2. BACKGROUND

This chapter presents the background information required to understand later chapters in the thesis. First, important definitions are introduced. Previous related work is then reviewed, followed by a description of the underlying control representation on which our balance control approach is based. Finally we describe our animation system and our biped models.

2.1 Definitions

A number of important terms and acronyms are used throughout the thesis. Their definitions and descriptions can be found in Appendix A. The majority of the terms are commonly used terms in the robotics and biomechanics literature. More in-depth information can be found in [HR86], [SV89], [Fr86] and [IRT81].

2.2 Previous Work

Bipedal locomotion is a topic of interest to a number of disciplines. This section describes a representative subset of the work in these fields. First, we provide an overview of the various approaches to motion generation in computer animation. This is followed by discussion of work specific to bipedal locomotion in computer animation, biomechanics and robotics. Finally, some relevant work in the control literature is addressed.

2.2.1 Kinematic Animation

Research in computer animation has evolved significantly in its relatively short life span. The earliest approaches to computer animation use *keyframing*, a technique based on classical animation. In keyframing, the configuration of the animated objects at various points in time is specified by the animator and the computer generates the in-between frames using linear or other forms of interpolation. In early systems, specification of keyframes required the animator to directly manipulate the DOFs of an object [Mez68, BW71, Csu71, KB84, Stu84, MTT85b, SB85, Las87]. Later systems allowed the animator to specify the position of specific points on the objects being animated (such as a hand or foot) and used inverse kinematics to determine the appropriate values for the creature's internal DOFs [KB82] [GM85] [BMW87]. Procedural descriptions of motion, often based on real-world data and observations, can be used to model very specific classes of movement, effectively "programming" the animated movement [Zel82] [GM85]. In all cases, the quality of the resulting motion is heavily dependent on the ability of the animator who is responsible for ensuring that the perceived dynamics of the motion are appropriate. This is a task which requires significant skill. It potentially distracts the animator from the primary task at hand, but it also allows him or her complete artistic control.

Rotoscoping and motion capture are techniques commonly used to obtain kinematic data from real-world sources. Directly recording a phenomenon to be animated guarantees realistic and natural-looking motion. Specialized hardware is generally required, but such equipment is becoming more accessible. A number of problems with this approach make the investigation of other motion generation techniques desirable. First, captured motions are limited to real-world motions that can easily be recorded. Essentially, motion capture has many of the same restrictions as live actors. Also, approaches to parameterizing captured motions often produce results that are no longer fully realistic. Physical constraint violations, such as ground interpenetration and sliding are common examples of failure. While solutions to enforcing such constraints for particular classes of motion have been demonstrated [BMTT90] [KB93], no general solution currently exists. More recent parameterization approaches seem oriented toward more broadly modifying captured motions and are likely to have similar problems [BW95] [WP95] [UAT95]. Finally, captured motions cannot easily be modified to respond realistically to environments different from the one in which they are obtained. Varying terrain and collisions are two examples of such potentially desirable changes. As the demand for fully interactive environments increases, this issue becomes more important. In recent years the interactive home-entertainment industry has begun to exceed the motion picture industry in consumer revenues. Today's interactive video games are being made with the same effort as low-budget movies, incorporating complex scripts and storylines, well-known actors, and mixing real and computer-generated imagery [Sni95].

2.2.2 Dynamics-based Animation

The techniques used to integrate physics into the generation of computer animations can be divided into two basic approaches, *trajectory-based* and *controller-based*.

Trajectory-based techniques such as [WK88], [BN88], [Coh92] and [LGC94] attempt to find a physically realistic or near-realistic trajectory from one point in the state space of a creature to another. Since the systems are typically highly underconstrained, the trajectory is usually optimized in some way, for example for smoothness, minimum control energy or minimum time. A disadvantage of the technique is that a new trajectory must be generated for each new desired instance of motion. Also, interactions with the environment such as collisions and friction are often difficult to properly incorporate into the dynamics specification. One decided advantage of trajectory-based techniques is that they relate well to keyframing. The animator can control details of the end result through the specification of keyframes and other trajectory-based constraints. Trajectory-based techniques are also able to find the most physically plausible solution, even if no completely physical solution is possible.

In the controller-based approach, a dedicated *controller* or *control algorithm* is used to activate the simulated muscles of a creature, causing it to perform some motion or action within a simulated environment. The use of such controllers has a number of advantages over the trajectory-based approach. In many cases, controllers can be designed to be *reusable* and *composable* [van89] [Hod91] [SC92]. Reusability implies that a controller can be used to achieve a given motion with a variety of initial states. Composability implies that a sequence of motions can be generated by switching between several controllers over time, possibly subject to some form of transition requirements.

Early versions of controllers for animation used force and torque functions, specified either by the user [Wil86] [AGL87] [FW88] or based on measured or observed data [MZ90] [Mil88]. Other controllers are based on state machines, dividing the motion into a number of phases, each of which is represented by a single state. Controllers are designed by hand in many cases [RH91] [SC92] [HSL92] [H+95]. Hand-designed controllers require the use of carefully chosen parameters to simplify the control program and are typically specific to a particular type of motion (e.g. hopping).

Controllers can also be automatically synthesized. Automatic synthesis uses various stochastic search strategies to explore the space of possible controllers [VF93] [vKF94] [vKF94b] [vL95] [NM93] [A+95] [Sim94] [GT95]. Each controller is assigned a fitness value which characterizes its "goodness" and a mechanism is provided for keeping and refining good controllers and eliminating poor ones. In [Sim94], the structure of the creature itself is allowed to evolve, as well as the controller. Current automatic synthesis techniques are best at finding controllers for relatively stable creatures and motions such as a crawling ant or motion in a single plane. This is because they rely on the fact that a good first guess can be stochastically determined with a reasonable amount of computation. A relatively smooth fitness function is also typically required to allow incremental progress toward an acceptable solution. Unstable motions such as human walking do not meet these requirements since the solution space is exceedingly small compared to that of a more stable creature and motion, especially when motion in 3 dimensions is desired.

The use of motion controllers increases the autonomy of the creature being animated, thereby requiring less direct animator intervention as compared to kinematic and trajectory-based approaches. The cost of this increased autonomy is in the degree of control the animator has over the final motion. Allowing the animator to specify that the biped "follow that taxi" often removes the animator's choice of which precise path to take. The tradeoff between autonomy and degree of animator control is illustrated in Figure 2.1.

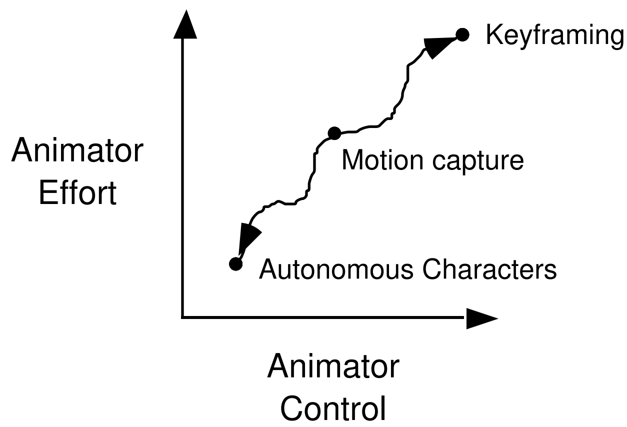


Figure 2.1 - Animator control vs autonomy

2.2.3 Bipedal Locomotion

The animation of bipedal locomotion has long been a topic of fascination to many. Zeltzer [Zel82] presents a hierarchical task-oriented animation system in which the low-level walking motions are implemented kinematically, based on measured human data. Girard and Maciejewski [GM85] use rules associated with dynamics (rather than dynamics simulations) for torso motion and inverse kinematics for leg motion to generate one of the first non-rotoscoped, natural looking walks. Bruderlin and Calvert [BC89] break each step into a number of kinematically-defined subphases based on known human gait mechanics and use simplified dynamics simulation to generate the motion in between each subphase. By allowing the user to vary a number of *gait determinants*, a wide variety of natural-looking walks can be generated. Since in this approach, the dynamics are highly constrained, replacing the dynamic interpolation with kinematic interpolation [BC93] is found to give results of similar quality while increasing performance significantly, allowing gait parameters to be adjusted interactively. This work currently represents the state-of-the-art in real-time, parameterized kinematic models of natural looking human walking motion. A similar technique has also been applied to human running [BC96]. Badler [BPW93] uses primarily kinematic techniques as well as rotoscoped data with dynamic enhancements to achieve many human motions and behaviours. Boulic, Magnenat-Thalmann and Thalmann [BMTT90] and Ko and Badler [KB93] present techniques to generalize rotoscoped or motion captured walking data to other subjects and step lengths while reducing or eliminating the resulting ground constraint violations.

Raibert and Hodgins [RH91] use full dynamical simulation with robust hand-crafted hopping control to attain various bounding gaits for biped and quadruped robot models. As well, a similarly controlled planar kangaroo model is shown to compare well to its real-world counterpart. Stewart and Cremer [SC92] use their flexible constraint-based approach to generate fully dynamic 3D bipedal walking on level terrain and up a flight of stairs. One of the required constraints, however, is a 0 DOF "magnetic boot" on the stance foot. Van de Panne, Fiume and Vranesic [VFV92] use optimal state-space control tables to control walking on level terrain and up and down ramps, smooth curved surfaces and stairs for a planar biped model. This approach requires a suitable control decomposition to make the generation of the state-space controllers tractable.

Auslander et al. demonstrate automatic synthesis of interesting 2D bipedal walking and tumbling motions but meet with difficulty in their initial attempts to extend this approach directly to 3D [Aus+95]. Van de Panne and Lamouret propose the use of guiding external forces to initially attain reasonable controllers using similar automatic synthesis [vL95]. The forces are then reduced in a number of steps and can sometimes be entirely eliminated to yield a fully-balanced, automatically synthesized motion. Examples of human walking, skipping and running and walking over varying terrain for a simple 3D biped are given. One difficulty with this approach is that the removal of guiding forces must be performed incrementally over the entire motion sequence (for example, each step of a walk). This process that can become prohibitively expensive for more complex creatures. Hodgins et al. [H+95] show how Raibert's hopping principles along with various forms of additional control can be applied to running for a fully-dynamic, complex human model. The ground model for the runner uses a 1 DOF constraint which allows motion only in the body's pitch DOF.

Legged locomotion has also received significant attention in the robotics and biomechanics literature. Research in the biomechanics literature is directed primarily toward gait analysis. Of particular interest is the efficiency of natural motion in humans and animals [McM84] [Ale84] and the identification of various *determinants of gait* and their role in normal and pathological gaits [TT76] [SCD80] [MM80] [IRT81] [SSH82] [PB89]. To this end, a number of dynamic locomotion models have been proposed [VJ69] [MM80] [McM84] [Tow85] [PB89]. These approaches generally make assumptions which limit their usefulness for animation. Typical assumptions include a simplified biped model and/or motion only in a plane [VJ69] [TT76] [Tow85] [McM84] [PB89]. Some only consider the open-loop motion over one or two steps [TT76] [McM84] [PB89].

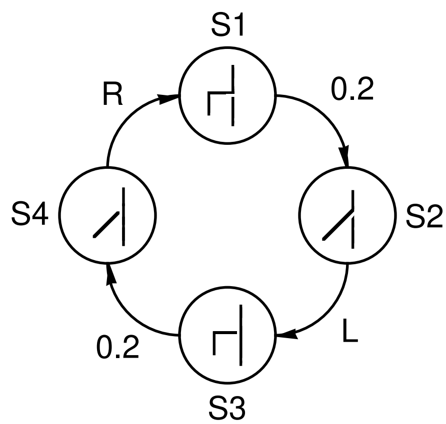
The robotics literature has more in common with the goals of physically-based computer animation than biomechanics does. It has as an objective the synthesis of legged locomotion, rather than analysis. While some works present only simulation results and others implement real robots, all systems, of necessity, incorporate some form of forward dynamics. Many approaches propose various reduced-order models for the equations of motion and rely on reference trajectories, typically based either on the motion of an inverted pendulum [FM84] [MS84] [KKI90] [KT91] or on measured human data [VS72] [HF77]. As with biomechanics, the complexity of the locomotion problem is often reduced through the use of simplified biped models. Constraining motion to the sagittal plane, is perhaps the most common simplification [HF77] [FM87] [KKI90] [KT91] [CHP92]. A number of approaches deal only with statically stable walking motions in which the biped is balanced at all points in the walk cycle [HF77] [ZS90] [SZ92]. While such motions may be quite useful for a robot due to their inherent stability, they are of less interest to animators. Miura and Shimoyama present a stilt-leg biped that walks dynamically in 3D on point feet [MS84]. Takanishi et al. [TIYK85] achieve a dynamic but very rough, lurching 3D walk for a robot with anthropomorphic legs. Furusho & Sano [FS90] demonstrate the use of sensor-based feedback to produce smoother motions from a similar gait. Raibert et al. [Rai+84] present an elegant three-way decomposition of control to accomplish robust one-legged hopping in three dimensions which is later extended to bipedal and quadrupedal models using the notion of a *virtual leg* [Rai86] [Rai86b].

2.2.4 Limit Cycle Control

A number of papers view bipedal walking and running motions as limit cycles in state space. These are most closely related to the work in this thesis. McGeer [McG89] [McG90] [McG90b] demonstrates that various forms of passive legged locomotion, such as walking with and without knees and running can exist as natural modes of a mechanical device. By using Newton's method to search for motions which have identical initial and final system states, stable gaits could be found for a system which uses only a small downhill slope as a source of energy. Katoh and Mori [KM84] use high-gain PD control to drive a biped's motion toward a prescribed cyclic state space trajectory. Hmam and Lawrence [HL91] use nonlinear feedback control to drive a running biped onto a prescribed trajectory which is based on the passive motion of the system. The feedback is used to improve the robustness of the system to perturbation. These latter two works use very simple biped models and all three assume strictly planar dynamics.

2.3 Pose Control

The fundamental control representation used throughout this thesis is the pose control graph, or PCG [vKF94]. Figure 2.2 shows a typical PCG, which is essentially a specialized type of finite state machine. Pose control provides a compact way to specify the torques to be applied to an articulated figure in order to attain a desired motion. Each state in the PCG specifies a set of desired joint angles for the creature with respect to some fixed reference position, called a *desired* *pose*, and transition information. A *pose table* is a sequential list of each desired pose and its transition information. Figure 2.3 illustrates a typical creature which might be controlled using pose control graphs.



(a)

DOF		right hip	right knee	left hip	left knee	transition info	
	S1	0	0	-90	90	0.2	
	S2	0	0	-45	0	0	L
state	S3	-90	90	0	0	0.2	
	S4	-45	0	0	0	0	R

(b)

Figure 2.2 - A periodic PCG for a simple planar, 4 degree of freedom biped model
(a) State diagram form.
(b) Pose table form. All DOF values are in degrees relative to a reference position with straight, vertical legs and upright torso. Time-based transitions are in seconds. For sensor-based transitions, L - left foot sensor, R - right foot sensor.

While pose control appears similar to keyframing, two distinctions should be emphasized. First, the PCG determines the *desired* joint angles, and not the *actual* joint angles. The joints must be driven toward the desired angles through the use of a low-level control mechanism. Second, the poses do not specify the creature's position and orientation with respect to the world frame of reference. Instead, these are determined by the creature's interaction with its environment.

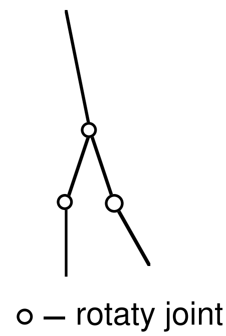


Figure 2.3 - Typical articulated creature model used with pose control graphs

In our implementation, joint angles are driven toward their desired values by joint actuator torques generated according to the following proportional-derivative (PD) control law:

$$\tau = k_p \cdot (\theta_{desired} - \theta) - k_d \cdot \dot{\theta}$$

where τ is the control torque applied at the joint, θ is the current joint angle, $\theta_{desired}$ is the desired joint angle, $\dot{\theta}$ is the relative angular velocity of the rigid links connected by the joint and k_p and k_d are proportional and derivative control constants.

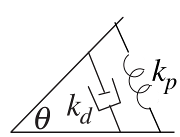
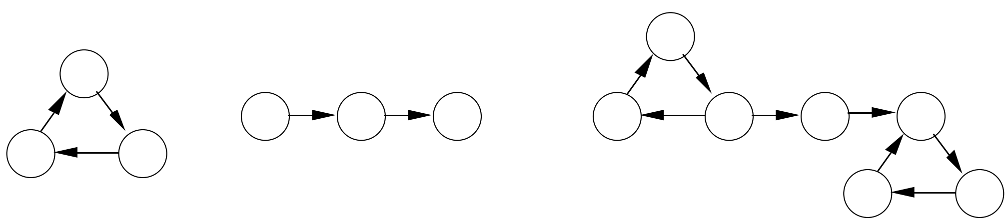


Figure 2.4 - Rotational PD controller for pose control

k_p and k_d specify the strength of a rotational spring and damper pair which acts as the "muscle" controlling the joint, as indicated in Figure 2.4. While this is a simple model, it is sufficient for our purposes in that it ensures that only internal control forces are used to generate the creature's motion. The actuator PD gain constants are held fixed for each joint and are considered to be part of the model specification.

The low-level mechanism for driving individual DOFs to their desired angles incorporates feedback, and is thus an example of closed loop control. The basic pose control mechanism does not make use of any other feedback to correct or adjust the overall motion of the creature. Therefore, we will refer to pose control using a fixed PCG as being a form of *open loop* control. Our definition is one of convenience and is appropriate since we choose to use pose control as our fundamental representation. In essence, the local closed loop control at the joints is treated as part of the system being controlled rather than part of the controller itself. In this context, *closed loop* refers to feedback used to modify the desired joint angle parameters of the PCG.

The transitions between the states of a PCG are based on a fixed *hold time* and/or are *sensor-based*. A hold-time transition occurs after a specified time has elapsed in the associated state. A sensor-based transition occurs when a particular binary sensor turns "on", or takes place immediately if the sensor is already on upon entering the state. The only form of sensors used in this work are ground contact sensors on the creature's feet, which are considered to be on when the foot is in contact with the ground.



(a) (b) (c)

Figure 2.5 - Pose control graph structures
(a) cyclic
(b) aperiodic
(c) composite

PCGs may be *periodic*, *aperiodic*, or *composite* (see Figure 2.5). A periodic PCG is one in which the state machine is cyclic. In this form, the desired poses are perfectly periodic. The actual motion and the applied torques need not be so; they do not typically repeat from one cycle to the next. As discussed in [vKF94], creatures controlled using periodic PCGs are analogous to simple windup toys. An aperiodic PCG is a chain of successive poses, useful for performing a single motion instance, such as recovering after a fall. A composite PCG is a more general form of state machine composed of periodic and aperiodic PCGs.

While pose controllers can be automatically synthesized with reasonable efficiency for relatively simple creatures and motions [vKF94], they work best for naturally stable motions. The base pose controllers presented in this thesis are reasonably complex and have all been designed by hand. The possibility of using automatic synthesis with the control techniques described is discussed later as future work.

To achieve the types of motions we are interested in, it is necessary to adjust our PCG to perform appropriate corrective actions on each cycle. An approach to the selection of these control actions is one of the key contributions of this thesis. Adjustments to the PCG are accomplished by applying linearly scaled perturbations to the PCG joint parameters during each cycle of motion.

2.3.1 Linear Parametric PCG Perturbations

To distinguish between the PCG providing our basic motion and the perturbations we apply to it, we will introduce the notion of a *base PCG*. A base PCG is a pose control graph which provides the fundamental cyclic motion of the creature we are trying to animate. For example, in the case of walking, the base PCG might consist of a left step followed by a right step. Ideally, the execution of a base PCG from a suitable initial state would result in the desired motion (e.g. a walk). However, the creatures we are interested in animating are unstable. The periodic open-loop actions of the base PCG invariably results in the creature falling. In order to generate balanced locomotion, additional control must be provided. Chapter 3 introduces one approach to providing such additional control, which uses the notion of *linear parametric perturbations* (LPPs), which we define below.

We begin by defining a *relative* PCG, which describes a change in the pose control to be applied to a creature, typically used to effect a desired change a motion. Consider a base PCG, B, to which we add a relative PCG, ΔP, scaled by an arbitrary scalar constant, k:

$$C = B + k\, \Delta P.$$

$k \cdot \Delta P$ is a linearly parameterized control perturbation where the parameter k is a *scaling factor* of ΔP. The PCGs B, ΔP and the overall control, C, are each similar in form to Figure 2.2 (b). The addition operation denotes the addition of the corresponding desired joint angle and hold time parameters of two PCG pose tables of equal dimensions. Scalar multiplication is applied to each desired joint angle and hold-time parameter of a PCG pose table. Sensor-based transitions in a perturbed PCG remain unaltered. As an example, an LPP could be used to turn a creature's head to look in a particular direction or to say "no". To accomplish this, ΔP might be chosen to vary only the desired angle of the neck and k would be chosen to turn the head to the right or left by the desired amount.

A variety of LPPs will be used in Chapters 3 and 5 as the mechanism to provide our biped with basic balance control and additional gait variations.

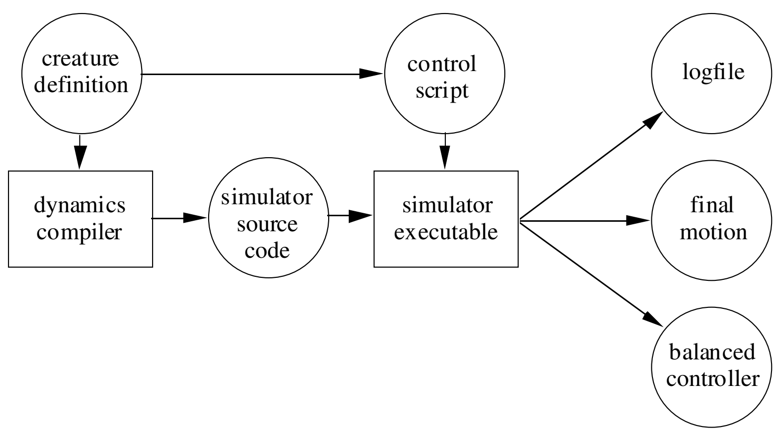


Figure 2.6 - Overview of the simulation process

2.4 The Animation System

Figure 2.6 provides an overview of the process of generating a physically-based animation with our system, beginning with a creature definition and a control script. The creature definition is used by a dynamics compiler to generate code which solves and integrates the equations of motion. The dynamics code, together with code implementing the ground model, basic pose control and the balance control formulation of Chapter 3 is then compiled into the simulator executable. The control script provides the particular control parameters for the desired simulation. Sample control scripts can be found in Appendices B and C. The primary outputs of each simulation are the final balanced motion of the creature and the aperiodic PCG which was ultimately responsible for generating it. Note that the resulting aperiodic PCG output provides open loop control. It is therefore only reusable given an identical initial state. In essence it is a record of the applied control actions for the motion, already complete with feedback actions.

The dynamics compiler used is a commercially available software package [SDFAST]. The animation environment currently supports the simulation and control of a single articulated creature consisting of rigid links in a tree structure with rotary joints of up to 3 DOF each and no joint limits. Each DOF has individual PD constants which remain fixed for the entire simulation. Collision forces due to interpenetration of the links of the articulated figure are not simulated.

The equations of motion are integrated using a fixed time step, fourth order Runge-Kutta integrator which is part of the dynamics compiler software. Performance of the simulator varies with model complexity with the most complex human model (described in Section 2.5) requiring approximately 1 minute of wall clock time to compute 1 second of simulated motion on a Sun Sparkstation 10. The use of a fixed integration time step has a significant impact on performance since the worst-case (i.e. smallest) time step for the complete simulation must be used. It is estimated that the use of a variable integration time step could improve performance by a factor of 5-10. Recorded simulation results can be played back in real-time on a Silicon Graphics Indigo^2 Workstation with GR3-XZ graphics hardware. Display functions are implemented using the using the SGI-GL graphics library.

2.4.1 Ground Model

The ground reaction forces are modeled for all simulations using a spring and damper model. Figure 2.7 illustrates the model for a 2D system. Ground forces are applied to specific, pre-defined *monitor points* on the model. The ground forces exerted on a monitor point, M, which has penetrated the ground surface are:

$$F = k_p \cdot (P - M) - k_d \cdot \dot{M}$$

where F is the ground reaction force, M is the position of the monitor point, $\dot{M}$ is its velocity, P is the initial point of contact with the ground, and k_p and k_d are proportional and derivative constants defining the stiffness and damping properties of the ground (see Figure 2.7).

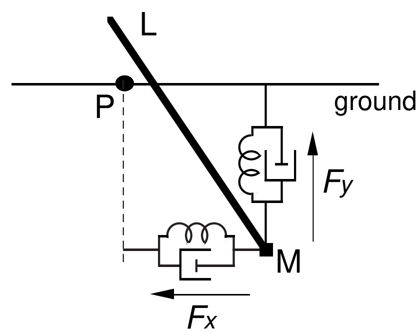


Figure 2.7 - Spring and damper ground force model (2D example)
L - a link of the creature
M - a monitor point attached to link L
P - point of initial contact of M with ground plane
F_x - simulated ground forces in x direction
F_y - simulated ground forces in y direction

The ground reaction forces are bounded such that the vertical component is always positive, thus never allowing the damping term to impede the lifting of the foot. Monitor point slippage is implemented using a Coulomb friction model, which is used to limit the tangent of the ground reaction force. Slippage is allowed when the ratio of the vertical and tangent forces exceeds a specified limit as illustrated in Figure 2.8 for the 2D case. Slippage is simulated by moving the point of initial contact, P, to be directly above monitor point.

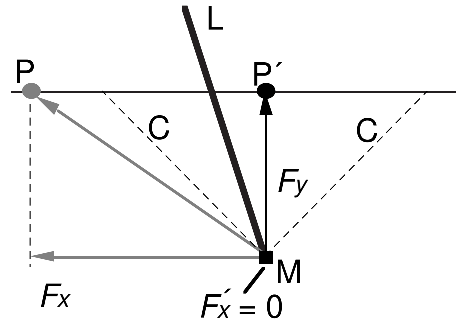
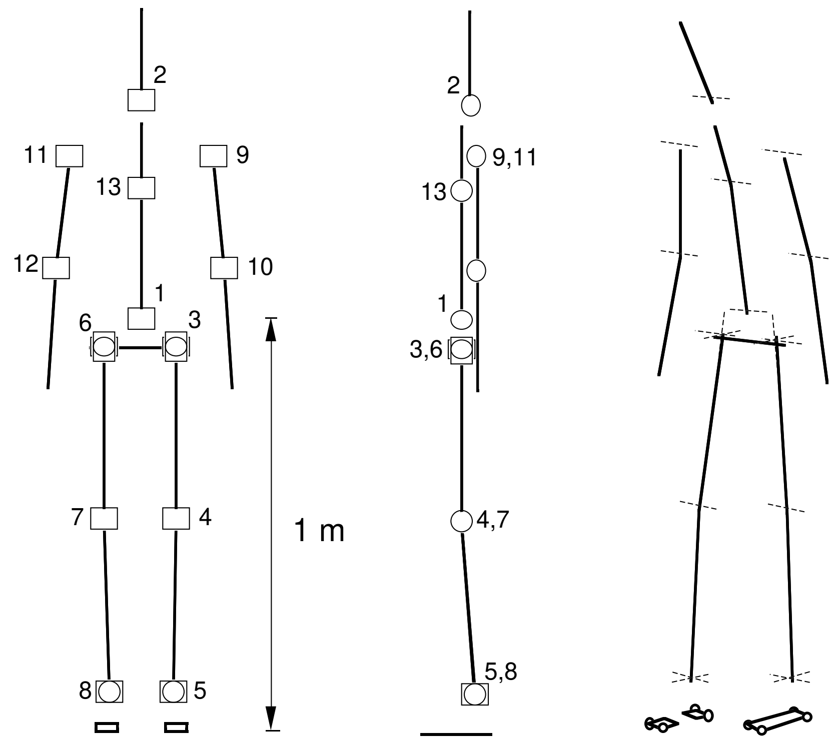


Figure 2.8 - Friction cone ground slip model (2D example)
L - a link of the creature
M - a monitor point on link L
C - boundaries of a friction cone for a ratio of 1.0
P - point of initial contact of M with ground plane
P´ - new "point of initial contact" of M with ground plane after slipping is applied

No other ground forces are applied to the creature to constrain movement in any direction. The foot may be in full or partial contact with the ground, depending on which monitor points are in ground contact. It is free to pitch, roll and yaw and to slip within the described constraints provided by the friction cones. This contrasts with many approaches in animation and in robotics, which use planar dynamics or place less realistic constraints on the motion of the foot while in contact with the floor [HF77, FM87, KKI90, KT91, CHP92,VFV92, SC92, Hod+9].

2.5 Biped Models

The most complex human model used, shown in Figure 2.9, has 19 degrees of freedom including ball-and-socket hips, 2 DOF ankles and a jointed torso. All other joints are modeled using 1 DOF. Mass and inertia parameters are realistic for a human model and are from taken from [WH95]. Several other simpler human models with fewer DOFs are also used throughout our experiments in order to reduce simulation time requirements. The simplest of these has 12 DOFs, with 2 DOF hips (pitch and roll but no yaw), no arms, and a rigid torso which incorporates the mass and inertia parameters for fixed arms in the reference position of Figure 2.9.



(a) (b) (c)

- rotary joints - rotary joint axes - monitor points

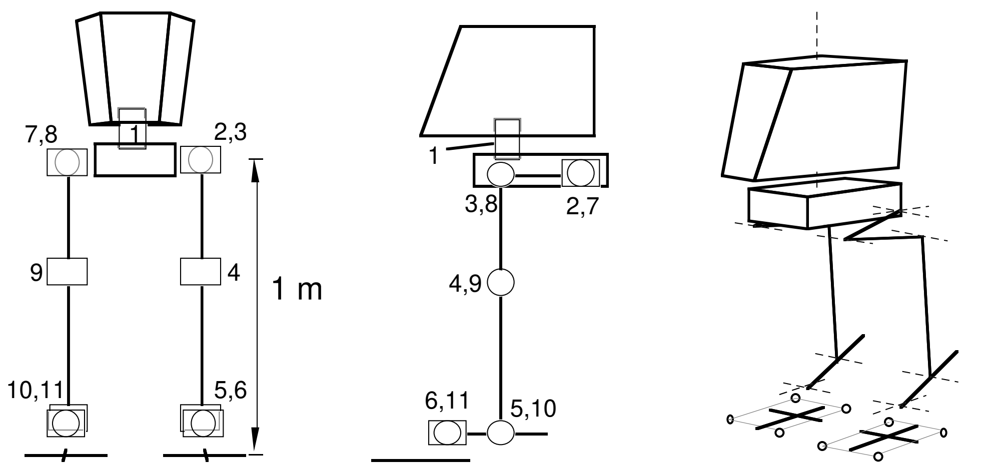
Degrees of Freedom

1	-	waist pitch (sagittal plane)			
2	-	neck pitch (sagittal plane)			
3:0	-	left hip roll (coronal plane)	6:0	-	right hip roll (coronal plane)
3:1	-	left hip pitch (sagittal plane)	6:1	-	right hip pitch (sagittal plane)
3:2	-	left hip yaw (transverse plane)	6:2	-	right hip yaw (transverse plane)
4	-	left knee pitch (sagittal plane)	7	-	right knee pitch (sagittal plane)
5:0	-	left ankle pitch (sagittal plane)	8:0	-	right ankle pitch (sagittal plane)
5:1	-	left ankle roll (coronal plane)	8:1	-	right ankle roll (coronal plane)
9	-	left shoulder pitch (sagittal plane)	11	-	right shoulder pitch (sagittal plane)
10	-	left elbow pitch (sagittal plane)	12	-	right elbow pitch (sagittal plane)
			13	-	mid-back pitch (sagittal plane)

Figure 2.9 - Complex human model
(a) front view (reference position)
(b) left side view (reference position)
(c) typical pose (with monitor points shown)

A second model has a bird-like structure similar to the biped All-terrain Scout Vehicle (ASV) robot in the motion picture "The Empire Strikes Back". The robo-bird creature, shown in Figure 2.10, has 15 degrees of freedom, including 2 DOF hips and 2 DOF ankles. The models do not incorporate any physical joint limits. Dimensions, mass and inertia parameters for both models can be found in Appendix B.

These models are quite complex compared to much of the previous work on bipedal systems which have often assumed very simple bipedal models. In the next chapter, we will present an approach to generating animations of balanced motion for these creatures, which represents one of the major contributions of this thesis.



(a) (b) (c)

- rotary joints - rotary joint axes - monitor point

Degrees of Freedom

1 - neck yaw (transverse plane)

2:0	- right hip roll (coronal plane)	7:0	- left hip roll (coronal plane)
2:1	- right hip pitch (sagittal plane)	7:1	- left hip pitch (sagittal plane)
3	- right knee1 pitch (sagittal plane)	8	- left knee1 pitch (sagittal plane)
4	- right knee2 pitch (sagittal plane)	9	- left knee2 pitch (sagittal plane)
5	- right knee3 pitch (sagittal plane)	10	- left knee3 pitch (sagittal plane)
6:0	- right ankle roll (coronal plane)	11:0	- left ankle roll (coronal plane)
6:1	- right ankle pitch (sagittal plane)	11:1	- left ankle pitch (sagittal plane)

Figure 2.10 - Robo-bird creature
(a) front view (reference position)
(b) left side view (reference position)
(c) typical pose (with monitor points shown)