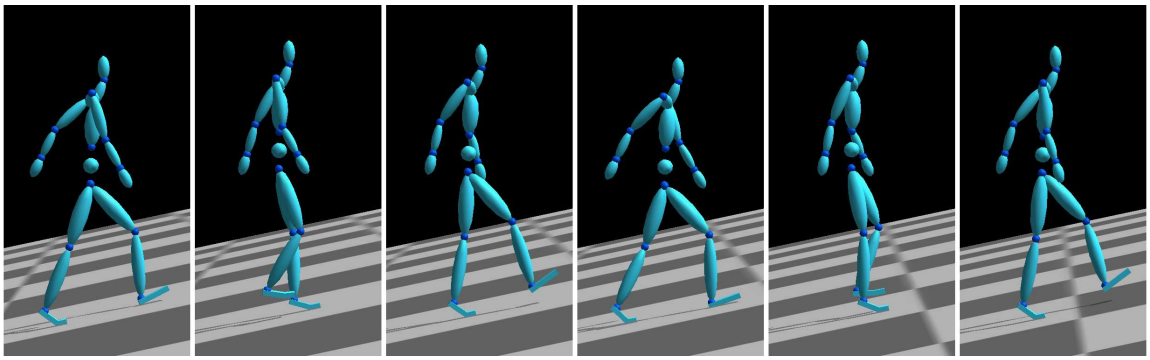


Figure 3.9: The short2 controller (Table 3.2) adapted to walk up a 12-degree slope.

although it was not run to convergence. For both the uphill walker and initialization, the target step length was set to 0.65 m. We then repeated this process for a sequence of walkers, increasing the slope until 12 degrees. The controller successfully walks up the slopes, leaning into the slope in a natural way, whereas the level-ground walker fails at the 3-degree slope. Similarly, we optimized a walker for a plane with a lower coefficient of friction. Beginning with an initial walker, we optimized for $\mu = 0.8$, and then optimized for $\mu = 0.3$. The resulting controller walks more gingerly, at a slower rate.

Other variations. As shown in the accompanying video, we can create controllers to walk at specified speed and/or step lengths via E_{user}, including specifying atypical speed and step length combinations. Alternatively, either speed or step length or both can be determined automatically. Our initial walker in Figure 3.9 is generated by specifying a target step length of 0.65 m without a target velocity. The accompanying video shows more examples of controllers synthesized in this fashion as well as results obtained by altering other terms in the objective function. If we redefine the cost of transport term as $1/v_x$, thereby omitting the penalty for total torque, then the objective function prefers motions that walk as fast as mechanically possible, without regard to energy consumption, yielding extremely fast walking.

3.5 Discussion

We have demonstrated a method for optimizing full-body 3D walking control that captures important features of natural walking without relying on mocap data. While the

basic idea of optimizing controllers is an old one, our results show that achieving good walking control requires careful choice of body parameterization, controller parameterization, and objective function. In particular, the results illustrate the importance of angular momentum minimization, relative magnitudes of lower-body joint torques, and head stabilization, among others. We believe these observations will be useful for designing more sophisticated controllers and objective functions, especially since our objective function and optimizer are independent of the choice of control parameterization. Our work also illustrates how subtle details in control parameterization — especially at the foot — can make a significant impact on the style of motion.

There are a number of limitations of our method that provide opportunities for future work. Our method requires an expensive optimization procedure, and depends on a reasonable initialization. As our goal in this chapter is natural-looking human locomotion, the controllers we produce are not as stable as SIMBICON. The difference is most likely due to the more conservative, less human-like strategy taken by SIMBICON that keeps the feet orientated parallel to the ground during the entire gait (Figure 3.6c), making toe-stubbing less likely. Another difference is that, while the balance parameters are tuned for robustness in Yin et al. [137], our objective function does not explicitly encourage robustness beyond the requirement of walking for 10 seconds. We explicitly address the robustness problem in the next chapter.

As is evident from comparison with mocap data, the motion generated by our controller still differs from human motion in noticeable ways. Figure 3.6 indicates that the thigh and knee rotation do not straighten as much as the data, especially when the stance leg is behind the COM. Note that when we only specified desired speed without desired step length, the resulting walker typically takes shorter steps than mocap data. This might be explained by the lack of hind leg stretching in our controllers.

Finally, our method does require some parameter tuning in order to achieve reasonable gaits and to achieve the desired style. We believe that adjusting energy parameters will be more intuitive than manually adjusting control parameters. Furthermore, we anticipate that it may be possible to learn the parameters from mocap data.

Chapter 4

Optimizing for Robustness to Uncertain Environments

We introduce methods for optimizing physics-based walking controllers for robustness to uncertainty. Many unknown factors, such as external forces, control torques, and user control inputs, cannot be known in advance and must be treated as uncertain. These variables are represented with probability distributions, and a return function scores the desirability of a single motion. Controller optimization entails maximizing the expected value of the return, which is computed by Monte Carlo methods. We demonstrate examples with different sources of uncertainty and task constraints. Optimizing control strategies under uncertainty increases robustness and produces natural variations in style.

4.1 Introduction

When designing controllers for character locomotion, one cannot know with certainty all factors that will influence the character's motion at run-time. Many unknown factors, such as external forces due to strong winds, interactive user inputs, or noise in the motor control system, can be significant and must be treated as uncertain. One might try to cope with such uncertainty by making the controller stiff and deliberate. However, it would be more desirable for the control strategy to adapt to different scenarios, much as humans do. For example, when walking on a slippery surface of variable roughness,

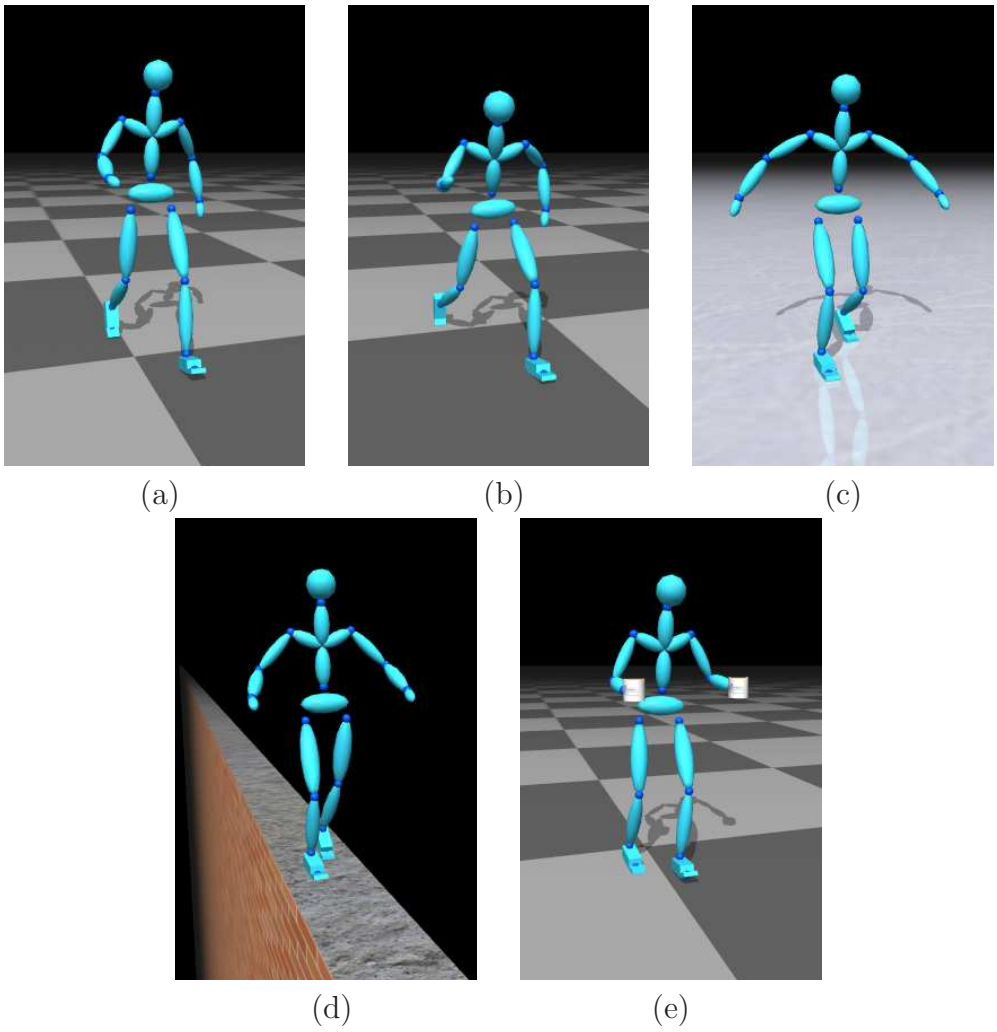


Figure 4.1: Walking controllers optimized for different environments with uncertainty. (a) Walking can be relaxed in a deterministic environment, without random external perturbations. (b) Under gusty conditions, the gait is more aggressive, with a wider stance. (c) On a slippery surface with internal motor noise, the gait is cautious with arms extended for balance. (d) Walking on a narrow wall on a windy day produces a narrower gait with small steps. (e) With internal motor noise, carrying hot beverages requires a slow gait with steady arms.

one might improve stability with a lower COM, outstretched arms, and smaller steps. Alternatively, when experiencing extreme gusty winds, one might walk more stiffly with a widened stance to avoid being blown over.

This chapter introduces a technique for automatically learning robust control strategies under different scenarios and various sources of uncertainty. We consider four sources of uncertainty. First, we incorporate unknown, varying external forces acting on the body, like wind on a gusty day. Second, we incorporate uncertainty due to user inputs.

When designing controllers for interactive simulation, *a priori* one cannot know a user's run-time control inputs, e.g., to change a character's heading or speed. A third source of uncertainty arises when transitions occur between controllers, where the start state for one controller depends on the state produced by the previous controller at the point of transition. Any variability in the timing of the transition, or in the motion produced by the first controller, will produce start states for the second controller that cannot be known *a priori*. The fourth source of uncertainty we consider is motor noise. In humans, it is believed that neuronal motor noise influences motor strategies, e.g., in the coordination of eye saccades, finger pointing, and in line drawing [33, 43, 109]. Accordingly, the inclusion of motor noise may help to produce human-like control strategies under different environmental conditions. Noise is also a convenient way to capture a wide range of otherwise unmodeled, complex sources of uncertainty that might significantly influence a character's motion.

Learning different control strategies is formulated in terms of optimizing 3D locomotion controllers in the presence of unknown environmental variables and controller inputs. We use a probabilistic formulation in which all prior beliefs over unknown quantities are modeled by probability distributions. Together with a controller and dynamics, they define a probability distribution over motions. A *return function* scores the quality of a given motion. Our goal then is to optimize a controller to maximize the *expected return*, a quantity not computable in closed-form. We use Monte Carlo methods to approximate the expected return. This approximation is optimized by CMA [31]. As a result, the character's control strategy and style of movement are determined automatically as a function of stochastic variables and the return function.

Our controllers exhibit increased robustness compared to *baseline controllers* that are optimized for scenarios where all significant factors are known *a priori*. For example, the amount of unexpected external force that a given walking controller can withstand can be greatly increased. Furthermore, the type and degree of robustness can be controlled through the specification of probability distributions over the unknowns. Different tasks and types of uncertainty together lead to different styles of movement. For example, a character walking on a slippery surface with increased motor noise tends to extend his arms and bend his knees for balance and stability. In contrast, a controller walking on a narrow beam with uncertain external forces will be conservative, taking relatively small steps with a narrow gait width to avoid stepping too close to the edge.

Optimization under uncertainty is also useful when composing controllers. In general, switching between controllers is unreliable, unless the controllers are highly robust, or they operate in similar regions of state space. We find that controllers optimized under uncertainty are generally more robust and therefore more tolerant to state variability at transitions. We also show that controllers optimized with uncertain start states can be used to create transition and recovery controllers to facilitate composition.

The approach we advocate here is conceptually intuitive and broadly applicable, supporting an extremely general class of uncertainty. Given a basic controller and CMA, this method is also very simple to implement. Optimization under uncertainty is expensive, often requiring overnight computations. Nevertheless, once optimized, the resulting controllers run at real-time rates.

4.2 Related work

As briefly discussed in Chapter 2, previous work in optimal control for physics-based animation assumed deterministic dynamical systems. The previous chapter is no exception; we optimized full-body locomotion controllers in deterministic settings. These controllers are robust only to limited external disturbances, and do not exhibit stylistic variation as a function of uncertainty. We extend this approach to stochastic environments and user inputs, and show that different sources of uncertainty lead to stylistically different control strategies.

Reinforcement learning has been used for control in kinematic motion graphs [53, 54, 64, 110]. These techniques assume completely deterministic systems. McCann and Pollard [67] proposed the use of a probability distribution over user inputs, and maximize expected return with respect to this distribution. Motion graph control is significantly different from low-level physical control, however, because actions in motion graphs are discrete and low-dimensional. Motion graph methods do not explicitly handle the possibility of control failure, such as tripping and falling.

4.3 Optimal control under uncertainty

Our approach to controller design combines probabilistic modeling of uncertain (i.e., unknown) quantities, and optimization under uncertainty. We begin with the formulation of the deterministic case.

4.3.1 Deterministic simulation and optimization

An articulated rigid-body character at time t is represented by a state vector $\mathbf{s}_t$, comprising joint angles, root position and root orientation. Articulated rigid-body dynamics and integration provides a function for mapping the state $\mathbf{s}_t$ at time t to the state at the next time instant $\mathbf{s}_{t+1}$:

$$\mathbf{s}_{t+1} = f(\mathbf{s}_t, \boldsymbol{\tau}_t, \mathbf{e}_t)\,. \tag{4.1}$$

This mapping is a function of state $\mathbf{s}_t$, the internal joint torques $\boldsymbol{\tau}_t$ produced by the character controller, and the relevant world parameters $\mathbf{e}_t$, such as external forces (e.g., wind) acting on the character.

The control torques $\boldsymbol{\tau}_t$ are determined by a controller. A controller, π, defines a mapping from the current state, $\mathbf{s}_t$, to the desired control signal:

$$\boldsymbol{\tau}_t = \pi(\mathbf{s}_t, \mathbf{c}_t; \mathbf{w})\,, \tag{4.2}$$

where $\mathbf{w}$ is a vector of control parameters, and $\mathbf{c}_t$ is a vector of user inputs at time t. Given a start state $\mathbf{s}_1$, the control parameters $\mathbf{w}$, any user inputs $\mathbf{c}_{1:T}$ and environmental parameters $\mathbf{e}_{1:T}$, a deterministic simulation is performed by recursively applying (4.2) to determine $\boldsymbol{\tau}_t$, and then (4.1) to compute $\mathbf{s}_{t+1}$ for each time-step $1 \leq t < T$. This yields an animation sequence $\mathbf{s}_{1:T}$.

We formulate the optimal control problem with the specification of a *return function* (e.g., see [104]), denoted $R(\mathbf{s}_{1:T})$, which measures the quality of a simulation. In the deterministic case, optimal control seeks the control parameters $\mathbf{w}$ which produce motions that maximize the return function [28, 117]. The return function is based on the energy function from the previous chapter. We use the term return here instead of energy for consistency with the policy search literature.

A controller optimized for a specific run-time environment without uncertainty can sometimes be robust to moderate perturbations, but typically not to large perturbations, changes in environmental conditions, or variations in start state that might occur when controllers are composed. For example, a carefully chosen $\mathbf{w}$ that allows the character to walk in a straight line may fail when the character is pushed in the chest, or when a user attempts to change the heading or the speed of the character.

Character and controller details. Our character model is identical to that of Chapter 3, with the same set of DOFs. However, the mass distributions are computed from an automatically generated mesh [34] that approximates a male with height 180 cm, and weight 70 kg. The controller has four states, each with linear PD joint control combined with SIMBICON-like balance control. The control parameterization $\pi(\mathbf{s}_t, \mathbf{c}_t; \mathbf{w})$ is like that in the previous chapter, but with several minor differences. First, the target angles that rotate the shoulders in the coronal plane are not manually specified; instead they are optimized. Second, to reduce the number of optimized parameters, and to encourage smoothness of the body's motion, several control parameters are constrained to share the same value in all four controller states. These include the damping and stiffness constants for each upper body DOF, and the target values for the elbow angles and the shoulder angle of rotation in the coronal plane.

We make one final change to the controller. Although the target angle for the stance hip in the transverse plane (θ_t) can be specified as a user input to control the character's heading direction [137], direct manipulation of this parameter, e.g., by the user, can easily lead to failure. Instead, we define an additional time-varying *target heading direction* θ_d, which determines the change in θ_t from one time-step to the next, and is better suited to be a user input. If the new θ_d is close to the current target heading, θ_c, we immediately change the target angle to θ_d. When θ_d differs significantly from θ_c, the target stance hip angle is adjusted gradually, to reduce instability. In particular, it is updated from one time-step to the next using

$$\theta_{t+1} = \alpha_{hip}(\theta_d - \theta_t) + (1 - \alpha_{hip})(\theta_c - \theta_t) + \theta_t \,, \tag{4.3}$$

where θ_t is the stance hip target angle at time t, and α_{hip} is a weight parameter. Each controller state has two values for α_{hip}, for when $\theta_c > \theta_d$ or $\theta_c < \theta_d$, respectively. All

these parameters are optimized along with the other controller parameters.

4.3.2 The return function

The complete return function, for motion $\mathbf{s}_{1:T}$, is the sum of *rewards* over time, plus terms E_{power}, E_{ratio} to encourage power usage similar to human walking:

$$R(\mathbf{s}_{1:T}) = \left(\sum_t r(\mathbf{s}_t)\right) - (w_p E_{power} + w_r E_{ratio}) \ , \tag{4.4}$$

where r is a scalar reward function of the current state, $w_p = 10^{-5}(200/mass), w_r = 5$.

In practice, there's no need to compute rewards at the frequency of the simulation. We set $r(\mathbf{s}_t) = 0$ except for when t is a multiple of 20, or when a state switch due to ground contact occurred at time t (the end of a stride). The reward is defined as the negative sum of a number of energy terms (i.e., $r(\mathbf{s}_t) = -\sum_i E_i$). The energy terms are similar to those defined in Chapter 3, with a few minor but important differences. In what follows, we will redefine the E_i terms in a more general and simplified fashion.

We have optional user gait constraints as in (3.5),

$$E_{user} = Q(v_x - \hat{v}_x; 0.05) + Q(s - \hat{s}; 0.05) \ , \tag{4.5}$$

where the average velocity and step length parameters are computed with respect to the target heading direction. This change is important for enabling optimization for scenarios with heading direction changes. If neither is specified, then we use

$$E_{step} = Q(s - \hat{s}; 0.05) \, , \quad \text{where} \ \ \hat{s} = l\left(v_x/\sqrt{gl}\right)^{0.42} \ , \tag{4.6}$$

and l is the leg length of the character. This helps to ensure the resulting gait has a human-like speed to step-length ratio [46].

The target heading direction may be time-varying, with energy

$$E_{facing} = Q(v_y; 0.05) + 0.005Q(\theta_d - \theta, 0.1) \ , \tag{4.7}$$

where v_y is the average simulation velocity (in m/s) of the COM in the y direction in the current stride, θ_d, is the desired heading, θ is the current heading, and v_x, v_y are both computed by with respect to θ_d.

As in the previous chapter, small angular momenta are also preferred:

$$E_{ang} = Q(L_x; 0.04) + Q(L_y; 0.05) + Q(L_z; 0.01) \ , \tag{4.8}$$

where L_{xyz} are the maximum normalized angular momenta about the COM in the previous stride. Note that unlike (3.10), since our arm swing is no longer restricted to the sagittal plane in this chapter, we prefer small angular momenta about all three axes. The thresholds for the quadratic penalties are selected based on empirical data [35].

Finally, we include the following terms for head stability, stable foot contact, and penalty for falling:

$$E_{head} = Q(v_{head}; 0.25) + 0.001(orient_t) \tag{4.9}$$

$$E_{land} = 0.001(stance_t + stubbed_t) \tag{4.10}$$

$$E_{fail} = 100(failed_t) \ , \tag{4.11}$$

where the binary variables are as defined in Section 3.3.1. We only compute the following terms at the end of every stride: $E_{user}, E_{step}, E_{ang}, Q(v_y; 0.05), Q(v_{head}; 0.25)$.

4.3.3 Random environments and optimal control

When user inputs and environmental variables are uncertain, we do not have specific values for $\mathbf{c}_t$ and $\mathbf{e}_t$ *a priori*. Rather, we characterize our limited prior knowledge by specifying a probability distribution over each unknown variable. As examples, here we consider four general types of uncertainty. First, environmental uncertainty is represented by $p(\mathbf{e})$; this might represent, for example, the distribution over wind forces applied to the character's torso. User inputs, such as commands to change the character's heading, are also unknown *a priori*. We therefore include a distribution over possible user inputs $p(\mathbf{c})$. Third, the precise initial state of a controller is often unknown during run-time applications. For example, we may wish to compose controllers where transitions from one controller to another occur at variable time instants. Accordingly, we can specify

a distribution over start states for a given controller with a second distribution $p(\mathbf{s}_1)$. Finally, human motor neurons are subject to signal-dependent noise [21], which is thought to play a significant role in human motion planning [33]. We incorporate motor noise by perturbing the joint torques produced by the controller. To this end we specify a distribution over joint torques, $\boldsymbol{\tau}$, given the parameters of the controller, the current state, and the user inputs, i.e., $\boldsymbol{\tau}_t \sim p(\boldsymbol{\tau}|\mathbf{s}_t, \mathbf{c}_t, \mathbf{w})$.

Together, these sources of uncertainty and noise, in combination with the dynamics (4.1), define a probability distribution over animations $p(\mathbf{s}_{1:T}|\mathbf{w})$. Despite the complexity of this distribution, it is rather straightforward to draw fair samples from it. To sample an animation sequence from $p(\mathbf{s}_{1:T}|\mathbf{w})$, one first samples a start state $\mathbf{s}_1 \sim p(\mathbf{s}_1)$. Then, for each time-step t, the environmental variables $\mathbf{e}$ and the user inputs $\mathbf{c}$, if desired, are sampled from their distributions: $\mathbf{e}_t \sim p(\mathbf{e})$ and $\mathbf{c}_t \sim p(\mathbf{c})$. Joint torques $\boldsymbol{\tau}_t$ are sampled as $\boldsymbol{\tau}_t \sim p(\boldsymbol{\tau}|\mathbf{s}_t, \mathbf{c}_t, \mathbf{w})$. Finally, the next state $\mathbf{s}_{t+1}$ is computed according to the dynamics (4.1).

Note that even for a deterministic system, simulations of multiple walk cycles will exhibit small deviations from perfect gait periodicity. Controllers optimized for long simulations will therefore be somewhat more robust than those optimized for short durations. But with random variables, we can no longer optimize the return of a single scenario, and then expect the controller to work during another simulation for which random variables take on different values. Instead, we optimize the *expected return*. The expected return of a controller, with control parameters $\mathbf{w}$, is

$$\begin{aligned} V(\mathbf{w}) &\equiv E_{p(\mathbf{s}_{1:T}|\mathbf{w})}\left[R(\mathbf{s}_{1:T})\right] \\ &= \int p(\mathbf{s}_{1:T}|\mathbf{w})\, R(\mathbf{s}_{1:T})\, d\mathbf{s}_{1:T} \ . \end{aligned} \tag{4.12}$$

The optimal control problem is to select $\mathbf{w}$ to maximize this expected return. By optimizing the expected return, we aim to find a controller that will work well, on average, with plausible values for the uncertain quantities. The same principles have been effective in modeling human motor control [43, 109].

4.3.4 Evaluation and optimization

Due to nonlinear dynamics and non-Gaussian noise, the expected return (4.12) cannot be computed analytically; hence we use Monte Carlo methods [104]. Specifically, N animation sequences are sampled, as described above. The approximation $\hat{V}(\mathbf{w})$ is then computed as the average return on these sequences:

$$\hat{V}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} R(\mathbf{s}_{1:T}^{(i)}) \ , \qquad (4.13)$$
$$\text{where} \quad \mathbf{s}_{1:T}^{(i)} \sim p(\mathbf{s}_{1:T}|\mathbf{w}) \ .$$

For example, suppose the only source of randomness is an external force, $\mathbf{f}$, applied to the character's torso at time t, where the direction and magnitude of the force are uniformly distributed. We run the simulation N times, each time applying different forces drawn at random from the uniform distribution. To optimize $\hat{V}(\mathbf{w})$, we use the CMA algorithm [31].

Note that, because new motions are sampled for each evaluation, $\hat{V}(\mathbf{w})$ is a random quantity. Hence, even if $\mathbf{w}$ is fixed, one obtains a different result each time $\hat{V}(\mathbf{w})$ is evaluated. This can cause problems for optimization algorithms. One issue is as follows. Suppose, when comparing two controllers $\mathbf{w}_1$ and $\mathbf{w}_2$, we obtain estimates for which $\hat{V}(\mathbf{w}_1) > \hat{V}(\mathbf{w}_2)$. We cannot tell whether this is because $\mathbf{w}_1$ is really better than $\mathbf{w}_2$, or if the random forces sampled for the second evaluation were more challenging than those for the first. We address this by using the method of common random numbers (CRN) [102], also known as PEGASUS [75]. In CRN, one reuses the same random seed each time $\hat{V}(\mathbf{w})$ is evaluated. This makes $\hat{V}(\mathbf{w})$ deterministic: for the example above, the same sample of N random forces would be used in each evaluation. This resolves difficulties with many optimizers, and, under certain conditions, can be shown to yield better results with high probability. We find that effective controllers can be optimized with small values of N (e.g., see Figure 4.4).

With each application, the controllers are optimized using the method of CRN with $N = 10$ simulations, and CMA. Following the previous chapter, the simulator frequency is 2400 Hz (time-steps are about 0.00042 s), each simulation run is 10 s long, and we run 19 CMA samples in parallel per iteration. The optimization here is, however, more expen-

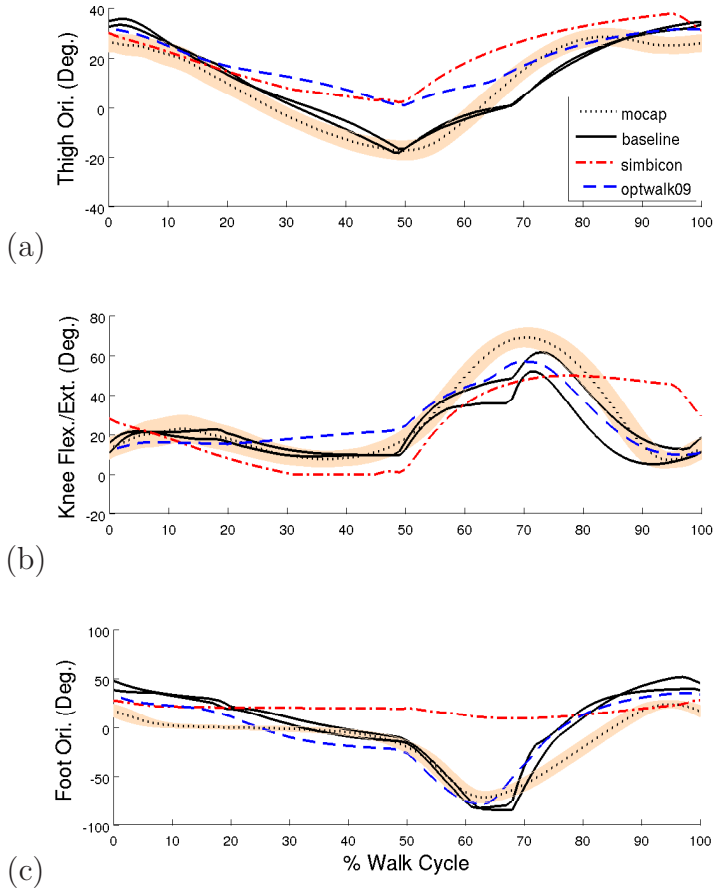


Figure 4.2: Comparison of baseline controllers to motion capture data. (a) Thigh orientation w.r.t. down vector. (b) Angle between thigh and shin. (c) Ankle orientation w.r.t. front vector. Thanks to Nikolaus Troje for providing the mocap data.

sive, because each evaluation of the return function requires N simulations. Convergence typically requires a few hundred iterations. Running the optimizations overnight on a cluster of 20 CPUs is usually sufficient. Further parallelization of the N independent samples is possible.

Because the optimization is nonconvex, a good initial guess is required and local minima are problematic. We first optimize a controller with no uncertainty, as in Section 4.3.1. This *baseline controller* is then used as the initial guess for controller optimization under uncertainty. When uncertainty or variability in environmental conditions is extreme, it is easy for the optimization to get trapped in poor local minima. In these cases, we first optimize controllers for smaller noise levels or perturbations (often with early stopping). These are then used to initialize optimizations for higher levels of noise. We find that incremental optimization, while slowly changing conditions from baseline controllers to the desired scenarios, typically produces effective controllers.

4.4 Applications

To demonstrate the impact and generality of optimization under uncertainty, we consider several applications where different environment conditions and sources of uncertainty combine to produce different strategies for robustness. These controllers are compared against their corresponding baseline controllers to evaluate the significance of uncertainty. The baseline controllers are similar to those described in the previous chapter, but with small differences in control parameterization, objective function, and mass distribution, the gaits produced by baseline controllers appear more natural. Many features, such as knee angle (Figure 4.2b) and foot orientation (Figure 4.2c) behave as before, while others, such as the thigh orientation trajectory (Figure 4.2a) bear greater similarity to mocap data. The dotted curves and orange regions represent the mean and standard deviation of walk cycles previously depicted in Figure 3.6. Dashed blue and red curves represent the mean of optimized controllers from the previous chapter and SIMBICON, respectively. Solid lines represent baseline controllers: optimized with no user constraints and the 1.6 m/s controller described in Section 4.4.4.

In general, however, the motion trajectories generated by our controllers (including ones from the previous chapter) are not as smooth as typically observed in motion capture data. For example, note the discontinuous change in the first derivative, which occurs between 65% and 70% of the walk cycle. It can most clearly be seen in Figure 4.2b, but is also apparent in Figure 4.2a. This is an artifact of state transition from 0/2 to 1/3, and can possibly be alleviated by interpolating the target angles between states. To the keen observer, the lack of smoothness also manifests itself visually in the generated motions. Indeed, qualifying the remaining gap between walking motions created by our controllers and motion capture data is an open problem, and the removal of state-switching artifacts would certainly reduce the gap.

4.4.1 External disturbances

We begin with a scenario in which external forces are applied to the torso of the character, but with unknown timing and direction, such as gusts of wind. Other quantities are assumed known: the generation of internal joint torques is deterministic (4.2), there are no user inputs $\mathbf{c}$, and the start state $\mathbf{s}_1$ is optimized along with the control parameters.

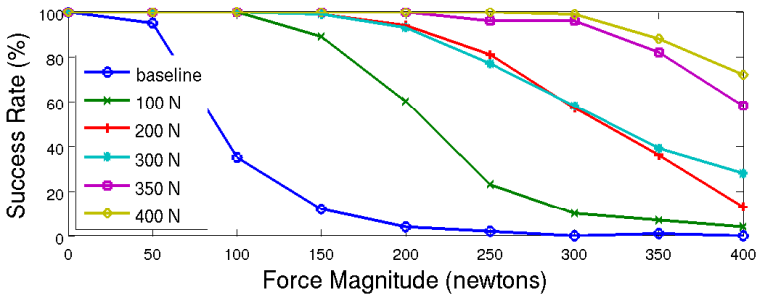


Figure 4.3: Controller success rate vs. F, averaged over 100 trials at each magnitude F. Controllers optimized with larger forces (see legend) are more robust to a wider range of pushes.

Walking under random pushes to the chest. Consider a baseline controller optimized for walking in a fixed direction, efficiently and with a human-like speed to step-length ratio (Chapter 3). While such controllers produce gaits that appear loose and relaxed (e.g., Figure 4.1a), they are not particularly robust. Strong pushes to the chest can easily make the character fall. Conversely, controllers optimized under uncertainty, where $p(\mathbf{e})$ represents random pushes to the chest, are significantly more robust. Further, the robustness of the resulting controllers increases with the magnitude of the pushes during optimization (see Figure 4.3).

To model pushes to the chest, forces of magnitude F newtons from random directions parallel to the ground are applied to the torso COM. During optimization and testing, they are applied with uniformly distributed directions over $(0, 2\pi]$. At each simulation time instant, a force lasting 0.4 s is initiated with probability 0.025% (approximately 6 pushes in a 10 s simulation). Controllers were optimized in sequence, with those for smaller force magnitudes F used to initialize optimizations for larger F. We learned controllers with $F = 100, 200, 300, 350$, and 400, all of which differ significantly from the baseline controller. The knee swings are less passive, making each step appear more deliberate. The arm swings are more pronounced, making gaits appear more energetic. At $F = 200$, the upper body leans forward slightly, resulting in a fast gait (e.g., see Figure 4.1b). At $F = 400$, the upper body is bent almost parallel to the ground, lowering the average COM from 1.02 m (baseline) to 0.96 m. We also find that, as F increases, the average squared torque over time E_{power} (3.13), grows quickly from 34671 (baseline) to 194847 ($F = 400$).

Figure 4.3 shows the success rate for different controllers under different force magnitudes during simulation, where a simulation trial is deemed successful if the character does not fall within 10 s. Controllers optimized for larger forces are clearly more robust. Table 4.1

direction	0 N (baseline)	100 N	200 N	300 N	350 N	400 N
$(1, 0)$	125	175	275	375	350	425
$(1, \pm 1)$	75	125	225	275	300	325
$(0, \pm 1)$	75	125	325	375	425	475
$(-1, \pm 1)$	25	100	175	225	250	275
$(-1, 0)$	75	200	350	325	375	375

Table 4.1: Maximum force (in newtons) tolerance for controllers pushed in different directions. Each column represents a controller trained for a particular F.

reports the maximum force tolerated by each controller from 8 directions. Following Chapter 3, we apply pushing forces to the torso once every 4 s. A controller succeeds if the character does not fall within 40 s.

Walking on a narrow beam. When walking on a narrow beam high above the ground (e.g., see Figure 4.1d), the consequence of even a slight misstep can be catastrophic. If the environment is deterministic and known, one's gait on the beam might not differ from that used on the ground plane. In the presence of uncertain forces applied to the body, however, one must be more conservative to avoid taking a bad step and falling. To demonstrate this, we first optimized a baseline controller to walk on a narrow beam that is 0.5 m in width. We enable collision detection between body parts here, so that gaits with legs passing through each other are not possible. As expected, the resulting controller is like that for walking on the ground plane, except that the width of its gait is less than 0.5 m.

We then apply random forces to the torso, like those in the previous experiment, but with $F = 30$. Under pushes of this magnitude, the baseline controller quickly takes a wrong step and falls off the beam. Unlike the ground plane, where extending the width of a step in the sagittal or coronal direction can prevent falling even with large forces, here, just a light push is enough to cause the character to fall. Nevertheless, a successful controller for this environment can be learned through optimization. As depicted in Figure 4.1d, the resulting controller takes smaller, more deliberate steps, and keeps the feet closer to center of the beam. The average step length decreased from 0.82 m (baseline) to 0.53 m, and E_{power} increased from 44998 to 100657.

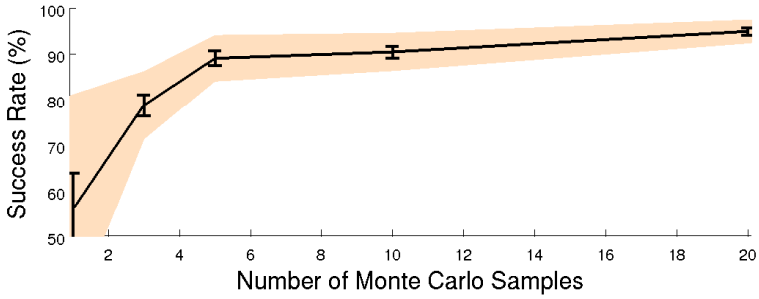


Figure 4.4: Effect of N on controller robustness. The curve shows the mean success rate for 10 controllers, at each value of N, with standard error bars. The orange region depicts the sample standard deviation for 10 controllers.

4.4.2 Interactive user control

The control parameterization allows a user to specify heading (Section 4.3.1) and hence the walking direction. Nevertheless, the baseline straight-walking controller does not handle changes in heading successfully. To improve this, we view user input as a source of uncertainty, and optimize a controller to cope with random changes in desired heading direction. The optimized controller is slower (1.1 m/s) than the baseline controller optimized without turning (1.6 m/s). Unlike our results in the pushing experiment, this controller has the upper body leaning back slightly. We created an interface where the user changes the desired heading direction at will using the keyboard, 0.5 radians at a time. When used with the baseline controller, the character falls frequently. When optimized with random orientation changes, however, the user can easily learn to interactively navigate the 2D plane without the character falling.

We also use this task to examine the effect of N on the optimized solutions. We optimized 10 controllers with stochastic heading changes for each of $N = 1, 3, 5, 10, 20$. To model $p(\mathbf{c})$, heading changes occur at each time instant with probability 0.05%, each of which is drawn from $\mathcal{N}(0, 0.5)$, a mean-zero Gaussian density with a standard deviation of 0.5 radians. To assess controller performance we use the fraction of 100 simulations that do not fall within 10 s. The results in Figure 4.4 show that for small values of N there is higher sampling variability, and hence less robustness on average. As N increases, the average performance increases, as does the reliability of the controllers. Around $N = 10$ the marginal gain in controller performance decreases significantly compared to the added computational expense during optimization for this task.

4.4.3 Motor noise

We now consider the effects of motor noise together with different environments and return functions. In a deterministic setting, control optimization may succeed at challenging tasks with relative ease, even if there is little margin for error. In the presence of randomness, however, controllers must become more careful.

Biological neural control systems exhibit noise. This seemingly random variability is found in the measurement of many biological quantities, even in highly repetitive tests [21]. For example, all neurons, including motor neurons, exhibit variability in their output potentials even when the same stimuli are presented on repeated trials. Such neural noise is often signal dependent. In the motor system, larger control signals exhibit greater noise; i.e., in motor neurons that control muscle activation, the standard deviation scales in proportion to firing rates. There is strong evidence that motor noise plays a major role in determining human motor control strategies [33]. Motor noise may also provide robustness under a wide range of otherwise unmodeled phenomena, including numerical errors in computer simulation.

We employ a simplified model in which motor noise affects the joint torques produced by the controller. Motivated by neural noise, we assume the standard deviation of the noise increases with torque, and decreases with the strength of the joint (i.e., the maximum torque that can be generated at that joint). The particular form of our model is a modified version of that developed by Hamilton et al. [30]. For each joint i, given a desired (noiseless) torque of the controller $\bar{\tau}_i$, the noisy torque τ_i is drawn from a Gaussian density with mean $\bar{\tau}_i$ and standard deviation $\sigma(\bar{\tau}_i)$:

$$\tau_i \sim \mathcal{N}(\bar{\tau}_i; \sigma(\bar{\tau}_i)) \tag{4.14}$$

$$\sigma(\bar{\tau}_i) = \bar{\tau}_i \beta \exp(-2.76) MVT^{-0.25} \ , \tag{4.15}$$

where MVT is the maximum voluntary torque output at the particular joint and β is a scale factor that allows one to adjust the noise level. For each joint DOF, we use $MVT = k_p$, the spring stiffness constant of the PD-controller at that joint DOF.

Walking on a slippery surface. Walking on a surface with a low coefficient-of-friction μ requires caution. When optimizing a baseline controller for walking in a straight line

on a surface with $\mu = 0.4$, with no noise or uncertainty in the environment, we obtain a controller that is somewhat more cautious, taking smaller steps than a comparable controller trained with greater friction. Nevertheless, more pronounced differences are evident when controllers are optimized with uncertainty due to motor noise. We optimized three controllers, each with a different amount of motor noise, by varying the scale factor in (4.15), i.e., $\beta = 50, 75, 100$. We find that, for high noise levels, the character raises his arms wide in the coronal plane and lowers his center of gravity, producing a gait much like that of a person treading carefully on ice (e.g., see Figure 4.1c).

Carrying hot beverages. When designing a controller to carry a mug of hot coffee, we want to ensure the character will not spill the coffee. Accordingly, the controller receives large penalties (negative rewards) at every time-step for which the orientation of the mug deviates too far from vertical. Figure 4.5a shows one pose of the baseline controller where the arms are not particularly stiff and the step length is relatively long. By comparison, a corresponding pose of the controller optimized with $\beta = 100$ shows a shorter step length with arms that remain steady and level. Figure 4.5b plots the mug tilt as a function of time. The baseline controller walks in a relaxed fashion, but allows the mug orientation to reach the spillage threshold. Even small disturbances will therefore cause coffee to spill. In contrast, the strategy optimized with motor noise creates a margin for safety below the threshold.

To model this task, let $\mathbf{\Phi}_t = [\phi_{cor}, \phi_{sag}, \phi_{trans}]$ represent the orientation of a hand in the world frame at simulation step t, such that $\mathbf{\Phi}_t = [0, 0, 0]$ corresponds to a level, upright mug. Suppose that spills occur whenever $m(t) = \sqrt{{\phi_{cor}}^2 + {\phi_{sag}}^2} > 0.1$. Note that this model ignores acceleration of the mug, which could also be included for realism. For the baseline controller, optimized in a deterministic setting, the mug orientation quickly exceeds the 0.1 threshold, even for small amounts of motor noise (e.g., with $\beta = 25$). For a controller optimized with $\beta = 50$, the walking speed slows, and the cup orientation is more stable. For $\beta = 100$, the gait appears cautious and hip-driven (63% of the lower body power output is from the hip, compared to 43% in the baseline and $\beta = 50$ case). The latter two controllers, optimized to handle larger noise levels, are both able to walk without spillage when simulated with $\beta = 25$, unlike the baseline controller.

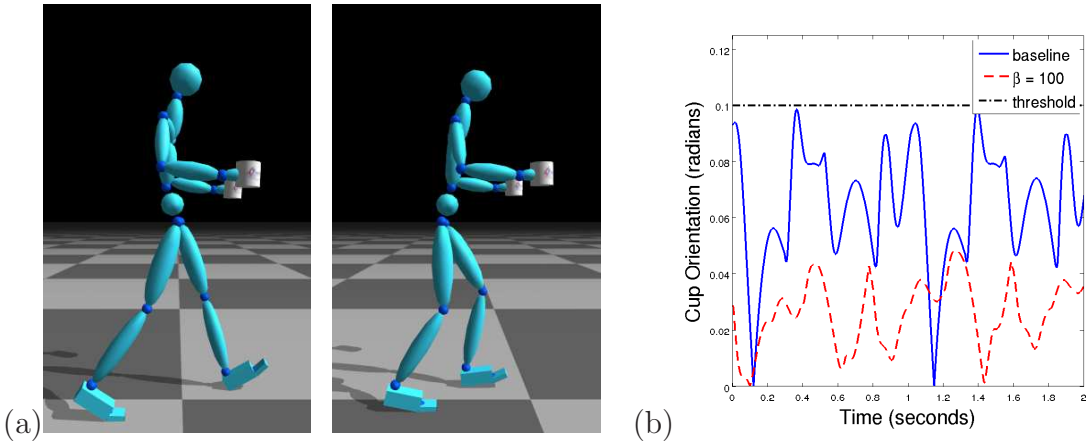


Figure 4.5: Carrying hot beverages. (a) Poses from controllers optimized without and with motor noise ($\beta = 100$). (b) Mug orientation as a function of time from the two controllers, both simulated without motor noise.

4.4.4 Recovery controllers

Optimization with random external perturbations, like that in Section 4.4.1, produces controllers that anticipate external disturbances. For example, they achieve robustness by remaining stiff and keeping the COM relatively low. A complementary approach is to design a reactive controller, where a basic controller π is active under normal circumstances, but a *recovery controller* π_r is invoked when a disturbance is detected. The recovery mechanism helps the character return to a normal gait so that π can be restarted. For example, one might walk with a relaxed gait until pushed, at which point the recovery controller takes over until the basic controller with the relaxed gait can resume control.

Ideally, π_r would bring the character to states with high expected return with respect to the basic controller π, but this is costly to evaluate. Instead, we aim for states that are typical of π. We define several key features $\hat{\mathbf{s}}$ of a character's state, and then estimate $p(\hat{\mathbf{s}})$, the distribution of features observed during normal walking under π. Here, $\hat{\mathbf{s}}$ comprises the horizontal distance from the stance ankle to the COM, and the COM velocity, projected into both the sagittal and coronal planes. Finally, we combine π and π_r to form a new reactive controller π_{new} as follows:

$$\pi_{new}(\mathbf{s}) = \begin{cases} \pi(\mathbf{s}) & p(\hat{\mathbf{s}}) > \kappa \\ \pi_r(\mathbf{s}) & \text{otherwise}\,, \end{cases} \tag{4.16}$$

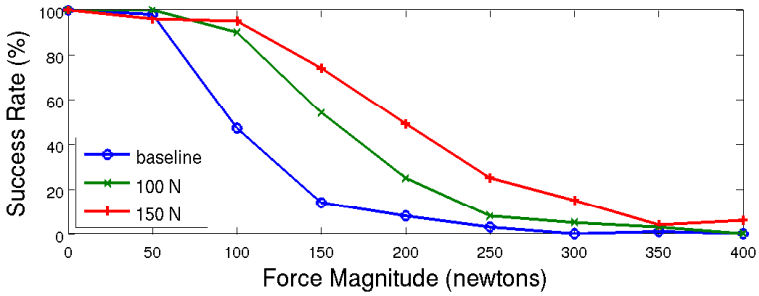


Figure 4.6: Recovery controllers optimized to return to the baseline controller (Section 4.4.4) for larger pushes show greater robustness. Success rate is estimated from 100 random trials.

where κ is a threshold. This controller runs π when the input is in π's typical states, and runs π_r otherwise.

For the recovery task we begin with a controller π optimized to walk comfortably at 1.6 m/s, with step-length 0.8 m. We generate motions of duration 100 s from π with random heading changes (see Section 4.4.2) that occur with probability 0.025% at each time-step, drawn from $\mathcal{N}(0, 0.3)$. We then fit an axis-aligned Gaussian to these motion features to approximate $p(\hat{\mathbf{s}})$.

The goal of the controller π_{new} is to walk using π where possible, using π_r to return the character to states with high $p(\hat{\mathbf{s}})$ (i.e., typical states from π). We model random external forces using random pushes to the torso (as in Section 4.4.1), of 100 N and 150 N. The optimization variables are κ and the parameters of π_r, where π_r is initialized to π, κ is initialized to e^{-7}. The reward for the optimization penalizes time-steps not spent in the basic controller, and heavily penalizes falling:

$$r(\mathbf{s}_t) = -0.01(recover_t) - 100(failed_t)\,, \tag{4.17}$$

where $recover_t$ is 1 if $p(\hat{\mathbf{s}}_t) < \kappa$, and 0 otherwise, and $failed_t$ is defined in the Appendix.

Following pushes, the recovery controllers successfully return the character to states with $p(\hat{\mathbf{s}}) > \kappa$ after a few steps at most, thus re-activating π. Adding the recovery controller improves robustness (Figure 4.6), to a degree comparable to directly optimizing the basic controller, as in Section 4.4.1. However, direct optimization still provides larger improvements in robustness, since the entire walking style is allowed to be modified.

direction	baseline	rec. 100 N	rec.150 N
$(1, 0)$	50	125	200
$(1, \pm 1)$	50	75	175
$(0, \pm 1)$	150	175	175
$(-1, \pm 1)$	25	50	125
$(-1, 0)$	75	125	125

Table 4.2: Maximum disturbance force components (in newtons) tolerance for recovery controllers pushed in different directions.

4.4.5 Transition between speeds

The ability to compose controllers is essential for characters to perform interesting activities in sequence. Previous work either assumes the existence of low-level controllers to or from which one can reliably transition, or has focused on identifying specific states where it is safe to switch [16, 22]. In general, between very different controllers (e.g., walking fast and slow), finding reliable switching states is difficult. We can however exploit the recovery controllers (4.16) above to facilitate such transitions.

More concretely, to facilitate transitions from controller π_A to controller π_B, we define a new controller

$$\pi_{AB}(\mathbf{s}) = \begin{cases} \pi_B(\mathbf{s}) & p_B(\hat{\mathbf{s}}) > \kappa_{AB} \\ \pi_{r,B}(\mathbf{s}) & \text{otherwise}\,, \end{cases} \tag{4.18}$$

where $p_B(\hat{\mathbf{s}})$ characterizes the key features of motions produced by π_B, and κ_{AB} is a threshold. When a command to switch from π_A to π_B is received, π_{AB} is activated. Since the states produced by π_A are not necessarily typical of π_B, i.e., $p_B(\hat{\mathbf{s}}) < \kappa_{AB}$, the transition will usually activate $\pi_{r,B}$ directly. The recovery controller $\pi_{r,B}$ will then be active until control under π_B can begin. To determine π_{AB}, like the recovery controller in (4.16), we optimize κ_{AB} and the parameters of $\pi_{r,B}$ to transition the character from a start state produced by π_A, to a state where π_B may be activated. To generate a range of possible start states for transitions, we simulate motions from π_A along with random switching times.

We demonstrate transition controllers to change speed while walking. First, we learned controllers for walking at 0.8, 1.6 and 2.4 m/s, and modeled their typical states with $p_B(\hat{\mathbf{s}})$ as above. For optimization and testing, we run each simulation for 7 s, with switching times drawn from a uniform distribution between 1 and 2 s. A transition is deemed successful if the character is still walking after 7 s. Unlike previous experiments, we use

CRN with $N = 20$.

Without transition controllers, transitions from high to low speeds fail frequently. Out of 500 random trials, switching from 2.4 m/s to 1.6 m/s failed 203 times (40.6%), while switching from 2.4 m/s to 0.8 m/s failed 238 times (47.6%). The number of failures is reduced dramatically by the transition controllers, down to 7 (1.4%) and 16 (3.2%), respectively. Solely in terms of failure rates, transition controllers do not make much difference in other cases. For example, the failure rates for 0.8 m/s to 2.4 m/s and 1.6 m/s to 2.4 m/s were lowered from 5.4% to 2.2% and from 6.2% to 5.8%. However, they often still served to bring the character into a stable state more quickly and gracefully than direct switching, resulting in smoother transitions. Comparisons are included in the supplemental video.

4.4.6 Composing many controllers at run-time

The ability to transition from one controller to another facilitates the implementation of a character that can switch control strategies on demand. In the supplemental video, we demonstrate a character switching between several walking controllers, each optimized using methods described for fast speeds, slow speeds, a slippery surface, recovery, turning, and the high beam. Most of the switching did not require explicit transition controllers, since the controllers optimized with uncertainty are often robust enough as they are. All switching was determined by user commands within a single interactive session.

4.5 Discussion

We have presented a unified framework that captures many sources of uncertainty in a single optimization process. Our work shows the value of explicitly representing uncertainty in character controllers: control strategies automatically adapt to specific sources of randomness, making them more robust and composable, while creating natural stylistic variations. A main limitation of our method is that the quality of results still falls short of kinematic methods, and it could be argued that some of our adaptations appear unusual. While we have tested with a simple four-state PD controller, we believe these

observations are general and should be useful for more sophisticated control parameterizations as well, but optimization may also become more difficult.

There remain other sources of uncertainty that could be handled with our model; each of these could lead to new forms of robustness and control strategies. Perhaps most significant is perceptual uncertainty [43], namely, the incomplete picture of the world we get from our senses. For example, visual estimation of depth and motion is inherently noisy, a crucial fact for anyone attempting to catch or avoid fast-moving objects. Other sources of uncertainty include proprioceptive error (e.g., pose uncertainty), ground roughness [12], and the behaviors of other agents.

Incorporating uncertainty into other approaches to optimization of character control should be straightforward. For example, many optimal control formulations used in recent animation research — including the Bellman equations, policy iteration, and the linear-quadratic regulator — can be formulated with probabilistic dynamics. However, restrictive dynamics and uncertainty models (e.g., linear dynamics and Gaussian noise) are normally required for optimal closed-form solutions in continuous models.

Part II

Gaussian Process Models for Human Motion

Chapter 5

Data-driven Prior Models of Human Motion

The idea of introducing additional constraints for motion synthesis can be made more general when discussed in terms of probabilistic models. In computer vision, where the use of Bayesian methods is commonplace, the problem of recovering 3D motion from 2D video observations can be thought of as modeling

$$p(motion|observation) \ , \tag{5.1}$$

and identifying (at least) the most likely motion for a given set of observations. In the animation context, the animator constraints constitute the observations. For a given set of constraints, the animation system generates a corresponding motion[1], which effectively specifies a mapping from constraint space to motion space.

More specifically, *motion* here consist of sequences of 3D pose configurations. If *observation* is available for each pose in the *motion*, which is often the case for computer vision, the main modeling task actually lies in the mapping from observations to poses. Strictly speaking, here the role of motion (relations between different poses) is potentially very limited. We will use *motion* in a broad sense, including both individual pose configurations as well as motions obtained from concatenating them.

[1]This motion trajectory could be thought of as a peak in the distribution $p(motion|observation)$.

Discriminative methods approach this problem directly, either by modeling the distribution or by specifying a mapping from observations to the pose/motion space as discussed before. However, this mapping could be highly nonlinear or even one-to-many, hence difficult to construct. On the other hand, the generative approach models (5.1) by relating it to $p(observation, motion)$, the joint distribution between observation and motion, via Bayes' rule:

$$p(motion|observation) \propto p(observation, motion) \tag{5.2}$$

$$= p(observation|motion)p(motion) \ . \tag{5.3}$$

In 3D tracking, $p(observation|motion)$ is called the likelihood model, which represents the mapping from 3D poses to 2D images. Fundamentally, given the camera configurations, this mapping is well understood as the rendering problem. Correspondingly, in animation, the mapping from 3D pose to animator handles such as end-effectors is also well defined. The challenge is mainly to model $p(motion)$, the prior belief on which 3D motion trajectories are likely. This term is often referred to as the motion model.

In the generative approach, given a set of observations or constraints, (5.3) can be evaluated with respect to a hypothesized motion. Some optimization algorithm will then be used to find the hypothesis that maximizes (5.3). Conceptually, this amount to finding a motion that simultaneously generates the observations (or satisfies user constraints) and is deemed likely by the motion model. In general, the optimization is expensive, prone to local minima, and needs to be done for each new observation. However, the motion model is independent from the observation model, allowing the problem of characterizing likely motions to be treated separately from the problem of selecting proper animator constraints. In the rest of the thesis, we will focus on the generative approach, and methods to model $p(motion)$ from motion capture (mocap) data.

Regardless of the specific application, the main purpose of motion models is to constrain the space of motions being considered. Much like the space of natural images is much smaller than the space of all possible combinations of pixel colours, the space of natural motions is also small relative to all possible motion trajectories. Two main aspects of constraining the space are characterizing the space of likely poses and characterizing the likely combinations between poses (e.g., smoothness constraints, physical plausibility).

The former class of models is more accurately called pose models, while the later motion models. As alluded to above, most models contain aspects of both, and we will refer to both as motion models.

5.1 Motion databases

One simple way to construct such a motion model is by keeping a database of valid motions, synthesized by other means, and restricts the search of solutions to motions in the database. This idea was first explored on simple 2D figures (Luxo lamp) [47], where a motion database of physically generated hops can be adapted to varying terrains. The motion model allows for hops to be concatenated as long as the final state of one hop is sufficiently close to the initial state of the next. Although the Luxo lamp is simple and low dimensional, the idea of transitioning between nearby poses became a common theme of more recent works on human motion models based on mocap data.

Mocap data typically come as a collection of vectors representing the pose configuration in a single frame. Natural looking motion trajectories can be synthesized by concatenating pose vectors that appear natural when placed next to each other, which is usually true when the vectors are numerically close. Given that similar poses can appear in multiple captured motions, it is possible to generate trajectories that make use of poses from different motion clips. This can be done by first constructing *motion graphs* [5, 44, 52] from data, which constructs a graph structure where nodes correspond to poses and edges correspond to allowable transitions. For any reasonably rich database, however, the graph could easily get extremely large. The motion graphs can be compressed by merging nodes from the same sequence [5, 44], or through cluster trees [52]. Dynamical programming techniques can then be applied on the graph to search for trajectories that satisfy user constraints such as position and direction. The structure of the graph can also be manually designed [24], allowing for better control over the transitions. Higher level controls such as run, walk, or jump can be composed and synthesized using annotated motion graphs [6].

A major challenge of using stock motion clips for animation is how they could be made more flexible to user constraints. The use of motion graphs allows for transitions in-between clips, which alleviates the problem somewhat. However, fundamentally if a

certain pose does not exist in the database, it cannot be satisfied as a user constraint. There have also been interests in combining interpolation and optimization with motion graphs [93, 97], which expands the range of poses that could be synthesized.

5.2 Statistical models

As discussed previously, it is difficult to constrain motions using a mocap database because we cannot hope to capture all possible variations of human motion. For animation, this means specific user constraints may not be always satisfied, and the model would only be suitable for applications that do not require fine control (e.g., games). For tracking, this limitation is even less appealing as trackers are often asked to track new individuals performing activities in new styles.

Of course, the need to build motion models that are capable of making predictions beyond observed data is not unique to animation and tracking. Indeed, the analysis of sequential data has been studied extensively in fields ranging from control engineering to economics. In computer science, the problem of learning from collections of data and generalizing from them is the central problem in machine learning. The use of statistical modeling is crucial to virtually all these applications. Although not exclusively, they have also been applied to building motion models.

One common approach is to learn a probability distribution over the space of possible poses and motions, parameterized by the joint angles of the body, as well as its global position and orientation. This corresponds to explicitly modeling $p(motion)$, mentioned in the start of this chapter. Such a density function provides a natural measure of plausibility, assigning higher probabilities to motions that are similar to the training data. The task is challenging due to the high-dimensionality of human pose data, and to the complexity of the motion. However, poses from specific activities often lie near a nonlinear manifold with much lower dimensionality than the number of joint angles. Motivated by this property, a common approach to define the generative model is to decouple the modeling of pose and motion. The motion is modeled by a dynamical process defined on a lower-dimensional latent space, and the poses are generated by an observation process from the latent space.

The current literature offers a number of generative models where the dynamics is not directly observed. Simple models such as linear dynamical systems (LDS) are efficient and easily learned, but are limited in their expressiveness for complex motions. Though they could still be useful as a weak prior for motion synthesis, give sufficient user constraints [13]. More expressive models, such as switching LDS [81], nonlinear dynamical systems (NLDS) [92], and restricted Boltzmann machines [105] are more difficult to learn, requiring many parameters that need to be hand-tuned and large amounts of training data.

Using hidden Markov models as a starting point, additional parameters associated with states or hierarchies can be introduced to model stylistic variations in motion [8,80], such as different dancers performing the same score in different styles. Impressive results in motion synthesis can be achieved with these models, but the learning process is complex and typically requires manually specifying parameters such as the number of hidden states and styles.

Alternatively, LDS can be used as a building block to more powerful models as well. Given a clip of human motion, it is almost always the case that a single LDS is insufficient to model its complexity. However, one would identify segments of the motion that can be well modeled with LDS. A two-level model can be learned in this fashion [58], and good synthesis results can be achieved. Closely related to this approach is switching LDS [81], which models each pose as a mixture of LDS, and have been applied to tracking and classification of motion.

Manifold learning techniques can be used to explicitly recover the low dimensional structure of the poses, which can make tasks such as density estimation (characterizing likely poses) easier. However, a mapping from the low dimensional representation to the pose space has to be learned separately [100]. Although the mappings between latent space and pose space, as well as the dynamics, are in general nonlinear. Locally linear approximations such as mixtures of factor analyzers [55, 56] can be used to approximate these mappings.

The statistical models discussed so far are all parametric models, which only uses the motion database as training data to tune the model parameters. These models allow the evaluation of motions outside of the database (i.e., given a new motion, the model can say something about how likely it is relative to training data). However, when used

for motion synthesis, they often have the effect of "smoothing" the motion, discarding important high frequency information, and leading to artifacts such as foot-skating.

More recently, a number of projects in graphics and vision made use of Gaussian process latent variable models (GPLVM) [49]. They were initially used for estimating the density of likely poses (but not motion) for the problem of inverse kinematics [27] and tracking [116]. In previous work, we extended the GPLVM to a dynamical model [127], which has lead to promising tracking and animation results [69, 113, 135]. Other extensions to the GPLVM include the addition of hierarchies [50], multifactor models (Chapter 6), and topological constraints [115]. Unlike most statistical models, Gaussian process models do not throw away training data after the learning process; instead, they are used directly to evaluate new motions. Consequently, good synthesis results can be obtained at the cost of a more expensive likelihood computation.

5.3 Gaussian process dynamical models

In the rest of this chapter, we describe Gaussian process dynamical models (GPDM) for nonlinear time series analysis, which is a specific approach to model $p(motion)$. We provide a summary of the work here as background for Chapter 6, interested readers are referred to our previous article [127] for a more complete treatment. A GPDM is a latent variable model. It comprises a low-dimensional latent space with associated dynamics, and a map from the latent space to an observation space. We marginalize out the model parameters in closed-form, using Gaussian process priors for both the dynamics and the observation mappings. This results in a non-parametric model for dynamical systems that accounts for uncertainty in the model.

5.3.1 Introduction

The GPDM is a Bayesian approach to learning NLDS, averaging over model parameters rather than estimating them. Inspired by the fact that averaging over nonlinear regression models leads to a Gaussian process regression model, we show that integrating over NLDS parameters can also be performed in closed-form. The resulting model is fully defined by a set of low-dimensional representations of the training data, with both observation and

dynamics processes learned from Gaussian process regression. As a natural consequence of Gaussian process regression, the GPDM removes the need to select many parameters associated with function approximators while retaining the power of nonlinear dynamics and observation.

As mentioned previously, this approach is directly inspired by the GPLVM [49]. The GPLVM models the joint distribution of the observed data and their corresponding representation in a low-dimensional latent space. It is not, however, a dynamical model; rather, it assumes that data are generated independently, ignoring temporal structure of the input. Here we augment the GPLVM with a latent dynamical model, which gives a closed-form expression for the joint distribution of the observed sequences and their latent space representations. The incorporation of dynamics not only enables predictions to be made about future data, but also helps to regularize the latent space for modeling temporal data in general [85].

The unknowns in the GPDM consist of latent trajectories and hyperparameters. Generally, if the dynamics process defined by the latent trajectories is smooth, then the models tend to make good predictions. We discuss a maximum *a posteriori* (MAP) algorithm for estimating all unknowns, and discuss cases where it fails to learn smooth trajectories.

5.3.2 Model formulation

The GPDM comprises a generative mapping from a latent space $\mathbf{x}$ to the observation space $\mathbf{y}$, and a dynamical model in the latent space (Figure 5.1). These mappings are in general nonlinear. For human motion modeling, a vector $\mathbf{y}$ in the observation space corresponds to a pose configuration, and a sequence of poses defines a motion trajectory. The latent dynamical model accounts for the temporal dependence between poses. The GPDM is obtained by marginalizing out the parameters of the two mappings, and optimizing the latent coordinates of training data.

More precisely, our goal is to model the probability density of a sequence of vector-valued states $\mathbf{y}_1, \ldots, \mathbf{y}_t, \ldots, \mathbf{y}_N$, with discrete-time index t and $\mathbf{y}_t \in \mathbb{R}^D$. As a basic

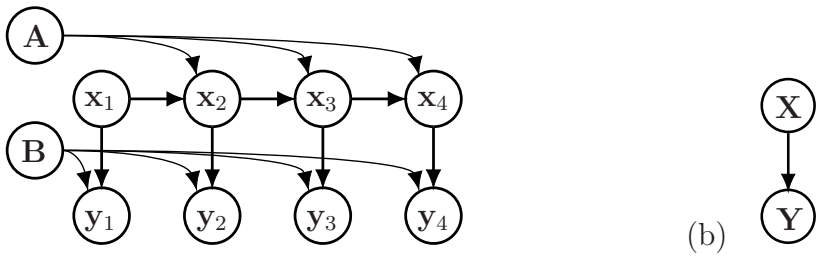


Figure 5.1: Time-series graphical models. (a) Nonlinear latent-variable model for time series. (Hyperparameters $\bar{\alpha}$, $\bar{\beta}$ and $\mathbf{W}$ are not shown.) (b) GPDM model. Because the mapping parameters $\mathbf{A}$ and $\mathbf{B}$ have been marginalized over, all latent coordinates $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^T$ are jointly correlated, as are all poses $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N]^T$.

model, consider a latent variable mapping (5.5) with first-order Markov dynamics (5.4):

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}; \mathbf{A}) + \mathbf{n}_{x,t} \tag{5.4}$$

$$\mathbf{y}_t = g(\mathbf{x}_t; \mathbf{B}) + \mathbf{n}_{y,t} \ . \tag{5.5}$$

Here, $\mathbf{x}_t \in \mathbb{R}^d$ denotes the d-dimensional latent coordinates at time t, f and g are mappings parameterized by $\mathbf{A}$ and $\mathbf{B}$, $\mathbf{n}_{x,t}$ and $\mathbf{n}_{y,t}$ are zero-mean, isotropic, white Gaussian noise processes. Figure 5.1a depicts the graphical model.

While linear mappings have been used extensively in auto-regressive models, here we consider the more general nonlinear case for which f and g are linear combinations of (nonlinear) basis functions:

$$f(\mathbf{x}; \mathbf{A}) = \sum_i \mathbf{a}_i \ \phi_i(\mathbf{x}) \tag{5.6}$$

$$g(\mathbf{x}; \mathbf{B}) = \sum_j \mathbf{b}_j \ \psi_j(\mathbf{x}) \ , \tag{5.7}$$

for basis functions ϕ_i and ψ_j, with weights $\mathbf{A} \equiv [\mathbf{a}_1, \mathbf{a}_2, \ldots]^T$ and $\mathbf{B} \equiv [\mathbf{b}_1, \mathbf{b}_2, \ldots]^T$. To fit this model to training data, one must select an appropriate number of basis functions, and one must ensure that there is enough data to constrain the shape of each basis function. After the basis functions are chosen, one might estimate the model parameters, $\mathbf{A}$ and $\mathbf{B}$, usually with an approximate form of expectation-maximization [92]. From a Bayesian perspective, however, the uncertainty in the model parameters is significant, and because the specific forms of f and g are incidental; the parameters should be marginalized out if possible. Indeed, in contrast with previous NLDS models, the general approach we

take in the GPDM is to estimate the latent coordinates while marginalizing over model parameters.

Each dimension of the latent mapping, g in (5.7), is a linear function of the columns of $\mathbf{B}$. Therefore, with an isotropic Gaussian prior on the columns of $\mathbf{B}$, and the Gaussian noise assumption above, one can show that marginalizing over g can be done in closed form [66, 73]. In doing so we obtain a Gaussian density over the observations, $\mathbf{Y} \equiv [\mathbf{y}_1, \ldots, \mathbf{y}_N]^T$, which can be expressed as a product of Gaussian processes (one for each of the D data dimensions):

$$p(\mathbf{Y} \,|\, \mathbf{X}, \bar{\beta}, \mathbf{W}) = \frac{|\mathbf{W}|^N}{\sqrt{(2\pi)^{ND}|\mathbf{K}_Y|^D}} \exp\left(-\frac{1}{2}\mathrm{tr}\left(\mathbf{K}_Y^{-1}\mathbf{Y}\mathbf{W}^2\mathbf{Y}^T\right)\right) \ , \tag{5.8}$$

where $\mathbf{K}_Y$ is a kernel matrix with hyperparameters $\bar{\beta}$ that are shared by all observation space dimensions, and hyperparameters $\mathbf{W}$. The elements of the kernel matrix, $\mathbf{K}_Y$, are defined by a kernel function, $(\mathbf{K}_Y)_{ij} \equiv k_Y(\mathbf{x}_i, \mathbf{x}_j)$. For the mapping g, we use the radial basis function (RBF) kernel,

$$k_Y(\mathbf{x}, \mathbf{x}') \ = \ \exp\left(-\frac{\beta_1}{2}||\mathbf{x} - \mathbf{x}'||^2\right) + \beta_2^{-1}\delta_{\mathbf{x},\mathbf{x}'} \ . \tag{5.9}$$

The width of the RBF kernel function is controlled by β_1^{-1}, and β_2^{-1} is the variance of the isotropic additive noise in (5.5).

Following Grochow et al. [27], we include D scale parameters, $\mathbf{W} \equiv \mathrm{diag}(w_1, \ldots, w_D)$, which model the variance in each observation dimension. This is important in many data sets for which different dimensions do not share the same length scales, or differ significantly in their variability over time. In effect, this assumes that each dimension of the input data should exert the same influence on the shared kernel hyperparameters, β_1 and β_2.

The dynamic mapping on the latent coordinates $\mathbf{X} \equiv [\mathbf{x}_1, \ldots, \mathbf{x}_N]^T$ is conceptually similar, but subtler. As above, one can form the joint density over the latent coordinates and the dynamics weights, $\mathbf{A}$, in (5.6). Then, one can marginalize over the weights $\mathbf{A}$ to obtain

$$p(\mathbf{X} \,|\, \bar{\alpha}) \ = \ \int p(\mathbf{X} \,|\, \mathbf{A}, \bar{\alpha})\, p(\mathbf{A} \,|\, \bar{\alpha})\, d\mathbf{A} \ , \tag{5.10}$$

where $\bar{\alpha}$ is a vector of kernel hyperparameters. Incorporating the Markov property (5.4)

gives

$$p(\mathbf{X} \mid \bar{\alpha}) = p(\mathbf{x}_1) \int \prod_{t=2}^{N} p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{A}, \bar{\alpha}) p(\mathbf{A} \mid \bar{\alpha})\, d\mathbf{A} \ . \tag{5.11}$$

Finally, with an isotropic Gaussian prior on the columns of $\mathbf{A}$, one can show that (5.11) reduces to

$$p(\mathbf{X} \mid \bar{\alpha}) = \frac{p(\mathbf{x}_1)}{\sqrt{(2\pi)^{(N-1)d} |\mathbf{K}_X|^d}} \exp\left(-\frac{1}{2} \mathrm{tr}\left(\mathbf{K}_X^{-1} \mathbf{X}_{2:N} \mathbf{X}_{2:N}^T \right) \right) \ , \tag{5.12}$$

where $\mathbf{X}_{2:N} = [\mathbf{x}_2, \ldots, \mathbf{x}_N]^T$, and $\mathbf{K}_X$ is the $(N-1) \times (N-1)$ kernel matrix constructed from $\mathbf{X}_{1:N-1} = [\mathbf{x}_1, \ldots, \mathbf{x}_{N-1}]^T$. Below we also assume that $\mathbf{x}_1$ also has a Gaussian prior.

The dynamic kernel matrix has elements defined by a kernel function, $(\mathbf{K}_X)_{ij} \equiv k_X(\mathbf{x}_i, \mathbf{x}_j)$, for which a linear kernel is a natural choice; i.e.,

$$k_X(\mathbf{x}, \mathbf{x}') = \ \alpha_1 \mathbf{x}^T \mathbf{x}' + \alpha_2^{-1} \delta_{\mathbf{x},\mathbf{x}'} \ . \tag{5.13}$$

In this case, (5.12) is the distribution over state trajectories of length N, drawn from a distribution of auto-regressive models with a preference for stability [69]. While a substantial portion of human motion (as well as many other systems) can be well modeled by linear dynamical models, ground contacts introduce nonlinearity [7]. We found that the linear kernel alone is unable to synthesize good walking motions (e.g., see Figure 5.2h-i). Therefore, we typically use a "linear + RBF" kernel:

$$k_X(\mathbf{x}, \mathbf{x}') = \alpha_1 \exp\left(-\frac{\alpha_2}{2} ||\mathbf{x} - \mathbf{x}'||^2 \right) + \alpha_3 \mathbf{x}^T \mathbf{x}' + \alpha_4^{-1} \delta_{\mathbf{x},\mathbf{x}'} \ . \tag{5.14}$$

The additional RBF term enables the GPDM to model nonlinear dynamics, while the linear term allows the system to regress to linear dynamics when predictions are made far from the existing data. Hyperparameters α_1, α_2 represent the output scale and the inverse width of the RBF terms, and α_3 represents the output scale of the linear term. Together, they control the relative weighting between the terms, while α_4^{-1} represents the variance of the noise term $\mathbf{n}_{x,t}$.

It should be noted that, due to the marginalization over $\mathbf{A}$, the joint distribution of the latent coordinates is *not* Gaussian. One can see this in (5.12), where latent variables occur both inside the kernel matrix and outside of it; i.e., the log likelihood is not quadratic in

$\mathbf{x}_t$. Moreover, the distribution over state trajectories in a nonlinear dynamical system is in general non-Gaussian.

Taken together, the latent mapping, and the dynamics define a generative model[2] for time-series observations (Figure 5.1b):

$$p(\mathbf{X}, \mathbf{Y}, \bar{\alpha}, \bar{\beta}, \mathbf{W}) = p(\mathbf{Y} \mid \mathbf{X}, \bar{\beta}, \mathbf{W})\, p(\mathbf{X} \mid \bar{\alpha}) \ . \tag{5.15}$$

5.3.3 GPDM learning

Learning the GPDM from measured data $\mathbf{Y}$ entails using numerical optimization to estimate some or all of the unknowns in the model $\{\mathbf{X}, \bar{\alpha}, \bar{\beta}, \mathbf{W}\}$. A model gives rise to a distribution over new poses and their latent coordinates [127]. We expect modes in this distribution to correspond to motions similar to the training data and their latent coordinates. We find that models with visually smooth latent trajectories $\mathbf{X}$ not only better match our intuitions, but also achieve better quantitative results. However, care must be taken in designing the optimization method, including the objective function itself [127]. We discuss the most intuitive MAP learning algorithm here.

A natural learning algorithm for the GPDM is to minimize the joint negative log-posterior of the unknowns, $-\ln p(\mathbf{X}, \bar{\alpha}, \bar{\beta}, \mathbf{W} \mid \mathbf{Y})$, that is given, up to an additive constant, by

$$\mathcal{L} = \mathcal{L}_{\mathcal{Y}} + \mathcal{L}_{\mathcal{X}} \ , \tag{5.16}$$

where

$$\mathcal{L}_{\mathcal{Y}} = \frac{D}{2} \ln |\mathbf{K}_Y| + \frac{1}{2} \mathrm{tr}\left(\mathbf{K}_Y^{-1} \mathbf{Y} \mathbf{W}^2 \mathbf{Y}^T\right) - N \ln |\mathbf{W}| \tag{5.17}$$

$$\mathcal{L}_{\mathcal{X}} = \frac{d}{2} \ln |\mathbf{K}_X| + \frac{1}{2} \mathrm{tr}\left(\mathbf{K}_X^{-1} \mathbf{X}_{2:N} \mathbf{X}_{2:N}^T\right) + \frac{1}{2} \mathbf{x}_1^T \mathbf{x}_1 \ . \tag{5.18}$$

We alternate between minimizing $\mathcal{L}$ with respect to $\mathbf{W}$ in closed form, and with respect to $\left\{\mathbf{X}, \bar{\alpha}, \bar{\beta}\right\}$ using scaled conjugate gradient (SCG). The latent coordinates are initialized using a subspace projection onto the first d principal directions given by principal components analysis (PCA) applied to mean-subtracted data $\mathbf{Y}$.

[2] We omit a discussion of priors on the hyperparameters $\bar{\alpha}, \bar{\beta}, \mathbf{W}$ here for compactness.

Figure 5.2 shows a GPDM on a 3D latent space, learned using MAP estimation. Each blue point in the latent space corresponds to a training pose in the high-dimensional space. The training data comprised two gait cycles of a person walking. The initial coordinates provided by PCA are shown in Figure 5.2a. Figure 5.2c shows the MAP latent space. Note that the GPDM is significantly smoother than a 3D GPLVM (i.e., without dynamics), shown in Figure 5.2b.

Figure 5.3b shows a GPDM latent space learned from walking data of four different walkers. In contrast to the model learned with a single walker in Figure 5.2, the latent trajectories here are not smooth. There are small clusters of latent positions separated by large jumps in the latent space. While such models produce good reconstructions from latent positions close to the training data, they often produce poor dynamical predictions. For example, the sample trajectories shown in Figure 5.3d do not resemble the training latent trajectories particularly well.

5.3.4 Discussion

Figure 5.2 shows 3D latent models learned from data comprising two walk cycles from a single subject. In all experiments here we use a 3D latent space. Learning with more than three latent dimensions significantly increases the number of latent coordinates to be estimated. Conversely, in two dimensions the latent trajectories often intersect which makes learning difficult. In particular, Gaussian processes are function mappings, providing one prediction for each latent position. Accordingly, learned 2D GPDMs often contain large "jumps" in latent trajectories as the optimization breaks the trajectory to avoid nearby positions requiring inconsistent temporal predictions.

Figure 5.2b shows a 3D GPLVM (i.e., without dynamics) learned from walking data. Note that, without the dynamical model, the latent trajectories are not smooth; there are several locations where consecutive poses in the walking sequence are relatively far apart in the latent space. In contrast, Figure 5.2c shows that the GPDM produces a much smoother configuration of latent positions. Here the GPDM arranges the latent positions roughly in the shape of a saddle.

Figure 5.2d visualizes the variance of the reconstruction as a function of the latent space position. This plot depicts the confidence with which the model reconstructs a pose as a

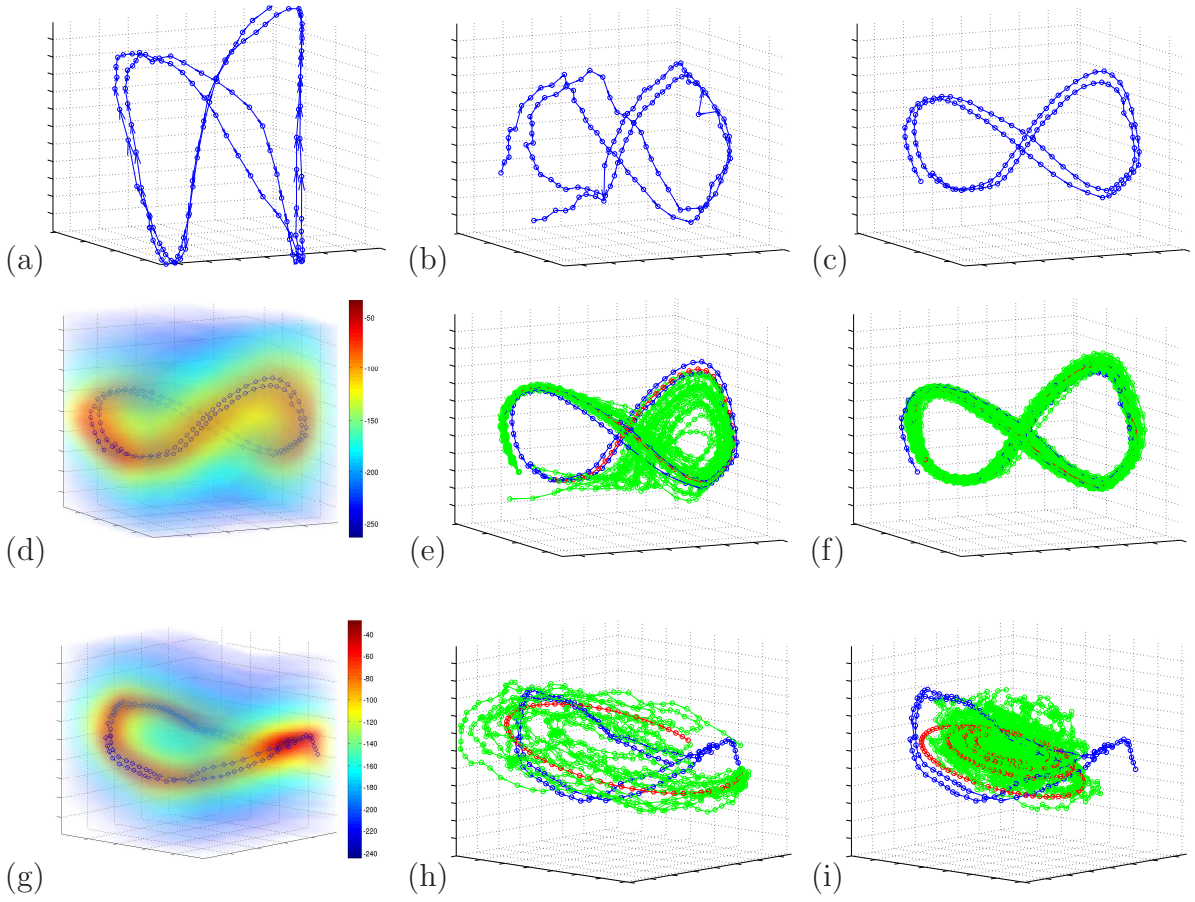


Figure 5.2: Models learned from a walking sequence comprising two gait cycles. The PCA initializations (a), latent coordinates learned with a GPLVM (b) and GPDM (c) are shown in blue. Vectors depict the temporal sequence. (d) $-\ln$ variance for reconstruction shows positions in latent space that are reconstructed with high confidence. (e) Random trajectories drawn from the dynamic predictive distribution using hybrid Monte Carlo are green, the red trajectory is the mean-prediction sample. (f) Longer random trajectories drawn from the dynamics predictive distribution. (g-i) $-\ln$ variance for reconstruction, random trajectories, and longer random trajectories created in the same fashion as (d-f), using a model learned with the linear dynamics kernel. Note that the samples do not follow the training data closely, and longer trajectories are attracted to the origin.

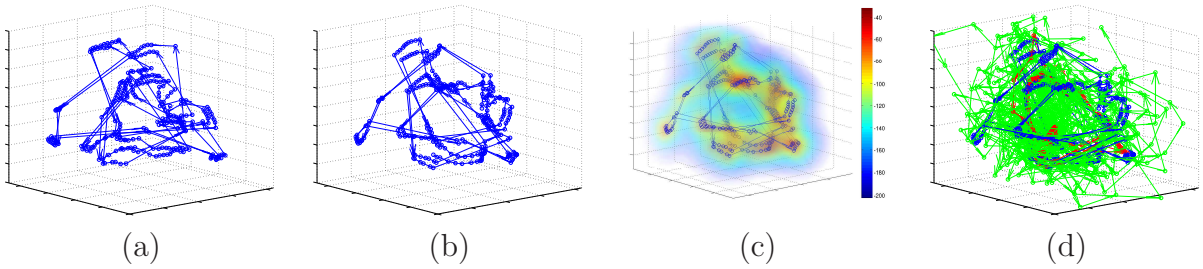


Figure 5.3: Models learned from walking sequences from four different subjects. The latent coordinates learned with a GPLVM (a) and GPDM (b) are shown in blue. (c) $-\ln$ variance plot shows clumpy high confidence regions. (d) Samples from the dynamic predictive distribution are shown in green, while the mean-prediction sample is shown in red. The samples do not stay close to the training data.

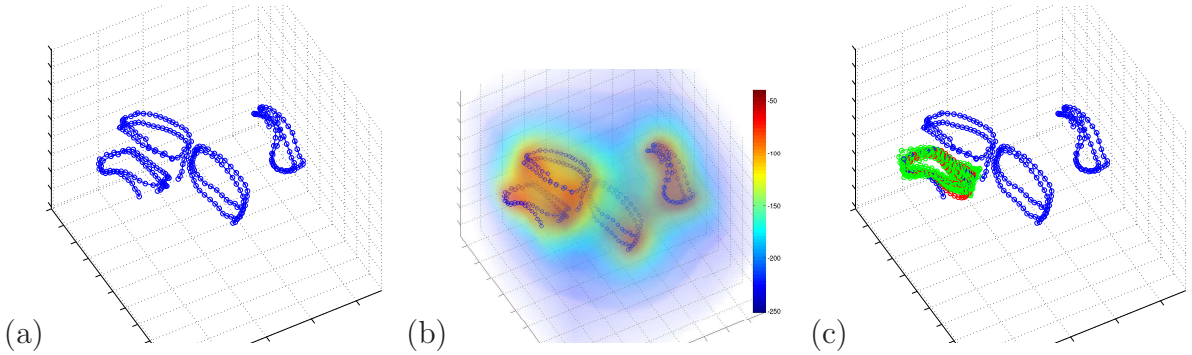


Figure 5.4: Models learned with two-stage MAP from four different walking subjects. (a) The learned latent coordinates shown in blue, note the walkers are separated into distinct portions of the latent space. (b) $-\ln$ variance plot shows smooth high confidence regions, and the variance near data is similar to Figure 5.3c. (c) Typical samples from the dynamic predictive distribution are shown in green, while the mean-prediction sample is shown in red.

function of latent position $\mathbf{x}$. The GPDM reconstructs the pose with high confidence in a "tube" around the region occupied by the training data.

To further illustrate the dynamical process, we can draw samples from the dynamic predictive distribution [127]. These sample trajectories are depicted as red and green trajectories. All samples are conditioned on the same initial state, $\mathbf{x}_1^{(*)}$, and each has a length of 62 time steps (i.e., drawn from $p(\mathbf{X}_{2:62}^{(*)} \,|\, \mathbf{x}_1^{(*)}, \Gamma)$). The length was chosen to be just less than a full gait cycle for ease of visualization. The resulting trajectories are smooth and roughly follow the trajectories of the training sequences. The variance in latent position tends to grow larger when the latent trajectories corresponding to the training data are farther apart, and toward the end of the simulated trajectory.

The bottom row of Figure 5.2 shows a GPDM with only a linear term in the dynamics

kernel (5.14). Here the dynamical model is not as expressive, and there is more process noise. Hence random samples from the dynamics do not follow the training data closely (Figure 5.2h). The longer trajectories in Figure 5.2i are attracted towards the origin.

The MAP learning algorithm produces good models for the single walker and the golf swings data. However, as discussed above, this is not the case with model learned with four walkers (Figure 5.3b). In contrast to the GPDM learned for the single walk data (Figure 5.2); the latent positions for the training poses in the four-walker GPDM consist of small clumps of points connected by large jumps. The regions with high reconstruction certainty are similarly clumped (Figure 5.3c); only in the vicinity of these clumps is pose reconstructed reliably. Also note that the latent positions estimated for the GPDM are very similar to those estimated by the GPLVM on the same dataset (Figure 5.3a). This suggests that the dynamical term in the objective function (5.16) is overwhelmed by the data reconstruction term during learning, and therefore has a negligible impact on the resulting model.

Figure 5.4 shows that it is possible to apply alternative learning algorithms (e.g., two-stage MAP [127]) to obtain smooth trajectories for all walkers. However, the model here places the trajectories far apart in the latent space, and no relations between them are captured. In the next chapter, we address this problem by employing additional information in the training data. Namely, a pose may have associated labels of subject identity, style, gait, or phase of motion. We explore a combination of Gaussian processes and multilinear models for modeling motion data from multiple individuals and styles. In particular, we derive the model by first assuming a Gaussian prior over parameters and then marginalizing over them, in the same fashion as this section.

Chapter 6

A Gaussian Process Extension of Multilinear Models

We introduce models for density estimation with multiple, hidden, continuous *factors*, for use with training data addressed by multiple indices. In particular, we propose a generalization of multilinear models using nonlinear basis functions. By marginalizing over the weights, we obtain a multifactor form of the GPLVM. In this model, each factor is kernelized independently, allowing nonlinear mappings from any particular factor to the data. We learn models for human locomotion data, in which each pose is generated by factors representing the person's identity, gait, and the current state of motion. We demonstrate our approach using time-series prediction, and by synthesizing novel animation from the model.

6.1 Introduction

Using prior models of human motion to constrain the inference of 3D pose sequences is a popular approach to improve monocular people tracking, as well as to simplify the process of character animation. The availability of mocap devices in recent years enables such models to be learned from data, and learning models that generalize well to novel motions has become a major challenge.

One of the main difficulties in this domain is that the training data and test data typically

come from related but distinct distributions. For example, we would often like to learn a prior model of locomotion from the mocap data of a few individuals performing a few gaits (i.e., walking and running). Such a prior model could then be used to track a new individual or to generate plausible animations of a related, but new gait not included in the training database. Due to the natural variations in how different individuals perform different gaits — which we broadly refer to as *style* — learning a model that can represent and generalize to the space of human motions is not straightforward. One approach is to learn a single model from all training data, without regard of our knowledge about their style. However, as seen in Section 5.3.4, this can lead to unrealistic models that either averages together all styles of motion, or else amount to a mixture model of styles. Neither approach can be expected to handle data from new styles well. Nonetheless, it has long been observed that interpolating and extrapolating mocap data yields plausible new motions, and it is reasonable to attempt building motion models that can generalize in style.

This chapter introduces a multifactor model for learning distributions of styles of human motion. We parameterize the space of human motion styles by a small number of low-dimensional *factors*, such as identity and gait, where the dependence on each individual factor may be nonlinear. This parameterization is learned in a semi-supervised manner from a collection of example motions with different styles. Given a new motion, identifying its stylistic factors defines that motion's style-specific distribution.

Our multifactor Gaussian process model can be viewed as a special class of GPLVM [49]. As in the GPLVM, we marginalize out the weights in the generative model, and optimize the latent variables that correspond to the different factors in the model. If used with linear factors, the complete model amounts to a Bayesian generalization of multilinear models [19, 120]. We also incorporate latent-space dynamics, and show that the use of the multifactor model improves time-series prediction results on human motion.

6.1.1 Background

The problem of style-content separation — modeling the interaction of multiple factors — was introduced by Tenenbaum and Freeman [107]. They employed a bilinear model, in which hidden "style" and "content" variables are multiplied along with a set of weights to

produce observations; their algorithm was used to model variations in images of human faces and in typefaces. Identifying which variables correspond to "style" or "content" is problem-dependent, and somewhat arbitrary.

The natural generalization of the bilinear model when more than two factors are present is the multilinear model. Multilinear factorizations have been used to model images of faces [120], 3D face geometry [122], mocap sequences [119], and texture and reflectance [121]. These models are multilinear in the factors, but linear with respect to any single factor. A multifactor generalization of kernel PCA have been proposed [57]. It is complementary to the model proposed here, as they kernelize the outputs while we kernelize the factors.

The main application in this chapter is learning models of human poses and motions. Perhaps the simplest approach to generating motion is to interpolate example poses [91] or mocap sequences [89], assuming that all examples are labeled with style parameters. Independent components analysis can be applied to sequences to obtain a linear style-space of sequences [95].

A few methods for nonlinear style-content separation of human pose and motion also exist. The style machines model [8] learns a linear space of style-specific hidden Markov models for different individuals. This method is limited to two factors, and must represent poses with a discrete state model plus temporal smoothing. Elgammal and Lee [20] learn a nonlinear manifold and a two-factor mapping to pose and silhouette data in a least squares setting.

As discussed in the previous chapter, several researchers have used the GPLVM [49] to model human poses [27, 116]. Given a set of high-dimensional training poses, the GPLVM provides a set of corresponding low-dimensional latent coordinates, along with a Gaussian process mapping from latent coordinates to pose observations. The mapping is in general nonlinear, and gives rise to a joint distribution over new data and the corresponding latent coordinates. The GPDM (Section 5.3) extends the GPLVM by including a dynamical model on the low-dimensional latent space. It thereby models time-series data for a single individual, but does not generalize well to multiple styles or activities. This chapter builds on these two models with the inclusion of factors to represent variation in gait and across individuals.

6.2 Multifactor Gaussian processes

The model we use is a probabilistic latent variable model, involving a low-dimensional latent space of hidden factors describing style and content, and a mapping to a high-dimensional observation space. In this section, we introduce the multifactor Gaussian process (GP) mapping that lies at the core of our approach. We will assume for now that the inputs are known, and only consider one-dimensional outputs. In Section 6.3, we describe how to learn the model in an unsupervised fashion, and apply the model to motion capture data, in which each observation is a high-dimensional human body pose associated with a particular person and a specific gait.

6.2.1 Gaussian processes

We begin by reviewing GP regression, using the "weight-space" view [88]. Suppose we have a one-dimensional function $y = g(\mathbf{x})$ of input vector $\mathbf{x}$, defined as a linear combination of J basis functions $\phi_j(\mathbf{x})$:

$$y = g(\mathbf{x}) = \sum_{j=1}^{J} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \mathbf{\Phi}(\mathbf{x}) \ , \tag{6.1}$$

where the vector $\mathbf{\Phi}(\mathbf{x}) = [\phi_1(\mathbf{x}), ..., \phi_J(\mathbf{x})]^T$ stacks the basis functions. Furthermore, we assume a weight decay prior: $\mathbf{w} \sim \mathcal{N}(0; \mathbf{I})$. Since the outputs y are a linear function of the weights, the outputs are also Gaussian. In particular, given known inputs $\mathbf{x}$ and $\mathbf{x}'$, the mean and covariance of their outputs y and y' are:

$$\begin{aligned} \mu(\mathbf{x}) &\equiv E[y] = E[\mathbf{w}^T \mathbf{\Phi}(\mathbf{x})] \\ &= E[\mathbf{w}^T] \mathbf{\Phi}(\mathbf{x}) \\ &= 0 \end{aligned} \tag{6.2}$$

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') &\equiv E[yy'] = E[(\mathbf{w}^T \mathbf{\Phi}(\mathbf{x}))^T (\mathbf{w}^T \mathbf{\Phi}(\mathbf{x}'))] \\ &= E[\mathbf{\Phi}(\mathbf{x})^T \mathbf{w}\mathbf{w}^T \mathbf{\Phi}(\mathbf{x}')] \\ &= \mathbf{\Phi}(\mathbf{x})^T E[\mathbf{w}\mathbf{w}^T] \mathbf{\Phi}(\mathbf{x}') \\ &= \mathbf{\Phi}(\mathbf{x})^T \mathbf{\Phi}(\mathbf{x}') \ , \end{aligned} \tag{6.3}$$

since $E[\mathbf{w}] = 0$ and $E[\mathbf{w}\mathbf{w}^T] = \mathbf{I}$. The functions $\mu(\mathbf{x})$ and $k(\mathbf{x}, \mathbf{x}')$ are referred to as the mean function and kernel function, respectively. If we choose linear basis functions (i.e., $\mathbf{\Phi}(\mathbf{x}) = \mathbf{x}$), then the kernel function is quadratic: $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T\mathbf{x}'$. It can be shown that, with appropriate choice of Gaussian basis functions for $\phi_j(\mathbf{x})$, the kernel function becomes the RBF kernel:

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\frac{\gamma}{2}||\mathbf{x} - \mathbf{x}'||^2) \ . \tag{6.4}$$

Other assumptions about the form of g lead to different kernel functions.

Given N training pairs $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$, the $N \times N$ kernel matrix $\mathbf{K}$ is defined such that $\mathbf{K}_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$. A Gaussian predictive distribution at a new input, $\tilde{\mathbf{x}}$ can then be derived i.e.,

$$\tilde{y} \,|\, \tilde{\mathbf{x}}, \mathcal{D} \ \sim \ \mathcal{N}(m(\tilde{\mathbf{x}}); \sigma^2(\tilde{\mathbf{x}})) \ , \tag{6.5}$$

where

$$m(\mathbf{x}) = [y_1, ..., y_N] \ \mathbf{K}^{-1}\mathbf{k}(\mathbf{x}) \tag{6.6}$$

$$\sigma^2(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x})^T\mathbf{K}^{-1}\mathbf{k}(\mathbf{x}) \tag{6.7}$$

$$\mathbf{k}(\mathbf{x}) = [k(\mathbf{x}, \mathbf{x}_1), \ ..., \ k(\mathbf{x}, \mathbf{x}_N)]^T \ . \tag{6.8}$$

6.2.2 A simple two-factor model

Suppose now we wish to model different mappings for different styles. One way to do this is to add a latent "style" parameter. Accordingly, consider a regression problem with inputs $\mathbf{x}$ and style parameters $\mathbf{s} \in \mathbb{R}^S$. We define the following mapping, in which the output depends linearly on style:

$$\begin{aligned} y &= f(\mathbf{x}; \mathbf{s}) = \sum_{i=1}^{S} s_i g_i(\mathbf{x}) + \varepsilon \\ &= \sum_{i=1}^{S} s_i \mathbf{w}_i^T \mathbf{\Phi}(\mathbf{x}) + \varepsilon \ , \end{aligned} \tag{6.9}$$

where each $g_i(\mathbf{x})$ is a mapping with weight vector $\mathbf{w}_i$, and ε represents additive i.i.d. Gaussian noise with zero mean and variance β^{-1}. Fixing just the input $\mathbf{s}$ specializes the

mapping to a specific style. If we hold fixed the input $\mathbf{x}$ and style $\mathbf{s}$, then, because ε and $\mathbf{w} \equiv [\mathbf{w}_1^T \dots \mathbf{w}_S^T]^T$ are Gaussian, and $f(\mathbf{x};\mathbf{s})$ is a linear function of $\mathbf{w}$, $f(\mathbf{x};\mathbf{s})$ is also Gaussian. Given two sets of inputs $(\mathbf{x},\mathbf{s})$ and $(\mathbf{x}',\mathbf{s}')$, this function has mean and covariance

$$E[y] = \sum_i s_i E[\mathbf{w}_i]^T \mathbf{\Phi}(\mathbf{x}) + E[\varepsilon] = 0 \tag{6.10}$$

$$\begin{aligned} E[yy'] &= E[\left(\sum_{i=1}^{S} s_i g_i(\mathbf{x}) + \varepsilon\right)\left(\sum_{j=1}^{S} s'_j g_j(\mathbf{x}') + \varepsilon'\right)] \\ &= \sum_i s_i s'_i \mathbf{\Phi}(\mathbf{x})^T E[\mathbf{w}_i \mathbf{w}_i^T] \mathbf{\Phi}(\mathbf{x}') + E[\varepsilon\varepsilon'] \\ &= (\mathbf{s}^T\mathbf{s}')\mathbf{\Phi}(\mathbf{x})^T\mathbf{\Phi}(\mathbf{x}') + \beta^{-1}\delta \ . \end{aligned} \tag{6.11}$$

The term δ is 1 when y and y' are the same measurement, and zero otherwise.

The simple two-factor model with linear dependence on style and nonlinear dependence on content can therefore be expressed as a GP, with the Bayesian integration of the weights derived in closed form. As discussed below, the two-factor model can be generalized to greater numbers of factors, each of which may be linearly or nonlinearly related to the training data, holding the other factors fixed.

6.2.3 General multifactor models

In general, suppose we wish to model the effect of M factors $\mathcal{X} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(M)}\}$ on the output independently, then

$$\begin{aligned} y &= f(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(M)}) + \varepsilon \\ &= \mathbf{w}^T(\mathbf{\Phi}^{(1)} \otimes \cdots \otimes \mathbf{\Phi}^{(M)}) + \varepsilon \ , \end{aligned} \tag{6.12}$$

where $\mathbf{\Phi}^{(i)}$ is a basis column vector for factor $\mathbf{x}^{(i)}$, $\mathbf{w}$ is a weight vector, ε is as defined in the previous section, and $\otimes$ denotes the Kronecker product. As an example, for $M = 3$ with indexing elements of $\mathbf{w}$ by (l,m,n), (6.12) can be written as

$$y = \sum_{l,m,n} w_{l,m,n} \phi_l^{(1)} \phi_m^{(2)} \phi_n^{(3)} + \varepsilon \ ,$$

where $\phi_j^{(i)}$ is an element of $\mathbf{\Phi}^{(i)}$, and is a function of $\mathbf{x}^{(i)}$. The lengths of $\mathbf{w}$ and $(\mathbf{\Phi}^{(1)} \otimes \cdots \otimes \mathbf{\Phi}^{(M)})$ are both equal to the product of the lengths of $\mathbf{\Phi}^{(i)}$'s.

As before, we assume a weight decay prior on $\mathbf{w}$. Hence, y is a GP with zero mean and covariance

$$\begin{aligned} k(\mathcal{X}, \mathcal{X}') &\equiv E[yy'] \\ &= E[(\mathbf{w}^T(\mathbf{\Phi}^{(1)} \otimes \cdots \otimes \mathbf{\Phi}^{(M)}) + \varepsilon)(\mathbf{w}^T(\mathbf{\Phi}^{(1)'} \otimes \cdots \otimes \mathbf{\Phi}^{(M)'}) + \varepsilon')] \\ &= (\mathbf{\Phi}^{(1)} \otimes \cdots \otimes \mathbf{\Phi}^{(M)})^T E[\mathbf{w}\mathbf{w}^T](\mathbf{\Phi}^{(1)} \otimes \cdots \otimes \mathbf{\Phi}^{(M)'}) + E[\varepsilon\varepsilon'] \\ &= (\mathbf{\Phi}^{(1)} \otimes \cdots \otimes \mathbf{\Phi}^{(M)})^T(\mathbf{\Phi}^{(1)'} \otimes \cdots \otimes \mathbf{\Phi}^{(M)'}) + \beta^{-1}\delta \\ &= (\mathbf{\Phi}^{(1)T}\mathbf{\Phi}^{(1)'}) \otimes \cdots \otimes (\mathbf{\Phi}^{(M)T}\mathbf{\Phi}^{(M)'}) + \beta^{-1}\delta \\ &= \prod_{i=1}^{M} k_i(\mathbf{x}^{(i)}, \mathbf{x}^{(i)'}) + \beta^{-1}\delta \ , \end{aligned} \tag{6.13}$$

where $k_i(\mathbf{x}^{(i)}, \mathbf{x}^{(i)'}) = \mathbf{\Phi}^{(i)T}\mathbf{\Phi}^{(i)'}$ is the kernel function for the i-th factor, and $\mathbf{\Phi}^{(i)'}$ is a function of $\mathbf{x}^{(i)'}$. For example, the kernel function in (6.11) has two factors, with $k_1(\mathbf{s}, \mathbf{s}') = \mathbf{s}^T\mathbf{s}'$ and $k_2(\mathbf{x}, \mathbf{x}') = e^{-\frac{\gamma}{2}||\mathbf{x}-\mathbf{x}'||^2}$.

Given N training pairs $\mathcal{D} = \{(\mathcal{X}_i, y_i)\}_{i=1}^N$, the kernel matrix $\mathbf{K}$ for the resulting GP is defined in the usual way; i.e., $\mathbf{K}_{i,j} = k(\mathcal{X}_i, \mathcal{X}_j)$. The kernel product may also be written as the element-wise product of M kernel matrices, one for each factor,

$$\mathbf{K} \ = \ \mathbf{K}^{(1)} \circ \mathbf{K}^{(2)} \circ \cdots \circ \mathbf{K}^{(M)} + \beta^{-1}\mathbf{I} \ . \tag{6.14}$$

Conditioned on the factors, the joint likelihood of a vector of outputs $\mathbf{y} = [y_1, ..., y_N]^T$ is Gaussian: $\mathbf{y}|\{\mathcal{X}_i\}_{i=1}^N \sim \mathcal{N}(0; \mathbf{K})$.

If all basis functions $\mathbf{\Phi}^{(i)}$ are linear, then the generative model is multilinear, and the GP represents a Bayesian form of multilinear regression. For general kernel functions, multifactor GP regression can be performed in the same manner as normal GP regression. Given training data $\mathcal{D}$, the predictive distribution for a new set of inputs in each of the factors, $\tilde{\mathcal{X}}$, is Gaussian, and are defined in terms of the kernel function as in (6.5) – (6.8). Generalizing the above discussion to non-zero mean functions is straightforward.

In the case where the inputs $\{\mathcal{X}_i\}_{i=1}^N$ are unknown and the outputs are high-dimensional, the model can be viewed as a GPLVM [49] with a structured latent space. As it is usually

assumed that different subsets of the observations are represented by the same vector in certain latent factors. For example, a set of distinct face images are assumed to share the same lighting direction, or a set of poses are assumed to share the same gait.

Since the product of valid kernel functions is also a valid kernel function [88,103], any valid kernels may be used for the individual factors. Although this is a known result, products of kernel functions are rarely used. The value of our formulation is that it leads to intuition as to how and why to multiply kernels, by considering the underlying generative model. Previous work provides guidance as to how to determine the generative model as well. For example, simple bilinear models have been used successfully to model stylistic variation in typefaces [107], and multilinear models have also been used to capture the dependence of facial images on identity, lighting, and pose [107,120]. Nonlinear manifolds are clearly useful for modeling the space of human poses [20,27,116], but we may wish to express the dependence of motion data on other factors with linear kernels. In the next section, we use such experience with simpler models of mocap data to guide the selection of kernel functions for more complex multifactor models.

6.3 A model for human motion

In this section, we apply the multifactor model to human mocap data consisting of sequences of poses. A single pose is represented as a feature vector $\mathbf{y}_t$ of 89 dimensions, including 43 angular degrees-of-freedom (DOF) (see Figure 6.1), their velocities, and the global translational velocity. Joints with three DOFs and the global orientation are represented as exponential maps [26]; other joints are represented as Euler angles. An entire motion is represented as a sequence of T poses, $\mathbf{y}_{1:T}$.

We focus on periodic human locomotion, such as walking and running, and model each pose in a motion sequence as arising from a combination of three independent factors:

- the identity of the subject performing the motion, represented as a 3D vector $\mathbf{s}$;
- the gait of locomotion (walk, stride, or run), represented as a 3D vector $\mathbf{g}$; and
- the current state in the motion sequence, represented as a 3D vector $\mathbf{x}$. For example, $\mathbf{x}$ corresponds to the phase of a cyclic gait.

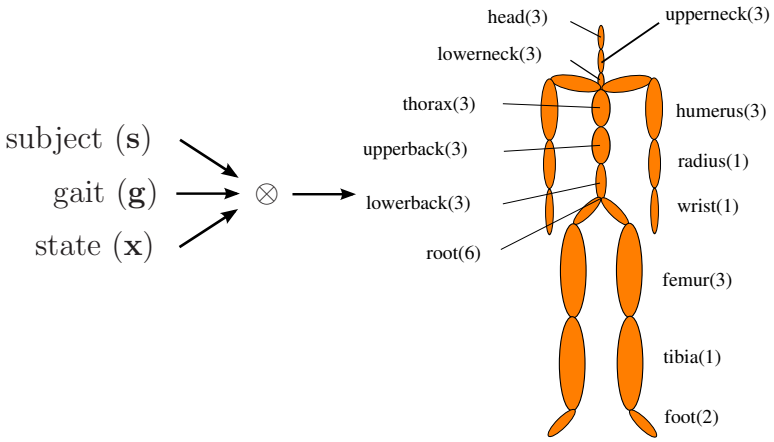


Figure 6.1: The skeleton used in our experiments is a simplified version of the default skeleton in the CMU mocap database. The numbers in parentheses indicate the number of DOFs for the joint directly above the labeled body node in the kinematic tree.

For the purpose of the discussion, We will refer to $\mathbf{s}$ and $\mathbf{g}$ as the *style*, and $\mathbf{x}$ as the *content* of the motion. These latent input coordinates are not normally provided in observed motions. Hence, the model is a form of the GPLVM, in which we estimate the latent coordinates.

We must also choose the type of kernel functions for each set of input coordinates. Fortunately, we can draw on experience from previous work to help select the mappings. In particular, it has been shown that, for a style-specific model of motion, a nonlinear GPLVM model with an RBF kernel provides excellent results [27, 116], whereas linear models (such as obtained by PCA) do not capture the nonlinearities of human poses. Second, stylistic parameters can often be modeled effectively using a linear space of styles [8, 98, 114] or multilinear in the case of multiple factors [119]. The representative power of linear mappings depends on the dimensionality of the latent space. We found 3D to be sufficient for representing style spaces containing three subjects and three gaits, respectively. Third, since each DOF y_d may have a very different variance, it is important to introduce scale terms w_d for individual DOFs [27]. Based on these observations, we employ the following kernel function for the d-th DOF:

$$k_d([\mathbf{x}, \mathbf{s}, \mathbf{g}], [\mathbf{x}', \mathbf{s}', \mathbf{g}']) = \\ \frac{1}{w_d^2}((\mathbf{s}^T\mathbf{s}')(\mathbf{g}^T\mathbf{g}')\exp(-\frac{\gamma}{2}||\mathbf{x} - \mathbf{x}'||^2) + \beta^{-1}\delta) \ . \tag{6.15}$$

This defines a Gaussian process $f_d(\mathbf{x}, \mathbf{s}, \mathbf{g})$ for each pose DOF, which is assumed to be independent conditioned on the inputs. Note that, if we fix values of $\mathbf{s}$ and $\mathbf{g}$, we get a style-specific GP over poses $\mathbf{y}$ conditioned on the content $\mathbf{x}$.

For any particular motion sequence, we assume the style stays constant over time, and only model dynamics in the content space. We consider two approaches: nonlinear GP dynamics and a circle dynamics model (CDM), where the content vectors are restricted to lie on a unit circle [20]. In the first approach, we assume the time-series obeys a nonlinear dynamical mapping:

$$\mathbf{x}_t = h(\mathbf{x}_{t-1}) + \varepsilon \ . \tag{6.16}$$

Furthermore, we assume that h is a GP with a linear + RBF kernel; hence, for any given $\mathbf{s}$ and $\mathbf{g}$, the model is a GPDM (Section 5.3).

In the CDM, low-dimensional coordinates are parameterized by a phase parameter θ_t, such that $\mathbf{x}_t = [\cos\theta_t, \sin\theta_t]^T$. Phase is linear as a function of time, parameterized by offset θ_0 and step-size $\Delta\theta$: $\theta_t = \theta_0 + t\Delta\theta$. Each sequence is then parameterized only by θ_0 and $\Delta\theta$. The step-size accounts for the different frequencies of different gaits. (The sampling rate of the mocap data is the same in all cases).

Given training sequences, we learn the model by maximizing the log-posterior of the unknown factors $\mathbf{x}$, $\mathbf{s}$ and $\mathbf{g}$ for each pose, as well as the kernel parameters. As mentioned before, each motion sequence has a single $\mathbf{s}$ and a single $\mathbf{g}$ for each pose; these factors are not allowed to vary through time. Furthermore, motions performed by the same subject are constrained to have the same $\mathbf{s}$ as each other, and motions with the same type of gait are constrained to have the same $\mathbf{g}$. The β and γ hyperparameters have prior $p(\beta, \gamma) \propto (\beta\gamma)^{-1}$; all other hyperparameters and factors have uniform priors. Numerical optimization is performed using L-BFGS-B [138]. Note that we do not constrain corresponding poses in different sequences to share the same $\mathbf{x}$, as we do not assume prior knowledge of the exact correspondences. It is desirable, however, to restrict the content of different styles to lie on the same trajectory, especially for motion synthesis. This is the main motivation for the CDM.

6.4 Experiments

We now evaluate the ability of learned multifactor models to perform time-series prediction from mocap data, and to synthesize new motion sequences in new styles.

Model	GPDM		B-GPDM		CDM
Style	no	yes	no	yes	yes
07-02	1.56	0.91	1.75	0.76	0.38
08-04	1.18	0.48	1.30	0.97	0.47
08-05	1.91	0.56	1.29	0.57	1.77
08-11	2.42	1.06	1.52	1.36	0.80
07-04	1.10	1.10	1.17	1.32	0.72
07-12	1.45	1.06	1.39	0.78	0.57
37-01	1.04	0.75	0.98	0.91	0.35
16-35	1.41	0.53	0.55	0.40	0.39
09-07	1.34	0.49	0.87	0.67	0.57
Avg.	1.49	0.77	1.20	0.86	0.67

Table 6.1: Root mean square (RMS) errors for long prediction. Sequence indices correspond to the sequences in the CMU mocap database.

Model	B-GPDM		CDM
Style	no	yes	yes
07-04	1.21 ± .035	0.92 ± .030	1.12 ± .048
07-12	1.48 ± .033	0.88 ± .037	1.14 ± .050
37-01	1.00 ± .026	0.70 ± .013	0.85 ± .019

Table 6.2: RMS errors for short prediction (averaged over 24 samples).

6.4.1 Prediction

In the prediction task, we first learn models from a collection of motion clips. The data are taken from the CMU mocap database (`http://mocap.cs.cmu.edu`), data sets 02_02, 02_03, 35_01, 35_18, 08_01, 08_07, downsampled by a factor of 4, and constitute 314 frames in total. Then, given a portion of a new sequence, we predict the subsequent frames of the sequence, and compare them against ground-truth. No time warping is done on any of the training or testing data. We compare single-factor dynamical models for $\mathbf{x}_t$ which do not explicitly model style ($\mathbf{s}$ and $\mathbf{g}$) with the multifactor models introduced in the previous section (but use the same dynamical model in $\mathbf{x}_t$). We will refer to the latter as stylistic models here. The models compared include the GPDM, the B-GPDM [113], and the CDM. The B-GPDM is a variant of GPDM, which heavily prefers smooth trajectories in the latent space. Following previous work, we use 3D latent spaces for $\mathbf{x}_t$ in the GPDMs. The CDM restricts $\mathbf{x}_t$ to lie on a circle, and is therefore 2D. For the stylistic models, 2 additional 3D latent spaces are introduced, corresponding to the $\mathbf{s}$ and $\mathbf{g}$ factors.

Given that a 2D circle latent space is unable to model any stylistic variations, the CDM without style is omitted as it performed very poorly. Single-factor GPDMs with higher

latent dimensionality are also omitted, as we have found that additional latent dimensions do not improve performance.

In prediction, the hidden factors are first estimated for the test subjects, by maximizing the joint posterior of all unknowns, conditioned on the test sequence and the learned model. Prediction is then performed by extrapolating the latent sequence of $\mathbf{x}_t$'s. For the GPDM variants, this is done by optimizing the joint dynamics distribution [127]; for the CDM, this is done by taking the appropriate number of linear steps in phase. In both cases, the subject and gait are assumed to stay constant for the new sequence. New poses are then generated as the mean of the conditional Gaussian given the computed factors for each time-step.

All five models are tested by a long prediction experiment and a short prediction experiment. In long prediction, poses for just over half a cycle are provided, and poses for the next cycle are predicted. For walking data, 25 frames are given, 40 are predicted. For running data, 13 are given, 20 are predicted. This is due to the difference in the number of poses per cycle. The mean RMS errors of all of the predicted frames are shown in Table 6.1. The stylistic versions the GPDMs perform better than the single-factor versions by 48% and 28%, respectively. The stylistic CDM model achieved the lowest average rates among all models in the long prediction test.

In short prediction, only about a quarter of a cycle is given (10 poses), and half a cycle (20 poses) is predicted. Here we selected three data sets — all from subjects not seen in the training data — consisting of walks of varying speeds. For each set, 24 random starting poses are selected (constrained by the need for there to be enough ground truth data after the start pose), and we show the mean and standard error of the average RMS for each pose. Here we do not test the GPDM models, as they performed worse than the B-GPDM models in the previous test. The stylistic model improved upon the original B-GPDM model in all three data sets. The stylistic CDM model had a higher mean error, as well as more variability than the stylistic B-GPDM model. This is in part due to its need for accurate estimation of the starting phase, as well as step size, which is less reliable for small numbers of input poses.

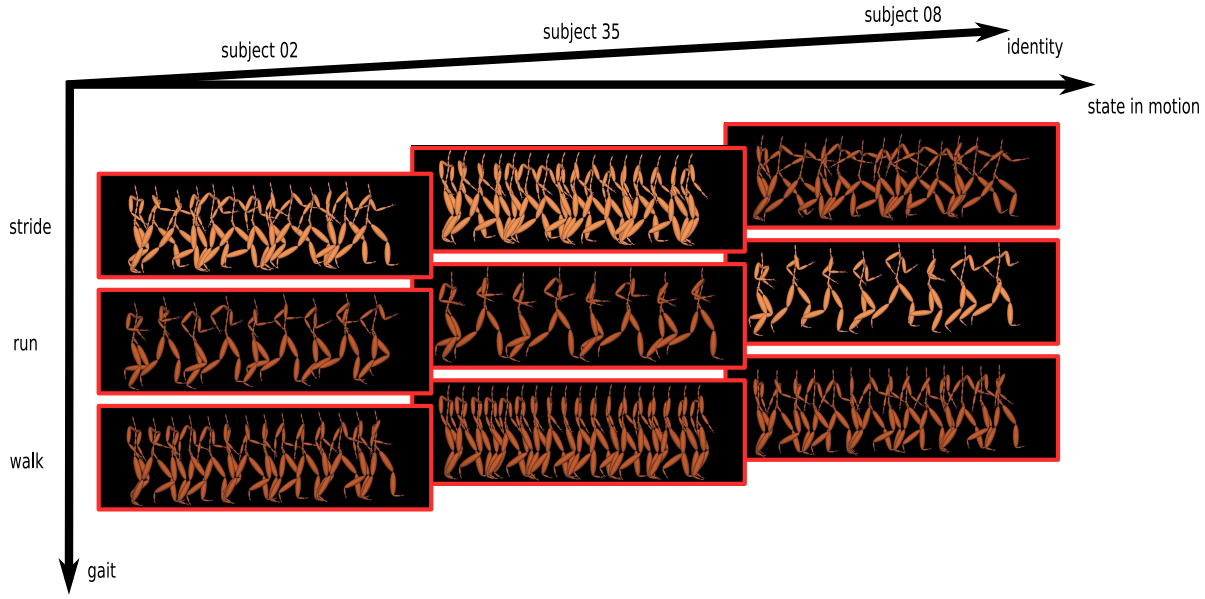


Figure 6.2: The structure of the multifactor model, where each sequence of poses are generated by an identity/subject vector, a gait vector, and a trajectory of states. Not all combinations of identity and gaits are available in the training data; the sequences (02, stride), (35, stride), and (08, run) are missing data inferred by the stylistic CDM.

6.4.2 Motion synthesis

The learned stylistic CDM can be used to generate motions not present in the training data. The model was learned from three subjects (02, 35, and 08 from the CMU database) all with some missing data. For subjects 02 and 35, the training data comprised examples of walking and running, but not striding. For subject 08, the training data included examples of walking and striding only. To generate new motion trajectories, a step size $\Delta\theta$ must be determined. We fit a bilinear model to the step sizes estimated during learning, mapping from $\mathbf{s}$ and $\mathbf{g}$ to $\Delta\theta$. Because the step-size determines the speed of the motion, we can generate motions of varying speeds.

Figure 6.2 depicts the structure of the multifactor model, including the inferred motions. These inferred motions are not simply copies of poses from a nearby gait or subject. In particular, the striding poses for subjects 02 and 35 contain stylistic elements of their respective walks: the inferred striding motion for 35 contains very little hand movement compared to the one striding training sequence (upper right), which is nevertheless consistent with subject 35's walking style (bottom center). Similarly, the bending of the left arm for subject 02 (upper left) is evident in that subject's walking style (lower left).

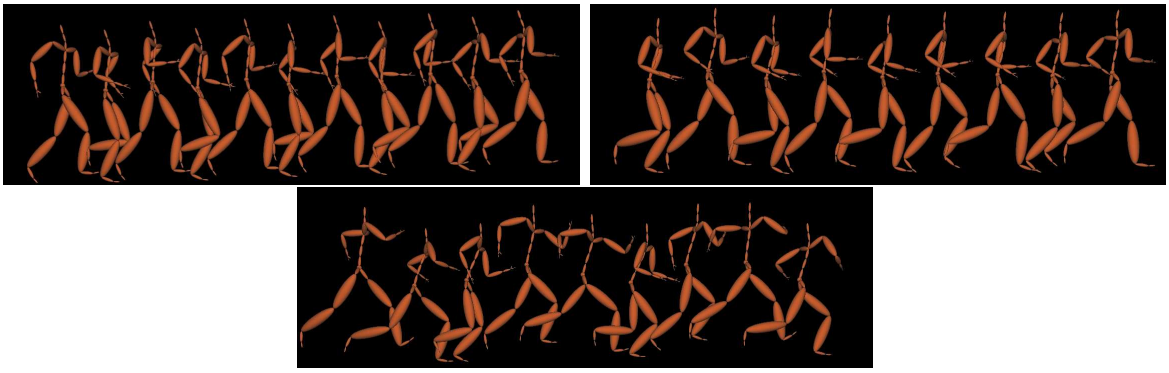


Figure 6.3: Motions generated from Gaussian sampling of the gait space and subject space. None of the poses are present in the training data.

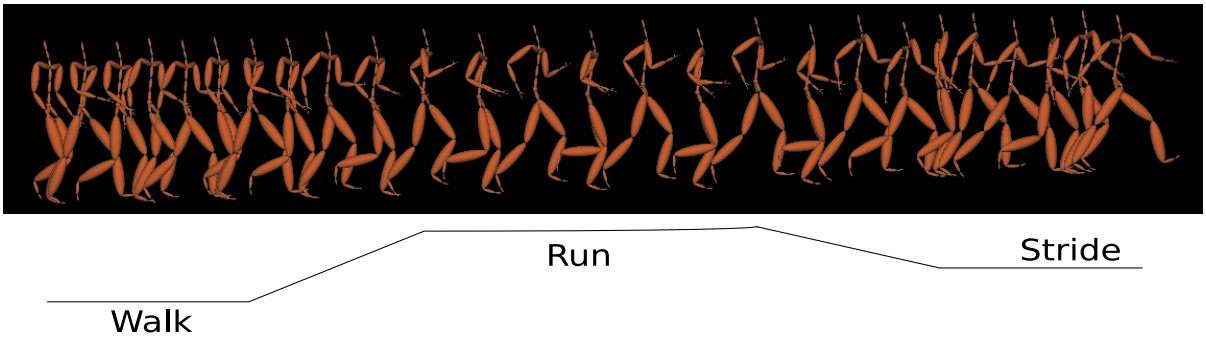


Figure 6.4: Transitions between different motions are achieved by linear interpolation in the gait space.

We can also generate new motions by random sampling. We fit one Gaussian distribution to the learned subject vector ($\mathbf{s}$) and another to the learned gait vector ($\mathbf{g}$), and then generate random new styles by sampling from these Gaussians. The step size $\Delta\theta$ can then be predicted by the bilinear model, and a sequence $\mathbf{x}_t$ of arbitrary length can then be generated.

The synthesized motions are shown in Figure 6.3. The top and middle rows are typical samples between a walk and a run. The bottom row is a slightly less typical, being a mixture of a run and a stride, which exaggerates the flight phase of running. In general, we find that convex combinations of styles produce reasonable motions.

Figure 6.4 demonstrates the ability of the model to generate smooth transitions from walking to running and from running to striding. The transitions are generated by linearly interpolating the gait vector with respect to the changing state vector. The subject vector is fixed to that of subject 02.

6.5 Discussion

We have described a multifactor regression and dimensionality reduction framework that unifies multilinear models with Bayesian regression and non-linear dimensionality reduction. The model can be viewed as a form of hierarchical Bayesian prior: modeling stylistic variation allows us to model a distribution of distributions, and thus generalize to new data with a style-specific distribution not included in the training data. In all of our experiments, we found that stylistic models performed better than generic models.

A number of potentially daunting choices are involved in determining which factors and kernels to use. We have made these choices by considering subproblems and special cases of the underlying generative model, such as "style-only" models and "content-only" models. Considering these cases sheds light on how to combine these models, and we recommend this approach. Alternatively, model selection techniques could be employed, assuming a large dataset is available. Either way, we believe that multiplicative combination of kernels will be useful for modeling many types of data sources with multiple factors.

The goal of models proposed in this chapter is to generate plausible motions that are different from training data, which is different from (though related to) a density estimation problem. One way to evaluate density estimation results is by first generating samples from the learned models, and then examining the difference between the sample and training distributions. However, the motion synthesis results we obtained through style interpolation in this chapter are not expected to come from the training distribution, and therefore need to be evaluated by a separate metric.

Alternatively, the multifactor models could be evaluated in the context of more realistic applications. The effectiveness of the proposed models can arguably be better demonstrated by improving the performance of say, an animation or a people tracking system than any evaluation of the models by themselves. The GPLVM has been applied to a number of such systems, as noted in Chapter 5. Being a special case of the GPLVM, the models proposed in this chapter can be applied and evaluated in a similar fashion. The evaluations could also be improved by training on a larger dataset, a different dataset (e.g., face data), or by comparing the style prediction results with ground truth data.

6.5.1 A special case for fast matrix inversion

The main computational cost of Gaussian process models, including the variant presented here, is the $O(N^3)$ inversion of the kernel matrix. The operation is necessary both to compute the likelihood of the data (5.8) and to perform prediction (6.6), (6.7). Furthermore, since our latent representations are not known and must be estimated, the inversion is performed for each iteration of the MAP estimation.

Consider a special case for the multifactor model where exactly one data point is available for each combination of factors. If we also assume no process noise, then up to proper ordering of the data vectors, the kernel matrix can be decomposed into a number of smaller matrices. Specifically,

$$\mathbf{K} = \mathbf{F}^{(1)} \otimes \mathbf{F}^{(2)} \otimes \cdots \otimes \mathbf{F}^{(M)} \ , \tag{6.17}$$

where $\mathbf{F}^{(m)}$ is a N_m by N_m matrix, $\mathbf{F}_{i,j} = k_m(\mathbf{x}_i^{(m)}, \mathbf{x}_j^{(m)})$, N_m is the number of different categories for the m-th factor, and $\mathbf{x}_i^{(m)}$ is the latent representation of the i-th category for the m-th factor. For example, the identity and gait factors in the example locomotion model have three categories each. Note that under our assumptions here, it must be true that $N = \prod_m^M N_m$, which suggests the size of the $\mathbf{F}^{(m)}$ matrices are generally much smaller than $\mathbf{K}$. It immediately follows from properties of the Kronecker product that

$$\mathbf{K}^{-1} = \mathbf{F}^{(1)^{-1}} \otimes \mathbf{F}^{(2)^{-1}} \otimes \cdots \otimes \mathbf{F}^{(M)^{-1}} \ , \tag{6.18}$$

which means the inversion of the kernel matrix can be achieved by inverting M much smaller matrices and taking their Kronecker product. Except for boundary cases, the size of the largest matrix that needs to be inverted is greatly reduced.

However, $\mathbf{K}$ is invertible if and only if matrices $\mathbf{F}^{(1)}, \ldots, \mathbf{F}^{(M)}$ are invertible. Furthermore, the requirement for the covariance function k_m, and therefore the matrix $\mathbf{F}^{(m)}$ to be positive-semidefinite is insufficient to guarantee invertibility. One solution is to redefine $\mathbf{K}$ so that a sufficiently large diagonal term is added to each of the $\mathbf{F}^{(m)}$ matrices before taking the tensor product:

$$\mathbf{K} = (\mathbf{F}^{(1)} + \beta_1^{-1}\mathbf{I}^{(1)}) \otimes (\mathbf{F}^{(2)} + \beta_2^{-1}\mathbf{I}^{(2)}) \otimes \cdots \otimes (\mathbf{F}^{(M)} + \beta_M^{-1}\mathbf{I}^{(M)}) \ . \tag{6.19}$$

Unlike in (6.14), where $\mathbf{K}$ is the sum of a tensor product term and a diagonal term that arises from the isotropic Gaussian process noise, expansion of (6.19) includes additional terms with lower-order dependencies on the factors. An interesting direction of future work is to investigate the effects of this modification with experiments.

Part III

Conclusion

Chapter 7

Conclusion and Future Work

In Part I of this thesis, we showed that walking controllers for 3D humanoid characters could be synthesized via an automatic optimization technique. In particular, Chapter 3 demonstrated that optimizing with a carefully chosen, biomechanically motivated objective function could result in controllers that create gaits significantly closer to human walking than hand-tuned alternatives from previous work. Chapter 4 showed that the same central idea — controller optimization based on CMA — can be combined with Monte Carlo simulation to generate controllers that are robust to different types of external disturbances and user inputs. Moreover, natural stylistic variations were generated by optimizing with different environmental constraints. For example, when significant motor noise is present, the optimization leads to a visually more careful walking style on slippery surface.

The idea of controller optimization based on derivative-free search techniques is conceptually simple, and is not new in itself. As we have discussed earlier, graphics researchers in the early 90s have optimized locomotion controllers for much simpler characters. However, it was never clear that the method would scale to a 3D, full-body human-like character. What we have shown here is that, when combined with a good control parameterization and objective function, this simple idea for automatic controller synthesis is effective for 3D humanoid walking. Moreover, it is important to note that our results do not rely on tracking motion capture or any other types of reference trajectory data. This not only provides increased robustness, but also allows us to explore stylistic changes under different types of environment and types of uncertainty.

The second part of this thesis dealt with the problem of human motion modeling from motion capture data. We presented a Bayesian approach to model existing motion data, which only assumes the model parameters has a prior distribution that is Gaussian, corresponding to a preference for smooth mappings. Since we do not make overly strong assumptions such as linearity of mappings, high-frequency details in the generally nonlinear human motion can be better modeled. Moreover, the Gaussian assumption allows the parameters to be analytically marginalized over, resulting in Gaussian process models that require fewer parameters to set compared to models with comparable expressiveness. These probabilistic motion models can not only be used for motion synthesis, but also are able to evaluate the likelihood of new motion trajectories, which enables applications such as forecasting and tracking.

The work in this thesis focused on the single activity of locomotion, which falls far short of the range of motions people perform in everyday life. Furthermore, while the optimization techniques we introduced here allow for high-level controls such as walking speed to be specified, the problem of satisfying animator constraints such as keyframes is not addressed. For highly specific user constraints, the *spacetime constraints* approach to trajectory optimization discussed in Section 2.3, where gradients can often be computed, is more suitable than our control optimization formulation. However, while convergence to a solution may be faster, the problem of multiple local minima persists. It would be interesting to see if a sampling-based method such as CMA can be combined with analytical gradients to improve the reliability of spacetime constraints on complex characters.

The development of physics-based constraints for human motion has traditionally been motivated from the character animation perspective, where the synthesis of high-quality motion is imperative. In contrast, data-driven constraints have at least in part been driven by applications in computer vision, where evaluating the quality of new motions is just as important as motion synthesis. Not surprisingly, in the latter domain, more attention has been paid to the use of motion models estimated from existing data, which can naturally be applied to evaluate new data. It is not until recently that the combination of the two — physical constraints and motion models, have started receiving attention in computer vision [9, 10, 123].

It is less clear whether adapting a more formal approach to motion modeling would be immediately beneficial to computer animation, but should still be a worthy direction for

exploration. For example, a clear and recognized [60] connection exists between spacetime constraints and energy-based models [51]. Can efficient learning and inference algorithms developed for the latter be applied to improve reliability of the former? On the other hand, can animation systems based on spacetime constraints be effectively converted to an energy-based model for applications such as motion classification?

If we are willing to approach both the character animation and tracking problems in terms of building prior models of motion, then it is instructive to consider what an ideal model would require. In particular, relating back to the flexibility versus plausibility issue discussed in Chapter 1, an ideal model for these applications should be

1. able to generate motions that satisfy any reasonable user constraints (or observations up to noise), and

2. generates only "natural" motions.

The definition of natural motion certainly depends on the application, but our knowledge of physics provides a rough guideline. For 3D people tracking, natural motions are almost certainly a subset of physically valid motions. On the other hand, in the production of artistic animation, hard physical constraints are less useful[1].

Depending on the approach, either one of the conditions can be trivially satisfied by itself. When considered together, however, motion model designers are forced to sacrifice either flexibility or perceived realism to find the right balance. Models defined from physics-based constraints can be extremely flexible in theory, but finding the force trajectory or control strategy to realize specific user constraints or observations is highly non-trivial. On the other hand, data-driven constraints, often combined with statistical models, can be highly effective for user constraints and observations that are close to previously available motion data. However, when this is not the case, they are limited in the ability to generalize, as even sophisticated statistical models can only make reliable predictions near observed data. The primary reason for this limitation is that they do not incorporate significant knowledge about the nature of human motion. Consequently, the most impressive demonstrations of generalization to unobserved data or environmental disturbances have relied on physical constraints, and this thesis is no exception.

[1]However, in this domain, it could be argued that strong motion models are not very useful in general.

Statistical modeling has been an effective tool in numerous domains of science, and should be effective for dealing with problems in motion analysis and synthesis as well. Moreover, by treating parameters of the equations of motion as parameters of a statistical model, it provides a natural medium to combine physics-based and data-driven constraints. The incorporation of physical constraints into the modeling process, despite its implication on computational costs, greatly increases the resulting model's potential for generalization. Indeed, finding efficient learning and inference algorithms for physics-based statistical models could be a highly rewarding direction for researchers in both character animation and human tracking.

Finally, the need to create animations for complex, articulated characters is almost as old as the field of computer animation itself. Today, animated feature films are commonplace and have become an important part of mainstream entertainment. Despite the impressive progress, there is no shortage of problems to work on for animation researchers. Unlike feature films that command budgets in the millions, films by independent artists often must be produced with minimal cost. The cost is highly dependent on the labour involved, and therefore the quality of motion synthesis algorithms. Applications in the rapidly growing video game industry offer even more technical challenges. Achieving the holy grail of creating interactive game characters capable of reacting to unexpected disturbances with a large repertoire of reliable motor skills, still remain elusive. As alluded to at the start of this thesis, addressing how prior knowledge about human motion, whether physics-based or data-driven, is utilized in the animation system is central to making progress.

In this work, we chose to focus on two subproblems, and dealt with both physics-based and data-driven constraints. We improved both the style and robustness of characters constrained by physical simulations, for the important activity of walking. We also proposed novel statistical models to better utilize available motion capture data. It is our hope that the methods proposed would serve as important building blocks for future motion synthesis algorithms. Algorithms that will significantly reduce the effort of animation creation, as well as get us one step closer to the holy grail of creating interactive characters that can walk, jump, play soccer, and perform other everyday motor skills that humans take for granted.

Bibliography

[1] Abbeel, P., Coates, A., Quigley, M., and Ng, A. Y. An application of reinforcement learning to aerobatic helicopter flight. In *Advances in Neural Information Processing Systems 19*. MIT Press, 2007, pp. 1–8.

[2] Agrawala, M., and Stolte, C. Rendering effective route maps: Improving usability through generalization. In *Proceedings of ACM SIGGRAPH 2001* (Aug. 2001), Computer Graphics Proceedings, Annual Conference Series, pp. 241–250.

[3] Alexander, R. M. *Principles of Animal Locomotion*. Princeton University Press, 2003.

[4] Anderson, F. C., and Pandy, M. G. Dynamic optimization of human walking. *Journal of Biomechanical Engineering 123*, 5 (Oct. 2001), 381–390.

[5] Arikan, O., and Forsyth, D. A. Interactive motion generation from examples. *ACM Transactions on Graphics 21*, 3 (July 2002), 483–490.

[6] Arikan, O., Forsyth, D. A., and O'Brien, J. F. Motion synthesis from annotations. *ACM Transactions on Graphics 22*, 3 (July 2003), 402–408.

[7] Bissacco, A. Modeling and learning contact dynamics in human motion. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2005), vol. 1, pp. 421–428.

[8] Brand, M., and Hertzmann, A. Style machines. In *Proceedings of ACM SIGGRAPH 2000* (July 2000), Computer Graphics Proceedings, Annual Conference Series, pp. 183–192.

[9] BRUBAKER, M. A., AND FLEET, D. J. The kneed walker for human pose tracking. In *Proceedings of the 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2008).

[10] BRUBAKER, M. A., FLEET, D. J., AND HERTZMANN, A. Physics-based person tracking using the anthropomorphic walker. *International Journal of Computer Vision 87*, 1 (Mar. 2010), 140–155.

[11] BURRIDGE, R. R., RIZZI, A. A., AND KODITSCHEK, D. E. Sequential composition of dynamically dexterous robot behaviors. *International Journal of Robotic Research 18*, 6 (1999), 534–555.

[12] BYL, K., AND TEDRAKE, R. Metastable walking machines. *International Journal of Robotic Research 28*, 8 (2009), 1040–1064.

[13] CHAI, J., AND HODGINS, J. K. Constraint-based motion optimization using a statistical dynamic model. *ACM Transactions on Graphics 26*, 3 (July 2007), 8:1–8:9.

[14] COHEN, M. F. Interactive spacetime control for animation. In *Computer Graphics (Proceedings of SIGGRAPH 92)* (July 1992), pp. 293–302.

[15] COLLINS, S., RUINA, A., TEDRAKE, R., AND WISSE, M. Efficient bipedal robots based on passive-dynamic walkers. *Science 307*, 5712 (Feb. 2005), 1082–1085.

[16] COROS, S., BEAUDOIN, P., AND VAN DE PANNE, M. Robust task-based control policies for physics-based characters. *ACM Transactions on Graphics 28*, 5 (Dec. 2009), 170:1–170:9.

[17] COROS, S., BEAUDOIN, P., YIN, K. K., AND VAN DE PANN, M. Synthesis of constrained walking skills. *ACM Transactions on Graphics 27*, 5 (Dec. 2008), 113:1–113:9.

[18] DA SILVA, M., ABE, Y., AND POPOVIĆ, J. Interactive simulation of stylized human locomotion. *ACM Transactions on Graphics 27*, 3 (Aug. 2008), 82:1–82:10.

[19] DE LATHAUWER, L., DE MOOR, B., AND VANDEWALLE, J. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications 21*, 4 (Apr. 2000), 1253–1278.

[20] ELGAMMAL, A., AND LEE, C.-S. Separating style and content on a nonlinear manifold. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)* (June/July 2004), vol. 1, pp. 478–485.

[21] FAISAL, A. A., SELEN, L. P. J., AND WOLPERT, D. M. Noise in the nervous system. *Nature Reviews Neuroscience 9*, 4 (Apr. 2008), 292–303.

[22] FALOUTSOS, P., VAN DE PANNE, M., AND TERZOPOULOS, D. Composable controllers for physics-based character animation. In *Proceedings of ACM SIGGRAPH 2001* (Aug. 2001), Computer Graphics Proceedings, Annual Conference Series, pp. 251–260.

[23] FANG, A. C., AND POLLARD, N. S. Efficient synthesis of physically valid human motion. *ACM Transactions on Graphics 22*, 3 (July 2003), 417–426.

[24] GLEICHER, M., SHIN, H. J., KOVAR, L., AND JEPSEN, A. Snap-together motion: Assembling run-time animations. In *Proceedings of the 2003 ACM Symposium on Interactive 3D Graphics (I3D)* (Apr. 2003), pp. 181–188.

[25] GOLDSTEIN, H., POOLE, C. P., AND SAFKO, J. L. *Classical Mechanics*, third ed. Addison-Wesley, 2001.

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

[27] GROCHOW, K., MARTIN, S. L., HERTZMANN, A., AND POPOVIĆ, Z. Style-based inverse kinematics. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 522–531.

[28] GRZESZCZUK, R., AND TERZOPOULOS, D. Automated learning of muscle-actuated locomotion through control abstraction. In *Proceedings of SIGGRAPH 95* (Aug. 1995), Computer Graphics Proceedings, Annual Conference Series, pp. 63–70.

[29] GRZESZCZUK, R., TERZOPOULOS, D., AND HINTON, G. NeuroAnimator: Fast neural network emulation and control of physics-based models. In *Proceedings of SIGGRAPH 98* (July 1998), Computer Graphics Proceedings, Annual Conference Series, pp. 9–20.

[30] HAMILTON, A. F. D. C., JONES, K. E., AND WOLPERT, D. M. The scaling of motor noise with muscle strength and motor unit in number in humans. *Experimental Brain Research 157*, 4 (2004), 417–430.

[31] HANSEN, N. The CMA evolution strategy: A comparing review. In *Towards a New Evolutionary Computation. Advances on Estimation of Distribution Algorithms.* Springer, 2006, pp. 75–102.

[32] HANSEN, N. The CMA evolution strategy: A tutorial. Homepage of Nikolaus Hansen (`http://www.lri.fr/~hansen/`), Mar. 2010.

[33] HARRIS, C. M., AND WOLPERT, D. M. Signal-dependent noise determines motor planning. *Nature 394*, 6695 (Aug. 1998), 780–784.

[34] HASLER, N., STOLL, C., SUNKEL, M., ROSENHAHN, B., AND SEIDEL, H.-P. A statistical model of human pose and body shape. *Computer Graphics Forum 28*, 2 (Apr. 2009), 337–346.

[35] HERR, H., AND POPOVIĆ, M. Angular momentum in human walking. *Journal of Experimental Biology 211*, 4 (Feb. 2008), 467–481.

[36] HIRAI, K., HIROSE, M., HAIKAWA, Y., AND TAKENAKA, T. The development of Honda humanoid robot. In *Proceedings of the 1998 IEEE International Conference on Robotics and Automation (ICRA)* (May 1998), vol. 2, pp. 1321–1326.

[37] HODGINS, J. K., AND POLLARD, N. S. Adapting simulated behaviors for new characters. In *Proceedings of SIGGRAPH 97* (Aug. 1997), Computer Graphics Proceedings, Annual Conference Series, pp. 153–162.

[38] HODGINS, J. K., AND RAIBERT, M. H. Biped gymnastics. *International Journal of Robotics Research 9*, 2 (Apr. 1990), 115–128.

[39] HODGINS, J. K., WOOTEN, W. L., BROGAN, D. C., AND O'BRIEN, J. F. Animating human athletics. In *Proceedings of SIGGRAPH 95* (Aug. 1995), Computer Graphics Proceedings, Annual Conference Series, pp. 71–78.

[40] KIM, J.-Y., PARK, I.-W., AND OH, J.-H. Walking control algorithm of biped humanoid robot on uneven and inclined floor. *Journal of Intelligent and Robotic Systems 48*, 4 (Apr. 2007), 457–484.

[41] KIM, M.-S., KIM, I., PARK, S., AND OH, J. H. Realization of stretch-legged walking of the humanoid robot. In *Proceedings of the 8th IEEE-RAS International Conference on Humanoid Robotics (Humanoids 2008)* (Dec. 2008), pp. 118–124.

[42] KIRKPATRICK, S., GELATT, C. D., AND VECCHI, M. P. Optimization by simulated annealing. *Science 220*, 4598 (May 1983), 671–680.

[43] KÖRDING, K. Decision theory: What "should" the nervous system do? *Science 318*, 5850 (Oct. 2007), 606–610.

[44] KOVAR, L., GLEICHER, M., AND PIGHIN, F. Motion graphs. *ACM Transactions on Graphics 21*, 3 (July 2002), 473–482.

[45] KUDOH, S., KOMURA, T., AND IKEUCHI, K. Stepping motion for a human-like character to maintain balance against large perturbations. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA)* (May 2006), pp. 2661–2666.

[46] KUO, A. D. A simple model of bipedal walking predicts the preferred speed-step length relationship. *Journal of Biomechanical Engineering 123*, 3 (June 2001), 264–269.

[47] LAMOURET, A., AND VAN DE PANNE, M. Motion synthesis by example. In *Proceedings of the 7th Eurographics Workshop on Computer Animation and Simulation (EGCAS 96)* (1996), pp. 199–212.

[48] LASZLO, J. F., VAN DE PANNE, M., AND FIUME, E. Limit cycle control and its application to the animation of balancing and walking. In *Proceedings of SIGGRAPH 96* (Aug. 1996), Computer Graphics Proceedings, Annual Conference Series, pp. 155–162.

[49] LAWRENCE, N. D. Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *Journal of Machine Learning Research 6* (Nov. 2005), 1783–1816.

[50] LAWRENCE, N. D., AND MOORE, A. J. Hierarchical Gaussian process latent variable models. In *Proceedings of 24th International Conference on Machine Learning (ICML 2007)* (June 2007), pp. 481–488.

[51] LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. Energy-based models. In *Predicting Structured Data*. MIT Press, 2006, pp. 191–246.

[52] Lee, J., Chai, J., Reitsma, P. S. A., Hodgins, J. K., and Pollard, N. S. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics 21*, 3 (July 2002), 491–500.

[53] Lee, J., and Lee, K. H. Precomputing avatar behavior from human motion data. *Graphical Models 68*, 2 (Mar. 2006), 158–174.

[54] Lee, Y., Lee, S. J., and Popović, Z. Compact character controllers. *ACM Transactions on Graphics 28*, 5 (Dec. 2009), 169:1–169:8.

[55] Li, R., Tian, T.-P., and Sclaroff, S. Simultaneous learning of nonlinear manifold and dynamical models for high-dimensional time series. In *Proceedings of the 11th IEEE International Conference on Computer Vision (ICCV 2007)* (Oct. 2007).

[56] Li, R., Yang, M.-H., Sclaroff, S., and Tian, T.-P. Monocular tracking of 3D human motion with a coordinated mixture of factor analyzers. In *Proceedings of the 9th European Conference on Computer Vision (ECCV 2006)* (May 2006), vol. 2, pp. 137–150.

[57] Li, Y., Du, Y., and Lin, X. Kernel-based multifactor analysis for image synthesis and recognition. In *Proceedings of the 10th IEEE International Conference on Computer Vision (ICCV 2005)* (Oct. 2005), vol. 1, pp. 114–119.

[58] Li, Y., Wang, T., and Shum, H.-Y. Motion texture: A two-level statistical model for character motion synthesis. *ACM Transactions on Graphics 21*, 3 (July 2002), 465–472.

[59] Li, Y., Wang, W., Crompton, R. H., and Gunther, M. M. Free vertical moments and transverse forces in human walking and their role in relation to arm-swing. *Journal of Experimental Biology 204*, 1 (Jan. 2001), 47–58.

[60] Liu, C. K., Hertzmann, A., and Popović, Z. Learning physics-based motion style with nonlinear inverse optimization. *ACM Transactions on Graphics 24*, 3 (July 2005), 1062–1070.

[61] LIU, C. K., HERTZMANN, A., AND POPOVIĆ, Z. Composition of complex optimal multi-character motions. In *Proceedings of the 2006 ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA)* (Sept. 2006), pp. 215–222.

[62] LIU, C. K., AND POPOVIĆ, Z. Synthesis of complex dynamic character motion from simple animations. *ACM Transactions on Graphics 21*, 3 (July 2002), 408–416.

[63] LIU, Z., GORTLER, S. J., AND COHEN, M. F. Hierarchical spacetime control. In *Proceedings of SIGGRAPH 94* (July 1994), Computer Graphics Proceedings, Annual Conference Series, pp. 35–42.

[64] LO, W.-Y., AND ZWICKER, M. Real-time planning for parameterized human motion. In *Proceedings of the 2008 ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA)* (July 2008), pp. 29–38.

[65] MACCHIETTO, A., ZORDAN, V., AND SHELTON, C. R. Momentum control for balance. *ACM Transactions on Graphics 28*, 3 (July 2009), 80:1–80:8.

[66] MACKAY, D. J. C. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[67] MCCANN, J., AND POLLARD, N. Responsive characters from motion fragments. *ACM Transactions on Graphics 26*, 3 (July 2007), 6:1–6:7.

[68] MCGEER, T. Passive dynamic walking. *International Journal of Robotics Research 9*, 2 (Apr. 1990), 62–82.

[69] MOON, K., AND PAVLOVIĆ, V. Impact of dynamics on subspace embedding and tracking of sequences. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2006), vol. 1, pp. 198–205.

[70] MORIMOTO, J., AND ATKESON, C. G. Nonparametric representation of an approximated poincaré map for learning biped locomotion. *Autonomous Robots 27*, 2 (Aug. 2009), 131–144.

[71] MUICO, U., LEE, Y., POPOVIĆ, J., AND POPOVIĆ, Z. Contact-aware nonlinear control of dynamic characters. *ACM Transactions on Graphics 28*, 3 (July 2009), 81:1–81:9.

[72] MUYBRIDGE, E. *Animal Locomotion*. University of Pennsylvania, 1887.

[73] NEAL, R. M. *Bayesian Learning for Neural Networks*. Springer-Verlag New York, Inc., 1996.

[74] NELDER, J. A., AND MEAD, R. A simplex method for function minimization. *Computer Journal 7*, 4 (1965), 308–313.

[75] NG, A. Y., AND JORDAN, M. I. PEGASUS: A policy search method for large MDPs and POMDPs. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI 2000)* (June/July 2000), pp. 406–415.

[76] NGO, J. T., AND MARKS, J. Spacetime constraints revisited. In *Proceedings of SIGGRAPH 93* (Aug. 1993), Computer Graphics Proceedings, Annual Conference Series, pp. 343–350.

[77] NOCEDAL, J., AND WRIGHT, S. J. *Numerical Optimization*. Springer-Verlag, 1999.

[78] NOVACHECK, T. F. The biomechanics of running. *Gait and Posture 7*, 1 (Jan. 1998), 77–95.

[79] OGURA, Y., SHIMOMURA, K., KONDO, H., MORISHIMA, A., OKUBO, T., MOMOKI, S., OK LIM, H., AND TAKANISHI, A. Human-like walking with knee stretched, heel-contact and toe-off motion by a humanoid robot. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Oct. 2006), pp. 3976–3981.

[80] PAN, W., AND TORRESANI, L. Unsupervised hierarchical modeling of locomotion styles. In *Proceedings of 26th International Conference on Machine Learning (ICML 2009)* (June 2009), pp. 99:1–99:8.

[81] PAVLOVIĆ, V., REHG, J. M., AND MACCORMICK, J. Learning switching linear models of human motion. In *Advances in Neural Information Processing Systems 13*. MIT Press, Cambridge, MA, 2001, pp. 981–987.

[82] PETERS, J., AND SCHAAL, S. Reinforcement learning of motor skills with policy gradients. *Neural Networks 21*, 4 (May 2008), 682–697.

[83] POPOVIĆ, Z., AND WITKIN, A. P. Physically based motion transformation. In *Proceedings of SIGGRAPH 99* (Aug. 1999), Computer Graphics Proceedings, Annual Conference Series, pp. 11–20.

[84] POZZO, T., BERTHOZ, A., AND LEFORT, L. Head stabilization during various locomotor tasks in humans. *Experimental Brain Research 82*, 1 (Aug. 1990), 97–106.

[85] RAHIMI, A., RECHT, B., AND DARRELL, T. Learning appearance manifolds from video. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2005), vol. 1, pp. 868–875.

[86] RAIBERT, M. H. *Legged Robots That Balance*. The MIT Press, 1986.

[87] RAIBERT, M. H., AND HODGINS, J. K. Animation of dynamic legged locomotion. In *Computer Graphics (Proceedings of SIGGRAPH 91)* (July 1991), pp. 349–358.

[88] RASMUSSEN, C. E., AND WILLIAMS, C. K. I. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[89] ROSE, C., COHEN, M. F., AND BODENHEIMER, B. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications 18*, 5 (September/October 1998), 32–40.

[90] ROSE, C., GUENTER, B., BODENHEIMER, B., AND COHEN, M. F. Efficient generation of motion transitions using spacetime constraints. In *Proceedings of SIGGRAPH 96* (Aug. 1996), Computer Graphics Proceedings, Annual Conference Series, pp. 147–154.

[91] ROSE, C., SLOAN, P.-P. J., AND COHEN, M. F. Artist-directed inverse-kinematics using radial basis function interpolation. *Computer Graphics Forum 20*, 3 (2001), 239–250.

[92] ROWEIS, S. T., AND GHAHRAMANI, Z. Learning nonlinear dynamical systems using the expectation-maximization algorithm. In *Kalman Filtering and Neural Networks*. Wiley, 2001, pp. 175–220.

[93] SAFONOVA, A., AND HODGINS, J. K. Construction and optimal search of interpolated motion graphs. *ACM Transactions on Graphics 26*, 3 (July 2007), 106:1–106:11.

[94] SAFONOVA, A., HODGINS, J. K., AND POLLARD, N. S. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 514–521.

[95] SHAPIRO, A., CAO, Y., AND FALOUTSOS, P. Style components. In *Proceedings of Graphics Interface 2006* (June 2006), pp. 33–39.

[96] SHARON, D., AND VAN DE PANNE, M. Synthesis of controllers for stylized planar bipedal walking. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA)* (Apr. 2005), pp. 2387–2392.

[97] SHIN, H. J., AND OH, H. S. Fat Graphs: Constructing an interactive character with continuous controls. In *Proceedings of the 2006 ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA)* (Sept. 2006), pp. 291–298.

[98] SIDENBLADH, H., BLACK, M. J., AND FLEET, D. J. Stochastic tracking of 3D human figures using 2D image motion. In *Proceedings of the 6th European Conference on Computer Vision (ECCV 2000)* (June/July 2000), vol. 2, pp. 702–718.

[99] SIMS, K. Evolving virtual creatures. In *Proceedings of SIGGRAPH 94* (July 1994), Computer Graphics Proceedings, Annual Conference Series, pp. 15–22.

[100] SMINCHISESCU, C., AND JEPSON, A. D. Generative modeling for continuous nonlinearly embedded visual inference. In *Proceedings of 21st International Conference on Machine Learning (ICML 2004)* (July 2004), pp. 759–766.

[101] SOK, K. W., KIM, M., AND LEE, J. Simulating biped behaviors from human motion data. *ACM Transactions on Graphics 26*, 3 (July 2007), 107:1–107:9.

[102] SPALL, J. C. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. Wiley, 2003.

[103] STITSON, M. O., GAMMERMAN, A., VAPNIK, V., VOVK, V., WATKINS, C., AND WESTON, J. Support vector regression with ANOVA decomposition kernels. In *Advances in Kernel Methods: Support Vector Learning*. The MIT Press, 1998.

[104] SUTTON, R. S., AND BARTO, A. G. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[105] TAYLOR, G. W., AND HINTON, G. E. Factored conditional restricted Boltzmann machines for modeling motion style. In *Proceedings of 26th International Conference on Machine Learning (ICML 2009)* (June 2009), pp. 129:1–129:8.

[106] TEDRAKE, R., ZHANG, T. W., AND SEUNG, H. S. Stochastic policy gradient reinforcement learning on a simple 3D biped. In *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Oct. 2004), vol. 3, pp. 2849–2854.

[107] TENENBAUM, J. B., AND FREEMAN, W. T. Separating style and content with bilinear models. *Neural Computation 12*, 6 (June 2000), 1247–1283.

[108] THORNTON, S., AND MARION, J. *Classical Dynamics of Particles and Systems*, fifth ed. Thomson-Brooks/Cole, 2004.

[109] TODOROV, E. Optimality principles in sensorimotor control. *Nature Neuroscience 7*, 9 (Aug. 2004), 907–915.

[110] TREUILLE, A., LEE, Y., AND POPOVIĆ, Z. Near-optimal character animation with continuous control. *ACM Transactions on Graphics 26*, 3 (July 2007), 7:1–7:7.

[111] TROJE, N. F. Decomposing biological motion: A framework for analysis and synthesis of human gait patterns. *Journal of Vision 2*, 5 (Sept. 2002), 371–387.

[112] TSAI, Y.-Y., LIN, W.-C., CHENG, K. B., LEE, J., AND LEE, T.-Y. Real-time physics-based 3D biped character animation using an inverted pendulum model. *IEEE Transactions on Visualization and Computer Graphics 16*, 2 (March/April 2010), 325–337.

[113] URTASUN, R., FLEET, D. J., AND FUA, P. 3D people tracking with Gaussian process dynamical models. In *Proceedings of the 2006 IEEE Computer Society*

Conference on Computer Vision and Pattern Recognition (CVPR) (June 2006), vol. 1, pp. 238–245.

[114] URTASUN, R., FLEET, D. J., AND FUA, P. Temporal motion models for monocular and multiview 3D human body tracking. *Computer Vision and Image Understanding 104*, 2–3 (November/December 2006), 157–177.

[115] URTASUN, R., FLEET, D. J., GEIGER, A., POPOVIĆ, J., DARRELL, T. J., AND LAWRENCE, N. D. Topologically-constrained latent variable models. In *Proceedings of 25th International Conference on Machine Learning (ICML 2008)* (June 2008), pp. 1080–1087.

[116] URTASUN, R., FLEET, D. J., HERTZMANN, A., AND FUA, P. Priors for people tracking from small training sets. In *Proceedings of the 10th IEEE International Conference on Computer Vision (ICCV 2005)* (Oct. 2005), vol. 1, pp. 403–410.

[117] VAN DE PANNE, M., AND FIUME, E. Sensor-actuator networks. In *Proceedings of SIGGRAPH 93* (Aug. 1993), Computer Graphics Proceedings, Annual Conference Series, pp. 335–342.

[118] VAN DE PANNE, M., AND LAMOURET, A. Guided optimization for balanced locomotion. In *Proceedings of the 6th Eurographics Workshop on Computer Animation and Simulation (EGCAS 95)* (1995), pp. 165–177.

[119] VASILESCU, M. A. O. Human motion signatures: Analysis, synthesis, recognition. In *Proceedings of the 16th International Conference on Pattern Recognition (ICPR 2002)* (Aug. 2002), vol. 3, pp. 456–460.

[120] VASILESCU, M. A. O., AND TERZOPOULOS, D. Multilinear analysis of image ensembles: TensorFaces. In *Proceedings of the 7th European Conference on Computer Vision (ECCV 2002)* (May/June 2002), vol. 1, pp. 447–460.

[121] VASILESCU, M. A. O., AND TERZOPOULOS, D. TensorTextures: Multilinear image-based rendering. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 336–342.

[122] VLASIC, D., BRAND, M., PFISTER, H., AND POPOVIĆ, J. Face transfer with multilinear models. *ACM Transactions on Graphics 24*, 3 (Aug. 2005), 426–433.

[123] VONDRAK, M., SIGAL, L., AND JENKINS, O. C. Physical simulation for probabilistic motion tracking. In *Proceedings of the 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2008).

[124] VUKOBRATOVIĆ, M., AND BOROVAC, B. Zero-moment point — Thirty five years of its life. *International Journal of Humanoid Robotics 1*, 1 (Mar. 2004), 157–173.

[125] WAMPLER, K., AND POPOVIĆ, Z. Optimal gait and form for animal locomotion. *ACM Transactions on Graphics 28*, 3 (July 2009), 60:1–60:8.

[126] WANG, J. M., FLEET, D. J., AND HERTZMANN, A. Multifactor Gaussian process models for style-content separation. In *Proceedings of 24th International Conference on Machine Learning (ICML 2007)* (June 2007), pp. 975–982.

[127] WANG, J. M., FLEET, D. J., AND HERTZMANN, A. Gaussian process dynamical models for human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence 30*, 2 (Feb. 2008), 283–298.

[128] WANG, J. M., FLEET, D. J., AND HERTZMANN, A. Optimizing walking controllers. *ACM Transactions on Graphics 28*, 5 (Dec. 2009), 168:1–168:8.

[129] WANG, J. M., FLEET, D. J., AND HERTZMANN, A. Optimizing walking controllers for uncertain inputs and environments. *ACM Transactions on Graphics 29*, 4 (July 2010), 73:1–73:8.

[130] WITKIN, A., AND BARAFF, D. Physically based modeling: Principles and practice (online SIGGRAPH '97 course notes). Physically Based Modeling (`http://www.cs.cmu.edu/~baraff/sigcourse/`), 1997.

[131] WITKIN, A., AND KASS, M. Spacetime constraints. In *Computer Graphics (Proceedings of SIGGRAPH 88)* (Aug. 1988), pp. 159–168.

[132] WOOTEN, W. L., AND HODGINS, J. K. Animation of human diving. *Computer Graphics Forum 15*, 1 (1996), 3–14.

[133] WU, J.-C., AND POPOVIĆ, Z. Realistic modeling of bird flight animations. *ACM Transactions on Graphics 22*, 3 (July 2003), 888–895.

[134] YANG, P.-F., LASZLO, J., AND SINGH, K. Layered dynamic control for interactive character swimming. In *Proceedings of the 2004 ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA)* (July 2004), pp. 39–47.

[135] YE, Y., AND LIU, C. K. Synthesis of responsive motion using a dynamic model. *Computer Graphics Forum 29*, 2 (May 2010), 555–562.

[136] YIN, K., COROS, S., BEAUDOIN, P., AND VAN DE PANNE, M. Continuation methods for adapting simulated skills. *ACM Transactions on Graphics 27*, 3 (Aug. 2008), 81:1–81:7.

[137] YIN, K., LOKEN, K., AND VAN DE PANNE, M. SIMBICON: Simple biped locomotion control. *ACM Transactions on Graphics 26*, 3 (July 2007), 105:1–105:10.

[138] ZHU, C., BYRD, R. H., LU, P., AND NOCEDAL, J. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software 23*, 4 (Dec. 1997), 550–560.

[139] ZORDAN, V. B., MAJKOWSKA, A., CHIU, B., AND FAST, M. Dynamic response for motion capture animation. *ACM Transactions on Graphics 24*, 3 (Aug. 2005), 697–701.