Thesis Proposal

Stuart O. Anderson

July 22, 2008

Abstract

Developing control systems for humanoid robots is a labor intensive process that typically depends on a simplified, task specific, model of the robot's dynamics. One reason for the difficulty of this process is the physical structure of these robots, with their numerous and often kinematically redundant degrees of freedom. Performance expectations, including balance, safety, and natural motion are an additional source of difficulty. These obstacles have resulted in a focused and successful research effort in the development of stable walking gaits. Because of this focus there are relatively few controllers for other task domains. Dynamic programming methods can find optimal policies for dynamic models of these robots without resorting to task specific models and controller designs. However, their usefulness is limited by the intractable computational costs associated with the large state spaces required by general purpose models. We propose that domain knowledge in the form of libraries of trajectory data can be leveraged to apply trajectory optimization-based policy generation methods to novel tasks for humanoid robots. In particular, libraries of motion capture data can provide examples of natural motion, constraints and initial guesses for trajectory optimization, statistical distributions for stochastic policy generation, and limits to the search space explored in the model learning process. These applications of trajectory libraries can replace or augment handcrafted models in much of the policy generation process. Using new trajectory-library-based methods, I will create controllers that allow the Sarcos humanoid biped to perform dynamic tasks, such as balancing on a bongo board. I will determine the utility of these methods by evaluating the naturalness and robustness of the resulting controllers.

Chapter 1

Introduction

Controller design is a difficult problem for all robots, but humanoid robots present a unique set of difficulties. These robots have many independent degrees of freedom and are designed with the kinematic redundancies and singularities present in human bodies. In the long term, humanoid robots are expected to interact with and operate near humans. The additional demands created by these expectations include safety, versatility, stability in unstructured environments, and natural motion. For these reasons, it can be difficult to directly apply classical planning and control techniques.

Many of the research questions resulting from these challenges have been studied extensively in the most active area of biped robot research, walking. Currently, state of the art humanoid robots can walk on simple uneven surfaces [Chestnutt et al., 2006] and use straight kneed gaits intended to better mimic natural human motion [Ogura et al., 2006]. Methods for dealing with kinematic singularities during gait and resolving kinematic redundancies [Sentis and Khatib, 2006] are rapidly maturing. Robots designed to exploit passive dynamics have demonstrated especially natural swing leg motions in the sagittal plane [McGeer, 1990]. However, much of this work is specific to the problem of producing stable gait on smooth hard floors. Additionally, no existing humanoid robot has demonstrated a three dimensional

gait that simultaneously appears natural and is robust to disturbances. Perhaps even more importantly, all methods for generating walking controllers for humanoid robots depend on handcrafted simplified models that are specific to the walking task. Enabling humanoid robots to perform the full variety of tasks required by many practical applications will require either a tremendous investment of engineering effort to develop new models for each new task, or the introduction of methods that simplify the process of creating controllers for new tasks.

This thesis proposes new methods for developing and improving controllers for humanoid robots that are intended to reduce or eliminate dependence on handcrafted models and control designs. These methods use information in databases of trajectories to reduce the effort required to produce a controller and to improve the naturalness of the robot's motion. The framework that these methods use is a model predictive control approach based on dynamic programming with offline model learning. I propose methods that generate sampling distributions, constraints, objective functions, and initial guesses for optimization by automatically building models that stochastically describe the behavior of the trajectories in a database.

The model predictive control framework has been used extensively in designing controllers for humanoid robots. Walking gaits refined by trajectory optimization and stabilized by linear feedback control laws are among the most common implementations of this design. Stilman and others [Stilman et al., 2005] have shown that dynamic programming based policy generation methods can be applied successfully to walking bipeds. A common feature of these and other humanoid robot control designs is the use of a handcrafted reduced model of the system that simplifies the optimization and control problem while preserving important features of the task domain.

I believe that these handcrafted models can be augmented or even replaced by the information in trajectory databases. In the case of stochastic dynamic programming (SDP),

the handcrafted model constrains the sampling distribution to a nonlinear manifold embedded within the system's full state space. A local model of the distribution of states in the trajectory library can also provide constraints on the sampling distribution, by biasing the sampling distribution toward states that tend to be involved in task solutions. Likewise, initial guesses used in trajectory optimization can be generated using a linear interpolation from the start state to the goal state in the state space of a reduced model. These simple trajectories in the reduced model can map to highly nonlinear trajectories in the full state space and, in doing so, capture important domain knowledge that improves the overall robustness of trajectory optimization. A trajectory library can be used to the same effect by interpolating between and concatenating trajectories that start near the initial state and eventually pass near the goal state.

Creating constraints and objective functions for trajectory optimization from trajectory data has been investigated by graphics researchers. The problem of animating humanoid characters performing a variety of actions without significant investment of time and effort is a central problem for animators working with human motion capture data. A rich collection of methods for concatenating, interpolating, and optimizing observed human motion to satisfy the constraints of new environments and behaviors has been developed by that community. Because this work typically requires that the resulting animations appear natural and satisfy physical constraints, the trajectory optimization methods developed for human animation are a promising resource for improving the optimization of trajectories for robot control. In particular, the automatic derivation of trajectory constraints from motion databases [Safonova et al., 2004] has the potential to augment or even replace the use of handcrafted reduced models for optimizing single-behavior trajectories.

Constrained trajectory optimization techniques, like those used by graphics researchers, are a key component of this thesis. Animators have shown convincingly that human motions can be optimized in constrained, task-specific, state spaces with fewer than ten dimensions.

Methods for deriving these constraints automatically from motion capture data have been widely demonstrated. While trajectory optimization methods are already used extensively to generate controllers for humanoid robots, the constraints used to make the optimization problems tractable are typically derived from the engineer's knowledge of the task domain rather than recorded data. I believe that this results in trajectory optimizers that produce unnatural motion because engineered constraints tend not preserve the coordination between multiple joints characteristic of natural motion, and may preclude some human-like motions entirely.

While efficient and robust trajectory optimization for human figures is an important component of the thesis work, the additional components described in this proposal are essential. In particular, the improvements to policy generation techniques, such as stochastic dynamic programming or differential dynamic programming, that use trajectory optimization as a core component are discussed in detail. We argue that, as with trajectory optimization, these methods can use information gathered from a trajectory database. Many of the techniques developed for bipedal walking controllers can be applied to the project of synthesizing controllers for new domains as well. In particular, the complementary use of reduced models, and the use of model learning to compensate for differences between a preexisting model of the robot and its true dynamics, are discussed.

The experimental platform for this work is the Sarcos Primus biped shown in Fig 1.1. The Sarcos Primus is a 1.6 meter tall humanoid robot weighing approximately 92 kilograms. It has 34 hydraulically actuated joints, seven in each limb, three in the trunk, and an additional three in the neck. Each joint is equipped with a position sensor and a load cell, and each foot contains a six axis load sensor. A five kHz local control loop allows compliant force control of all joints.

I have worked with the Sarcos Primus platform extensively. In the course of my previous work, I developed a number of controllers for the robot. These include a dynamic walking

gait, sensory adaptation during balance, and motion capture playback with dynamic stabilization. I have also worked extensively in simulation, developing methods for analyzing libraries of trajectories and transferring control policies between similar systems.

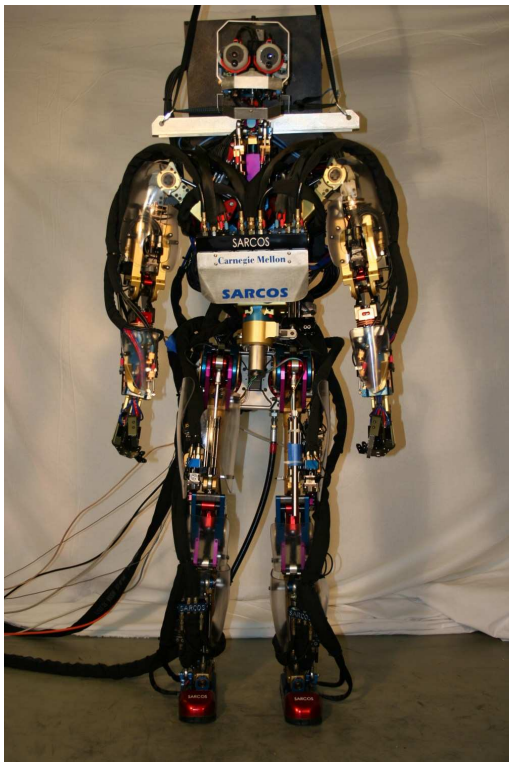


Figure 1.1: Sarcos Primus Biped.

Chapter 1 of this proposal clearly defines my thesis statement. Chapter 2 discusses research related to the work I am proposing. Chapter 3 reviews my own research in preparation for this work. Chapter 4 describes in detail the approach I am proposing, including components of the thesis that are already complete. Chapter 5 establishes in detail the process I will use to implement the approach described in Chapter 4. It also describes the evaluation metrics I will use to determine the success of this project.

1.1 Thesis Statement

Trajectory libraries, such as motion capture databases containing human motion, provide important information for developing and improving controllers for humanoid robots. The information in trajectory libraries can replace or augment task specific information present in handcrafted, task-specific, simplified models. This information can be leveraged in the form of direct kinematic trajectory tracking, constraints and initial guesses used during trajectory optimization, and as statistical distributions used for stochastic policy generation methods. I will demonstrate this thesis by showing that the Sarcos hydraulic humanoid's performance and naturalness while performing novel tasks such as balancing on a bongo board can be improved using these techniques.

Chapter 2

Related Research

Prior research related to the method outlined in this proposal spans several areas. These areas are humanoid robot control, the use of motion capture data for animating human characters, model reduction, trajectory optimization methods, sampling-based policy generation methods, and model learning. This chapter discusses existing research in each of these areas and concludes by discussing specific work done by three research groups that combines elements from all these areas and is most similar to the work that I am proposing.

One common approach to finding control policies for robotic systems with many degrees of freedom is to decompose a task into subtasks that are individually less difficult to solve. Methods like state space funnels [Mason, 1985; Erdmann and Mason, 1988], behavioral primitives [Mataric et al., 1998; Fod et al., 2002; Jenkins and Mataric, 2003], hybrid systems [Branicky et al., 2000], and reinforcement learning based on behavioral primitives [Bentivegna and Atkeson, 2001] are all variations of this approach. In practice, however, finding a useful decomposition is most often not automatic and relies heavily on expert human knowledge of a particular problem domain. The areas of research discussed in this chapter are related by the idea that trajectory libraries can provide information for factoring a difficult control problem into tractable component subproblems automatically.

2.1 Humanoid Robot Control

Research with robotic humanoids has focused primarily on locomotion, most often walking gaits on smooth level surfaces, and has relied heavily on task-specific models. A popular approach to developing a walking controller for a humanoid is to find a gait trajectory that satisfies a set of constraints such as actuator torque limits and the need to maintain balance. That trajectory can then be tracked using a controller designed to maintain stability in the presence of small perturbations and modeling errors. Variations on this method, using the Zero Moment Point (ZMP) balance constraint, are used by most humanoid robots with electric motors, including the HRP, ASIMO, HUBO, KHR, HSR, and the WABIAN series. Exceptions include the electric biped developed by CMU and ATR [Morimoto et al., 2003] and Delft's biped Meta [Schuitema et al., 2005]. Open questions in this area include the design of reference trajectories that increase the natural appearance, efficiency, or robustness of the resulting gait, and the development of new trajectory tracking controllers that enable recovery from larger perturbations that the local balance controller cannot stabilize.

The primary challenge to stable control of humanoid robots during gait is loss of balance. Existing approaches to maintaining stability can be categorized by the definition of balance. Quasi-static balance constraints require that the robot only enter states that are stable in a static analysis. Given a planar support surface this constraint implies that the robot's center of mass (CoM) lies above the convex hull of the points of contact between the robot and the support surface. If this constraint is obeyed, and the robot moves sufficiently slowly, it cannot fall unless pushed. A more common class of balance constraints requires that the robot only perform actions that allow its feet to remain in flat contact with the ground [Vukobratovic, 1973]. The ZMP constraint is the most commonly used member of this class. It is limited to the case of a robot with flat feet moving on a flat surface. The final and most general class of balance definitions are those that have no definition as instantaneous constraints.

These methods include the multi-step Poincare stability analysis of passive dynamic walking systems, point foot walkers, humans, and robots with control policies that do not produce gaits that obey instantaneous constraints [Garcia et al., 1998; Goswami et al., 1996].

The ZMP is the point on the ground plane at which the net moment of the forces acting on the robot have no horizontal component. When the ZMP falls within the region of support it is equivalent to the center of pressure (CoP). If the ZMP is maintained within the region of support then the feet will not rotate about a non-vertical axis. A more intuitive understanding of the ZMP constraint is provided by the idea that as long as the robot's motions do not violate the constraint, the feet can be treated as if they are rigidly attached to the ground. Because natural human gait involves rolling contact with the ground, some researchers believe that this constraint precludes natural walking motion. The utility of the ZMP model derives from its ability to reduce the problem of balance to the problem of controlling the change in total system momentum. This decoupling allows the control problem to be factored into two decoupled stages, first determining the center of mass trajectory and then the joint trajectory that produces the current desired center of mass acceleration.

Another tradition in controlling legged robots was pioneered for running hoppers [Raibert, 1986] and later extended to other gait styles [Hodgins and Raibert, 1991; Hodgins, 1991]. These controllers are based on an engineered model, the single link inverted pendulum (SLIP) [Blickhan and Full, 1993] that describes the exchange between potential and kinetic energy observed in many animal gaits using a pogo-stick-like model. A further simplification of that model is used to design the hopping controller. This simplification assumes that weakly coupled modes of the full nonlinear dynamics are in fact fully decoupled and linear. In particular, the forward velocity of the center of mass, the angular momentum, and the vertical velocity of the center of mass are all considered to be decoupled and are controlled by independent linear controllers. As a result, the results of the actions taken by each independent controller are assumed to be completely decoupled from inputs to the

other controllers.

Both constraint-based trajectory optimization approaches, like ZMP walking, and approaches based on finite state machines and linear controllers, like the Raibert hoppers, use factoring as a key component of their approaches. The ZMP approach separates balancing from the remainder of gait by ensuring that any motion used to achieve gait satisfies the ZMP constraint. This assumption permits a controller with a decoupled, tiered architecture, where the lowest level computes joint commands that stabilize the robot as it tracks a trajectory computed by a higher level. The controllers for the Raibert hoppers decouple the problem of biped hopping into the problems of maintaining body pitch, maintaining forward velocity, and maintaining hopping height.

2.2 Motion Capture Databases

Generation of humanoid motion is an active area in both the robotics and graphics research communities. Although the emphasis in the two communities differs, an increasing number of techniques that rely on motion capture data are now used productively by both fields. Roboticists are interested in creating motions for humanoid robots that appear natural [Kurazume et al., 2005] and in using humanoid robots for increasingly diverse tasks [Asfour et al., 2006]. Research in the graphics community on motion capture data focuses largely on reusing, modeling, and adapting previously captured motions to generate new animation. Quickly creating natural and physically realistic motion for animated human figures has been a primary focus for many graphics researchers [Rose et al., 1998; Pollard and Reitsma, 2001; Safonova et al., 2004; Kovar et al., 2002]. This work is of particular interest because it is intended to be applied to a range of tasks without significant changes to the underlying algorithms. The specific uses of human motion capture data that are relevant to this proposal are dynamic filtering, re-targeting, motion graphs, and constrained optimization.

The use of existing motion capture data to create new animations has been the basis for a range of graphics research. Relevant aspects of this research include methods for reusing segments of captured trajectories to create new motions by interpolating or concatenating the original motions, filtering captured motion to match the dynamic constraints imposed by a particular character model, clustering groups of similar motions, and generating new motions for characters using trajectory optimizers informed by motion capture data. Many of these techniques already have analogs in the humanoid robotics literature, but treat motion capture data either as a disturbance to be compensated for by an existing control system or as an additional term in the objective function used by an already functional trajectory generator.

Dynamic filtering refers to the process by which an input trajectory is altered to respect the constraints imposed by environmental constraints and system dynamics. The term filter indicates that no information about the future actions of the character are known, allowing the system to operate in real time on streaming data. The methods described in [Yamane, 2006; Pollard and Reitsma, 2001] modify the accelerations or torques at a character's joint to obey physical constraints. These constraints include torque or acceleration limits, and frictional contact constraints. The methods attempt to optimize tracking of joint angles and key points on the character's body, such as the chest or toes. Dynamic filtering can fail to provide a solution for some input trajectories, in the case where no action allows the constraints to be matched or the character motion diverges from the input motion.

Motion re-targeting allows captured motions to animate characters with different kinematic or dynamic properties than the actor who initially performed those motions. Automatic retargeting of motion capture data to produce upper body trajectories for humanoid robots has been demonstrated [Safonova et al., 2003]. This work uses trajectory optimization techniques to modify the input trajectory to respect constraints such as joint limits, maximum joint velocities, and self intersection. The objective function used for optimiza-

tion attempts to preserve the end effector position and small oscillations seen in the original motion as well as the overall joint angles. This work provides an example of how motion capture data could be transformed into trajectories expressed in the state space of a humanoid robot.

Motion graphs are the primary means by which motion capture data is reused through resequencing and concatenation of existing motion segments [Kovar et al., 2002; Lee et al., 2002]. Motion graph implementations must solve two problems. These problems are the construction of a graph that represents the possible transitions between segments of motion, and the identification of these segments within a motion capture database. To produce a finished animation, an interpolation method must be used to blend between the end of one segment and the beginning of the next.

Motion segmentation [Barbic et al., 2004; Beaudoin et al., 2007] is a technique for identifying groups of similar motions within a motion database. These techniques identify statistics of the states in the database and identify groups of trajectories that share similar statistics. For example, changes in the PCA can be used to identify when a change in motion type has occurred, or a Gaussian mixture model can be fit to all states in the database and changes in behavior detected by changes in the most likely cluster associated with the current state.

2.3 Model Reduction

Trajectory optimization and policy generation for humanoid figures is difficult because these systems contain a large number of internal degrees of freedom. Finding a reduced representation with many fewer degrees of freedom is a commonly used technique in both graphics [Safonova et al., 2004; Liu and Popović, 2002] and robotics [Park and Youm, 2007; Golliday and Hemami, 1977]. The degree to which an engineer's knowledge of the task domain is used to determine the problem decomposition provides a useful metric for catego-

rizing different methods of model reduction used for human figures. These methods range from fully automatic decompositions based on the model's dynamics or the content of the trajectory library, through task-specific, hand-designed reduced models intended to capture the important features of the task at hand.

Eigenvector decomposition can be used to find reduced representations of linear systems automatically. Each eigenvector with a real eigenvalue, and each pair of eigenvectors with complex conjugate eigenvalues represent an independent mode of the system. Energy added to one of these independent modes will not transfer to another mode. Because of this property it is possible to reduce a linear system by removing eigenvectors or pairs of coupled eigenvectors without affecting the behavior of the remaining system. However, as a system evolves in time it passes through larger regions of state space in which its dynamics become nonlinear and energy transfer between modes that are locally isolated becomes possible.

In the graphics community, motion editing and optimization in low dimensional spaces is an active topic of research. Three automatic dimensionality reduction methods that have been used successfully in the literature to produce low dimensional spaces that capture the structure of human motions used in specific tasks are Principal Component Analysis (PCA) [Safonova et al., 2004], Multidimensional Scaling (MDS) [Shin and Lee, 2006], and ISOMAP [Jenkins and Matarić, 2004]. In many cases researchers include some domain knowledge in the reduction by detecting contacts with the environment or tracking important physical quantities such as angular momentum or the character's center of mass.

Model reductions based heavily on engineering insight are common in both animation and robotics. Popovic [Popović and Witkin, 1999] suggests a model reduction based on eliminating particular joints by fixing their pose, eliminating entire branches of the kinematic hierarchy using the same method, and enforcing symmetry constraints between the left and right half of the body during some motions. The balance and gait generation strategies described in [Buschmann et al., 2007; Kajita et al., 2003; Rebula et al., 2007; Kemalettin Er-

batur, 2007] illustrate some of the common simplifications used to represent the dynamics of a humanoid biped, including reducing the robot to a single mass without angular momentum, a single mass with an attached flywheel, and a three mass model, where each leg has a mass independent from the body mass. These methods currently appear to be the most popular and successful in developing controllers for humanoid robots.

Another problem, equal in importance to the design of a reduced model, is the design of the method that projects the full state of the system to and from the reduced state of the model. Often, multiple sensors or actuators are redundant in the reduced model, and this redundancy must be resolved robustly. Even when this situation is not the case, the mapping between state spaces is a non-trivial problem. The state of the reduced model may not be directly computable from current sensor readings, as in the case of foot contact detection, which can require a hysteresis threshold-based state estimator. Likewise, a reduced model with an action space that includes motion of particular points on the robot's body, for example, the hips or the center of mass, may be difficult to map back to the action space of the robot due to redundant or singular kinematic configurations. We have previously explored a case where torque redundancy, in addition to kinematic redundancy can also challenge the mapping from a reduced model to a full humanoid robot [Anderson et al., 2006].

2.4 Trajectory Optimization

The techniques I am interested in that are shared between robotics and graphics research are related to trajectory optimization and dynamic filtering. These methods can be used to transform an unstable, inefficient trajectory that violates physical constraints into a physically plausible, balanced, and efficient trajectory, if the initial guess is sufficiently close to a viable solution. While trajectory optimization is already used to design trajectories for

humanoid robots, the methods developed by the graphics community are of particular interest because they are explicitly designed to make use of motion capture data both for the initial guess at an optimal trajectory and to constrain the search space that the optimizer must explore. Additionally, because animators desire characters that can perform a range of actions in varied environments, trajectory optimizers developed for human animation tend to be designed to operate on a more general set of inputs than domain specific optimizers developed for periodic tasks such as gait. These features suggest that the trajectory optimization techniques developed for creating physically plausible animations of human figures may also be well suited for creating viable reference trajectories for humanoid robots.

Trajectory optimization techniques are used by both animators and roboticists to create motions for humanoid figures. Trajectory optimization refers to a class of methods for determining solutions to a system of differential equations that minimize a functional of the solution while satisfying a set of constraints. A taxonomy of these methods based on salient features in their design is useful in a discussion of their application to robotics and animation. One such feature is whether a method minimizes the objective function directly by modifying the dependent variables subject to the constraints, or indirectly by using the calculus of variations to find a solution to the ODEs that satisfy a necessary condition for optimality, such as the Hamiltonian described by Pontryagin's Minimum Principal [Stengel, 1994]. Because indirect methods are often numerically unstable and require a more accurate initial guess [Schwartz, 1996], we consider only a selection of direct methods in this work.

Direct methods alter some or all of the independent variables to minimize the objective function. While they produce solutions that are strictly integrable, computing derivatives of the objective and constraint functions can become numerically unstable because the effects of actions taken earlier in time can grow exponentially over time. Shooting methods are a popular class of direct methods. These methods integrate the equations of motion forward or backward in time from a known state, varying the control input to achieve an optimal

trajectory. Computing the gradient of the objective function is challenging because the influence of control inputs at earlier times can be amplified by the integration, resulting in poor numerical conditioning in the resulting Jacobian. Multiple shooting methods can ameliorate this problem by dividing time into a series of epochs that are integrated separately [Morrison et al., 1962].

Differential dynamic programming is another method that integrates forward in time [Jacobson and Mayne, 1970]. This method iteratively improves a trajectory using quadratic approximations of the value function computed using linear models of the system at many points along the trajectory. It has the advantage of producing not only the optimal control inputs, but also an optimal linear policy at each discretization point. Recently, researchers suggested that improvements to the value function approximation could be made by fitting a quadratic model to values obtained by forward simulation of the system dynamics [Tessa et al., 2007].

A separate class of direct trajectory optimization methods, referred to as direct collocation methods, vary both the sequence of states and the control inputs. The equations describing the system dynamics are treated as additional constraints. All constraints are enforced only at discretization points, meaning that the system cannot necessarily be integrated forward in time. Direct collocation methods are popular because they are numerically stable and tend to converge from less accurate initial guesses than other methods [Schwartz, 1996]. In the graphics community, trajectory optimization methods were introduced by Witkin and Kass [1988] under the name spacetime constraints. The optimization approach presented in that work and used often in the graphics literature is a direct collocation method [von Stryk and Bulirsch, 1992].

Trajectory optimization for high dimensional systems can be computationally intensive due to the need to compute the Jacobian of cost and constraint parameters with respect to the free parameters. In the traditional formulation of the problem, joint torques appear in

the objective function and require $O(n^2)$ time to compute, where n is the number of degrees of freedom in the system. Fang and Pollard [Fang and Pollard, 2003] demonstrated a method for computing the force and torque about a single joint or the kinematic root in $O(n)$ time. They showed that if the objective function is reformulated in terms of joint accelerations rather than torques, and physical constraints are specified in terms of wrenches applied to the kinematic root, then a significant speedup to trajectory optimization for systems with many degrees of freedom can be achieved.

2.5 Sampling Based Methods

Stochastic dynamic programming [Atkeson and Stephens, 2007; Atkeson and Morimoto, 2003] and rapidly exploring random trees [LaValle, 1998] use random sampling of states to find policies or trajectories in high dimensional spaces.

Stochastic dynamic programming builds a piecewise linear policy incrementally using trajectory optimization. The process iteratively improves an approximate policy and value function. At each iteration, a random point is selected and the current policy is used to compute an initial trajectory that is then refined by differential dynamic programming (DDP). The initial point is selected such that the current approximation of the value function at that point is less than some threshold value. That threshold value is slowly increased over many iterations. The empirical value of the randomly sampled state is computed from this refined trajectory. If the expected value and true value differ significantly then the linear policy and quadratic value function approximation at the new point are added to the global policy. Periodically, the system replaces the policy at a point with other linear policies selected from the set of the policies associated with nearby points. This replacement is made permanent if it further minimizes the cost of a trajectory starting at that point. This replacement step serves to help globally optimize the approximate policy over time. Because this method only

adds points with unexpected values to the policy, it constructs a sparse representation of the true value function. However, because the exploration of the state space is random it must still fill the entire state space, and must employ a computationally expensive global search step to produce a globally optimal policy, because the DDP trajectory optimizer finds only local minima.

Rapidly Exploring Random Trees are a path planning technique that can be applied to systems with dynamic constraints [LaValle and Kuffner, 2001]. These methods find feasible trajectories for a system, often with many degrees of freedom, by building a trajectory tree that efficiently explores state space. Efficient exploration is achieved by sampling randomly in state space and choosing the nearest point in the tree to the randomly sampled state. The tree is then extended toward the randomly selected state by selecting an action that drives the system in the direction of that point. Successful search in high dimensional spaces in the presence of significant constraints requires well chosen sampling distributions [Kuffner et al., 2003]. The problem of finding good distributions from which to draw samples is a significant challenge when using RRTs to plan trajectories for humanoid robots performing general tasks.

One approach to generating human motion with RRTs uses a motion capture database to constrain the results of an inverse kinematics solution [Yamane et al., 2004]. The system generates motion for humanoid characters placing objects in constrained spaces. The RRT planner operates in a task space that represents the position of the characters hands. The inverse kinematics solver finds joint angles for the full character that produce the desired hand positions while avoiding self collision and attempting to produce poses that are similar to poses observed in a database of motions captured from actors performing a similar task. The path is converted into a trajectory by applying a velocity profile interpolated from nearby motions in the database. In this way the motion generator uses both domain specific knowledge, in the choice of representation for the RRT planner, and motion capture data,

in the inverse kinematics, to produce natural human motions.

An additional example of the combination of trajectory libraries with stochastic methods was developed by Choi and colleagues [Choi et al., 2003]. This domain specific method used a probabilistic roadmap method to choose footstep locations for an animated character. The local planner that found motions to connect sequential footsteps used motions from a motion capture database as the basis for its motion generation.

2.6 Model Learning

Model learning is a type of machine learning that allows the system to improve the model of its dynamics over time. These techniques can be classified by their use of parametric or non-parametric models of the dynamics. One form of model learning involves improving estimates of these parameters over time to better match observed behavior. Parametric models of the system are powerful because they can describe nonlinear behavior over large regions of state space using relatively few parameters. While this method can make efficient use of existing knowledge about the behavior of the system, it is often the case that parametric models cannot describe the behavior of a system in precise detail. For example, the Sarcos biped has many flexible hoses and wires that span its joints. These hoses and wires shift as the robot moves, changing mass distribution, joint friction, and other parameters. These effects cannot be captured by the rigid body dynamics formulation we use to represent the robot. However, these unmodeled effects do impact the performance and stability of the robot. Nonparametric model learning addresses this problem by storing a large amount of data recording the observed behavior of the system in many states [Atkeson et al., 1997a,b]. In the methods we consider, nonparametric methods provide precise modeling but use data only locally, while parametric methods cannot reproduce all observed effects but use local observations to improve models globally.

Learning unmodeled robot dynamics while constructing a global control policy has been studied with other types of robots, including helicopters [Abbeel et al., 2006; Bagnell and Schneider, 2001] and robotic arms [An et al., 1988]. Because the robot is expected to spend the majority of its time near the reference trajectories, the algorithm can focus on modeling these regions of state space. This focus limits the amount of data and number of states at which local models will need to be built.

2.7 Similar Approaches

Other researchers are pursuing work similar in many respects to the research outlined in this proposal. This section discusses that work in detail and identifies differences and similarities with the work being proposed.

Using motion capture data as either the initial trajectory for subsequent optimization and constraint enforcement [Safonova et al., 2003; Dasgupta and Nakamura, 1999] or to provide other input to the motion generation process [Shiratori et al., 2007; Kim et al., 2006] has been explored in the context of robotics. The work proposed in this thesis extends these previous examples. Specifically, I propose to combine many individual reference trajectories into a library of reference motions and to use that library to assist in the creation of a global control policy for the robot. While some of the existing methods also use relationships between trajectories, these methods have not been applied to policy generation.

Recent work on receding horizon differential dynamic programming [Tessa et al., 2007] also investigates building global policies from libraries of trajectories for systems with many degrees of freedom. This work introduces a potentially important improvement to the local approximation of the value function based on co-location vectors. Their method computes the value function by fitting a quadratic model to a cloud of state/action pairs. These pairs are sampled in the neighborhood of the point about which the approximation is constructed.

At each sampled point, the value of the state/action pair near each point in the trajectory is computed by simulating forward one timestep and using the value function approximation at the next timestep to provide an estimate of the value at the sampled point. The authors claim that this method provides a more accurate estimate of the local value function because it uses the full nonlinear dynamic of the system rather than a local approximation to the dynamics, and can be more efficient because it avoids the need to compute the seconds state space partial derivative of the dynamics model, F_{xx}.

Although this work does use a trajectory library with an associated piecewise linear policy to control a high dimensional system, there are significant differences between it and the work that I am proposing. The trajectory library is generated by the receding horizon differential dynamic programming and is used to provide real time access to these results, but does not inform the trajectory or policy generation process.

The SIMBICON controller developed by Yin and colleagues [Yin et al., 2007] produces robust two and three dimensional behavior for a simulated humanoid. The authors show that its motions can be tuned automatically from human motion capture data. The controller design is similar to earlier work [Raibert, 1986], in that it uses a finite state machine to transition between target poses, controls some link orientations with respect to the world frame, and achieves robust performance using tuned linear feedback. The work is novel because it produces a large variety of behaviors in two and three dimensions, presents a method for semi-automatically altering controllers based on motion capture data, and uses feedback error learning to generate feedforward torques. The state machine transitions between desired poses on foot contact or after fixed time intervals. The swing leg femur orientation and the torso orientation are controlled relative to world coordinates. Linear feedback is used to continuously vary desired joint angles, often the swing leg hip and stance ankle, as a function of center of mass position (relative to the stance ankle) and velocity. Motion capture data is used to design controllers by using Fourier analysis to extract a periodic movement.

That periodic movement is then tracked, with the phase of the tracking being reset on foot contact. Finally, feedback error learning is used to improve the behavior of the system over time and permit better performance with lower gains. The particular feedback error learning method employed computes the feedforward torque as a function of gait phase.

The significant difference between the SIMBICON controller and the system that I will propose is that the SIMBICON approach to motion capture data reduces the database to a single implicit trajectory during the Fourier analysis phase. By identifying only a single periodic motion all information about the variance of the gait cycle is lost. Hand tuned, task-specific balance control is required to maintain stability.

The work of Sok and colleagues [Sok and Lee, 2007] is similar to the SIMBICON work in that it animates two dimensional figures using motion capture data. It differs significantly in its implementation. In particular, this method uses a trajectory optimization method to modify motion capture data so that it can be tracked stably by the animated character. Once a collection of these trajectories has been computed, they can be combined into a single controller using a linear regression method. This regression method computes the current desired state. The regression is based on the weighted interpolation of the subsequent states of the k nearest neighbors to the current state. This control method is then adaptively improved by simulation. The improvement method is designed such that when a failure is detected in the simulated controller the system creates a new trajectory in the database that starts at the beginning of the last viable gait cycle and blends smoothly back to a stable gait already in the database. This warped trajectory is then rectified by the trajectory optimizer and added to the trajectory database used by the controller, ensuring that the same mistake will not be made again.

This approach is quite similar to my own in that is keeps many viable trajectories in the database. However, it does not explicitly compute a policy based on these trajectories, rather it uses a PD servo to track the observed trajectories. The method suggested for adaptively

improving the performance of the system over time is similar to the method I suggest for model learning and policy refinement.

Chapter 3

Initial Work

This chapter discusses completed work that is related to the proposed thesis. I discuss the real time control system developed for use with the Sarcos biped, a number of controllers developed using that system, methods for transferring policies between similar physical systems, and a method for identifying classes of trajectories in a library of trajectories. The intent of this section is to convince the reader that I have sufficient experience working with the Sarcos system to carry out the work proposed and to estimate the amount of time and effort that work will require.

3.1 Real Time Control Architecture

I developed a real time control architecture for the Sarcos biped that includes interfaces to rigid body dynamics simulation (SDFAST), nonlinear optimization (SNOPT), MATLAB, and command line and graphical interfaces (Fig 3.1). This software incorporates my experience regarding how to efficiently and safely control the robot, maintain configuration parameters, store data, and develop new task controllers. Its continuing development and refinement has made the development of new control systems for the Sarcos biped faster and

less error prone.

3.2 Robot Controllers

During the course of my research I have developed several task controllers for the Sarcos Biped. These tasks were walking with a gait inspired by passive dynamic walking, sensory adaptation during balance, and dynamic balance during motion capture playback.

Developing a controller that produces a walking gait for the Sarcos Biped provided essential experience in tuning and programming control systems for this robot [Anderson et al., 2005; Wisse et al., 2006]. In this work, we developed a controller based on the principles of passive dynamic walking, such as a straight stance leg and swing leg retraction prior to impact. The controller operated by replaying a hand-tuned trajectory, while additional linear controllers maintained the upper body pitch and produced spring-like forces at the ankles. The phase of the trajectory was reset on each heel strike.

While the passive dynamic 'compass gait' model was important to this work, it was not part of the control system. Rather, it influenced out decisions in designing the structure of the controller and the general shop of the hand-tuned trajectory the system followed during each step. By not strictly enforcing constraints based on a simplified model, we were able to include features, such as push-off using the stance ankle, that are not possible in an implementation that always satisfies the ZMP constraint.

In addition to the walking gait, I have developed a controller that implements a model of sensory adaptation (Fig 3.2) during human balance [Anderson et al., 2008] and methods for fine tuning ankle pose calibration [Anderson et al., 2006]. These controllers are both based on simplified models of the robot's dynamics. They provide evidence of my ability to develop viable control systems based on handcrafted models and to implement a variety of controllers on the robot successfully.

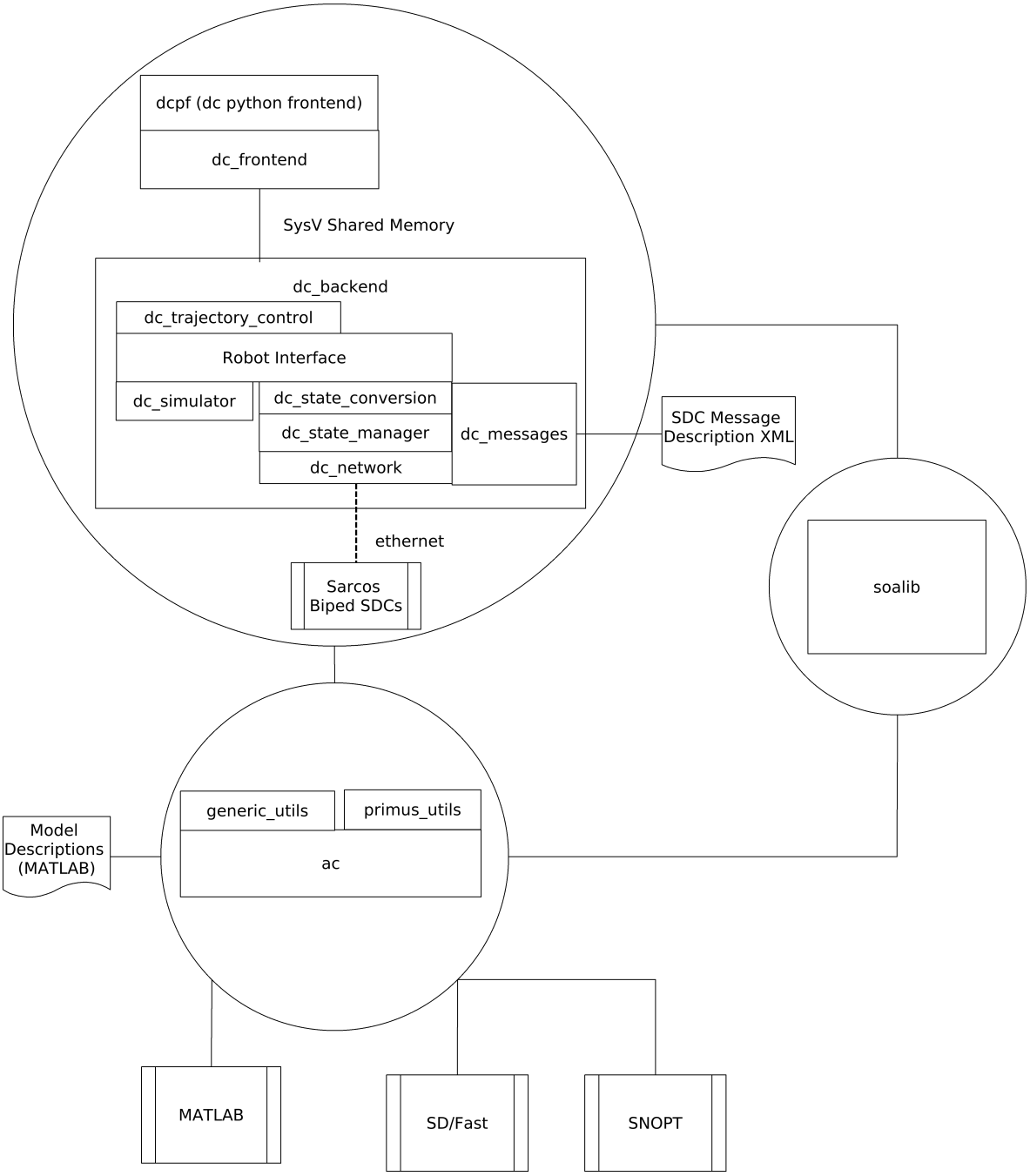


Figure 3.1: Real Time Control Architecture.



Figure 3.2: The Sarcos biped on an Equitest platform as used for sensory reweighting experiments.

Finally, I developed a system that allows the robot to replay a motion capture trajectory using its upper body joints while moving its lower body to compensate for the disturbances caused by that playback. This system also uses a simplified model of the robot, cart-table dynamics, but more importantly, demonstrates both the integration of motion capture data into a robot controller and iterative learning of a nonparametric model of the robot's dynamics.

Learning the appropriate lower body trajectory to compensate for displacements resulting from the upper body motion is an iterative process (Fig. 3.3)). The system learns by recording the center of pressure deviations caused by slowly performing the current upper and lower body trajectories. The system solves a sequential quadratic programming problem to find a center of mass trajectory for the cart table model [Kajita et al., 2003] that should

compensate for these deviations. The system then finds a new lower body joint trajectory that produces these additional center of mass motions using inverse kinematics. The inverse kinematics solution is computed using a nonlinear optimizer operating on small windows of the trajectory. This updated trajectory is then played back and the process is repeated. The speed of the playback is increased once the robot is sufficiently stable at a given speed, until the desired playback speed is reached or the system can no longer compensate for the introduced disturbances.

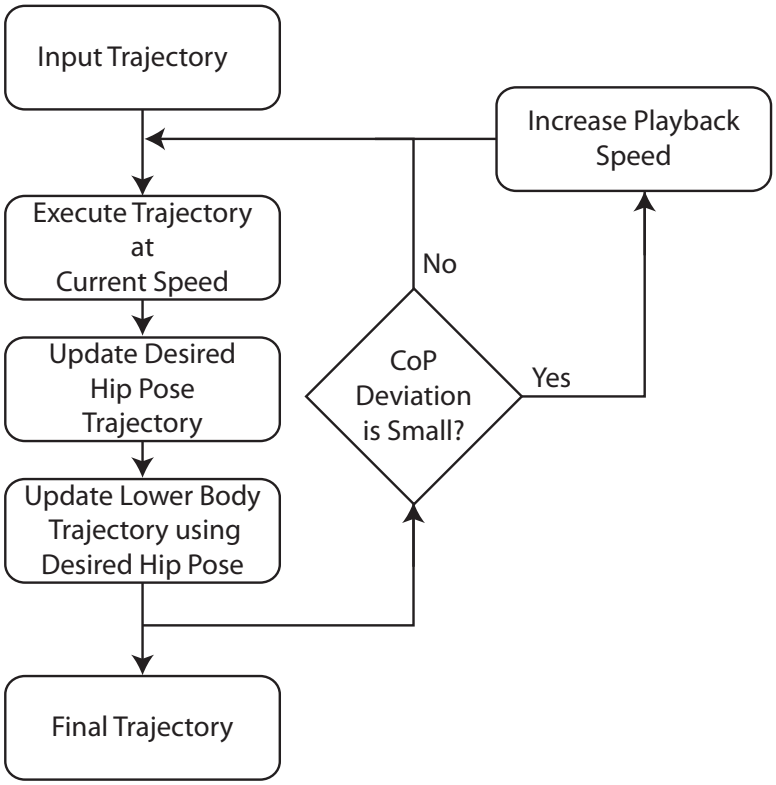


Figure 3.3: Iterative process for constructing lower body motion to compensate for upper body disturbances.

3.3 Policy Transfer

I have also studied methods for transferring value functions between similar systems [Anderson et al., 2007]. This work proposed two methods that allowed information from a known policy for an example system to be transferred to a similar experimental system. One method required that a feature vector be defined over the state spaces of both systems, as well as a loss function defining the salience of a feature space displacement to control of the system. We defined a mapping of the value function of the example system into the state space of the experimental system. If the features are well defined, this mapped value function should approximate the unknown true value function of the example system. We showed that by using the approximate value function to define the terminal cost of a receding horizon control trajectory optimizer we could create an implicit policy that was stable in situations where neither the receding horizon control alone nor the approximate value function alone was stable.

The second method for transferring information between systems defined an objective function for the experimental system based on the observed trajectories of the example system. Specifically, we used the inverse of the PCA basis of all the states in a trajectory library to define the state cost matrix Q in the experimental system's objective function. A stochastic dynamic programming implementation was then used to find a policy based on this objective function. The experimental system in our example was a bongo board balance task. This task was modeled using a fixed revolute joint attached to a translational joint that was in turn attached to a two link inverted pendulum (Fig. 3.4). The two link pendulum modeled the human's body and the remaining two links modeled the bongo board. We demonstrated that motion capture data of a human balancing on the bongo board could be projected into the state space of this reduced model by finding model parameters that best matched the observed motions of the estimated centers of mass of the human's upper

and lower body. We also showed that stochastic dynamic programming could be used to find policies for this difficult balance task with eight state variables.

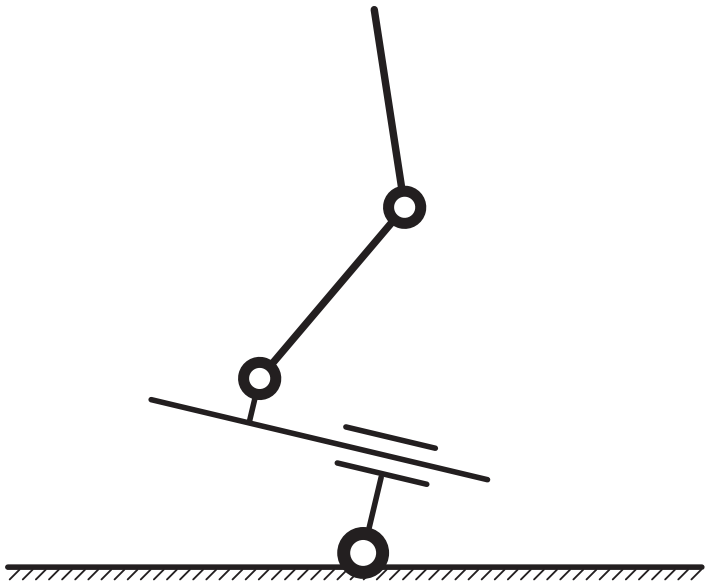


Figure 3.4: Simplified bongo board model used for policy transfer experiments.

This work allowed me to integrate an implementation of stochastic dynamic programming into my workflow and familiarize myself with its implementation. Using that implementation to develop a functioning policy for an eight dimensional system provided significant experience in analyzing and debugging the performance of this algorithm.

3.4 Identifying Trajectory Classes

We presented a method for identifying discrete classes of trajectories within a two dimensional trajectory library [Anderson and Srinivasa, 2007]. Each class of motion was intended to contain trajectories that generally converged toward each other. The method computes the curvature of manifolds defined to be normal to the observed trajectories at every point. This curvature provides a measure of the degree to which the trajectories in a local region are converging or diverging from each other. Integrating this metric along the length of each

trajectory in the database, and identifying trajectories for which local maxima and minima of this integral are obtained, provides central and limiting examples of each trajectory class. This process is diagrammed in Fig. 3.5. Trajectories that maximized the convergence integral were defined as "exemplar" trajectories, toward which all nearby trajectories tended to converge. Likewise, trajectories that minimized this integral divided classes of trajectories.

Fig 3.6 shows this segmentation algorithm applied to a single link inverted pendulum swingup task. This task requires a pendulum to reach its unstable (upright) equilibrium while minimizing a cost functional that includes terms reflecting the torque applied to the pendulum and distance from the goal state. The algorithm has identified four distinct classes of trajectories. The system appears to have selected a classification that reflects whether the pendulum starts with more or less total energy than it has when it reaches the goal, and whether the system enters the goal from the left or the right.

The blue lines radiating from the origin are examples of the flow-normal manifolds. Note that they form a right angle with the trajectories of the system at every point of intersection. The background intensity indicates the curvature of these manifolds according to the scale to the right of the image. Finally, the red trajectories maximized the integral of this curvature over their length, while the black trajectories minimized that same integral. This particular

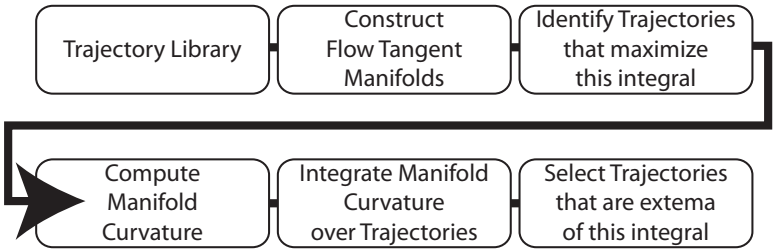


Figure 3.5: Flow chart illustrating the operation of the trajectory class identification process.

algorithm is of limited use to the problem presented in this thesis because it is limited to systems with only two state variables. However, the work proposed in the next section aims to achieve many of the same goals using related techniques. Specifically, the approach of

detecting regions of convergence and divergence is related to the local PCA an extension proposed in the following section, because converging trajectories will have reduced variance in the dimensions in which they converge.

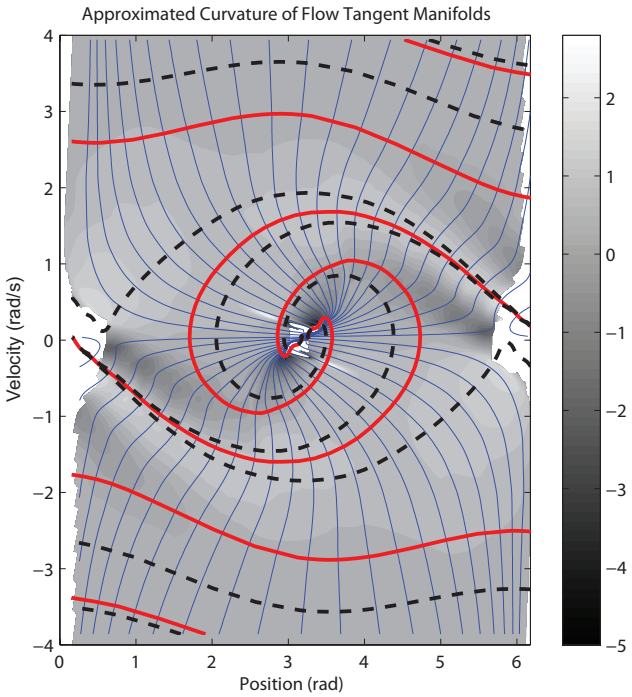


Figure 3.6: Trajectory classes for an inverted pendulum swingup policy. Thick red lines are exemplar trajectories, black dashed lines are class divisions, blue lines are flow normal manifolds. The local curvature of the flow tangent manifold is indicated by the lightness of the background. The visualization of any given trajectory is facilitated by noting the positions at which the trajectory crosses the horizontal (velocity) axis. In the case of the trajectories that start at the origin, the initial condition represents hanging vertically with no velocity. The first crossing of the velocity axis represents a reversal of the direction of motion at a relatively small displacement. The second crossing is a second reversal of direction at a larger displacement, and the final crossing occurs as the pendulum comes to rest at the goal.

Chapter 4

Approach

I will demonstrate the use of trajectory libraries in controller design by building a series of controllers for the Sarcos biped using the techniques proposed in this thesis. There are three components in these controllers that can be developed largely independently: a suitable trajectory library, an initial control policy derived using that library, and refinement of that policy and the dynamic model over time. Constructing the trajectory library requires either that motion capture data be remapped to the state space of the robot or that data be collected from an existing controller. Additionally, it may be useful to identify distinct groups of trajectories within the library. Developing an initial policy requires a policy generation technique that in turn will rely on a trajectory optimization method. The improvement of the policy over time requires both parametric and nonparametric model learning methods.

The principal idea guiding this approach concerns the role that handcrafted models and constraints currently play in controller design. I believe that not only are these task-specific models and constraints difficult to design, but that even when well designed they often over-constrain the resulting controllers. Because a reduced model is equivalent to an implicit nonlinear constraint on the states of the system, both models and explicit constraints are stiff limits to the behavior of a system. My goal is to replace these rigid constraints with

soft constraints in the form of stochastic models. That these stochastic constraints can be created automatically from trajectory libraries adds further value to their use.

While the key ideas and motivations guiding the approach I will take to demonstrate my thesis are fixed, some of the particular methods I will use are likely to change as I proceed with the research. This chapter discusses the methods I am considering for each of the three components of the process.

4.1 Building a Trajectory Library

Constructing a library of trajectories is the first step in my approach. I consider two sources for trajectories, human motion capture data and an existing control implementation on the robot. Constructing a trajectory library from human motion capture data requires consideration of what behaviors to capture, and the transformation of the captured data into the state space of the robot. This transformation from motion capture data to robot trajectory is intended to capture the covariance in the robot's state space rather than reproduce stable motions. The mapping process may be as simple as fitting joint angles to match limb orientations, or as complex as applying a dynamics filter and checking for self collision. Constructing a database from an existing robot controller requires only that data be logged while that controller is active. A significant challenge in constructing trajectory libraries in either case is ensuring that the library covers the full range of behaviors the system must exhibit. In both cases it may be advantageous to identify groups or classes of trajectories that can be processed separately. For example, if two distinct behaviors occupy the same region of state space, it is important to identify the distinct subspaces that each behavior occupies. If the subspaces corresponding to each behavior are not separated and a single subspace is used for policy generation, then either the dimensionality of that subspace must be increased to capture all relevant aspects of the system's motion or one or more relevant

dimensions must be dropped.

The motion capture database that I propose can serve several functions. It is a repository of reference trajectories that can be interpolated and concatenated to produce extended and novel motions using techniques borrowed from the literature on motion graphs [Kovar et al., 2002; Lee et al., 2002]. In particular, the similarity between motion graphs and sequential policy composition [Mason, 1985] suggests that by treating each trajectory or class of trajectories as an individual policy, motion graph techniques could be used to construct global policies. The database can also be a source of information used to find reduced dimensionality approximations of the system that can improve trajectory optimization results, similar in some respects to the method described in Safonova's work [Safonova et al., 2004]. It can aid policy generation by providing statistical metrics of local variance in trajectories useful for stochastic sampling-based methods, or by providing a library of trajectories about which linear policies can be defined [Sok and Lee, 2007].

It may be necessary to augment the motion capture system with additional sensors for some tasks. For example, pressure or contact sensors on the bongo board might help identify when the roller reaches the end of its range of motion or when the board touches the ground. In addition, it may be desireable to estimate the value of additional states that cannot be sensed directly.

Trajectories recorded while an existing controller is operating can also be used to create a trajectory library. This data can be especially useful because the internal state of the controller, representing information such as estimated foot contact state, can be directly recorded rather than reconstructed from observed motion. Mixing motion capture trajectories with trajectories recorded from an existing controller in a single library is also possible.

For the bongo board controller I will use two sources of trajectory data. The first source will be a hand designed controller built by having the robot track a nonlinear control policy designed for a simplified model of the problem. This controller is being implemented

on the robot and, when complete, the data resulting from its execution will provide an initial trajectory library for controller design. The seconds source of data is the motion capture dataset that I captured for my paper on policy transfer between similar mechanical systems [Anderson et al., 2007]. This dataset contains a few dozen examples of a human mounting, balancing, and dismounting a bongo board. The bongo board's state is included in this capture data.

4.2 Building a Policy

The computational cost of policy generation through a general optimization process such as dynamic programming is prohibitively expensive when the system's state has high dimensionality. For example, the stochastic dynamic programming implementation that I use has found viable policies for problems with up to 8 dimesnions. Algorithmic problems of this nature, where cost grow exponentially with input dimensionality, are often addressed by dividing the problem into simpler subproblems. The solutions to these subproblems can later be recombined into a solution to the full problem. Depending on the specifics of the problem at hand this factoring process may yield exact or approximate solutions to the full problem.

Consider the problem of building a globally optimal policy from the combination of several optimal policies defined for subspaces of the full system's state space. Finding the optimal policy for a subspace provides the globally optimal policy on that subspace if optimal trajectories of the full system do not leave that subspace. Stated more precisely:

Given a system with state space X and a globally optimal policy $U(x)$, let X_s be a manifold in X and $u(x)$ be a policy that is defined on X_s. Let $T_s(x,t)$ be the state of the system at time t when U is applied starting at state x and $t = 0$. If $T_s(x,t)$ is an element of $X_s \forall x \in X_s$ and $t > 0$ and if $u(X)$ is an optimal policy under the constraint $x \in X_s$, then:

$u(x) = U(x) \forall x \in X_s$.

One method of factoring a difficult policy generation problem into many subproblems is to constrain the system's state to low dimensional manifolds within the original state space. Such an approach depends on these manifolds containing, or being near, the paths of optimal trajectories of the system. The selection of constraint manifolds that follow optimal trajectories of the system appears to be, in general, a 'chicken and egg' problem, in that a global policy is required to find the optimal trajectories that define the constraint manifolds that are in turn required to find an approximate global policy.

However, as discussed, other sources of optimal or near optimal trajectories are available, especially for humanoid robots. If human motion is believed to be close to an optimal motion of the system, then using the trajectories from motion capture data as the basis for a model of the distribution of states used in the policy makes it likely that a globally optimal policy can be found by drawing states from this distribution for stochastic dynamic programming (SDP). The degree to which captured motions are, in practice, close to stable or desireable trajectories of the system depends on both the particular task and the details of the method used to project motion capture data into the joint space of the robot. It is possible that because the covariance between joint angles, rather than absolute joint positions, are the primary information extracted from the capture database, the system will be relatively robust to the design of the projection methods. Additionally, if a controller based on a constrained model of the system exists, the trajectories produced by this controller can be used to define the constraint manifolds.

Once such a library of trajectories has been built, it can be used to guide the development of a control policy. I am considering two possible methods for this step, stochastic dynamic programming and differential dynamic programming. Both of these methods benefit from well-chosen trajectory optimization constraints.

4.2.1 Policy Generation

There are two models of policy generation that I will explore, one based on stochastic dynamic programming, and the second based on differential dynamic programming. The stochastic dynamic programming method uses models of the distribution of states in the database when selecting samples. The differential dynamic programming approach is similar to the work of Sok and Tessa, in that it uses local linear policies defined about trajectories in the database and adds new trajectories as needed [Sok and Lee, 2007; Tessa et al., 2007]. Both systems rely on efficient and robust trajectory optimization methods that also use information from the trajectory library.

I propose using a stochastic dynamic programming method that samples new points from a distribution defined by the trajectory library. There are multiple ways in which this distribution could be computed. Three possible models for the distribution of points in the trajectory database are a PCA computed from all the states observed in the database, a mixture of Gaussian models derived from EM clustering as in Beaudoin's work [Beaudoin et al., 2007], or a PCA basis computed from the k nearest neighbors of a local point. In this final case, the local point would be chosen randomly from the points already stored in the SDP policy or the trajectory library, and basis computation could be performed efficiently by computing it only once, when the point is added to the database. I will begin by implementing the global PCA method, but expect that moving to the local PCA method will prove more effective, because it is better able to follow the local behavior of the manifold that trajectories in the database tend to lie in. Additionally, temporal coherence information from the database can be used to model the distribution of states that are likely to lead to a given state, rather than simply identifying those states that are occur near a particular state.

A PCA basis computed from states in the trajectory library is not the only source of

potential sampling distributions. It is possible that alternate dimensionality reduction techniques such as ST-ISOMAP or multidimensional scaling will produce more useful subspaces, although this alternative is not an avenue I intend to investigate unless my experiments with the PCA basis approach fail. The stochastic model of the expected distribution of states near a point can also reflect other sources of prior information. If a reduced model has been defined, it could be used as a statistical prior on the distribution of points throughout state space. Likewise, constraints, including the ZMP constraint, can be treated as statistical priors, rather than hard constraints, on the distribution of states the system will occupy.

An additional alternative is to base the distribution on the model's linearized dynamics. A linear model of the system about a point can be derived from the full nonlinear dynamics. Taking the form $\mathbf{x_{n+1}} = A\mathbf{x_n} + B\mathbf{u_n}$, the eigendecomposition of the matrix A can be computed. The eigenvectors of A form an orthonormal basis of the full state space. The associated eigenvalues are either real valued or form conjugate pairs. Each of the eigenvectors with a real eigenvalue and each pair of eigenvectors with a conjugate pair of eigenvalues is an independent mode of the linearized system. The trajectories produced by an optimal linear controller will not exit any subspace constructed from the combination of one or more of these independent modes. This statement implies that, locally, the eigendecomposition of the system provides a good choice for both optimization constraints and weighted bases.

Another way that the trajectory library could be used to improve the performance of stochastic dynamic programming is the construction of initial guesses at optimal trajectories for new points in the policy. In the standard implementation, the existing policy is used to compute an initial trajectory from a sampled point to the goal. If the current approximate policy does not converge, the sampled point is discarded. Interpolation and concatenation of nearby trajectories in the database could be used to produce an alternate initial guess at an optimal trajectory that could then be further refined by trajectory optimization. In this way points further from the existing policy can be selected and the policy can be grown

faster because fewer points will be discarded.

Many possible methods for the generation of an initial guess satisfying the initial and goal constraints are suggested by the graphics literature. Constrained interpolation and motion-graph-based approaches seem to be particularly appropriate. The local statistical model of the distribution of states near the trajectories to be interpolated can be used to improve interpolation by penalizing statistically unlikely motion displacements. Thus if several trajectories are being interpolated, the resulting motion should attempt to maximize the statistical likelihood of the resulting trajectory given the local distributions about each input trajectory.

As an alternative to a full stochastic dynamic programming approach, differential dynamic programming can also be used to produce piecewise linear policies. By using a trajectory optimization method that incorporates differential dynamic programming to process the observed motions in the trajectory library, we can find stable policies near each example trajectory. This approach is very similar to that already suggested by other researchers [Tessa et al., 2007]. Unlike that work however, I am suggesting a two step trajectory optimization method. Because trajectories in the database may violate physical constraints, a more robust trajectory optimization method, such as direct collocation, may be required as a first pass to obtain convergence. Once a trajectory that satisfies the dynamic constraints has been found differential dynamic programming can be used to further refine the trajectory and generate a linear policy in its vicinity.

An iterative method, similar to that used to improve robustness in a purely simulated walking controller [Sok and Lee, 2007], could be used to improve the performance of this method over time. When the system fails, either in simulation or hardware, a new trajectory can be computed by finding a warp between the failed trajectory and a stable motion of the system, starting at a time when the system is believed to have been recoverable.

While modeling the local covariance of state variables can be used to reduce the amount of

sampling needed during policy generation, identifying statistically independent components of the example motions can also be advantageous. Specifically, enforced decomposition of the optimal control problem into two or more largely independent problems can potentially be achieved using independent component analysis (ICA). Enforced decomposition allows weakly coupled systems to be factored into component subsystems that are solved separately [Bertsekas, 2005]. Using either the eigendecomposition of the local linear model of the system dynamics or the result of ICA performed on a local neighborhood could be used to define the bases used for this decoupling. Enforced decoupling and biased sampling can be applied simultaneously, with samples being drawn either from a single non-decoupled distribution or from one or more independent distributions corresponding to the decomposed subsystems.

4.2.2 Trajectory Optimization

My approach requires an efficient method for transforming input trajectories into physically plausible motions. The optimization method I use needs to be robust to poor starting guesses for the optimization. I have three strategies for providing this robustness. The first strategy is to perform an initial optimization on a reduced model that can then be mapped into the full state space without violating physical constraints. Second, the optimization method employed can be switched as the solution converges, as in a Levenberg-Marquardt solution. Last, constraints derived from the trajectory library can guide the optimizer toward convergence. I now discuss each of these strategies in more detail.

If an engineered reduced model of the system exists, it can be used to improve the robustness of the trajectory optimization. This method relies on a mapping between the full system and the reduced system such that physically consistent trajectories in the reduced system map to physically consistent trajectories in the full system. Simple examples of this

type of mapping include fixing the position of particular joints, forcing particular joints to track predetermined trajectories, forcing the system to stay in a particular linear subspace, forcing the parts of the system not in a particular linear subspace to follow a predetermined trajectory, forcing the parts of the system in in a particular linear subspace to be a function of the state of the remainder of the system, and varying that linear subspace as a function of either time or state. In addition to these relatively simple reductions, a hand-designed reduction that permits a direct mapping from the reduced to full system dynamics may be especially useful.

The optimization methods itself can also be switched as the optimization progresses. Both policy generation methods I will discuss require that the final optimizer produce a linear policy about the optimal trajectory. I will use differential dynamic programming (DDP) as the final phase of my trajectory optimization method to produce this policy. Before applying DDP, I will use a direct collocation trajectory optimizer that is able to converge more robustly. It may prove expedient to add a first pass through a dynamic filter algorithm to attempt to generate a feasible but suboptimal trajectory before transitioning to the global trajectory optimization methods. I have already implemented and used both DDP and direct collocation optimizers in my previous work. My DDP implementation is part of Atkeson's stochastic dynamic programming implementation and my direct collocation method is based on the SNOPT nonlinear optimizer.

I will constrain the trajectory optimization further using the basis constraint method described by Safonova [Safonova et al., 2004]. It may be possible to improve this method by computing a weighted k nearest neighbor local PCA basis at each knot point along the trajectory, then constraining the optimization of each knot point to lie in this subspace. If a low dimensional state space manifold that a behavior tends to occupy is nonlinear, then locally linear approximations can capture the manifold as well as a global linear approximation with fewer degrees of freedom at each point. These local constraints can then be

periodically updated as the optimization progresses. This method should work at least as well as the global PCA method, because as k becomes large the local and global PCA bases become identical. I will also consider additionally constraining the optimization using the momentum-curve-based approach described by Abe and colleagues [Abe et al., 2004]

Independent of the constraints used during optimization, the objective function can also be used to improve the quality of the resulting motions. In particular, the objective function criteria used by Safonova and colleagues [Safonova et al., 2003] to preserve general body pose and small oscillations may help preserve natural motion in the new policy. Additionally, adding a term to the objective function that biases the policy to stay near trajectories in the database may have the same effect.

Another avenue for improvement of the trajectory optimization method is the use of methods that vary the dimensionality of the state and action spaces at different times in a manner similar to receding horizon control. For example, instead of selecting the n largest vectors to form a local PCA basis at every node, the number of vectors could vary such that the local basis reconstructed at least some fixed percentage of the local variance. A similar method based on leave one out cross validation has been employed successfully in reconstructing human motion [Chai and Hodgins, 2005]. In this way, the constraints can vary to allow more freedom in regions where the trajectories in the library are highly variable and more constraint in regions where the trajectories lie primarily in a small subspace.

4.2.3 Handcrafted Model Reductions

As discussed above, it may be necessary to use a reduced model of the robot dynamics adapted for the specific task at hand. We consider a number of reduced models developed using our domain specific knowledge of the bongo board. These models can define subspaces for policy generation, provide constraints for trajectory optimization, and aid the design of

hand built controllers.

In earlier work we used a model of the bongo board consisting of a fixed revolute joint connected to a translational joint. The balancer's body was modeled with a two link pendulum connected to that translational joint [Anderson et al., 2007]. This model allows the center of mass of the robot to move both vertically and horizontally and can be stabilized by a linear controller. However, alternate models may reproduce other important features of the bongo board task more accurately.

The bongo board model we now consider begins as an extension of the linear cart-table model proposed by Kajita and colleagues [Kajita et al., 2003]. In this initial, minimal, model of the bongo board, the board is assumed to have negligible mass. Any force applied to the robot's center of mass must pass through the board-roller contact point, because this joint cannot exert any torque. If the board were moved such that the point of contact between the roller and the board was located at $p = x - \frac{z}{g}\ddot{x}$ then there is some force f_c that can be applied at the board-roller contact point to produce the desired horizontal acceleration and no angular or vertical acceleration. Furthermore, if the board is inclined such that the line of action between the center of mass and the roller-board contact point passes through the roller-ground contact point, then there will be no net acceleration of the board and roller. This model is shown in Fig. 4.1.

If it were possible to constantly control the bongo board to match these constraints on its position and orientation, then the problem of balancing on the bongo board could be trivially reduced to the cart-table balance problem. This reduction would be accomplished by using the current offset of the body's center of mass as the input to a controller based on the cart-table model. The output of a cart-table model controller is a horizontal acceleration of the center of mass that corresponds to a desired center of pressure location and ground reaction force. For any center of pressure location and ground reaction force there is a unique pose of the bongo board that results in no acceleration of the bongo board, as described previously.

By placing the bongo board at this location and applying the desired ground reaction force the center of mass can position can be controlled. Of course, the bongo board cannot be moved fast enough or accurately enough to always match these constraints, and its dynamics must be taken into account. An offset between the actual and desired roller position or a board orientation will create additional, undesired, accelerations of the body CoM and the board when the specified reaction force is applied.

A linear model of this system about its equilibrium point can be expanded from the cart-table model:

$$\ddot{x} = \frac{g}{z}(x - c) \tag{4.1}$$

$$\theta = \tan\left(\frac{m\ddot{x}}{mg}\right) = tan\left(\frac{x - c}{z}\right) \tag{4.2}$$

where c is the roller position, x is the center of mass position, and z is the CoM height. The model shown in Fig. 4.1 illustrates the extension of this model to the case where the bongo board cannot be moved instantaneously. Note that we maintain the assumption that there is no net torque about the robot's center of mass. In the diagram, the robot's center of mass is acted on by gravity and a force applied by the bongo board at the roller-board contact point. The force applied by the bongo board is determined by the constraint applied to the center of mass height and angular momentum. The coordinate d is used to represent the translation of the roller, and because of the rolling contact constraint, the offset of the board ($2d$) as well. We derived the full nonlinear dynamics of this system using the Lagrangian method and built a simulation and controller based on these equations. An LQR policy based on a model of the dynamics linearized about a stable pose did not prove to be a sufficiently robust controller for our simulated system. We developed a nonlinear controller that has significantly better performance (Fig. 4.2). A computed torque solution based on inverse dynamics cannot be used directly because the system is underactuated by two degrees

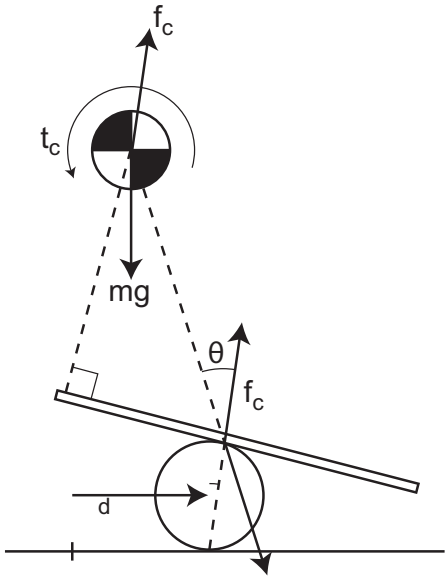


Figure 4.1: Bongo board forces.

of freedom due to the bongo board. The nonlinear controller instead determines only the desired horizontal, vertical, and angular acceleration of the robot's center of mass using a linear control policy. The controller then computes a bongo board position that will allow the force required to produce this desired acceleration to be applied to the robot without accelerating the bongo board. This position is used as the desired position for a second linear control policy that determines a desired acceleration for the bongo board. The inverse dynamics model is then used to compute torques that produce this bongo board acceleration and drive the center of mass height towards its desired position. Because of the dynamic singularity at the desired bongo board position, as the bongo board approaches its desired position the angular and horizontal acceleration of the body center of mass approach their desired values as well. This controller is capable of stabilizing the simulated bongo board system with large initial disturbances that include static resting poses in which one of the ends of the board is resting on the ground.

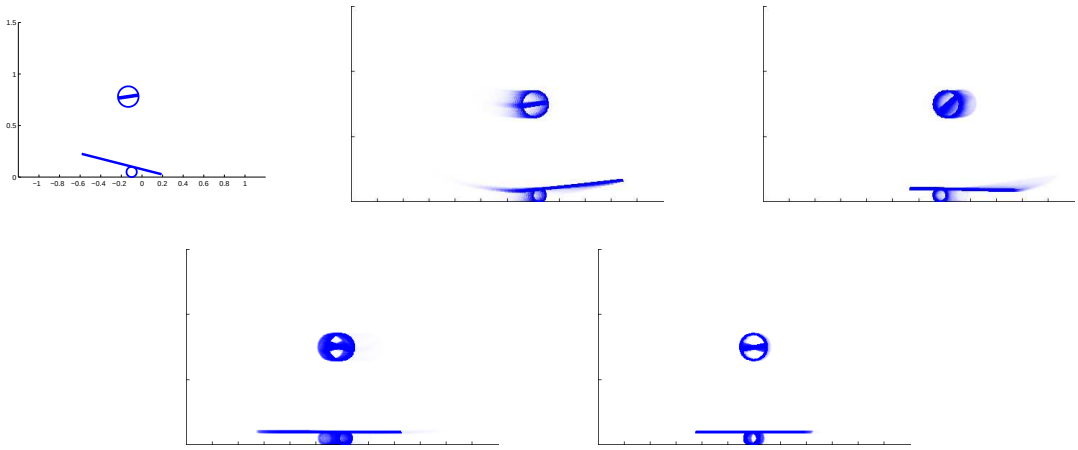


Figure 4.2: Simulated trajectory of the nonlinear bongo board controller. The images, from left to right, show the system in its initial state, then at times rangine from from zero to one seconds, from one to three seconds, from three to ten seconds, and, finally, from ten to twenty seconds.

4.3 Transferring to Hardware and Improving Performance

Having identified methods for creating a trajectory library and generating a linear piecewise policy from that library and a model of the robot, a final component of this approach is needed to create a controller than can stabilize the physical robot. I suggest two methods of model learning and two approaches to incrementally gathering data and improving the existing policy.

I intend to use both parametric and non-parametric model learning simultaneously and to update models offline. The full model of the dynamics I will use is

$$\dot{x} = F(x, u) = F_p(x, u, p) + F_l(x, u, D) \tag{4.3}$$

where F_p is a parametric rigid body model of the robot's dynamics, F_l is a local nonparametric model of the robot's residual dynamics, x is the current state of the robot, u is the current

valve command, p is the vector of model parameters, and D is the database of observations. The parametric model of the rigid body dynamics will be learned first using the least squares regression methods described by An, Atkeson, and Hollerbach [An et al., 1988] and the parametric model of the actuator dynamics will be based on the actuator compensation method developed by Bentivegna [Bentivegna et al., 2007]. Once a parametric model of the robot dynamics has been identified, the residual errors between the predicted and observed dynamics will be represented using a receptive field weighted regression model [Schaal and Atkeson, 1997].

In order to employ these model learning techniques the robot has to perform actions to gather data. If an existing controller is available then I will use it to gather data for model building. If no such controller is available, but the task affords some equilibrium state, then a linear controller can be automatically derived to collect an initial dataset. Once an initial dataset has been created it may be necessary to iteratively acquire new data and update the control policy based on this data.

It is possible that, as suggested by Atkeson in [Atkeson et al., 1997b], the nonparametric component of the model may be aggressively exploited by a trajectory optimizer resulting in undesired results. If this occurs then I will attempt to modify the trajectory optimization to use only the parametric model to compute the trajectory cost, while using the full model to compute the output policy, and ensuring that constraints are satisfied by both the parametric and full models. This change should have the effect of smoothing the objective function while still producing trajectories and policies that are viable in the full model. Additionally, it may be necessary to alter the cost functional to include a penalty on the distance from the set states for which data was collected.

Chapter 5

Workplan

I will begin by developing a controller based on a reduced model similar to the augmented cart-table model discussed above that allows the Sarcos biped to balance on the bongo board. This process will familiarize me with the task as well as produce data for building a better model of the system. I will document and publish the details of the control implementation that enables this performance. The bongo board may be augmented with additional sensing (i.e. motion capture markers or a force plate) as needed.

As described in the previous chapter, this controller has already been implemented and tested with a simulation using a lump mass model for the robot dynamics. Transferring this controller to operate on the full robot requires a mapping from a model of the full robot to the reduced model. We construct this mapping using locally weighted linear regression. We designed quasistatic trial trajectories for the robot that move its center of mass horizontally and vertically. Recording the robot's center of pressure as it performs these trajectories provides information about its center of mass location in the directions perpendicular to gravity. By performing these trajectories both on level ground and on known inclines we can reconstruct the 3D position of the center of mass. The data from the performance of these trajectories provides a mapping between joint positions and center of mass position.

In order for the mapping to remain accurate, the robot's pose must remain close to the poses recorded in the hand designed trajectories. This constraint is met by applying a joint acceleration that minimizes a weighted sum of the difference between the desired and actual CoM accelerations and the difference between an acceleration directly towards a pose in the hand designed joint trajectory and the actual joint acceleration.

As described, this process does not produce a model of angular acceleration. I plan to approximately determine, via experiment, the ratio between upper and lower body angular inertia in a reference posture. This ratio will then be used to drive the upper body in an attempt to maintain a near-zero angular momentum of the whole body. The effect of upper body rotation on center of mass position will be included in the LWR model of the dynamics.

Next, I will experiment with producing a simulated controller based on the trajectory data collected from the robot as it balanced on the bongo board using the policy generation approach discussed previously. The important result from the experiments will be a method that gives a policy that is stable in a large region of the task space.

Once a working simulation of the piecewise linear policy exists I will begin learning better parametric and nonparametric models of the robot based on data collected from the hand written controller. I will the use the already completed policy generation tool with this new model, and confirm that it operates as expected in simulation. Next, I will run the handwritten controller and the linear piecewise policy simultaneously on the robot, comparing the output commands to ensure that their behavior is similar.

Once I am satisfied that the policy is stable in simulation and produces commands roughly similar to the handwritten controller when run on the robot, I will use the policy to stabilize the robot directly. This step may require several iterations of model improvement and policy adaptation or other unforeseen changes to the method.

At this point I will introduce motion capture trajectories to the trajectory library. I will try generating a policy using only the motion capture trajectories, generating a policy using

both the motion capture and directly recorded trajectories as the initial set, and updating the policy generated from only the directly recorded trajectories using the motion capture data.

I will attempt to develop the second controller for a different task without developing a hand-designed controller first. I may, however, design a state space reduction and mappings to and from that reduction. The task will be chosen to facilitate model development without the need for bootstrapping data for the initial model. This task will be a whole body task involving contact state changes. Examples of this type of task include dance, soccer ball manipulation, and walking using rolling foot contact. I will select this task after completing the bongo board task asa that work will provide greater insight into the power and limitations of my approach.

5.1 Evaluation

I will evaluate my success by providing indicators of the degree to which the use of the methods described in this proposal improved the naturalness and stability of the robot's behavior, and the ease with which those behaviors were implemented.

Because I will have a hand written controller based on a reduced model for the bongo board task, I can perform meaningful comparisons between this controller and the results of my policy generation method. In particular, I will be able to compare the performance of the handwritten controller against a controller derived from a trajectory library containing only data generated by the handwritten controller, or human motion capture data, or both. I will use both qualitative and quantitative measures of stability in my evaluation, such as maximum recoverable lateral impulse. I will evaluate the naturalness of the motion qualitatively and may present videos to human subjects in a perceptual experiment to provide some measure of the perceived difference between the controllers and the original motion

capture data.

By adding and removing specific segments of a captured database, I will be able to determine whether each component of the motion database contains important information. For example, removing disturbance and recovery motions from a database might result in recoveries that appear unnatural or are less robust.

In addition to evaluating the overall success of the policy generation method I develop, I will examine the behavior of the individual components that comprise the method. The trajectory database's coverage can be evaluated by comparing the average distance between states of the system as it executes a computed policy and the states in the initial database.

A direct evaluation of the quality of the approximation of the policy produced by stochastic dynamic programming to the true optimal policy is impossible without knowing the true optimal policy. However, the effect that the sampling bias has on the resulting policy can be evaluated by generating policies with increasingly uniform sampling distributions and comparing the change in average cost-to-goal for a set of start conditions. The value of using motion interpolation to seed trajectory optimization can be evaluated by comparing the percent of convergent runs over a set of start and goal conditions for several methods of generating an initial guess. These methods can include interpolation based on local variance estimates, linear interpolation, the trajectory that minimizes the sum distance to the start and goal location, and a simple linear interpolation between the start and goal location. Finally, policy generation methods that employ enforced decomposition can be compared against other methods that use coarser or finer decompositions of the system.

The model learning procedure can be evaluated by comparing the system's performance using each component of the modelling process individually and in combination. These comparisons would entail building policies based on only the parametric model, only the local learning model, and the combination of both the parametric and local models. Additionally, the effects of the initial model data on the eventual policy can be evaluated by constricting

the initial model to a small region around the goal and comparing the resulting policy after additional model learning with the policy resulting from a larger initial data set.

5.2 Contributions

This work presents a new approach to developing control policies for humanoid robots. This approach is based on using trajectory libraries in the role that a hand designed, task-specific, simplified model of the robot now serves. Successfully demonstrating the viability of this approach will decrease the time and expense required to develop a new behavioral controller for a humanoid robot. The work makes the following specific contributions:

- The use of statistical models of trajectory libraries to bias the sampling distribution used in stochastic dynamic programming.
- The use of statistical models of trajectory libraries to constrain interpolation between trajectory examples.
- Enforced decoupling during optimization incorporating information from a trajectory library and a model of the system dynamics.
- Potentially, novel techniques for optimizing and generating humanoid motion trajectories.

I will produce the following end products to demonstrate these contributions:

- A working robot controller for a balance task, hopefully the bongo board, possibly standing and stepping. The controller will be based on the policy generation methods described in this paper.
- A working robot controller for a whole body task involving contact state changes. This task may be a dance and may be very simple and involve only minimal change

in contact state, but will ideally involve multiple changes in contact state coupled with dynamic motions of the upper body. Alternative possibilities involve basic soccer ball manipulation (kick and block) and walking with use of degenerate contact states (heel and toe contacts). Any object manipulation may require additional sensing to determine the state of the object.

- An analysis of the utility of the trajectory-library-based techniques in improving performance, as measured by disturbance rejection, and ease of implementation, and perception of naturalness.

Bibliography

Pieter Abbeel, Varun Ganapathi, and Andrew Y. Ng. Learning vehicular dynamics, with application to modeling helicopters. In *NIPS 18*, 2006.

Y. Abe, C. Liu, and C. Popovic. Momentum based parameterization of dynamic character motion. In *In Symposium on Computer Animation*, pages 173–182, 2004.

Chae H. An, Christopher G. Atkeson, and John Hollerbach. *Model-Based Control of a Robot Manipulator*. MIT Press, 1988.

S. O. Anderson and S. S. Srinivasa. Identifying trajectory classes in dynamic tasks. In *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 172–177, April 2007. doi: 10.1109/ADPRL.2007.368185.

S. O. Anderson, M. Wisse, C. G. Atkeson, J. K. Hodgins, G. J. Zeglin, and B. Moyer. Powered bipeds based on passive dynamic principles. In *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, pages 110–116, December 2005. doi: 10.1109/ICHR.2005.1573554.

S.O. Anderson, C.G. Atkeson, and J.K. Hodgins. Coordinating feet in bipedal balance. In *6th IEEE-RAS International Conference on Humanoid Robots*, pages 624 – 628, November 2006.

S.O Anderson, C.G. Atkeson, Hodgins J.K., A. Mahboobin, P.J. Laughlin, and M.S. Redfern. Sensory adaptation in human balance control: Lessons for biomimetic robotic bipeds. In *Unpublished*, 2008.

Stuart O Anderson, Jessica K Hodgins, and Christopher G Atkeson. Approximate policy transfer applied to simulated bongo board balance. In *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*, 2007.

T. Asfour, K. Regenstein, P. Azad, J. Schroder, A. Bierbaum, N. Vahrenkamp, and R. Dillmann. ARMAR-III: An integrated humanoid platform for sensory-motor control. In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 169–175, Genova, December 2006. doi: 10.1109/ICHR.2006.321380.

C. G. Atkeson and J. Morimoto. Non-parametric representation of a policies and value functions: A trajectory based approach. In *Advances in Neural Information Processing Systems*, volume 15, 2003.

C. G. Atkeson and B. Stephens. Random sampling of states in dynamic programming. In *Advances in Neural Information Processing Systems*, 2007.

C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11:11–73, 1997a.

Christopher G. Atkeson, Andrew W. Moore, and Stefan Schaal. Locally weighted learning for control. *Artificial Intelligence Review*, 11(1-5):75–113, 1997b. URL `citeseer.ist.psu.edu/article/atkeson96locally.html`.

J. Andrew Bagnell and Jess G. Schneider. Autonomous helicopter control using reinforcement learning policy search methods. In *Proc. Int. Conf. on Robotics and Automation*, May 2001.

J. Barbic, A. Safonova, J. Pan, C. Faloutsos, J. Hodgins, and N. Pollard. Segmenting motion capture data into distinct behaviors. In *Proceedings of Graphics Interface*, 2004. URL `citeseer.ist.psu.edu/article/barbic04segmenting.html`.

Philippe Beaudoin, Michiel van de Panne, and Pierre Poulin. Automatic construction of compact motion graphs. Technical Report 1296, DIRO, Universite de Montreal, May 2007.

D.C. Bentivegna and C.G. Atkeson. Learning from observation using primitives. In *IEEE International Conference on Robotics and Automation, Proceedings of.*, pages 1988–1993, 2001.

D.C. Bentivegna, C.G. Atkeson, and J.Y. Kim. Compliant control of a hydraulic humanoid joint. In *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*, 2007.

Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 2005.

R. Blickhan and R. J. Full. Similarity in multilegged locomotion: Bouncing like a monopode. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, 173(5):509–517, November 1993.

M.S. Branicky, T.A. Johansen, I. Petersen, and E. Frazzoli. On-line techniques for behavioral programming. In *Proceedings of the 39th IEEE Conference on Decision and Control*, pages 1840–1845, 2000.

Thomas Buschmann, Sebastian Lohmeier, Mathias Bachmayer, Heinz Ulbrich, and Friedrich Pfeiffer. A collocation method for real-time pattern generation. In *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*, 2007.

Jinxiang Chai and Jessica K. Hodgins. Performance animation from low-dimensional control signals. *ACM Transactions on Graphics (SIGGRAPH 2005)*, 24(3), August 2005.

Joel Chestnutt, Philipp Michel, Koichi Nishiwaki, James Kuffner, and Satoshi Kagami. An intelligent joystick for biped control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 860 – 865, May 2006.

Min Gyu Choi, Jehee Lee, and Sun Yong Shin. Planning biped locomotion using motion capture data and probabalistic roadmaps. *ACM Transactions on Graphics (SIGGRAPH 2003)*, 22(2):182–203, April 2003.

A. Dasgupta and Y. Nakamura. Making feasible walking motion of humanoid robots from human motion capture data. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 2, pages 1044–1049, Detroit, MI, May 1999. doi: 10.1109/ROBOT.1999.772454.

M. Erdmann and M. Mason. An exploration of sensorless manipulation. *IEEE J. Robot. Automat.*, 4(4), August 1988.

A. Fang and N. Pollard. Efficient synthesis of physically valid human motion. In *ACM Transactions on Graphics, 22, 2003.*, 2003.

A. Fod, M.J. Mataric, and O.C. Jenkins. Automated derivation of primitives for movement classification. *Autonomous Robots*, 1:39–54, January 2002.

M. Garcia, A. Chatterjee, A. Ruina, and M. J. Coleman. The simplest walking model: Stability, complexity, and scaling. *ASME J. Biomech. Eng.*, 120(2):281–288, April 1998.

Jr. Golliday, C. and H. Hemami. An approach to analyzing biped locomotion dynamics and designing robot locomotion controls. *IEEE Transactions on Automatic Control*, 22(6): 963–972, December 1977. ISSN 0018-9286.

A. Goswami, B. Espiau, and A. Keramane. Limit cycles and their stability in a passive bipedal gait. In *Proc., IEEE Int. Conf. on Robotics and Automation*, Piscataway, NJ, 1996.

J. K. Hodgins and M. N. Raibert. Adjusting step length for rough terrain locomotion. *IEEE Trans. Robotics and Automation*, 7(3):289 – 298, 1991.

J.K. Hodgins. Biped gait transitions. In *Proc., IEEE Int. Conf. on Robotics and Automation*, 1991.

D.H. Jacobson and D.Q. Mayne. *Differential Dynamic Programming.* Elsevier, 1970.

O.C. Jenkins and M.J. Mataric. Automated derivation of behavior vocabularies for autonomous humanoid motion. In *Autonomous Agents and Multi Agent Systems, Proceedings of.*, pages 225–232, 2003.

Odest C. Jenkins and Maja J Matarić. A spatio-temporal extension to isomap nonlinear dimension reduction. In *Proc., Int. Conf. on Machine Learning*, pages 441–448, July 2004.

Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Kazuhito Yokoi, and Hirohisa Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *Proc., IEEE Int. Conf. on Robotics and Automation*, 2003.

Utku Seven Kemalettin Erbatur. An inverted pendulum based approach to biped trajectory generation with swing leg dynamics. In *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*, 2007.

Jung-Yup Kim, Ill-Woo Park, and Jun-Ho Oh. Experimental realization of dynamic walking of biped humanoid robot khr-2 using zmp feedback and inertial measurement. *Advanced Robotics*, 20(6):707 – 736, June 2006.

Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. *ACM Transactions on Graphics*, 21(3):473–482, July 2002.

James Kuffner, Koichi Nishiwaki, Satoshi Kagami, Masayuki Inaba, and Hirochika Inoue. Motion planning for humanoid robots. In *Proc. 11th Intl. Symp. of Robotics Research*, 2003.

Ryo Kurazume, Shuntaro Tanaka, Masahiro Yamashita, Tsutomu Hasegawa, and Kan Yoneda. Straight legged walking of a biped robot. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2005.

S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, May 2001.

Steven M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical Report 98-11, Iowa State University, Ames, IA, 50011, USA, 1998.

Jehee Lee, Jinxiang Chai, Paul Reitsma, Jessica Hodgins, and Nancy Pollard. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics (SIGGRAPH 2002)*, 21(3), July 2002.

C. Karen Liu and Zoran Popović. Synthesis of complex dynamic character motion from simple animations. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 408–416, New York, NY, USA, 2002. ACM Press. ISBN 1-58113-521-1. doi: http://doi.acm.org/10.1145/566570.566596.

M. Mason. The mechanics of manipulation. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 544–548, March 1985.

M.J. Mataric, M. Williamson, J. Demiris, and A. Mohan. Behavior-based primitives for articulated control. In *Fifth Interational Conference on Simulation of Adaptive Behavior, Proceedings of.*, pages 165–170. MIT Press, 1998.

T. McGeer. Passive dynamic walking. *Int. J. Robot. Res.*, 9(2):62–82, April 1990.

J. Morimoto, G. J. Zeglin, and C. G. Atkeson. Minimax differential dynamic programming: application to a biped walking robot. In *Proc., Int. Conf. on Intelligent Robots and Systems*, volume 2, pages 1927 – 1932. IEEE, 2003.

David D. Morrison, James D. Riley, and John F. Zancanaro. Multiple shooting method for two-point boundary value problems. *Commun. ACM*, 5(12):613–614, 1962. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/355580.369128.

Yu Ogura, Kazushi Shimomura, Hideki Kondo, Akitoshi Morishima, Tatsu Okubo, Shimpei Momoki, Hun ok Lim, and Atsuo Takanishi. Human-like walking with knee stretched, heel-contact and toe-off motion by a humanoid robot. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3976–3981, Beijing,, October 2006. doi: 10.1109/IROS.2006.281834.

Jonghoon Park and Youngil Youm. General zmp preview control for bipedal walking. In *Proc., IEEE Int. Conf. on Robotics and Automation*, 2007.

N. S. Pollard and P. S. A. Reitsma. Animation of humanlike characters: Dynamic motion filtering with a physically plausible contact model. In *Yale Workshop on Adaptive and Learning Systems*, 2001.

Zoran Popović and Andrew Witkin. Physically based motion transformation. In *SIGGRAPH*, pages 11–20. ACM, 1999. ISBN 0-201-48560-5.

M. H. Raibert. *Legged robots that balance*. The MIT Press, Cambridge, Massachusetts, 1986. ISBN 0-262-18117-7.

John Rebula, Fabián Ca nas, and Jerry Pratt. Learning capture points for humanoid push recovery. In *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*, 2007.

C. Rose, M. F. Cohen, and B. Bodenheimer. Verbs and adverbs: multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, 18(5):32–40, September/October 1998. ISSN 0272-1716. doi: 10.1109/38.708559.

Alla Safonova, Nancy S. Pollard, and Jessica K Hodgins. Optimizing human motion for the control of a humanoid robot. In *Proceedings of the 2nd International Symposium on Adaptive Motion of Animals and Machines (AMAM2003)*, March 2003.

Alla Safonova, Jessica K Hodgins, and Nancy Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics (SIGGRAPH 2004)*, 23(3), August 2004.

S. Schaal and C. Atkeson. Receptive field weighted regression. Technical Report 209, 1997. URL `citeseer.ist.psu.edu/schaal97receptive.html`.

E. Schuitema, D. G. E. Hobbelen, P. P. Jonker, M. Wisse, and J. G. D. Karssen. Using a controller based on reinforcement learning for a passive dynamic walking robot. In *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, pages 232–237, December 2005. doi: 10.1109/ICHR.2005.1573573.

Adam L. Schwartz. *Theory and Implementation of Numerical Methods Based on Runge-Kutta Integration for Solving Optimal Control Problems*. PhD thesis, University of California at Berkeley, Berkeley, CA, 1996.

L. Sentis and O. Khatib. A whole-body control framework for humanoids operating in human environments. *2006. ICRA 2006. Proceedings 2006 IEEE International Conference on Robotics and Automation*, pages 2641–2648, May 2006. ISSN 1050-4729.

Hyun Joon Shin and Jehee Lee. Motion synthesis and editing in low-dimensional spaces. *Computer Animation and Virtual Worlds (Special Issue: CASA 2006)*, 17(3-4):219 – 227, July 2006.

Takaaki Shiratori, Shunsuke Kudoh, Shin'ichro Nakaoka, and Katsushi Ikeuchi. Temporal scaling of upper body motion for sound feedback system of a dancing humanoid robot. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007)*, 2007.

Kwang Won Sok, Manmyung Kim, and Jehee Lee. Simulating biped behaviors from human motion data. *ACM Transactions on Graphics*, 26(3):105:1–105:10, July 2007.

R. F. Stengel. *Optimal Control and Estimation*. Dover, New York, 1994.

M. Stilman, C. G. Atkeson, J. Kuffner, and G. Zeglin. Dynamic programming in reduced dimensional spaces: Dynamic planning for robust biped locomotion. In *Proc., IEEE Int. Conf. on Robotics and Automation*, Barcelona, Spain, 2005.

Yuval Tessa, Tom Erez, and William D. Smart. Receding horizon differential dynamic programming. In *Advances in Neural Information Processing Systems*, 2007.

O. von Stryk and R. Bulirsch. Direct and indirect methods for trajectory optimization. *Ann. Oper. Res.*, 37(1-4):357–373, 1992. ISSN 0254-5330.

M. Vukobratovic. How to control the artificial anthropomorphic systems. *IEEE Trans. System, Man and Cybernetics SMC-3*, pages 497–507, 1973.

M. Wisse, D.G.E Hobbelen, R.J.J Rottevell, S.O. Anderson, and G.J. Zeglin. Ankle springs instead of arc-shaped feet for passive dynamic walking. In *6th IEEE-RAS International Conference on Humanoid Robots*, pages 110 – 116, November 2006.

Andrew Witkin and Michael Kass. Spacetime constraints. In *Computer Graphics (Proceedings of SIGGRAPH 88)*, pages 159–168, August 1988.

Katsu Yamane. *Simulating and Generating Motions of Human Figures*. Springer tracts in advanced robotics 9. Springer, 2006.

Katsu Yamane, James J. Kuffner, and Jessica K. Hodgins. Synthesizing animations of human manipulation tasks. *ACM Transactions on Graphics (SIGGRAPH 2004)*, 23(3), August 2004.

Kangkang Yin, Kevin Loken, and Michiel van de Panne. Simbicon: Simple biped locomotion control. *ACM Transactions on Graphics*, 26(3):105:1–105:10, July 2007.