Evolution of Legged Locomotion

by

Dirk Arnold

Dipl.-Inform., Universität Dortmund, 1995

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
July 1997

Approval

Name: Dirk Arnold

Degree: Master of Science

Title of Thesis: Evolution of Legged Locomotion

Examining Committee:

Dr. Tiko Kameda
Chair

Dr. F. David Fracchia
Senior Supervisor

Dr. Thomas W. Calvert
Supervisor

Dr. John C. Dill
Examiner

Date Approved: ---

Abstract

The realistic animation of human and animal figures has long been a prime goal in computer graphics. A recent, physically-based approach to the problem suggests modeling creatures as actuated articulated bodies equipped with a "virtual brain" which generates the control signals required by the actuators to produce the desired motion. The animation of such a creature is simply a forward simulation of the resulting motion under the laws of physics in time.

While using this approach ensures physically realistic motion, there is no obvious solution to the problem of devising a control system that leads to the desired motion. Even though good results have been achieved by carefully handcrafting control systems on the basis of biomechanical knowledge and physical intuition, it is desirable to produce control system automatically. Evolutionary algorithms which iteratively improve randomly generated initial control systems have shown to be a promising approach to this problem.

This thesis introduces spectral synthesis as a tool for generating control systems to be optimized in an evolutionary process and demonstrates the viability of the approach by evolving creatures for the task of legged locomotion. Other than representations of control systems that have previously been used for evolving useful behavior, spectral synthesis guarantees evolvability, improving the chances of the evolutionary search to succeed. Virtual creatures exhibiting a great variety of modes of locomotion, including hopping, crawling, jumping, and walking, have been evolved as part of this work. The incorporation of more goal-directed components remains as a future goal.

A second accomplishment of this thesis is the derivation of the equations describing the effect of applying a contact force to an articulated body on its acceleration, making it possible to generalize the common algorithms for handling contacts in systems of rigid bodies to articulated bodies. The physical simulation algorithm described in this thesis allows for real time simulation of articulated creatures of up to about twenty degrees of freedom. Efficient simulation algorithms are especially important as evolutionary optimization requires the evaluation and therefore simulation of the behavior of a great number of creatures.

Acknowledgements

I'd like to thank Dave for encouragement, support, and helpful advice, and the other members of my examining committee for useful input which helped shaping the final version of this thesis.

Contents

Chapter 1

Introduction

The realistic modeling of the locomotion of human and animal figures has long been a prime goal in computer graphics. In a direct derivation from traditional animation techniques, the problem of animating figures in physically realistic and visually convincing ways has long been put into the hands of highly skilled animators who prescribe the positions and orientations of the objects in motion for a number of key frames, with intermediate frames constructed by computer interpolation. Lasseter [27] gives a comprehensive account of how the principles of traditional animation apply to 3D keyframe computer animation. However, producing motion kinematically by determining positions of objects over time while neglecting the forces and torques that generate the motion in the real world often causes the movements to have a somewhat unrealistic appearance, and figures may look as if being pulled by invisible strings instead of locomoting as a result of their limb movements.

Dynamic methods do not suffer from this problem as they model the behavior of the objects in motion under the laws of physics and therefore guarantee physically realistic and visually pleasing results in the limits of the physical model being employed. Using a physically-based approach, articulated figures are usually modeled as collections of rigid bodies connected by joints. Virtual "muscles", in the following referred to as actuators, apply forces across the joints to make the creatures move. The motion of the rigid bodies comprising an articulated figure is governed by the laws of Newton and Euler, with the joints imposing kinematic constraints. To generate the successive time frames required for an animation sequence, the system combining the equations of motion with any constraint equations that might be needed, for example for avoiding interpenetration of solid objects, is numerically integrated.

Unfortunately, however, the gain in realism resulting from the use of dynamic simulation

as opposed to kinematic methods comes at the cost of a loss of control. The problem of specifying a physically realistic sequence of positions and orientations has been replaced by the even less intuitive problem of specifying a system which, when subjected to the laws of physics and simulated over time, shows the desired behavior. In particular, to produce animations of actuated articulated figures, the actuator forces and torques leading to the desired motion have to be given. As the number of degrees of freedom is potentially quite large, the problem of finding and coordinating appropriate control algorithms is rather complex.

Related Work

In recent years, a number of studies have dealt with the problem of automating the creation of animations of articulated figures. The following overview does not attempt to cover all of the work done in the area, but rather concentrates on instances where dynamic concepts have been used as a tool to achieve realistically looking motion. For readers interested in a more comprehensive treatment of the subject or related higher level approaches, the anthology *Making Them Move* [2] can serve as a starting point.

Several authors have introduced methods for blending kinematic constraints with dynamic simulation. Isaacs and Cohen [20] and Barzel and Barr [5] suggested exerting explicit control over some of the degrees of freedom by kinematically specifying positional constraints on parts of a body while allowing other parts to react in a correct dynamic fashion. Technically, this can be achieved by treating the kinematic constraints as consequences of constraint forces which are solved for and subsequently added into the simulation, cancelling out the part of the dynamics that leads to the violation of the constraint. The approach permits parts of an articulated body, such as a character's hands or feet, to be moved along predefined trajectories. However, it provides no help in defining those trajectories, which is the central problem in creating character animation, and unrealistically large forces can occur as a consequence of badly chosen constraints. It has been noted that while allowing a character to be dragged around manually like a marionette, constraint forces sidestep the central issue of deciding how the character should move.

In an attempt to overcome these limitations, Witkin and Kass [53] introduced space-time constraints, permitting the imposition of constraints throughout the time course of the motion, with the effects of constraints propagating freely backward as well as forward in time. The approach conceptually deviates from the simulation paradigm by treating the laws of physics as constraints on the motion rather than as the primary driving force for simulation. Instead of progressing sequentially through time, the problem of producing an

animation sequence is considered as a trajectory optimization problem with the laws of physics as constraints, and is solved by iteratively refining an initial guess at a possible trajectory. As the optimization problem requires that all forces and all of a creature's degrees of freedom during the entire time interval of interest are solved for simultaneously, the computational requirements grows rapidly with the number of degrees of freedom of the objects being modeled and with the length of the simulation time interval. Moreover, the approach requires exact advance knowledge of the time steps at which non-interpenetration constraints are active, making it difficult if not impossible to handle multiple and frequently changing contacts realistically. In an attempt to alleviate these problems, COHEN [12] extended the original idea by introducing spacetime windows, subdividing spacetime into discrete pieces, and created an interactive framework for specifying constraints and objectives for the motion, and for guiding the numerical solution of the optimization problem.

An entirely different approach was chosen by BRUDERLIN and CALVERT [10] who made use of detailed knowledge of the biomechanics of walking to create a simplified dynamic model of the human gait and to control its degrees of freedom. Rather then relying on a general dynamic model, they tailored a leg model based upon a telescoping structure with two degrees of freedom for the stance phase and a compound pendulum for the swing phase, and analytically constrained the motion to allow for only a specific range of movements. Having thus reduced the number of degrees of freedom as compared to a general dynamic model, they used knowledge about locomotion cycles to construct a hierarchical process controlling the remaining variables. Feet, upper body, and arms were added to the model kinematically, and were made to move in an oscillatory pattern similar to that observed in humans. The approach resulted in a procedural model allowing an animator to effortlessly control the motion by simply varying some parameters affecting the gait.

The remainder of this survey concentrates exclusively on related work employing general dynamic models with no kinematic constraints other than those stemming directly from the laws of physics. No advance assumptions are being made about the forms of motion that emerge during the course of a simulation. Given an accurate physical model, the motion that can be generated using this approach is extraordinarily realistic and naturally incorporates effects resulting from phenomena such as surface friction in a convincing manner. Two major problems pertaining to it are the high computational demands of a general physics simulator which is sufficiently robust to cope convincingly with all constellations that might occur during the course of a simulation, and the difficult question of how to design a control system which generates input to the actuators leading to the desired form of motion.

While the problem of physical simulation has often been dealt with in the past by using

simplified dynamic models, creating statically stable systems, or restricting the motion to two dimensions, a considerable amount of work exists on the problem of designing powerful control systems. Instead of just producing periodic time series as inputs for the actuators, control systems can make use of sensor information to compute appropriate signals on the basis of data on the current position and velocity of the creature and parts thereof, information on contact with the ground or other objects, or even visual data about the environment.

As the design based upon physical intuition or knowledge of biomechanics requires a considerable amount of effort on the part of the animator, much of which consists of tweaking parameter values, attempts have been made to generate such systems automatically in an evolutionary process. The performance of initially randomly generated control systems is evaluated, and in a process mimicking features of biological evolution, promising systems are used as starting points for further generations while poorly performing ones are discarded. The approach derives part of its appeal from its potential to generate interesting and often surprising results that would be hard to achieve by using other methods. The disadvantage that the influence an animator can exert on the behavior of the creatures to be evolved is limited to the specification of fitness criteria, and therefore is indirect and comparatively weak, is often more than compensated for by the fact that evolved characters frequently exhibit behaviors which give them the appearance of intention and personality without it being explicitly designed.

The following list describes interesting features of some related work in which control systems have been generated that lead to successful locomotion behavior.

- McKenna and Zelzer [29] created a three-dimensional, six-legged virtual insect with 38 degrees of freedom which is capable of statically stable walking. The problem of generating a control system was divided into one of coordinating the motion of the links and one of generating the actual forces. Coordination was achieved by a gait controller consisting of a number of coupled oscillators — one for each leg of the creature — which rhythmically trigger the legs to step. Additionally, reflexes such as a step reflex which triggers a leg to step if it has nearly reached its maximum rearward extension, and a load-bearing reflex which prevents a leg from stepping if it is currently supporting the body, were added to reinforce the stepping pattern. The problem of computing the forces was solved by carefully handcrafting motor programs for stepping and stance which compute the forces necessary to lift the leg up and forward and place it in a position to take up the load of the body when stance begins, and the forces needed to support the body via the legs and propel it forward,

respectively. The resulting control system can adapt to various speeds as a reaction to changes in an exogenous parameter and proved to be robust enough to enable the creature to negotiate its way in uneven terrain.

- NGO and MARKS [35] evolved control systems for two-dimensional stick figures treated as autonomously deforming objects without dynamic internal degrees of freedom. As the deformation of such creatures is purely kinematic, the task of physical simulation is restricted to modeling interactions with the ground plane. Sensors including proprioceptive sensors for the joints, tactile sensors delivering information on the forces exerted by the endpoints of the links on the ground, kinesthetic sensors giving the vertical velocity of the center of mass of a creature, and position sensors supplying the vertical position of the center of mass relative to the floor provide input to the control systems of the creatures. The control system itself uses a number of stimulus-response pairs and a metric defined on the sensor space to map sensor readings to the response associated with the closest stored sensor tuple.

- VAN DE PANNE and FIUME [49] pursued similar goals by connecting the sensors of their two-dimensional creatures to the actuators through a neural network which was designed to incorporate internal delays, thereby giving it dynamic properties.

- SIMS [44] made use of a highly parallel computer to create virtual creatures for walking, swimming, and jumping in a simulated three-dimensional world, with both the morphologies and the control systems being generated automatically in an evolutionary process. His articulated figures are composed entirely of three-dimensional, box-shaped rigid bodies connected by a variety of joint types. Control systems are networks of nodes computing simple logic and arithmetic functions, resembling dataflow computers in their operation. Both morphologies and control systems were encoded as graphs, and evolutionary operators applicable to this representation were specifically designed.

- VENTRELLA [50] enhanced the automatic evolutionary algorithm by an interactive component, making it possible for the animator to let subjective judgement influence the exploration of emergent locomotion behavior in articulated three-dimensional stick figures evolved for walking. Both morphologies and the control systems which were made to generate simple sinusoidal output without making use of sensor input were subject to evolutionary change.

- HODGINS, WOOTEN, BROGAN, and O'BRIEN [19] produced animations of human

athletics. They created kinematic models of humans with up to 30 controlled degrees of freedom and designed control algorithms based upon physical intuition about the behaviors, observations of humans performing the task, and biomechanical data to produce animations of running, bicycling, and vaulting.

Goals

The goal of this thesis is to produce realistic animations of three-dimensional, fully dynamic, legged articulated creatures. Two separate challenges are related to this task. First, algorithms for the dynamic simulation of three-dimensional articulated figures have to be developed. While efficient algorithms for the simulation of unconstrained figures exist, the situation is less satisfactory when interpenetration constraints have to be considered. One contribution of this thesis is the generalization of an algorithm for computing contact forces from systems of rigid bodies to articulated bodies. The resulting simulator is sufficiently powerful and robust to convincingly simulate the motion of rounded three-dimensional objects with up to about twenty degrees of freedom in real time.

The second challenge is the problem of finding encodings describing virtual creatures which are suitable for evolutionary optimization, in that they increase the chances of the evolutionary search to succeed. Such genotypic descriptions have to contain all information on the kinematic structure of the links and joints, the mass and inertia properties of the links, and the parameters describing the control system of a creature. The encodings of most current control systems such as neural networks or dataflow architectures suffer from the defects that for most values of the parameters, the resulting behaviors are meaningless, and that the dependence of creature behavior on the parameters is usually highly discontinuous. In this thesis, spectral synthesis is introduced as a useful tool for the design of motor programs, and problem specific evolutionary operators are suggested.

Overview

The remainder of this thesis is organized as follows. Chapter 2 introduces the physical groundwork for modeling articulated creatures. In particular, after outlining the basic concepts and equations of motion, the articulated body method as first formulated by FEATHERSTONE [16] is derived as an efficient algorithm for computing the dynamics of articulated bodies without kinematic loops. Then, algorithms for handling collisions and static contacts are presented and their appropriateness for the problem at hand is discussed. Chapter 3 starts with an introduction into the field of evolutionary optimization before outlining im-

portant factors for the design of scripting languages for the morphology and control systems of creatures which are to be optimized in an automatic evolutionary process. Then, results from experiments in which virtual legged creatures have been evolved for locomotion behavior are presented. Chapter 4 concludes, and Appendix describes the software system developed for conducting the experiments.

Chapter 2

Physical Simulation of Actuated Articulated Bodies

Actuated articulated bodies are systems of rigid bodies connected by joints which are powered by force generators — henceforth referred to as actuators — applying forces in the free directions of the joints. Often, as is the case here, articulated bodies are required to not have internal loops, meaning that a graph with vertices corresponding to the links comprising an articulated body and edges corresponding to the connecting joints is cycle-free. Acyclic articulated bodies are sufficiently general to encompass models for legged creatures as required in Chapter 3. Admitting cyclic topologies would lead to redundant systems of equations, thereby greatly increasing the mathematical and computational difficulties arising during simulation.

The state of an articulated body can be described by means of a set of joint position coordinates, q, and their derivatives with respect to time, joint velocities $\dot{q}$. The simulation problem for articulated bodies consists of solving for the joint accelerations $\ddot{q}$, given the current positions and velocities of the joints and the torques and forces applied internally by the actuators and externally by contact and friction forces. A numerical integration procedure can then be used to compute new positions and velocities, advancing the simulation in time.

Effectively, the real problem is the practical one of finding formulations of articulated body dynamics and schemes for solving the equations of motion that lead to efficient algorithms. Two different paradigms are commonly used. Non-recursive algorithms, such as the *composite rigid body method* by Walker and Orin [51], obtain and then solve a set of simultaneous linear equations in the unknown joint accelerations. That is, they construct

a system of equations of the form $J(q)\ddot{q} = f(q, \dot{q})$, where J is the joint space inertia matrix of the articulated body and depends on the current joint positions, and f is a linear function of q and $\dot{q}$, and then solve for $\ddot{q}$ by inverting J. Since such algorithms require the inversion of the $N \times N$ matrix J, where N is the number of degrees of freedom of the body, the computational requirements are high. On the other hand, recursive algorithms such as the *articulated body method* by FEATHERSTONE [15] make explicit use of the topological structure of the system to structure the computational process by propagating motion and force constraints along the mechanism, and require computational resources growing only linearly with N.

While the problem of generating and solving the equations of motion for an articulated body without internal loops or contact to other bodies is a matter of efficiency, the more general problem of simulating an articulated body, parts of which are in contact with other parts of the body or with the environment, is more complex. Mathematically, contacts introduce additional constraint equations, eliminating some of the degrees of freedom of the body, and giving rise to the problem of finding efficient schemes for incorporating the constraint equations into the system of equations of motion. Unfortunately, the additional equations can lead to redundancies which result in inconsistencies or ambiguities. It is a well established fact that combining the principles of rigid body mechanics with the usual assumptions about the constraints at contact points suffers from this kind of problem. Moreover, none of the current contact handling algorithms is always able to avoid the trend to intolerably small integration step sizes as a consequence of configurations it is not suited for. As a general rule, the more complex the articulated body to be simulated, the more demanding the task of physical simulation.

The purpose of this chapter is to present computationally efficient algorithms for calculating the dynamics of actuated articulated bodies. After introducing terminology and notation for the description of articulated bodies in Section 2.1, the equations governing the motion of articulated bodies are outlined in Section 2.2 and a recursive scheme for their solution — devised by FEATHERSTONE [15] and generalized and refined by BRANDL, JOHANNI, and OTTER [8] — is described in Section 2.3. This algorithm is included here because the equations derived in its development form the basis for the algorithms dealing with static contacts and collisions discussed in Section 2.4. The derivation of the equations of motion of an articulated body subject to applied impulses in Subsection 2.4.1 and the proposed algorithm for handling multiple static contacts in Subsection 2.4.4 are original contributions of this thesis. The chapter concludes with a summary and a discussion of how to integrate the outlined algorithms with a numerical integration procedure.

2.1 Preliminaries

This section first introduces spatial notation, a useful tool for formulating the equations of motion for rigid and articulated bodies. Then, a general formalism for describing constraining mechanisms between rigid bodies which is well suited for modeling the joints connecting the components of articulated bodies is presented. Finally, the conventions on notation and coordinate systems used throughout the remainder of this thesis are outlined. This section does not attempt to derive the equations needed in the following, but simply presents them without proof. For its understanding, basic familiarity with the mechanics of rigid bodies is useful. For an introductory treatment of the subject see for example Craig [13]. For a more extensive treatment of the problem of modeling constraining mechanisms and a great number of examples of joint types see Roberson and Schwertassek [41]. Additional information on spatial notation and on efficient implementations of spatial matrix operations can be found in Featherstone [16].

2.1.1 Spatial Notation

Spatial notation was first introduced by Featherstone [15] to unite the rotational and translational aspects of motion into a single vector quantity, thereby leading to concise and elegant descriptions of physical systems. In this thesis, a version revised by Brandl, Johanni, and Otter [8] who clarified notational details is used. In spatial notation, position is a generic term, embracing both location and orientation of an object. Accordingly, velocities, accelerations, and forces are described by six-dimensional vectors, each incorporating three angular and three linear components. Throughout the following,

$$\mathbf{v} = \begin{pmatrix} \omega \\ v \end{pmatrix}$$

denotes a spatial velocity with angular component ω and linear component v. Similarly,

$$\mathbf{a} = \begin{pmatrix} \dot{\omega} \\ \dot{v} \end{pmatrix}$$

stands for the spatial acceleration with angular component $\dot{\omega}$ and linear component $\dot{v}$, and

$$\mathbf{f} = \begin{pmatrix} \tau \\ f \end{pmatrix}$$

is a spatial force with τ representing a torque and f a force vector. The spatial inertia matrix of a rigid body is a 6×6 matrix of the form[1]

$$\mathcal{I} = \begin{pmatrix} I & m\tilde{d} \\ m\tilde{d}^T & m1_{3\times 3} \end{pmatrix} \tag{2.1}$$

where I is the 3×3 moment of inertia tensor of the object with respect to the origin of the coordinate system, d is the position of the center of mass of the body, and m is the body's mass.

A spatial vector quantity can be transformed from one coordinate frame to another by multiplication with a spatial transformation matrix. More specifically, let α and β denote two coordinate frames, and let

$$X_{\alpha,\beta} = \begin{pmatrix} A & 0_{3\times 3} \\ 0_{3\times 3} & A \end{pmatrix} \begin{pmatrix} 1_{3\times 3} & 0_{3\times 3} \\ \tilde{b}^T & 1_{3\times 3} \end{pmatrix} = \begin{pmatrix} A & 0_{3\times 3} \\ A\tilde{b}^T & A \end{pmatrix}$$

where A is the 3×3 rotation matrix aligning α-coordinates with β-coordinates and b is the origin of the β-coordinate system in α-coordinates. Then, if $\mathbf{x}^{(\alpha)}$ is the representation of a spatial vector quantity $\mathbf{x}$ expressed in α-coordinates, the representation of that same quantity in β-coordinates is

$$\mathbf{x}^{(\beta)} = X_{\alpha,\beta}\mathbf{x}^{(\alpha)}.$$

Similarly, a spatial inertia matrix $\mathcal{I}$ can be transformed from α-coordinates to β-coordinates by means of

$$\mathcal{I}^{(\beta)} = X_{\alpha,\beta}^T \mathcal{I}^{(\alpha)} X_{\alpha,\beta}.$$

As a special case, choosing a coordinate system centered at the center of mass of an object shows that $\mathcal{I}$ can be expressed as

$$\mathcal{I} = D^T \begin{pmatrix} I^{CM} & 0_{3\times 3} \\ 0_{3\times 3} & m1_{3\times 3} \end{pmatrix} D$$

[1] For a vector $c = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix}$, let $\tilde{c} = \begin{pmatrix} 0 & -c_2 & c_1 \\ c_2 & 0 & -c_0 \\ -c_1 & c_0 & 0 \end{pmatrix}$. As a consequence, the cross product $c \times d$ of three-dimensional vectors c and d can be written as the matrix-vector product $\tilde{c}d$. In the same manner, the vector-matrix cross product $c \times M$ of three-dimensional vector c and 3×3 matrix M is defined to be $\tilde{c}M$, the 3×3 matrix with columns being the cross products of c with the columns of M.

Furthermore, throughout the following, $1_{n\times n}$ denotes the $n \times n$ unity matrix. Similarly, $0_{m\times n}$ stands for the $m \times n$ zero matrix.

where I^{CM} is the moment of inertia tensor of the object with respect to the coordinate system centered at its center of mass, and

$$D = \begin{pmatrix} 1_{3\times3} & 0_{3\times3} \\ \tilde{d}^T & 1_{3\times3} \end{pmatrix}.$$

As I^{CM} is symmetric and positive definite, m is positive, and D is regular, it follows that $\mathcal{I}$ is symmetric and positive definite and therefore invertible. This property will be used in Section 2.3. Furthermore, it is worth noting that spatial transformation matrices are transitive in the sense that

$$X_{\alpha,\gamma} = X_{\beta,\gamma} X_{\alpha,\beta}$$

for any three coordinate frames α, β, and γ.

2.1.2 Joints

The joint model presented in this section provides a uniform mathematical description of the kinematically constraining interconnection between two contiguous links of an articulated body. In this model, joints have at least zero and at most three rotational and three translational degrees of freedom. That is, even a rigid connection and the relative motion of two free bodies in space can be described. The model is a simplification of the general joint model introduced by Roberson and Schwertassek [41].

If a joint has N $(0 \leq N \leq 6)$ degrees of freedom, the joint state can be described at any instant by a set of N position variables forming the joint position vector q, and N velocity variables forming the joint velocity vector $\dot{q}$. For all joints used in the following, a set of joint state variables can be found such that the relative velocity of the moving body with respect to the base body can be written in spatial notation as

$$\mathbf{v} = \phi \dot{q}.$$

As $\dot{q}$ is a minimal set of variables, the $6 \times N$ matrix ϕ has full column rank and a $6 \times (6-N)$ matrix $\overline{\phi}$ can be found such that the columns of 6×6 matrix $\begin{pmatrix} \phi & \overline{\phi} \end{pmatrix}$ form a basis of $\mathbb{R}^6$. The constituents of this basis are called the *mode vectors* of the joint, with the columns of ϕ being the vectors of the *free modes* of the joint spanning the motion space while the columns of $\overline{\phi}$ are the vectors of the *constrained modes*. Additionally, for all joints used in the following the relationship

$$\begin{pmatrix} \phi^T \\ \overline{\phi}^T \end{pmatrix} \begin{pmatrix} \phi & \overline{\phi} \end{pmatrix} = \begin{pmatrix} 1_{N\times N} & 0_{N\times(6-N)} \\ 0_{(6-N)\times N} & 1_{(6-N)\times(6-N)} \end{pmatrix} \tag{2.2}$$

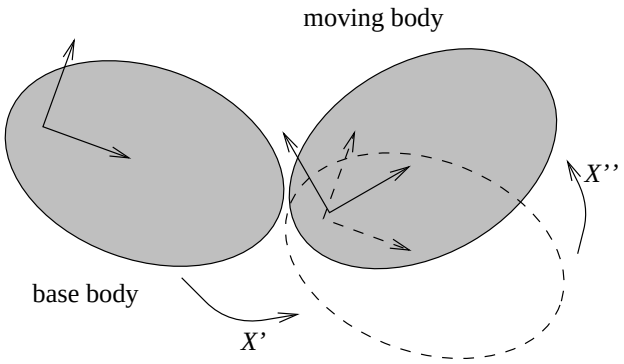

Figure 2.1: *A general joint connecting a moving body to a base body. The rest position of the moving body is indicated by broken lines. Spatial transformation X' transforms spatial vectors from the coordinate frame of the base body to the coordinate frame of the moving body in its rest position, and spatial transformation X'' transforms from there to the coordinate frame of the moving frame in its actual position.*

holds.

Figure 2.1 shows a general joint connecting a moving body to a base body. On both bodies, a fixed hinge point is at the origin of a local coordinate frame which moves with the body. A transformation X' comprised of a translation b' followed by a rotation A' transforms spatial quantities from the coordinate frame of the base link to the coordinate frame of the moving link in its rest position, and a transformation X'' comprised of a translation b'' followed by a rotation A'' which are generally both dependent on the joint state completes the transformation to the coordinate frame of the moving body. The dependency of A'' and b'' on the joint position q formally reflects the constraints on the position of the moving body imposed by the joint.

The dynamic action between the base body and the moving body can be described by a resultant spatial force $\mathbf{f}$ acting on the moving body and, with negative sign, on the base body. With $\mathbf{f}$ resolved in the coordinate system of the moving body, the free and constrained modes of the joint can be separated by writing

$$\mathbf{f} = \phi\lambda + \overline{\phi}\ \overline{\lambda}, \tag{2.3}$$

where λ is the vector of known, applied forces in the unconstrained directions of the joint, including actuator forces as well as damping forces and possibly spring forces enforcing joint limits, and $\overline{\lambda}$ is the vector of unknown constraint forces acting in the constrained directions.

For a complete description of a joint, the state variables for the relative motion across the joint, the constraint equations on the position variables manifested in transformation X'', the mode vectors, and the kinematical equations of motion have to be given. The following examples of frequently used joints with the exception of the rigid connection and

the unconstrained joint are taken from ROBERSON and SCHWERTASSEK [41].

Rigid Connection As a rigid connection has no degrees of freedom, it is trivially described the the following information.

state variables: none

positional constraint equations: $b''(q) = 0_{3\times 1}, \quad A''(q) = 1_{3\times 3}$

mode vectors: $\phi = 0_{6\times 0}, \quad \overline{\phi} = 1_{6\times 6}$

kinematical equations of motion: none

Rotational Joint The state of the single rotational degree of freedom of a rotational joint, the axis of which is along the common base vectors $(1,0,0)^T$ of both the base frame transformed by X' and the moving link frame, is adequately represented by φ, the angle of rotation, and its derivative with respect to time.

state variables: $q = \varphi, \quad \dot{q} = \dot{\varphi}$

positional constraint equations: $b''(q) = 0_{3\times 1}, \quad A''(q) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\varphi) & -\sin(\varphi) \\ 0 & \sin(\varphi) & \cos(\varphi) \end{pmatrix}$

mode vectors: $\phi = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad \overline{\phi} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$

kinematical equations of motion: $\dfrac{dq}{dt} = \dot{q}$

Prismatic Joint The state of the single translational degree of freedom of a prismatic joint, with the axis of the joint being along the common base vectors $(0,0,1)^T$ of both the base frame transformed by X' and the moving link frame, is adequately represented by z, the amount of extension along that axis, and its derivative with respect to time.

state variables: $q = z, \quad \dot{q} = \dot{z}$

positional constraint equations: $b''(q) = \begin{pmatrix} 0 \\ 0 \\ z \end{pmatrix}, \quad A''(q) = 1_{3\times 3}$

$$\textit{mode vectors: } \phi = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \quad \overline{\phi} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\textit{kinematical equations of motion: } \frac{dq}{dt} = \dot{q}$$

Spherical Joint As a spherical joint has three rotational degrees of freedom, its velocity is adequately described by the relative angular velocity ω of the moving body. However, there is no set of three position variables describing the joint position such that its derivative with respect to time is the relative angular velocity of the moving body. Moreover, as ROBERSON and SCHWERTASSEK [41] note, any representation of arbitrary rotations in space by three parameters degenerates for certain values of those parameters.

Therefore quaternions, sets of four parameters s_φ, s_x, s_y, and s_z satisfying the normality constraint $s_\varphi^2 + s_x^2 + s_y^2 + s_z^2 = 1.0$ which prescribe a rotation of $2\arccos(s_\varphi)$ radians about the axis $(s_x, s_y, s_z)^T$, are chosen for the representation of arbitrary rotations. Even though strictly speaking quaternions are not state variables, they can effectively be used as such by defining their time derivative as done below.

$$\textit{state variables: } q = (s_\varphi, s_x, s_y, s_z),\ \dot{q} = \omega$$

$$\textit{positional constraint equations: } b''(q) = 0_{3\times 1},$$

$$A''(q) = 2 \begin{pmatrix} \frac{1}{2} - s_y^2 - s_z^2 & s_x s_y + s_\varphi s_z & s_x s_z - s_\varphi s_y \\ s_x s_y - s_\varphi s_z & \frac{1}{2} - s_x^2 - s_z^2 & s_y s_z + s_\varphi s_x \\ s_x s_z + s_\varphi s_y & s_y s_z - s_\varphi s_x & \frac{1}{2} - s_x^2 - s_y^2 \end{pmatrix}$$

$$\textit{mode vectors: } \phi = \begin{pmatrix} 1_{3\times 3} \\ 0_{3\times 3} \end{pmatrix}, \quad \overline{\phi} = \begin{pmatrix} 0_{3\times 3} \\ 1_{3\times 3} \end{pmatrix}$$

$$\textit{kinematical equations of motion: } \frac{dq}{dt} = \frac{1}{2} \begin{pmatrix} -s_x & -s_y & -s_z \\ s_\varphi & -s_z & s_y \\ s_z & s_\varphi & -s_x \\ -s_y & s_x & s_\varphi \end{pmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}$$

Unconstrained Joint As an unconstrained joint has three translational and three rotational degrees of freedom, its velocity is appropriately represented by the relative

spatial velocity of the moving body. The joint position can described by a translation represented by vector $(x, y, x)^T$ followed by a rotation represented by quaternion $(s_\varphi, s_x, s_y, s_z)$. When computing the rate of change of joint position q from joint velocity $\dot{q}$ it has to be taken into account that the linear component of the position is resolved in the frame of the base body while the linear component of the joint velocity is resolved in the frame of the moving body.

state variables: $q = (s_\varphi, s_x, s_y, s_z, x, y, z)^T, \quad \dot{q} = (\omega, v)^T$

positional constraint equations: $b'' = \begin{pmatrix} x \\ y \\ z \end{pmatrix},$

$$A'' = 2 \begin{pmatrix} \frac{1}{2} - s_y^2 - s_z^2 & s_x s_y + s_\varphi s_z & s_x s_z - s_\varphi s_y \\ s_x s_y - s_\varphi s_z & \frac{1}{2} - s_x^2 - s_z^2 & s_y s_z + s_\varphi s_x \\ s_x s_z + s_\varphi s_y & s_y s_z - s_\varphi s_x & \frac{1}{2} - s_x^2 - s_y^2 \end{pmatrix}$$

mode vectors: $\phi = 1_{6\times 6}, \quad \overline{\phi} = 0_{6\times 0}$

kinematical equations of motion:

$$\frac{dq}{dt} = \begin{pmatrix} \frac{1}{2} \begin{pmatrix} -s_x & -s_y & -s_z \\ s_\varphi & -s_z & s_y \\ s_z & s_\varphi & -s_x \\ -s_y & s_x & s_\varphi \end{pmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \\ (A'')^T \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} \end{pmatrix}$$

2.1.3 Recursive Description of Articulated Bodies

Throughout the following, articulated bodies are required to be acyclic. By choosing one of the links as the base link, the topology of an acyclic articulated body is that of a tree with the base link at the root. Effectively, this means that each link — with the exception of the base link — possesses one joint at which it is attached to its parent link and possibly one or more joints at which child links are attached. If n is the number of links of which an articulated body is comprised, let the links be numbered from 1 to n such that the number of a link is always greater than the number of its parent link, and let pnt a mapping such that $pnt(i)$ is the number of the parent link of link i. The joint connecting link i to its parent link is called joint i. Furthermore, formally assume the existence of an additional body 0 of infinite mass and inertia which acts as the parent link of link 1, the base link, to which it is connected by an unconstrained joint. Figure 2.2 shows an articulated creature

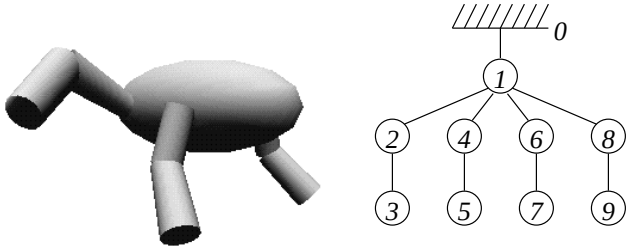

Figure 2.2: *A four-legged creature as evolved in Chapter 3 and a tree describing its structure. Each of the legs of the creature corresponds to a branch of the tree, and the trunk corresponds to node* 1.

as evolved in Chapter 3 and a tree describing its structure in which the links are numbered according to this convention.

Rather than resolving all variables in a global reference frame, every link has a local coordinate system attached to it. An immediate benefit from this convention is the fact that the inertia of a link and the mode vectors of its joint are constants in local coordinates. In all of the following, velocity, acceleration, and the torques and forces acting on a link will be given in the link's local coordinate system without it being explicitly indicated. Moreover, as transformations will be required only between coordinate systems of consecutive links, $X_{pnt(i),i}$ can be written shorter as X_i.

2.2 Equations of Motion

In this section, the equations governing the motion of the links comprising an articulated body are presented in recursive form. That is, the velocity and acceleration of a link are given in terms of velocity and acceleration of the parent link and the relative joint motion and acceleration, and the equations of Newton and Euler are formulated in terms of the forces exerted on a link by its neighbors. Detailed derivations of the equations presented in this section are beyond the scope of this thesis and can be found in Craig [13] or Featherstone [16].

Velocities

The spatial velocity of a link incorporates both the linear velocity of a link's hinge point and the rotational velocity of the link. For frame i it can be written as

$$\mathbf{v}_i = X_i \mathbf{v}_{pnt(i)} + \phi_i \dot{q}_i, \tag{2.4}$$

showing that it is composed of the velocity of the parent link and the new velocity components added by the motion of joint i.

Accelerations

The spatial acceleration of a link is comprised of its linear and angular acceleration. For frame i it can be written as

$$\mathbf{a}_i = X_i \mathbf{a}_{pnt(i)} + \zeta_i + \phi_i \ddot{q}_i \tag{2.5}$$

where the term

$$\zeta_i = \begin{pmatrix} 0 \\ A_i(\omega_{pnt(i)} \times (\omega_{pnt(i)} \times b_i)) \end{pmatrix} + \begin{pmatrix} A_i \tilde{\omega}_{pnt(i)} & 0 \\ 0 & 2A_i \tilde{\omega}_{pnt(i)} \end{pmatrix} \phi_i \dot{q}_i$$

results from the fact that the coordinate frame in which the acceleration is resolved is moving. Therefore, the acceleration of link i is the sum of that of its parent link taking into account the motion of the coordinate frame and the new acceleration components added by the acceleration of joint i.

Laws of Newton and Euler

The force balance equation in spatial notation unites the laws of Newton and Euler into a single vector equation. Each link has spatial forces exerted on it by its neighbors, and in addition experiences an inertial spatial force and possibly external forces, such as those arising due to contact between two links. As $\mathbf{f}_i$ acts by definition with positive sign and the spatial forces $\mathbf{f}_j$, where j is a child link of i act with negative sign on link i, the laws of Newton and Euler can be written as

$$\mathcal{I}_i \mathbf{a}_i = \mathbf{f}_i - \sum_{\{j|i=pnt(j)\}} X_j^T \mathbf{f}_j + \beta_i \tag{2.6}$$

where

$$\beta_i = \mathbf{f}_i^{ext} - \begin{pmatrix} \omega_i \times I_i \omega_i \\ m_i \omega_i \times (\omega_i \times d_i) \end{pmatrix}$$

is a spatial bias force accounting for centripetal and Coriolis forces and any external forces $\mathbf{f}_i^{ext}$ that may be applied to the articulated body.

Equations (2.4), (2.5), and (2.6) in connection with the constraint equations imposed by the joints capture all of the laws of physics governing the motion of articulated bodies. More specifically, after substitution of Equation (2.3) into Equation (2.6) and having computed all angular velocities using Equation (2.4), Equations (2.5) and (2.6) form a system of $12n$ linear equations in the $12n$ unknowns $\ddot{q}_i$, $\overline{\lambda}_i$, and $\mathbf{a}_i$ for $i = 1, \ldots, n$. In the next section, an efficient algorithm for the solution of this system is presented.

2.3 Recursive Formulation of Forward Dynamics

In this section, FEATHERSTONE's [15] articulated body algorithm is derived in the formulation of BRANDL, JOHANNI, and OTTER [8] who clarified some notational details and generalized it from handling only joints with a single degree of freedom to use the general joint model of ROBERSON and SCHWERTASSEK [41]. LILLY [28] referred to it as the most efficient known algorithm for solving the equations of motion of an unconstrained articulated body without internal loops and interaction with the environment. So as to facilitate its development, the equations underlying the algorithm are first derived for chain-structured systems before they are generalized to arbitrary tree-structured systems.

2.3.1 Chain-Structured Systems

For an articulated body with the structure of a chain, the numbering conventions outlined in Subsection 2.1.3 mean that the links of the body are numbered in consecutive order, starting at the base, from 1 to n, with link i being connected to link $i-1$ by joint i. Therefore, $pnt(i) = i-1$ and Equation (2.6) can be replaced by

$$\mathcal{I}_i \mathbf{a}_i = \mathbf{f}_i - X_{i+1}^T \mathbf{f}_{i+1} + \beta_i \tag{2.7}$$

where $\mathbf{f}_{n+1} = 0$ since the outermost link is unconstrained.

The key to an efficient recursive solution of the resulting system is to summarize the inertial properties of the subchains comprised of links $i \ldots, n$ into a new inertial quantity $\mathcal{I}_i^*$ called the *articulated body inertia*. As a consequence of the equations of motion given in Section 2.2 a linear relationship exists between a force applied to a component of an articulated body and the resulting acceleration of this component or of any other component of the body. Therefore, articulated body inertias can be represented using matrices, and it can be written

$$\mathcal{I}_i^* \mathbf{a}_i = \mathbf{f}_i + \beta_i^*. \tag{2.8}$$

In this equation, β_i^* is an accumulated bias force term reflecting the force which has to be exerted on link i if it is not to accelerate. It should be noted that the articulated body inertia $\mathcal{I}_i^*$ does not conform to the special form of rigid body inertias given in Equation (2.1), physically implying that there is no such thing as a center of mass for an articulated body.

For $i = n$ it is obvious from Equation (2.7), taking into account that $\mathbf{f}_{n+1} = 0$, that

$$\mathcal{I}_n^* = \mathcal{I}_n$$

and

$$\beta_n^* = \beta_n.$$

To compute $\mathcal{I}_{i-1}^*$ and β_{i-1}^* for $i \leq n$, assume that $\mathcal{I}_i^*$ is symmetric and positive definite. This is true for $i = n$ as shown in Section 2.1.1 and will be demonstrated below for $i < n$. Both sides of Equation (2.8) can be projected onto the free modes of the joint by premultiplication with ϕ_i^T, effectively eliminating the dependence on $\overline{\lambda}_i$. Using Equations (2.5), (2.3), and (2.2) leads to

$$\phi_i^T \mathcal{I}_i^* (X_i \mathbf{a}_{i-1} + \zeta_i + \phi_i \ddot{q}_i) = \lambda_i + \phi_i^T \beta_i^*.$$

As $\mathcal{I}_i^*$ is symmetric and positive definite and ϕ_i has full column rank, the inverse of matrix $\mathcal{M}_i = \phi_i^T \mathcal{I}_i^* \phi_i$ exists and $\ddot{q}$ can be expressed as

$$\ddot{q}_i = \mathcal{M}_i^{-1}(\lambda_i + \phi_i^T(\beta_i^* - \mathcal{I}_i^*(X_i \mathbf{a}_{i-1} + \zeta_i))). \tag{2.9}$$

Making use of Equation (2.5) and substituting Equation (2.9) into Equation (2.8) yields, after some simple transformations,

$$\mathbf{f}_i = \mathcal{N}_i X_i \mathbf{a}_{i-1} - \gamma_i \tag{2.10}$$

where

$$\mathcal{N}_i = \mathcal{I}_i^* - \mathcal{I}_i^* \phi_i \mathcal{M}_i^{-1} \phi_i^T \mathcal{I}_i^*$$

and

$$\gamma_i = \beta_i^* - \mathcal{N}_i \zeta_i - \mathcal{I}_i^* \phi_i \mathcal{M}_i^{-1}(\lambda_i + \phi_i^T \beta_i^*). \tag{2.11}$$

Therefore, using Equations (2.7) and (2.10) for link $i - 1$, it follows that

$$\begin{aligned} \mathbf{f}_{i-1} &= \mathcal{I}_{i-1} \mathbf{a}_{i-1} + X_i^T \mathbf{f}_i - \beta_{i-1} \\ &= (\mathcal{I}_{i-1} + X_i^T \mathcal{N}_i X_i) \mathbf{a}_{i-1} - X_i^T \gamma_i - \beta_{i-1}. \end{aligned}$$

Comparison with Equation (2.8) shows that

$$\mathcal{I}_{i-1}^* = \mathcal{I}_{i-1} + X_i^T \mathcal{N}_i X_i \tag{2.12}$$

and

$$\beta_{i-1}^* = \beta_{i-1} + X_i^T \gamma_i. \tag{2.13}$$

Furthermore, Equation (2.2) can be used together with the fact that $\mathcal{I}_i^*$ is symmetric to show that $\mathcal{N}_i$ can be written as

$$\mathcal{N}_i = \overline{\phi}_i \overline{\mathcal{M}}_i^{-1} \overline{\phi}_i^T$$

where $\overline{\mathcal{M}}_i = \overline{\phi}_i^T \mathcal{I}_i^{-1} \overline{\phi}_i$. As $\overline{\phi}_i$ has full column rank, it follows that $\mathcal{N}_i$ is positive semidefinite, and as $\mathcal{I}_{i-1}$ is known to be symmetric and positive definite and X_i is regular, $\mathcal{I}_{i-1}^*$ is symmetric and positive definite.

2.3.2 Tree-Structured Systems

The generalization of the equations to arbitrary tree-structured systems is straightforward. It has to be taken into account that a link can now have several child links, making it necessary to use Equation (2.6) instead of Equation (2.7). As a consequence, Equations (2.12) and (2.13) have to be replaced by

$$\mathcal{I}_i^* = \mathcal{I}_i + \sum_{\{j|i=pnt(j)\}} X_j^T \mathcal{N}_j X_j \tag{2.14}$$

and

$$\beta_i^* = \beta_i + \sum_{\{j|i=pnt(j)\}} X_j^T \gamma_j, \tag{2.15}$$

respectively.

Equations (2.14), (2.15), (2.9), and (2.5) form the basis for a recursive simulation algorithm for articulated bodies. The required articulated body inertias $\mathcal{I}_i^*$ and the accumulated force terms β_i^* can be computed from Equations (2.14) and (2.15), respectively, by starting at the leaves and progressing towards the root. The numbering convention on the links introduced in Subsection 2.1.3 makes it possible to simply proceed in order of decreasing i. Subsequently, the motion of link i can be computed from that of its parent link by means of Equations (2.9) and (2.5). Since the motion of reference frame 0 is known, the accelerations for the entire articulated body can be obtained.

The resulting algorithm is given in Algorithm 2.1. The first loop computes coordinate transformations, velocities, and bias terms for all links and initializes the inertia matrices. The second loop computes articulated body inertias and accumulates forces. The third loop propagates accelerations from the base outwards. To include gravity, instead of applying an acceleration of $-9.81 m/s^2$ to all links of the articulated body, it is possible to apply an equal but opposite acceleration to the inertial reference frame. It is obvious that both time

$\mathbf{v}_0 = 0_{6\times 1}$;

$\mathbf{a}_0 = (0.0, 0.0, 0.0, 0.0, 9.81, 0.0)^T$;

`for` $i = 1$ `to` n { `/* compute transformations, velocities, and bias terms */`

$X_i = X_i'' X_i'$;

$$\zeta_i = \begin{pmatrix} 0 \\ A_i(\omega_{i-1} \times (\omega_{i-1} \times b_i)) \end{pmatrix} + \begin{pmatrix} A_i\tilde{\omega}_{pnt(i)} & 0 \\ 0 & 2A_i\tilde{\omega}_{pnt(i)} \end{pmatrix} \phi_i \dot{q}_i;$$

$\mathbf{v}_i = X_i \mathbf{v}_{pnt(i)} + \phi_i \dot{q}_i$;

$\mathcal{I}_i^* = \mathcal{I}_i$;

$$\beta_i^* = \mathbf{f}_i^{ext} - \begin{pmatrix} \omega_i \times I_i\omega_i \\ m_i\omega_i \times (\omega_i \times d_i) \end{pmatrix};$$

}

`for` $i = n$ `to` 1 { `/* propagate forces and inertias */`

$\mathcal{M}_i = \phi_i^T \mathcal{I}_i^* \phi_i$;

`if` $(pnt(i) > 0)$ {

$\mathcal{N}_i = \mathcal{I}_i^* - \mathcal{I}_i^* \phi_i \mathcal{M}_i^{-1} \phi_i^T \mathcal{I}_i^*$;

$\gamma_i = \beta_i^* - \mathcal{N}_i\zeta_i - \mathcal{I}_i^* \phi_i \mathcal{M}_i^{-1}(\lambda_i + \phi_i^T \beta_i^*)$;

$\mathcal{I}_{pnt(i)}^* = \mathcal{I}_{pnt(i)}^* - X_i^T \mathcal{N}_i X_i$;

$\beta_{pnt(i)}^* = \beta_{pnt(i)}^* + X_i^T \gamma_i$;

}

}

`for` $i = 1$ `to` n { `/* propagate accelerations */`

$\ddot{q}_i = \mathcal{M}_i^{-1}(\lambda_i + \phi_i^T(\beta_i^* - \mathcal{I}_i^*(X_i \mathbf{a}_{pnt(i)} + \zeta_i)))$;

$\mathbf{a}_i = X_i \mathbf{a}_{pnt(i)} + \zeta_i + \phi_i \ddot{q}_i$;

}

Algorithm 2.1: *Multibody algorithm in the formulation of* Brandl, Johanni, *and* Otter *[8]. Relative joint accelerations $\ddot{q}_i$ are computed for $i = 1, \ldots, n$ from joint positions q_i and joint velocities $\dot{q}_i$, and a description of the articulated structure by means of the joint types and the transformations A', b' between consecutive links.*

and storage requirements of the algorithm are linear in the number of links and therefore in the number of degrees of freedom.

Algorithm 2.1 efficiently solves the problem of computing the joint accelerations for an acyclic articulated body without interaction with the environment. The subject of the following section is methods of incorporating kinematic constraints imposed by contacts between different bodies into the simulation.

2.4 Contacts and Collisions

A realistic simulation of articulated bodies demands that no two bodies interpenetrate. Both contacts between different links of one articulated body such as different legs of an artificial creature and contacts with the environment such as the ground plane or other bodies have to be considered. Contacts impose kinematic constraints on the relative velocity and acceleration of the bodies involved. In order to enforce these constraints, a simulation program has to detect contacts between bodies and then take appropriate action. If the detected amount of interpenetration is within some tolerable range, it is sufficient to apply counter-acting forces computed on the basis of some contact model; if it is not, the previous time step has to be repeated using a smaller step size.

Two kinds of contact can be distinguished. Static contacts extend over a finite period of time and require that the normal component of the force opposing interpenetration does no work on the bodies in contact. Modes of static contact include resting contact as well as rolling and sliding contact. On the other hand, dynamic contacts, henceforth referred to as collisions, are of infinitesimal duration and result in instantaneous changes in the velocity of the colliding bodies. So as to enforce the constraints imposed by collisions, impulses, a mathematical idealization of very large forces applied over very short intervals of time, have to be employed.

This section first discusses contact detection and some geometrical issues at contact points in Subsection 2.4.1 before deriving the equations describing the effect of applying a contact force to an articulated body in Subsection 2.4.2. The derivation of these equations is an original contribution of this thesis and allows for the generalization of the standard method for handling frictional collisions between rigid bodies to acyclic articulated bodies described in Subsection 2.4.3. Finally, Subsection 2.4.4 presents a classification of current algorithms handling multiple simultaneous static contacts, outlines the problems inherent in them, and then suggests a new contact handling algorithm for articulated bodies to be used in Chapter 3.

2.4.1 Contact Geometry

The purpose of this subsection is to introduce the problem of contact detection and to list some common assumptions about geometrical conditions at contact points which, in combination with the assumption that the links an articulated body is comprised of are perfectly rigid and therefore propagate forces and impulses instantaneously, allow for the possibility of restricting the attention to events at the contact point alone.

Contact Detection

The problem of detecting contacts is the most time consuming task in many simulations. To be able to check whether two given objects are colliding at a particular point in time, the positions of both objects have to be known with respect to a common coordinate system. Frame 0, the world coordinate system, can serve as such a common frame. The transformation matrix between frame 0 and an arbitrary link frame i can be computed as

$$X_{0,i} = X_i \dots X_1,$$

with the multiplication extending over all transformation matrices of links on the path from the root of the articulated structure to link i. The components $b_{0,i}$ and $A_{0,i}$ of $X_{0,i}$ can be used to compute the location and orientation of link i in world coordinates.

A variety of algorithms aim at reducing the number of collision tests from $O(n^2)$, where n is the number of objects in the scene, by using bounding boxes or spatial decomposition schemes, or attempt to make the collision tests themselves more efficient. In applications such as dynamic simulation, geometric or temporal coherence can often be exploited by making use of the fact that objects move on continuous trajectories in space. For example, a plane separating two objects at one time step is often likely to separate them during the next time step as well. The details of the collision test are dependent on the shape of the objects being tested.

Efficient collision detection is of minor importance in the current work. The number of bodies making up the virtual creatures evolved for walking is rather small, and the number of collision tests to be performed can be restricted by making use of the fact that legs from opposite sides of a creature are not likely to collide. For all simulations carried out for this thesis, it turned out to be sufficient to employ a very simple approach, testing for collisions only between segments of adjacent limbs of a creature. The collision tests themselves consist in an analytical computation of the minimal distance between the frustums making up the limbs. For larger simulations with several articulated creatures

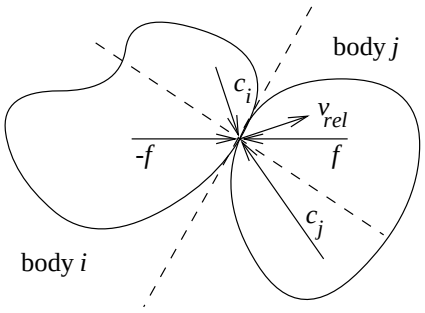

Figure 2.3: *A contact between bodies i and j rendered in two dimensions. The broken lines indicate the orientation of the contact coordinate system, the z-axis of which is aligned with the surface normals at the contact point.*

involved, a more sophisticated approach would have to be employed. BARAFF [4] gives an extensive overview of such algorithms.

Collision Coordinates

So as to derive the equations of motion required in the following, it will be assumed that all contacts are point contacts. If extended contact occurs, it will be modeled as a finite number of point contacts. For example, if the bodies in contact are composed of polyhedra, imposing non-penetration constraints at the vertices of the polygonal contact area is sufficient to prevent interpenetration over the entire area. It will also be assumed that there always is a common tangent plane for the bodies in contact. If one of the features in contact is a plane, then it is taken to be the tangent plane. If both features in contact are edges, the tangent plane is defined to be the plane spanned by the edges. The unit normal vector of the tangent plane is referred to as the contact normal.

Relative velocities, contact forces, and the equations of motion for a contact will be given with respect to a contact coordinate system centered at the point of contact, which is defined in such a way that its z-axis coincides with the contact normal. Formally, for a contact between bodies i and j, the contact coordinate system is defined by a spatial transformation

$$X_{i,con} = \begin{pmatrix} A_{i,con} & 0 \\ A_{i,con}\tilde{c}_i^T & A_{i,con} \end{pmatrix}$$

which transforms i-coordinates into contact coordinates, where c_i is the contact point in i-coordinates and $A_{i,con}$ is a rotation matrix transforming the collision normal expressed in i-coordinates into the unit vector $\begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^T$. A corresponding spatial transformation $X_{j,con}$ for link j transforms j-coordinates into contact coordinates. Figure 2.3 illustrates the location and orientation of the contact coordinate system and some of the quantities used during contact analysis.

By convention, the contact normal is directed such that it points inwards for link i and consequently outwards for link j, and the relative velocity between bodies i and j at the point of contact in contact coordinates is defined as

$$\mathbf{v}_{rel} = X_{i,con}\mathbf{v}_i - X_{j,con}\mathbf{v}_j. \tag{2.16}$$

If the z-component of the linear component $v_{rel} = (v_x, v_y, v_z)^T$ thereof, the relative normal velocity v_z, is non-negative, the bodies are receding and no action has to be taken to prevent interpenetration. If $v_z < 0$, a force f has to be applied at the point of contact to body i in the direction of the contact normal and to body j in the opposite direction.

2.4.2 Contact Equations

This subsection derives the equations describing the effect that applying an external spatial force at a particular point of an articulated body has on the acceleration of the body. These equations make it possible to generalize the common algorithms for handling collisions and static contacts between rigid bodies to articulated bodies. Like the articulated body method outlined in Section 2.3, the computational process leading to the inertial quantities required here is recursive and requires time growing linearly in the number of links of the articulated body. The derivation is an original contribution of this thesis. Recently, MIRTICH [33] found a similar algorithm for computing the same quantities.

For the following derivation it is assumed that the components of the acceleration due to forces other than the applied external force are computed in a separate process and therefore do not have to be considered here. This assumption will prove valid for both the algorithms for handling collisions and for handling static contacts described below. Due to the linearity of the equations for propagating forces and accelerations introduced in Section 2.3, the total acceleration is the sum of the two parts. A result of the assumption is considerably simplified equations. In particular, making use of $\lambda_i = 0$ and $\zeta_i = 0$, Equations (2.11) and (2.15) can be combined to yield

$$\beta^*_{pnt(i)} = \eta_i^T \beta_i^*, \tag{2.17}$$

where

$$\eta_i = (1_{6\times 6} - \phi_i \mathcal{M}_i^{-1} \phi_i^T \mathcal{I}_i^*) X_i,$$

and from Equations (2.9) and (2.5) it follows

$$\mathbf{a}_i = \eta_i \mathbf{a}_{pnt(i)} + \phi_i \mathcal{M}_i^{-1} \phi_i^T \beta_i^*. \tag{2.18}$$

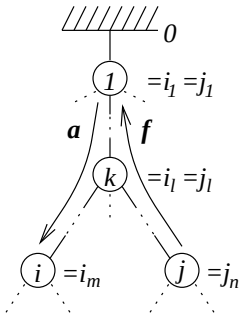

Figure 2.4: *Part of a tree representing an articulated body. The inverse spatial inertia matrix $\varkappa_{ij}$ which determines the effect of a spatial force $\mathbf{f}_j$ applied to link j on the acceleration $\mathbf{a}_i$ of link i can be found by propagating the force from link j to the root link and subsequently the resulting acceleration from the root to link i. Link k is the outermost link on the paths from the root to both link i and link j.*

Equations (2.17) and (2.18) can be used to compute the effect that applying a force $\mathbf{f}_j$ to link j of an articulated body has on the acceleration $\mathbf{a}_i$ of link i. More specifically, Equation (2.17), together with $\beta_j^* = \mathbf{f}_j$, can be used to propagate the force from link j recursively to the base link, and Equation (2.18) in combination with $\mathbf{a}_0 = 0$ can be used to propagate the resulting acceleration from the base link to link i. As will be shown, the relationship between the two quantities is linear and can be written as

$$\mathbf{a}_i = \varkappa_{ij}\mathbf{f}_j. \tag{2.19}$$

The task at hand is to compute the inverse inertia matrix $\varkappa_{ij}$ which depends solely on the joint positions and the mass and inertia properties of the articulated body.

Figure 2.4 illustrates the situation. Let $i_1 = 1, i_2, \ldots, i_m = i$ and $j_1 = 1, j_2, \ldots, j_n = j$ be the links on the paths from the base link of the articulated body to links i and j, respectively. Furthermore, let k be the number of the outermost link which is in both the paths. Then there is an l such that $i_l = j_l = k$ and $i_{l+1} \neq j_{l+1}$. The force term β_μ^* is non-zero only for links $\mu \in \{j_1, \ldots, j_n\}$ and can be computed for link j_ν from Equation (2.17) as $\beta_{j_\nu}^* = \eta_{j_{\nu+1}}^T \ldots \eta_{j_n}^T \mathbf{f}_j$. Therefore, from Equation (2.18) it follows that

$$\varkappa_{1j} = \phi_1 \mathcal{M}_1^{-1} \phi_1^T \eta_{j_2}^T \ldots \eta_{j_n}^T$$

and

$$\varkappa_{i_\nu j} = \begin{cases} \eta_{i_\nu} \varkappa_{i_{\nu-1} j} + \phi_{i_\nu} \mathcal{M}_{i_\nu}^{-1} \phi_{i_\nu}^T \eta_{j_{\nu+1}}^T \ldots \eta_{j_n}^T & \text{for } 1 < \nu \leq l \\ \eta_{i_\nu} \varkappa_{i_{\nu-1} j} & \text{for } l < \nu \leq m \end{cases}.$$

As $i_m = i$, it follows trivially that $\varkappa_{ij} = \varkappa_{i_m j}$. In the following subsections, the quantity $\varkappa_{ij}$ will be used to compute collision impulses and contact forces.

2.4.3 Frictional Collisions

Collisions between hard and compact bodies are highly complex processes involving vibration waves propagating through the bodies, local deformations in the vicinity of the contact

area, and other highly nonlinear phenomena. So as to make their simulation tractable, a physical model which is not only physically reasonably accurate but also computationally efficient has to be found.

Collisions result in changes in momentum and some loss of kinetic energy for the bodies involved during a brief contact period. Mathematically, this can be achieved by applying reaction impulses to the colliding bodies at the point of impact which are equal in magnitude but opposite in direction. The central problem in collision resolution is to determine the magnitude and direction of the impulses required to achieve realistic behavior. Generally, the reaction impulse that develops during a collision depends on the initial relative velocity and material and inertia properties for both bodies at the impact point. The following describes a procedure for computing reaction impulses for articulated bodies, generalizing a similar argument for rigid bodies which can be found in a number of references, including Mirtich and Canny [31], Keller [21], and Stronge [47].

Collision Model

Generalizing a statement on rigid bodies made by Baraff [4], treating the links of an articulated body as perfectly rigid bodies leads to the instantaneous propagation of forces and the possibility of the replacement of complex "micromechanical" processes by simple "macromechanical" results. As a consequence, the analysis of a collision can be confined to events at the contact point. However, in reality no body is perfectly rigid, and the rigid body assumption has to be adjusted to arrive at a contact model allowing an analytical treatment of the collision process. The usual procedure is to postulate infinitesimal collision time, prescribe a simple deformation history, and define tangential forces by Coulomb's friction law as outlined below.

Infinitesimal collision time: The duration of a collision is assumed to be negligible on the simulation time scale. However, for the sake of computing an appropriate magnitude of the impulse to be applied, the moment of impact has to be regarded as a time interval of finite length on a compressed time scale. On this time scale, interpenetration is prevented by a finite force f applied to the colliding bodies for the duration of the collision, which gives rise to continuous changes in velocity. Macroscopically, the collision impulse p can be computed as

$$p = \int f dt = \int dp. \tag{2.20}$$

The postulate of infinitesimal collision time allows for two important approximations. First, the positions of the colliding objects can be regarded as constant during the entire collision,

and second, the effects of other forces acting on the bodies can be disregarded as they are negligible compared to the large impulsive forces. As a consequence, Equation (2.19) can be used to compute the change in relative velocity which results from applying a collision impulse. More specifically, if links i and j of an articulated body are colliding and an impulse p expressed in contact coordinates is applied to link i and the opposite impulse $-p$ to link j at the point of contact, then the relative velocity

$$v_{rel} = A_{i,con}(v_i + \omega_i \times c_i) - A_{j,con}(v_j + \omega_j \times c_j)$$

experiences a change which can be computed by forming the derivative of Equation (2.19) with respect to time as

$$\Delta v_{rel} = \Lambda p, \tag{2.21}$$

where

$$\Lambda = \begin{pmatrix} 0_{3\times3} & 1_{3\times3} \end{pmatrix} (X_{i,con}\varkappa_{ii}X_{i,con}^T - X_{i,con}\varkappa_{ij}X_{j,con}^T - X_{j,con}\varkappa_{ji}X_{i,con}^T + X_{j,con}\varkappa_{jj}X_{j,con}^T) \begin{pmatrix} 0_{3\times3} \\ 1_{3\times3} \end{pmatrix}$$

is an inverse inertial quantity which combines the dynamic properties of the entire articulated body and projects them to the point of contact.

Postulated deformation history: A coefficient of restitution serves as an approximation to the complex deformations and energy losses which occur when two real bodies collide. On the collision time scale, the collision process is regarded as consisting of two different phases which can be distinguished by the sign of the relative normal contact velocity v_z. During the compression phase, which is marked by negative relative normal velocity, a deformation of the bodies occurs, and part of the kinetic energy of the two bodies is transformed into elastic strain energy which is stored in the bodies. When $v_z = 0$, the point of maximum compression has been attained. During the subsequent restitution phase, the bodies return to their original shapes and the stored energy is released, restoring part of the kinetic energy the bodies had before the collision, and thereby driving them apart. The work done by the normal component p_z of the collision impulse on the normal component of the relative velocity at the point of impact is

$$E = -\int v_z dp_z, \tag{2.22}$$

the elastic strain energy. The work done by the tangential component of p is frictional energy and irrevocably lost.

The duration of the restitution phase is determined by a constant ϵ, the *coefficient of restitution*, which is assumed to depend only on material properties of the colliding bodies. If $\epsilon = 1.0$, the collision is completely elastic and no energy is lost. If $\epsilon = 0.0$, the collision is totally plastic and the objects do not separate after the collision. There are three competing hypotheses as for which quantities the coefficient of restitution relates, all of which are in widespread use. The kinematic hypothesis, also termed Newton's hypothesis, prescribes final normal velocity, defining the coefficient of restitution as the negative of the ratio of normal component of relative velocity between contact points at separation to that at incidence. Poisson's hypothesis prescribes the normal impulse applied during restitution, defining the coefficient of restitution as the negative of the normal reaction impulse during restitution divided by normal reaction impulse during compression. A more recent hypothesis suggested by STRONGE [47] demands that the coefficient of restitution is the square root of the ratio of elastic strain energy released at the contact points during restitution to the energy absorbed by deformation during compression. Formally,

$$\epsilon^2 = -\frac{\int\limits_{rest} v_z dp_z}{\int\limits_{comp} v_z dp_z} \tag{2.23}$$

where the integral in the numerator extends over the restitution phase and denotes the elastic strain energy released during restitution while that in the denominator extends over the compression phase and denotes the elastic strain energy absorbed during compression. All three hypotheses are equivalent for collinear or non-frictional collisions. However, STRONGE [46] was able to show that only Equation (2.23) is always energetically consistent for non-collinear collisions with finite friction.

Coulomb friction: The Coulomb friction model describes a well accepted empirical relationship between the normal and tangential components of the reaction impulse at the contact point. Effectively, it will be used for defining the frictional component of the contact force. It states that at any point in time the tangential component of the collision force is directed to oppose the tangential velocity between the colliding bodies, and that the magnitude of the tangential force is limited by the product of a constant μ representing material behavior and the magnitude of the normal force; i.e. that

$$\sqrt{dp_x^2 + dp_y^2} \leq \mu dp_z. \tag{2.24}$$

While the tangential component of the relative velocity v_{rel} between the two bodies is

non-zero, this leads to the equations

$$dp_x = -\frac{\mu v_x}{\sqrt{v_x^2 + v_y^2}} dp_z \qquad \text{and} \qquad dp_y = -\frac{\mu v_y}{\sqrt{v_x^2 + v_y^2}} dp_z. \tag{2.25}$$

If the tangential component of the relative motion between the two bodies is zero, the frictional force still acts to oppose sliding. However, if a tangential force less than μ times the normal force is sufficient to prevent sliding, only that force will be applied. Equations (2.25) define what is commonly known as the friction cone. In the case of dynamic friction, the friction force can be found on the surface of this cone while static friction forces are in the interior.

Sliding Mode

If the tangential component of the relative velocity between the two bodies is non-zero, Equations (2.25) can be used to express the rate of change of tangential impulse with respect to normal impulse as

$$\frac{dp_x}{dp_z} = -\frac{\mu v_x}{\sqrt{v_x^2 + v_y^2}} \qquad \text{and} \qquad \frac{dp_y}{dp_z} = -\frac{\mu v_y}{\sqrt{v_x^2 + v_y^2}}. \tag{2.26}$$

Differentiating Equation (2.21) with respect to the normal component of reaction impulse p_z, it follows that

$$\begin{pmatrix} dv_x/dp_z \\ dv_y/dp_z \\ dv_z/dp_z \end{pmatrix} = \Lambda \begin{pmatrix} -\mu v_x/\sqrt{v_x^2 + v_y^2} \\ -\mu v_y/\sqrt{v_x^2 + v_y^2} \\ 1.0 \end{pmatrix}. \tag{2.27}$$

Equation (2.27) is a nonlinear differential equation which cannot be solved in closed form in the general case. So as to track v_{rel} during the course of the collision, Equation (2.27) has to be numerically integrated with p_z as the independent variable. The total reaction impulse p can be computed by using Equation (2.25) and summing up the differential normal impulses dp_z using Equation (2.20). The elastic strain energy stored in the colliding bodies can be tracked using Equation (2.22).

Sticking Mode

If there is no relative tangential motion between the two colliding bodies the frictional force acts to maintain sticking. That is, if the force required to achieve $dv_x/dp_z = dv_y/dp_z = 0.0$

is within the friction cone, it acts to prevent any tangential motion between the two bodies. To compute the force required to maintain sticking, differentiating Equation (2.21) with respect to p_z and setting $dv_x/dp_z = dv_y/dp_z = 0.0$ can be used to compute

$$\frac{dp_x}{dp_z} = \frac{k_0}{k_2} \qquad \text{and} \qquad \frac{dp_y}{dp_z} = \frac{k_1}{k_2}$$

where

$$\begin{aligned} k_0 &= \lambda_{01}\lambda_{12} - \lambda_{02}\lambda_{11} \\ k_1 &= \lambda_{10}\lambda_{02} - \lambda_{00}\lambda_{12} \\ k_2 &= \lambda_{00}\lambda_{11} - \lambda_{01}\lambda_{10} \end{aligned}$$

with the λ_{ij} being the components of matrix Λ. Inequality (2.24) shows that

$$\sqrt{k_0^2 + k_1^2} \leq \mu k_2 \tag{2.28}$$

is sufficient to guarantee that the sticking condition can be maintained. In this case Equation (2.27) has to be replaced by

$$\begin{pmatrix} dv_x/dp_z \\ dv_y/dp_z \\ dv_z/dp_z \end{pmatrix} = \Lambda \begin{pmatrix} k_0/k_2 \\ k_1/k_2 \\ 1.0 \end{pmatrix}. \tag{2.29}$$

It is worth noting that this force stays constant for the remainder of the collision, making it possible to discard the numerical integration and compute the remaining impulse to be applied in a single step.

If Condition (2.28) is not fulfilled, the frictional force is not sufficient to maintain sticking, and sliding resumes. Immediately after the resumption of sliding, the behavior of the tangential impulse is again governed by Equations (2.26). However, BHATT and KOECHLING [6] have shown that in the case of resumed sliding the direction of sliding is constant and given by

$$\frac{dp_y}{dp_x} = \frac{v_y}{v_x}.$$

Thus, using Equation (2.21) to express v_x and v_y in terms of dp_x and dp_y and substituting $\gamma = dp_y/dp_x$, it follows

$$\gamma = \frac{\lambda_{10} + \lambda_{11}\gamma + \lambda_{12}\sqrt{1+\gamma^2}/\mu}{\lambda_{00} + \lambda_{01}\gamma + \lambda_{02}\sqrt{1+\gamma^2}/\mu}. \tag{2.30}$$

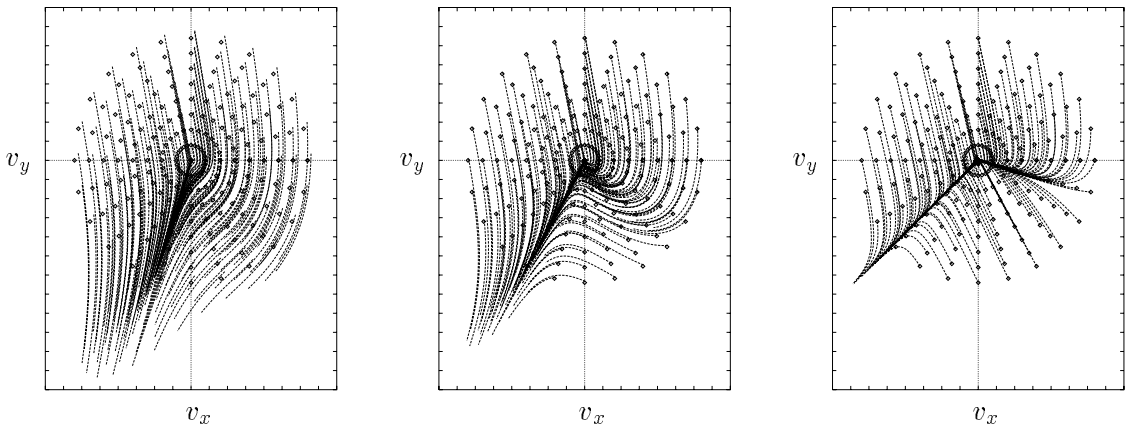

Figure 2.5: *Influence of friction constant μ on the flow in tangent velocity space. The horizontal and vertical axes represent v_x and v_y, respectively. The values of friction constant μ are, from left to right,* 0.3, 0.4, *and* 0.7. *The dots indicate the inital values of the trajectories.*

Bhatt and Koechling [6] have demonstrated that Equation (2.30), a quartic equation in parameter γ, always has exactly one solution for which sliding velocity and applied impulse are opposed. This solution can be found by Newton's method and subsequently used to compute the reaction impulse for the rest of the collision, making it possible to discard the numerical simulation as in the case of continued sticking.

The Algorithm

Summarizing the procedure from the previous paragraphs, the algorithm for computing reaction impulses is now clear. First compute the relative velocity v_{rel} between the two bodies and verify that its normal component v_z is negative. Then numerically integrate differential Equation (2.27), tracking the work E done by the normal component of the reaction impulse using Equation (2.22). When v_z reaches zero, the point of maximum compression has been attained. Multiply E by ϵ to determine the amount of elastic strain energy to be released during restitution and continue the integration until $E = 0.0$. If the relative tangential velocity vanishes at some point during the integration, use Inequality (2.28) to determine whether the frictional force is sufficient to maintain sticking. If it is, employ Equation (2.29) to compute the tangential component of the impulse for the remainder of the collision, otherwise solve Equation (2.30) to compute dp. Algorithm 2.2 illustrates the procedure.

Figure 2.5 shows a projection of the trajectories of the relative velocity v_{rel} onto the plane defined by $v_z = 0.0$ for a particular matrix Λ and a number of initial relative velocities

```
phase = compression;
E = 0.0;
dp_z = 1.0;
while (phase ≠ restitution  or  E > 0.0) {
    if (v_x^2 + v_y^2 = 0.0) {                                   /* stuck */
        k_0 = λ_01 λ_12 − λ_02 λ_11;
        k_1 = λ_10 λ_02 − λ_00 λ_12;
        k_2 = λ_00 λ_11 − λ_01 λ_10;
        if (√(k_0^2 + k_1^2) < μ k_2) {                          /* keep sticking */
            dp_x = k_0/k_2;   dp_y = k_1/k_2;
        }
        else {                                                   /* resume sliding */
            ...
        }
        dv_z = λ_20 dp_x + λ_21 dp_y + λ_22;
        if (phase = compression) h = (√(ε(v_z^2 − 2E dv_z)) − v_z)/dv_z;
        else h = (√(v_z^2 − 2E dv_z) − v_z)/dv_z;
        p = p + h dp;
        break;
    }
    dp_x = −μ v_x/√(v_x^2 + v_y^2);   dp_y = −μ v_y/√(v_x^2 + v_y^2);
    dv = Λ dp;
    E = E + (v_z + h dv_z/2.0)h;
    p = p + h dp;
    v = v + h dv;
    if (phase = compression  and  v_z ≥ 0.0) {
        phase = restitution;
        E = ε ∗ E;
    }
}
```

Algorithm 2.2: *Collision algorithm computing p from v_{rel} and Λ. For an effective implementation, step sizes h have to be suitably chosen, and the direction of the reaction impulse after the resumption of sliding (indicated by dots) has to be computed.*

and different frictional coefficients μ. For most initial conditions, the direction of relative tangential velocity keeps changing continuously throughout the duration of impact. It is also obvious that the flow depends qualitatively on the frictional coefficient μ. While Equation (2.30) has two roots for $\mu = 0.3$ and $\mu = 0.4$, it has four for $\mu = 0.7$, leading to two additional invariant directions. However, in any case there is exactly one outward directed invariant direction.

As the independent variable in the integration, the normal component of the reaction impulse p_z monotonically increases during the integration. However, several authors, including WANG and MASON [52] and BARAFF [4], have pointed to the possibility that applying a positive normal impulse leads to an acceleration of the bodies towards each other instead of preventing interpenetration. In that case, v_z decreases during the collision and the termination criterion will never be reached. The integration computing the reaction impulse continues indefinitely. This counterintuitive behavior is a deficiency of the contact model resulting from the attempt to use Coulomb's law in conjunction with the principles of rigid body mechanics, and it has been speculated by STRONGE [47] that it is important for surface damage due to abrasive wear during impact. While the problem does not seem to be of great interest in rigid body mechanics — MIRTICH and CANNY [32] assert that it has not occurred during any of their simulations — it is quite frequent if articulated bodies are involved. For the purpose of this thesis, the problem has been solved somewhat arbitrarily, but with visually pleasing results, by applying an impulse which zeroes the tangential component of the relative velocity without influencing the normal component before computing the reaction impulse, according to Algorithm 2.2, whenever it occurs.

An additional problem can arise when parts of the articulated body are in static contact with each other or with the ground during a collision. In that case, the impulse which enforces the non-interpenetration constraint at the point of impact can lead to a violation of constraints at other contact points, creating the necessity to apply additional reaction impulses wherever the normal component of the relative velocity becomes negative.

2.4.4 Static Contacts

Due to their non-singular nature, static contacts are more difficult to handle than collisions. They cannot be treated as isolated phenomena on a different time scale with all other events being disregarded, but have to be handled as finite in duration and occurring simultaneously. Moreover, a simulator dealing with articulated bodies subject to kinematic constraints resulting from contacts has to be able to cope with changing modes of contact and with changing topologies as a consequence of the fact that contacts can be established

and broken frequently during the course of a simulation.

As for collisions, the normal component and the tangential component of a contact force are usually related by Coulomb's friction law. Normal forces prevent interpenetration by acting perpendicularly to the contact surfaces while friction forces act tangentially and oppose slipping motion. The friction force is called dynamic friction if the two bodies are slipping at the contact point; otherwise it is called static friction. In contrast to collisions, the normal force for static contacts does no work on the bodies in contact. The algorithms dealing with the problem of computing static contact forces can roughly be grouped into three classes: analytical methods, penalty methods, and impulse-based methods.

Analytical Methods

A variety of methods attempt to model contact forces analytically. The motion constraints imposed by contacts are accounted for by formulating the equations of motion of an unconstrained mechanism subject to unknown contact forces. In particular, using Equation (2.19), if n forces $\mathbf{f}_{j_1}, \ldots, \mathbf{f}_{j_n}$ are applied simultaneously to links $j_1, \ldots, j_n$ of an articulated body, the resulting acceleration $\mathbf{a}_i$ of link i can be computed by linear superposition as

$$\mathbf{a}_i = \sum_{k=1}^{n} \varkappa_{ij_k} \mathbf{f}_{j_k}. \tag{2.31}$$

Solving for the accelerations at the contact points and substituting the result into the constraint equations results in a system of equations which, together with Coulomb's law, can be used to compute the contact forces. Conceptually, the contacts may be regarded as temporary joints, creating closed loops in the articulated body. Brandl, Johanni, and Otter [9] present an algorithm tailored for articulated bodies which uses constraint propagation to effectively extend the articulated body algorithm to handle closed loops, leading to an especially efficient way of generating the system of equations.

While theoretically the most satisfying solution, the analytical approach suffers from a number of problems. Baraff [3] has shown that the problem of computing frictional contact forces for a system of rigid bodies with contacts subject to dynamic friction governed by Coulomb's law amounts to solving a non-convex quadratic program. In addition to the computational difficulty of finding a solution, it is possible that either no valid set of contact forces or several distinct sets leading to different outcomes exist, raising the issue of finding consistent solutions during consecutive time steps. Moreover, Baraff [4] has shown that the problem of deciding whether a unique set of contact forces exists is NP-complete, and that the problem gets even harder if some of the contacts are subject to static friction.

Furthermore, analytic methods assume that the direction of sliding is constant during an integration time step. In the simulations carried out for this thesis, this requirement often turned out to be impractical for articulated bodies as it required extremely small step sizes to avoid substantial error.

Penalty Methods

Penalty methods are an attempt to circumvent the problem of generating and solving the quadratic system of equations by converting the constrained problem into an unconstrained one where deviation from the constraint is penalized. The constraint is not strictly enforced, but only encouraged. Moore and Wilhelms [34] pioneered the method by suggesting insertion of temporary springs at all contact points which act to repel interpenetrating bodies, with the normal component of the contact force linearly dependent on the interpenetration depth. The implementation of this scheme is rather simple compared with the analytical method, making it by far the most commonly used algorithm for handling static contacts.

The main drawback of this approach is that it leads to a set of very stiff equations, requiring extremely large spring constants and thereby small step sizes from the numerical integration routine used to compute relative joint accelerations if deep interpenetration is to be avoided. Moreover, physically and visually unrealistic behavior can result from the fact that time proceeds in steps rather than continuously, leading to contacts not being detected until a finite amount of interpenetration has occurred and therefore being subject to unreasonably large forces which may cause "jumps". Furthermore, it is unclear how to solve the problem of handling parallel static contacts subject to static friction. Altogether, this makes penalty methods a rather unattractive choice for handling static contacts.

Impulse-based Methods

The impulse-based method, first suggested by Hahn [18], models all types of contact through series of impulses between the objects in contact. Contact forces are not computed explicitly, but occur only as time averages of reaction impulses. Mirtich and Canny [31] took up the idea and revised it to identify static contacts by the relative normal velocity being below some threshold and suggested to handle them separately from ordinary collisions. Fully elastic collisions can be employed to make sure that the normal impulse does no work on the bodies in contact and eliminate unrealistic effects such as a block slowly creeping down a ramp in spite of friction. Mirtich [33] extended the method to articulated bodies, circumventing the problem of having to handle multiple contacts simultaneously by reduc-

ing the integration step size to have at most a single contact at any point in time. While there is empirical evidence that the impulse-based method produces physically accurate results in simple cases, it is inappropriate for handling multiple simultaneous or temporally extended contacts efficiently. Generally, the impulse-based approach is a promising alternative to analytical solutions and penalty methods in the case of transient contacts, but it leads to intolerably small step sizes and a large amount of unnecessary computation in situations where static contacts dominate the system dynamics.

Summary

To summarize, none of the three approaches offers a completely satisfactory solution to the problem of handling multiple simultaneous static contacts, a fact which has led GOYAL, PINSON, and SINDEN [17] to substantially deviate from the common approach by suggesting a soft contact solution which attempts to model the micromechanical processes associated with the contact in greater detail. However, this approach produces very stiff equations and is too inefficient for most applications. A solution which is both efficient and satisfactorily realistic has yet to be found.

Due to the inefficiency of the analytical approach and its inability to handle changes in the direction of sliding, the shortcomings of the other approaches, and the need for high simulation speeds due to the necessity to reliably evaluate the performance of a large number of creatures, a variation of the above methods was chosen for this thesis. After every integration step, the inverse inertia matrices Λ_{ij} for every pair of contacts i and j are computed. Then, in an iterative process, forces are applied to the contact with the largest remaining negative relative normal acceleration until it is ensured that no further interpenetration occurs at any contact point. The direction of the tangential component of a contact impulse is governed by Coulomb's law and is free to change during the iterative process. Simulations show comparatively high efficiency, making real time simulation of creatures with up to about twenty degrees of freedom possible while producing reasonably accurate results. However, it has to be noted that occasionally the iterative process of adding contact impulses continues indefinitely as decreasing the absolute acceleration at one contact can increase the absolute contact acceleration at another, and that the handling of contacts subject to static friction is less than optimal. Nonetheless, given the often large number of simultaneous contacts — up to about twenty can occur at a time — and the sometimes rapidly changing directions in the tangent velocity flow space for some of the contacts, the results that can be obtained by the approach developed here have proved quite satisfactory.

2.5 Summary

To summarize, the algorithm for solving the equations of motion discussed in Section 2.3 and the methods for handling collisions and static contacts outlined in Section 2.4 form a basis for a dynamic simulation program for actuated articulated bodies. A numerical integration procedure advances the simulation in time. As collisions introduce discontinuities into the motion of the simulated bodies, predictor-corrector methods are not a wise choice for an integration routine. Instead, for this thesis a fourth order Runge-Kutta algorithm with adaptive step sizes as described in PRESS et al. [37] has been used. Collisions have to be handled between two steps as integrating over collisions would lead to intolerably small step sizes.

In the next chapter, the algorithms outlined above will be used for simulating the motion of three-dimensional legged creatures in a very simple environment. On an SGI Indigo^2 workstation with R4400 processor, simulation speeds about 60% faster then real time could be obtained for single creatures with up to eighteen degrees of freedom.

Chapter 3

Evolution of Virtual Creatures

When dynamic simulation based on general physical models of articulated figures is used to produce animation sequences, physical simulation algorithms ensure that the creatures' motion is realistic and that all physical constraints are met. However, so as to animate a character, its internal degrees of freedom have to be controlled. For that purpose, an articulated body is equipped with a mechanism generating forces to be applied in the free directions of its joints. In the notation of Chapter 2, the force generators, often called actuators or effectors, supply the values of the forces λ_i of Algorithm 2.1.

A control system is a device for controlling the actuators of a creature. For every simulation time step, for every internal joint i with n degrees of freedom, it computes an n-tuple h_i, possibly by making use of a variety of sensors providing information on the current state of the creature and its environment, such as proprioceptive sensors, tactile sensors, kinesthetic sensors, or photosensors, which is used by the actuator of the joint to generate forces

$$\lambda_i = k_i \cdot (h_i - q_i)$$

in the free directions of the joint. As usual, q_i denotes the joint position of joint i, and k_i is a joint specific constant.

A variety of approaches ranging from using simple sinusoidal functions without sensor input to biologically inspired combinations of coordination schemes including reflexes and motor programs, neural networks, and dataflow architectures have been used as control systems for virtual creatures as described in Chapter 1.

If control systems are to be generated in an evolutionary process, it is favorable to have a controller space which is densely populated with useful controllers. Moreover, if the

evolutionary process is not to degenerate into a random search, control systems which are close according to some metric in the controller space have to show similar, but not identical, behavior. While the use of sinusoidal functions fulfills these requirements, it is too restrictive and does not allow for a wide range of behaviors that may be of interest. More powerful control mechanisms such as neural networks and dataflow architectures are well capable of generating these behaviors, but in using one of these approaches most control systems fail to generate useful behavior at all, and there is no simple metric ensuring continuity in controller space. In this thesis, spectral synthesis is suggested as a useful tool for generating control systems which are sufficiently powerful to generate a wide range of motion and at the same time well suited for evolutionary optimization.

This chapter first outlines evolutionary algorithms and the principles they are based upon in Section 3.1 and then introduces the legged, insect-like virtual creatures used in this thesis and the evolutionary operators used to evolve them in Section 3.2. As an original contribution, spectral synthesis as a method for generating evolvable control systems is introduced in Subsection 3.2.2. The chapter concludes with results and a selection of image sequences from computer simulations of creatures which have been evolved for locomotion behavior in Section 3.3.

3.1 Evolutionary Algorithms

Evolutionary algorithms form a class of direct search and optimization algorithms making use of some of the principles which have been identified to underlie organic evolution. They attempt to model the collective learning process of a population of individuals under selective pressure. Though only a crude abstraction of biological reality, evolutionary algorithms have proven to be robust and powerful alternatives to conventional optimization techniques where many of the latter tend to fail, as for example in the case of multimodal, discontinuous, or non-differentiable objective functions or in dynamical optimization. These properties, together with their potential to emphasize the explorative aspect of the search without losing sight of the goal of the optimization, have made evolutionary algorithms a popular choice for the generation of virtual creatures with life-like properties.

This section introduces evolutionary algorithms from a very general point of view. Many details and theoretical results can be found in Bäck [1] who provides a comprehensive overview of the subject upon which the following treatment is based. The particular evolutionary operators used for evolving virtual creatures for locomotion behavior which have been developed as part of the work underlying this thesis are described in Section 3.2.

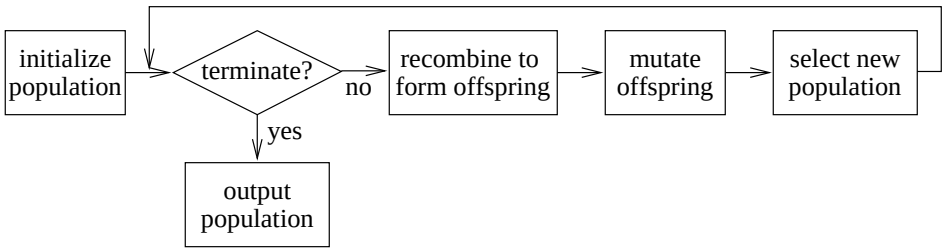

Figure 3.1: *Flow chart of an evolutionary algorithm.*

3.1.1 Evolutionary Search and Optimization

RUDOLPH [42] suggested a classification of optimization techniques distinguishing between volume-oriented and path-oriented methods. While volume-oriented algorithms, typical examples of which are exhaustive search and Monte Carlo methods, emphasize the idea of exploring different regions of the search space, path-oriented methods, such as gradient ascent algorithms, rely on information gathered during the optimization process, making small steps towards successively better solutions. Generally, volume-oriented methods are superior as far as convergence reliability towards a global optimum is concerned as they are not likely to get stuck in local optima, while they are outperformed by path-oriented methods with respect to convergence velocity.

Evolutionary algorithms combine features from both of these classes and can change their characteristics during the course of the search process as a response to endogenous conditions as well as to exogenous parameters. They benefit from a high diversity of the genetic material present in the population, emphasizing the explorative, volume-oriented aspect, while making use of previous solutions when generating offspring, resulting in a path-oriented component. Ideally, several paths are explored in parallel while maintaining the chance to mix information from paths pursued by different subpopulations so as to preserve the possibility of escaping local optima.

Historically, three threads in the field of evolutionary optimization which until recently have developed very much independently from each other can be discerned: genetic algorithms, evolution strategies, and evolutionary programming. They differ in the representation of the object variables and in the evolutionary operators applied to them, each one of them emphasizing different features as being most important to successfully modeling an evolutionary process. More recently, genetic programming has been devised by KOZA [25, 26] as a technique for the evolutionary optimization of computer programs. Several authors, including PETERSON [36] and VENTRELLA [50], add interactive features to the

evolutionary approach so as to be able to incorporate subjective judgement into the search process.

However, all methods share the common idea of interpreting a number of points in the search space as the genotypes of individuals forming a population which is subject to recombination, mutation, and selection so as to evolve it towards successively better regions of the search space. More specifically, at every time step, individuals are selected randomly from the population to form offspring by means of recombination. Recombination results in a mixing of parental information which is subsequently subject to mutation. From the total offspring generated in this manner, a number of individuals is selected on the basis of their phenotypic properties determined by the objective function to form the population of the next generation. By favoring individuals of higher fitness, i.e. those for which the objective function applied to their genotypic description yields lower values if the task is minimization and vice versa, over those with lower fitness, convergence towards the global optimum can be ensured under certain conditions. Figure 3.1 outlines the general procedure. The following subsections introduce generic versions of the genetic operators.

3.1.2 Selection

The selection operator selects a given number of individuals from the total offspring which has been produced to form the population of the next generation. It differs from mutation and recombination in that it is independent of the genetic representation of the individuals, merely making use of phenotypical properties of individuals. In the Darwinian theory of evolution, the fitness of an individual is determined only indirectly by its propensity to survive and reproduce in the particular environment it is living in. Bäck [1] notes that in that sense natural selection is not an active driving force but only manifested in differential growth rates of individuals. In evolutionary algorithms, however, fitness is a direct, well-defined, and evaluable property of individuals, making it an artificial abstraction of the biological struggle for existence which implies survival and reproduction behavior, opposite to biological reality.[1]

Bäck suggested a characterization of selection mechanisms by means of the concept of takeover time. The takeover time is the number of generations after which repeated ap-

[1] Recently, attempts have been made to reverse this causality by letting individuals compete with each other without defining a static fitness function. Reynolds [40], Tu and Terzopoulos [48], and Sims [45] report experiments in which behavior is evolved by direct competition between individuals. In all of these examples, the reproducibility of a virtual organism depends on other evolving organisms and is continuously in flux, leading to a dynamically changing fitness landscape.

plication of the selection operator to a population which is not subject to recombination and mutation yields a population consisting exclusively of copies of the best individual contained in the initial population. The takeover time can serve as a measure for the selective pressure a particular selection operator exerts on a population, with large takeover times reflecting low selective pressure and vice versa. High selective pressure means a high directedness of the search process and leads to a path-oriented search while low selective pressure corresponds to a "soft", volume-oriented search.

Two selection schemes frequently employed are proportional selection and deterministic selection of the best n offspring individuals where n is the fixed population size. The former scheme is the usual selection mechanism for genetic algorithms. Using it, the probability of an individual being selected for survival is proportional to its relative fitness within the population. The scheme has been shown by BÄCK [1] to have large takeover times compared with other selection mechanisms, and therefore to exert relatively low selective pressure on the population. The latter scheme, on the other hand, is the selection mechanism exerting the highest selective pressure. It is the default selection algorithm for evolution strategies.

3.1.3 Mutation

The mutation operator models the small errors that occur when genetic information in biological systems is replicated by recombination. In evolutionary algorithms, it has the important function of introducing innovation into the population. As a general rule, small errors are more likely to occur than large ones, making normally distributed random variables with zero means a natural candidate for the mutation of real-valued parameters and recommending Grey codes for the representation of binary data. It is important to ensure that in the absence of selective pressure successive mutations are neutral on average to avoid numerical drift which would lead to biased outcomes of the evolutionary search.

A general problem related to mutation is that of step size control, where step size is used as a generic term, embracing quantities such as the mutation probability for binary data and the standard deviation of a normal distribution used to mutate real-valued data. If the step size chosen is too small, an unnecessarily large number of iterations is required to make substantial progress, whereas too large a step size continuously destroys information, leading to a degeneration of the evolutionary process into a random search. Generally, according to BÄCK [1], optimal mutation rate and convergence velocity tend to increase as selective pressure increases. It is wrong to say that one selection mechanism is superior to another, but mutation rates have to be carefully tuned to ensure satisfactory performance.

Different algorithms have been proposed for controlling step sizes and adjusting muta-

tion rates to the local structure of the objective function. RECHENBERG [39] analytically computed optimal step sizes for a very simple evolution strategy applied to several simple objective functions. He observed that the probability with which an offspring individual outperforms its parent in the case of optimally chosen step sizes is approximately the same for different objective functions. On the basis of this observation, he derived a simple step size control algorithm which calls for an increased step size if the success rate during the past generations was too high and vice versa. As an alternative, SCHWEFEL [43] suggested self-adaptation of strategy parameters, making the step sizes part of the genetic information of an individual itself and thereby optimizing them in the course of the evolution. Self-adaptation has been shown to successfully adjust step sizes to the local structure of the objective function for several examples. However, care is advised as it increases the dimensionality of the search space.

3.1.4 Recombination

The recombination operator models the process of generating the genetic material of an offspring individual from that of its parent individuals. It can either act on two individuals randomly chosen from the parent population, making the genetic information of the offspring a combination of the information carried by the two, or choose a new pair of parents for every component to be generated. The latter algorithm deviates from biological reality and emphasizes the point of view that the population as a whole forms a gene pool from which new individuals are constructed.

Recombination on the components can be either discrete, randomly choosing genetic information from either of the parents, or — for real-valued parameters — intermediate, forming an arithmetic mean of the parent information. It is worthwhile noting that different recombination operators can be used for object variables and strategy parameters in the case of self-adaptation. Experiments reported by BÄCK [1] have shown that for evolution strategies discrete recombination for object variables and intermediate recombination for strategy parameters yield good results.

3.1.5 Parallelism

Evolutionary algorithms are a natural candidate for parallel implementation. The population can be distributed over a number of processors, with the evaluation of the fitness of an individual, which is often the most time consuming task, and mutation not requiring any communication between different processors. However, the benefits from simply increasing

the population size are limited. The expected gain in convergence velocity and reliability decreases as population size increases, in part due to the fact that many individuals will carry almost identical genetic information. Therefore, efforts have to be made to maintain genetic diversity of the population. BÄCK [1] distinguishes two ways of accomplishing this: the migration model and the diffusion model.

In the migration model, the population is divided into a number of subpopulations which evolve in isolation from each other most of the time, with recombination and selection being local to the subpopulations. Occasionally, however, individuals migrate between different subpopulations. The desired effect is to have different subpopulations evolving towards different locally optimal regions of the search space, effectively avoiding degeneration into a path-oriented search. Obviously, this approach is highly suitable for parallel implementation on a cluster of workstations.

The diffusion model uses a more fine-grained form of parallelism which requires a considerably higher communication bandwidth. It defines a metric on the population by arranging it for example a two- or three-dimensional lattice structure and has the recombination and selection operators applied only to neighboring individuals. In this way, genetic information can propagate, or diffuse, only slowly through the population, effectively favoring subpopulations with widely differing genetic material to claim their niches in different parts of the lattice.

3.2 Virtual Creatures

Virtual creatures are described by two separate aspects: their morphology and their control system. The morphology determines the physical appearance and properties of a creature, the control system its behavior. When there are no stringent needs which completely define the physical appearance of a creature, its morphology can be subject to evolutionary optimization along with its control system. Generally, evolving three-dimensional creatures not only makes the task of physical simulation considerably more difficult than using two-dimensional ones, but also creates higher demands on the control systems due to an increased number of degrees of freedom.

To generate either of the two components by automated evolution, an appropriate genotypic description for creatures has to be found, and recombination and mutation operators have to be given. So as to increase the chances of the evolutionary search to succeed, the creatures have to be evolvable in the sense that small changes to the genotype of a creature result in small changes of the phenotype. Systems in which small changes of parameters

result in either no change at all or in a very large change of the dynamics of the system make evolutionary search an arduous task. Furthermore, it is useful to devise an encoding which leads to a search space of not too high a dimension. Unnecessarily many parameters make the evolutionary search likely to fail while a "constrained embryology" as suggested by DAWKINS [14], in which a few high-level genes control relatively powerful features of the phenotype, greatly increases the chance of success.

3.2.1 Morphology

Creature morphologies that have been explored in the past range from VAN DE PANNE and FIUME's [49] simple two-dimensional creatures to MCKENNA and ZELTZER's [29] 38-degree-of-freedom cockroach, and from VENTRELLA's [50] kinematic stick figures to SIMS's [44] three-dimensional dynamic creatures. Obviously, both the value of three-dimensional, fully dynamic creatures for animation purposes and the difficulties involved in the simulation of their behavior are higher than for two-dimensional or kinematically deforming characters.

The creatures evolved in this thesis are three-dimensional, fully dynamic articulated bodies. Explicit attempts have been made to limit the number of degrees of freedom while preserving the ability to generate creatures of a complexity which makes them useful for animation purposes. More specifically, the creatures' morphology has been restricted to a symmetric, insect-like shape, with each creature consisting of a trunk, modeled as an ellipsoid with a volume constraint so as to prevent the evolution of creatures successful only due to a weight advantage, and two or three pairs of limbs, each consisting of two frustum-shaped segments. All links have an identical and invariant specific mass. The diameters of two segments connected by a joint are required to match at the junction. The limbs are connected to the trunk at fixed positions, but at variable orientations. All joints are rotational with joint limits enforced by spring forces. Actuator forces are proportional to the diameter of the links at the respective joints.

Consequentially, the parameters required for specifying the morphological properties of a creature are a binary variable for the number of limb pairs, and for every link, real-valued variables specifying the dimensions of the link and the position where it is attached to its parent link, and a quaternion determining its relative orientation with respect to its parent link. This morphology was chosen due to the possibility of encoding it in just a few bytes, making it appropriate for evolutionary optimization, its potential to produce rolling motions — a factor which has not yet been explored by other researchers and which was hoped to lead to interesting forms of locomotion — and because it is reminiscent of some animal shapes while having only a small number of degrees of freedom.

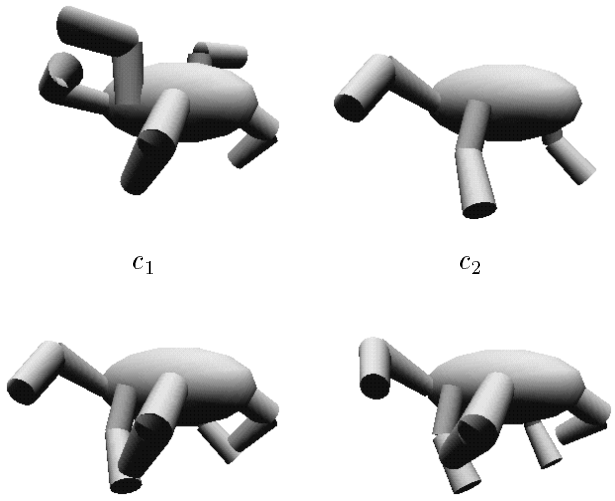

Figure 3.2: *The top row shows two creatures c_1 and c_2, the bottom row on the left a typical recombination of the two, and on the right a mutation thereof.*

So as to evolve creature morphologies, a recombination operator combining the genetic information describing the morphological properties of two parent individuals and a mutation operator making small changes to the result have to be defined. The recombination operator used for the computer experiments to be described below sets the number of limb pairs of the offspring individual to that of either of the parent individuals, uses the arithmetic mean of the dimensions of the parent individuals' trunks to generate the corresponding quantities of the offspring individual, and copies the genotypic descriptions of complete limb pairs chosen randomly from either of the parent individuals to the offspring individual. Subsequently, all real-valued parameters and both the angles and the axes of rotation of the quaternions are mutated by adding small, normally-distributed random displacements with zero mean. Figure 3.2 shows two evolved creatures along with the results of the application of the recombination operator and a typical mutation.

3.2.2 Control Systems

A wide range of control systems has been explored in related research. VENTRELLA [50] used oscillators with no sensor input for animating three-dimensional stick figures. While this scheme is highly evolvable, the output is limited to simple sinusoidal functions. FIUME and VAN DE PANNE [49] employed small linear networks of weighted connections with delays between sensors and actuators and noted that most of the control systems that can be generated fail to produce sustained locomotion. Moreover, they observed that small changes

to the weights of the network often resulted in either no change at all or a very large change in the dynamics of the system. Similar remarks apply to the dataflow computer-like architecture developed by SIMS [44]. MCKENNA and ZELTZER [29] divided the problem of generating suitable input to the actuators into a coordination problem, solved by a gait controller consisting of a number of coupled oscillators for the different limbs which were further synchronized by reflexes, and a control problem, which was solved by carefully handcrafted motor programs gleaned from observations of biological systems and physical intuition. While highly effective, it is unclear how to employ evolutionary optimization to automate the design process. NGO and MARKS's [35] control systems based on stimulus-response pairs promises to be both powerful and evolvable, but has not been shown to be useful for generating motion behavior in three dimensions yet.

The approach taken in this thesis differs from all of the above. Spectral synthesis is introduced as a tool for generating control systems in an attempt to encompass a potentially large range of functions while at the same time providing evolvability by ensuring both that the search space is densely populated with useful controllers and that a small change of the genotype of a creature results in a small change of the generated behavior.

In what follows, the genotypic description of an actuator function consists of N complex numbers H_n, $n = 0, \ldots, N-1$, forming a discrete Fourier spectrum which describes the function in the frequency domain. As actuator functions are required to be real-valued, it has to be ensured that $H_n = H^*_{N-n}$ for $n = 1, \ldots, N/2$. Discrete values $h_k = h(t_k)$, $k = 0, \ldots, N-1$, for the actuator function at times $t_k = k\Delta$ in the time domain can be obtained by means of the discrete Fourier transform

$$h_k = \frac{1}{N} \sum_{n=0}^{N-1} H_n e^{-2\pi i k n / N}.$$

For time values t between the discrete time steps t_k, simple linear interpolation can be used to recover $h(t)$. For times after $N\Delta$ the pattern repeats periodically. Effectively, this scheme can be seen as a generalization of VENTRELLA's [50] control algorithm which corresponds to the special case that there is only a single non-zero frequency component.

As for the morphology, a symmetry requirement has been been imposed on the control system of a creature. The actuator functions for opposing limbs of a creature are either identical or offset by a time interval equal to $N\delta/2$, leading to hopping and alternating gaits, respectively.

Control systems are initialized by randomly generating the set of $N/2$ complex frequency components $H_0, \ldots, H_{N/2-1}$ and a time scale Δ. Good results have been obtained

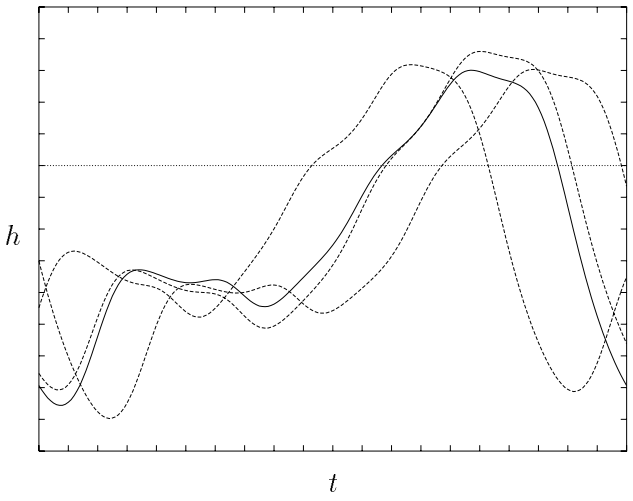

Figure 3.3: *An actuator function $h(t)$ (solid line) and several mutations (broken lines) over time t. The mutated functions have been generated by shifting and scaling time and making small changes to the spectrum of the original function.*

by generating spectra dominated by low frequency components, for example by using a normal distribution with zero mean and a variance which decreases exponentially with the frequency for generating the H_n.

The recombination operator combining two parental Fourier spectra is discrete in that it simply selects the spectrum of the parent whose morphology was chosen to be inherited by the offspring individual. This choice has been made in the belief that the morphological properties of a limb and the control functions determining its behavior form a unit which has been fine tuned in previous generations. Mutation is accomplished by three separate processes. First, the frequency components H_n of a spectrum are modified by adding normally distributed increments with zero mean. Then, a normally distributed time shift with zero mean is performed, and finally, the overall time scale on which actuator forces are computed is mutated by adding an increment to the time resolution Δ which is treated as part of the genetic information of an individual. Figure 3.3 shows different actuator functions generated by an evolved Fourier spectrum and mutations thereof, demonstrating that the mutation operator thus defined has indeed the desired property of producing small errors.

3.3 Results

This section presents a number of three-dimensional virtual creatures which have been evolved for locomotion behavior on a flat ground plane. The objective function used to determine the quality of a creature simply maps the creature to the absolute distance it is capable of traveling within a given time span. For evaluating a creature, it is dropped onto the ground plane and its motion is simulated for a period of 20 seconds of simulated

time. As it usually takes creatures a few seconds to fall into their natural gait, the total horizontal distance between the location of the center of the creature's trunk after 10 and after 20 seconds was used as the fitness measure. In future experiments, it is conceivable to add other fitness measures, for example to minimize the total energy spent or to award for a large average height of a creature's trunk during locomotion.

Some populations were initialized with creatures the morphologies of which were thought to be promising in that they seemed to suggest certain locomotion strategies, others in a completely random fashion. Interestingly, the creatures stemming from populations with totally random initial conditions most often turned out to be more interesting and also more successful than those with "promising" initial conditions. The population sizes of the evolutionary algorithm ranged from 16 to about 64, following advice from BÄCK [1] with six times that number generated as offspring in every generation. In separate runs, both deterministic choice of the best offspring individuals and proportional selection were used as selection mechanisms; neither of the two schemes could be identified as superior to the other. Neither the migration model nor the diffusion model were implemented. Most populations were evolved for about 50 to 100 generations before the most successful creatures evolved in the course of the evolution were inspected.

As a consequence of the lack of a mechanism for preserving genetic diversity of the population, most computer runs quickly converged towards homogeneity, with populations dominated by a large number of very similar and fairly successful creatures. However, separate computer runs most often ended up with completely different locomotion strategies and widely varying creature characteristics. For that reason, and because the goal of the evolutionary search was not just to find the most successful locomotion behavior, but to generate creatures capable of "interesting behavior", a term which is not always properly captured by the rather simple evaluation metric, an interactive evolution overlay was created to provide the animator with the possibility of letting subjective judgement influence the evolutionary process and to mix the genetic information of creatures evolved in separate runs. In particular, populations with different characteristics can be merged and individual creatures which are deemed not to be interesting or which occur in similar form in a great multitude in a population can be removed. Substantial further improvement resulted from these procedures. It can be expected that comparatively good results can be achieved with less need for interactive manipulation by implementing the methods outlined in Section 3.1.5.

A selection of some creatures which have been evolved is shown in Figures 3.4, 3.5, and 3.6, demonstrating that a variety of gaits has been generated. Some of the evolved creatures

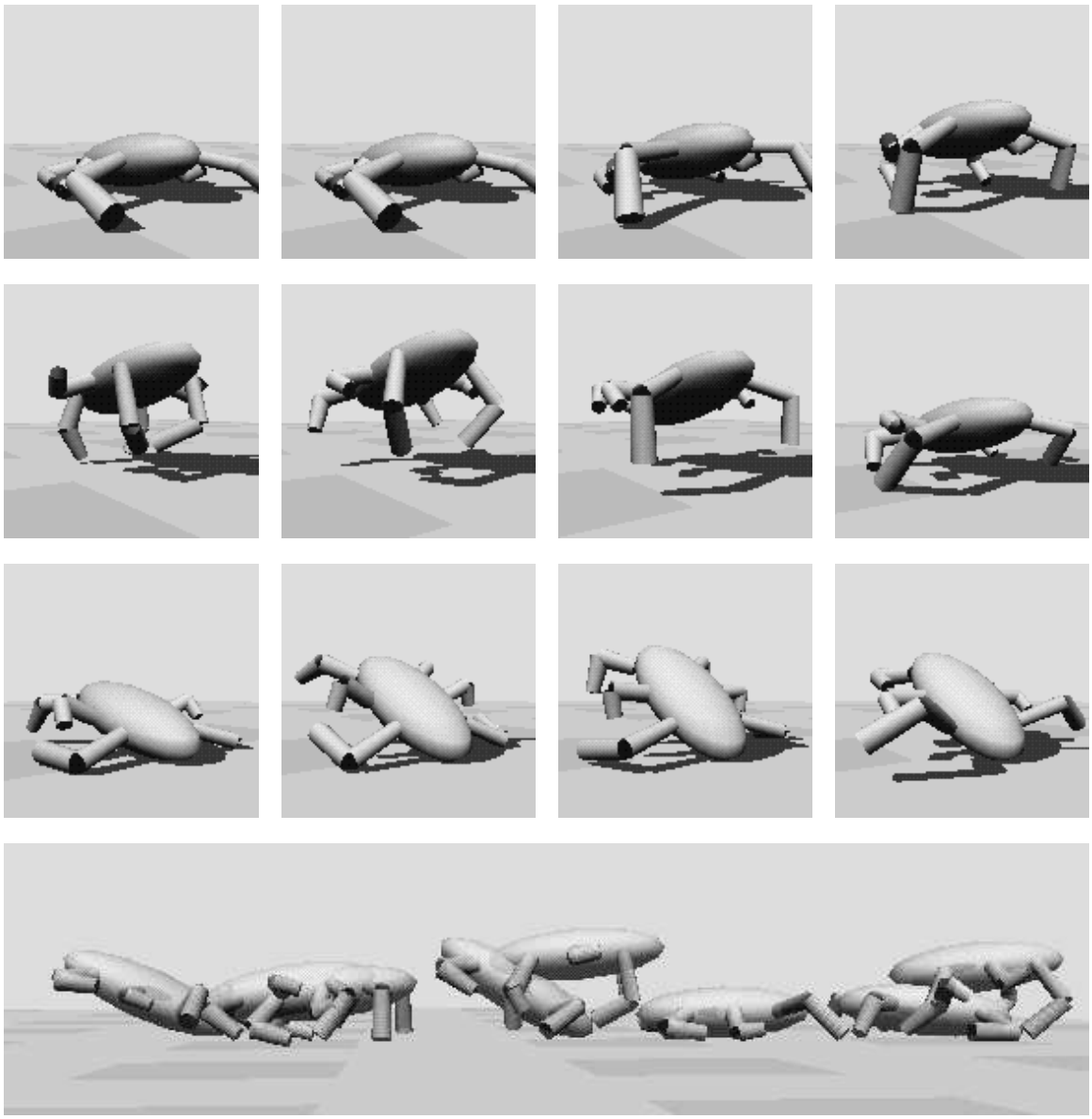

Figure 3.4: *A six-legged creature which uses its rear legs and the trunk for locomotion, with the remaining limbs supporting the motion and ensuring balance.*

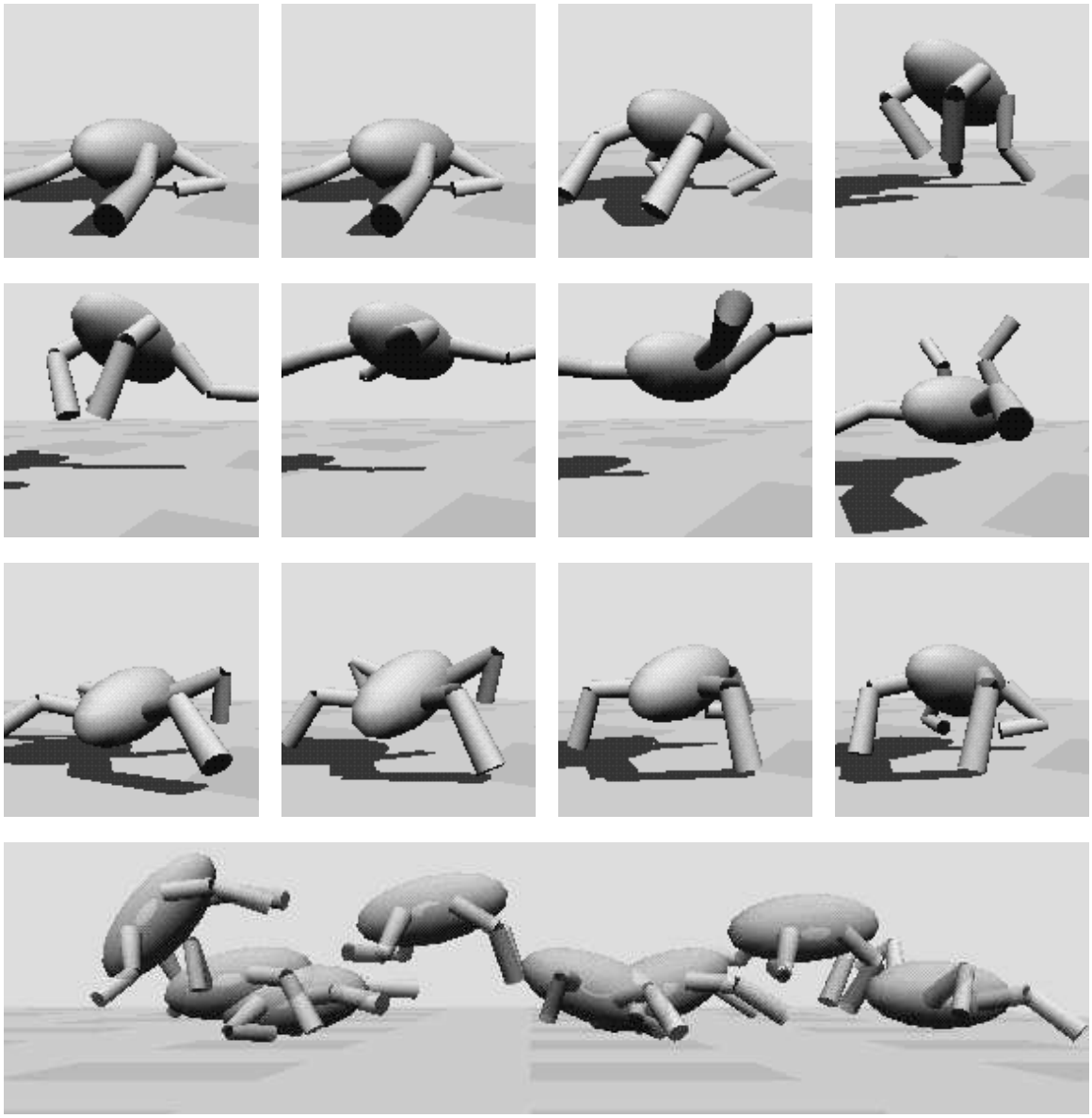

Figure 3.5: *A four-legged creature which propels itself forward by frog-like jumps.*

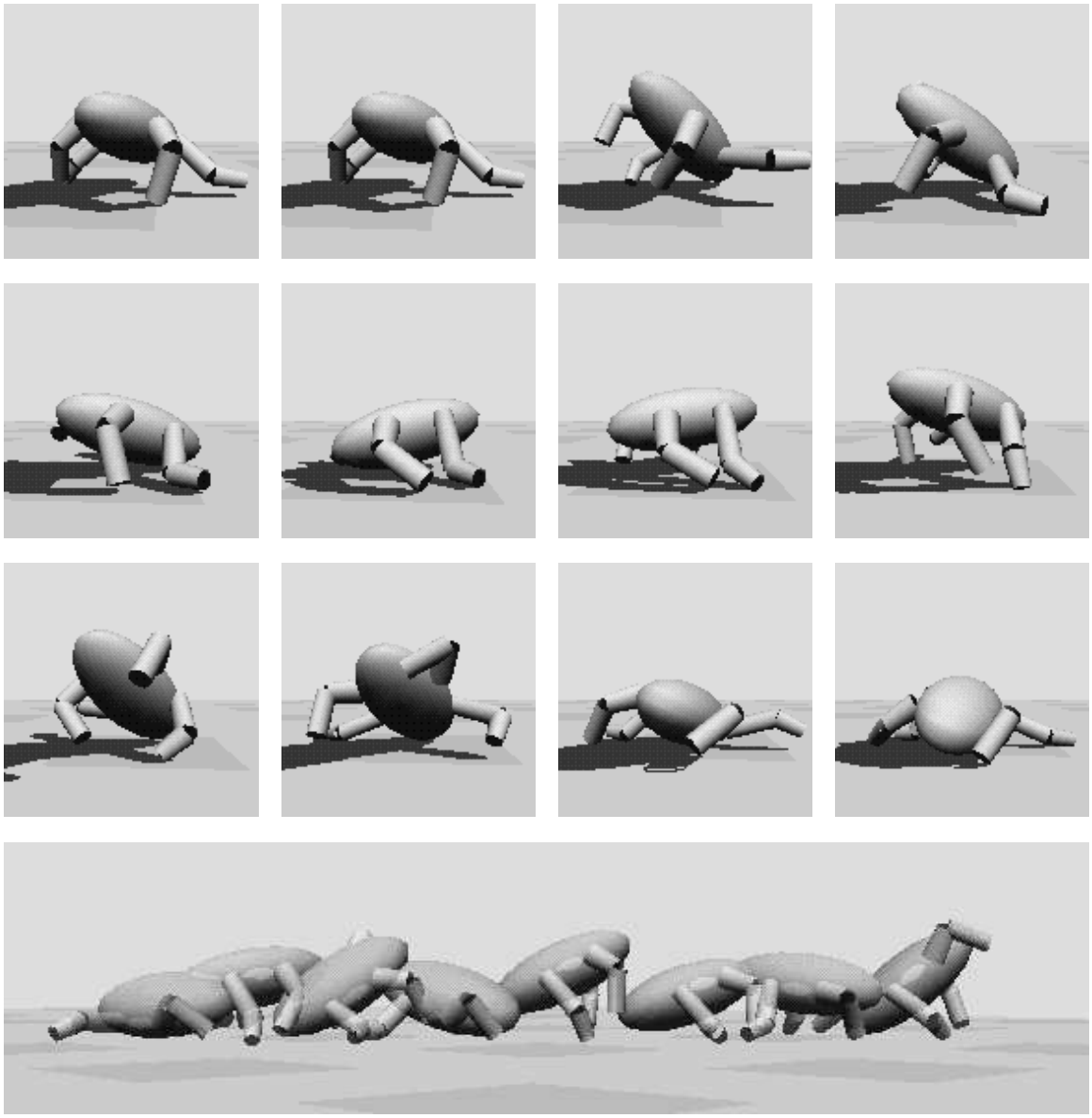

Figure 3.6: *A four-legged creature which exhibits an alternating gait pattern. Its locomotion behavior relies mainly on the front legs and is rather inefficient compared with that of the hopping creatures.*

displayed some qualities reminiscent of familiar animals and exhibit apparently goal-directed behavior. However, as morphologies and compatible motion styles evolve together, many creatures exhibit rather unfamiliar locomotion styles, making use of morphological and environmental properties that cannot be observed in nature. For example, some creatures exploit the fact that there is little friction between an ellipsoid and the perfectly even ground plane by using their trunk in a phase of rolling motion.

While most of the randomly generated creatures quickly ended up on their back with the legs helplessly moving in the air, the most successful ones of the few capable of sustained locomotion traveled about five times the length of their body within ten seconds. The most successful evolved creatures traveled about five times that distance. Given the current restrictions on creature morphologies, four-legged creatures which use their rear legs for hopping and the front legs for supporting the motion are most successful. As a general rule, six-legged creatures are harder to evolve due to their higher number of degrees of freedom, and alternating gates are less stable than hopping ones, emphasizing demand for a stimulus-response system to reinforce stepping patterns.

Chapter 4

Conclusion

This thesis has explored some aspects of the use of fully general dynamic simulation for the generation of physically realistic animations of articulated figures and of the use of evolutionary search algorithms to produce meaningful behaviors. The problems related to the task of physically simulating three-dimensional articulated figures subject to frictional collisions and static contacts have been introduced, and algorithms for their solution have been outlined. In particular, the derivation of the equations describing the effect of a contact force on the acceleration of an articulated figure is an original contribution of this thesis, and the algorithm for handling multiple static contacts developed here is, apart from MIRTICH's [33] impulse-based method, the first of its kind which does not have to resort to penalty spring forces to enforce interpenetration constraints. Moreover, it allows for interactive simulation speeds. Then, some issues accompanying the automatic evolution of useful behavior were introduced. Particular stress was put on the need for evolvable encodings for virtual creatures, and as another contribution of this thesis, spectral synthesis was suggested as a useful tool for generating creature control systems. Subsequently, the ability of this method to lead to useful creature behavior was demonstrated by reporting results from computer experiments in which virtual legged creatures with a comparatively large number of controlled degrees of freedom were evolved for land-based locomotion behavior.

A result of this thesis is the insight that the amount of physical accuracy required to produce realistic behavior grows with the complexity of the morphology of a creature, and that the simple models which have been used to simulate locomotion behavior of stick figures, in statically stable cases, or in two dimensions cannot always be easily generalized to cope with dynamic, three-dimensional motion with multiple concurrent contacts. While systematic errors may not be immediately obvious in single simulations, an evolutionary

algorithm inevitably finds deficiencies in the physical model or its implementation and will ultimately exploit them.

Virtual creatures generated as the result of an automatic evolutionary search process are not likely to replace other approaches to character animation in the near future. While a range of interesting and highly realistic looking motion can be generated without requiring cumbersome user specifications, design efforts, or any knowledge of physics or biomechanics, the amount of control that the animator has on the resulting animation is insufficient for most applications. However, the approach is capable of producing virtual creatures of a complexity that would be hard to devise for a human animator.

Future Work

A number of steps can be taken to improve on and extend this work. First, a physical model able to handle multiple static contacts with static friction in a more exact manner can be expected to further enhance the physical realism of the generated motion. Moreover, it is conceivable that also the quality of the locomotion behavior that can be achieved would be improved as less creatures would have to be discarded simply because the simulation of their behavior went astray.

Second, while the control systems evolved in this thesis are versatile enough to allow for a wide range of forms of locomotion, due to the choice of generating control signals for the actuators without using sensor input there are natural limits to the potential capabilities of a creature. The lack of a stimulus-response system makes it impossible for creatures to develop a sense of balance or to alter their gait pattern as a response to changing environmental conditions or a command by the animator. Creatures would have to be equipped with sensors delivering information on the current state of the creature, on contacts with other objects, or on other external stimuli. Possibly the most promising approach of modeling a stimulus-response system while preserving evolvability is to incorporate the spectral synthesis approach introduced in this thesis into the framework of gait controllers and motor programs as used by McKenna and Zeltzer [29]. In particular, Fourier spectra are a prime candidate for the representation of motor programs which are to be automatically evolved.

Third, incorporating terms awarding energetic efficiency and smoothness of the generated motion into the fitness function may help producing creatures more akin to real animals and reduce the need for interactive evolution. Other objectives, such as maximum average height of the center of a creature's trunk, are conceivable too.

Fourth, the rendering of the virtual creatures could be improved by surrounding them

with a flexible skin and by adding surface details such as fur, eyes, or hair.

Lastly, creature morphologies other than the simple four or six legged shapes used in this thesis are worth exploring. An obvious extension is to allow for different types of joints, such as prismatic or spherical joints in addition to rotational joints. Even more interesting is the possibility of constraining morphologies to ranges of geometric and inertial properties as measured in real animals or human beings. However, more potent control systems and improved strategies for coordinating motions of different parts of a creature are necessary to control the increased number of degrees of freedom needed for a biomechanically sufficiently exact simulation of the human body.

Appendix

Implementation Issues

A software system for the evolution of articulated three-dimensional creatures capable of locomotion has been implemented in C on SGI workstations running IRIX 5.3. Most experiments were run on Indigo^2 computers with R4400 processors. A client-server approach has been used to decouple the task of physical simulation from the evolutionary component of the program and thus to allow for parallel execution. While at any point in time there is a single client process implementing the evolutionary algorithm and a user interface for interactive evolution and displaying features, a multitude of server processes performing physical simulation can run on different machines, with each one simulating its own creature. Communication between the client and the server processes is via sockets.

The client side can run in either batch or interactive mode. In batch mode, it runs as a background process, performing an automatic evolutionary search on a population of creatures. For that purpose, the process keeps track of the available hosts running server processes. Whenever a host is known to be idle, the client process chooses two creatures for recombination, mutates the result, and sends a description of the creature to the server process for evaluation. By periodically polling the connections, the client process can receive evaluations from server processes that have finished their job and time out connections that have run astray. In interactive mode, the client process provides the user interface for interactive evolution shown in Figure A.1. Its features include the ability to interactively load, save, and merge populations, thereby mixing genetic information from independently evolved gene pools, or to delete single creatures.

For displaying the simulated motion, the creatures are rendered using smooth shading, and the shadow they cast on the ground plane is added to give a better idea of their location. The display can be continuous or advanced in single frames both forwards and backwards

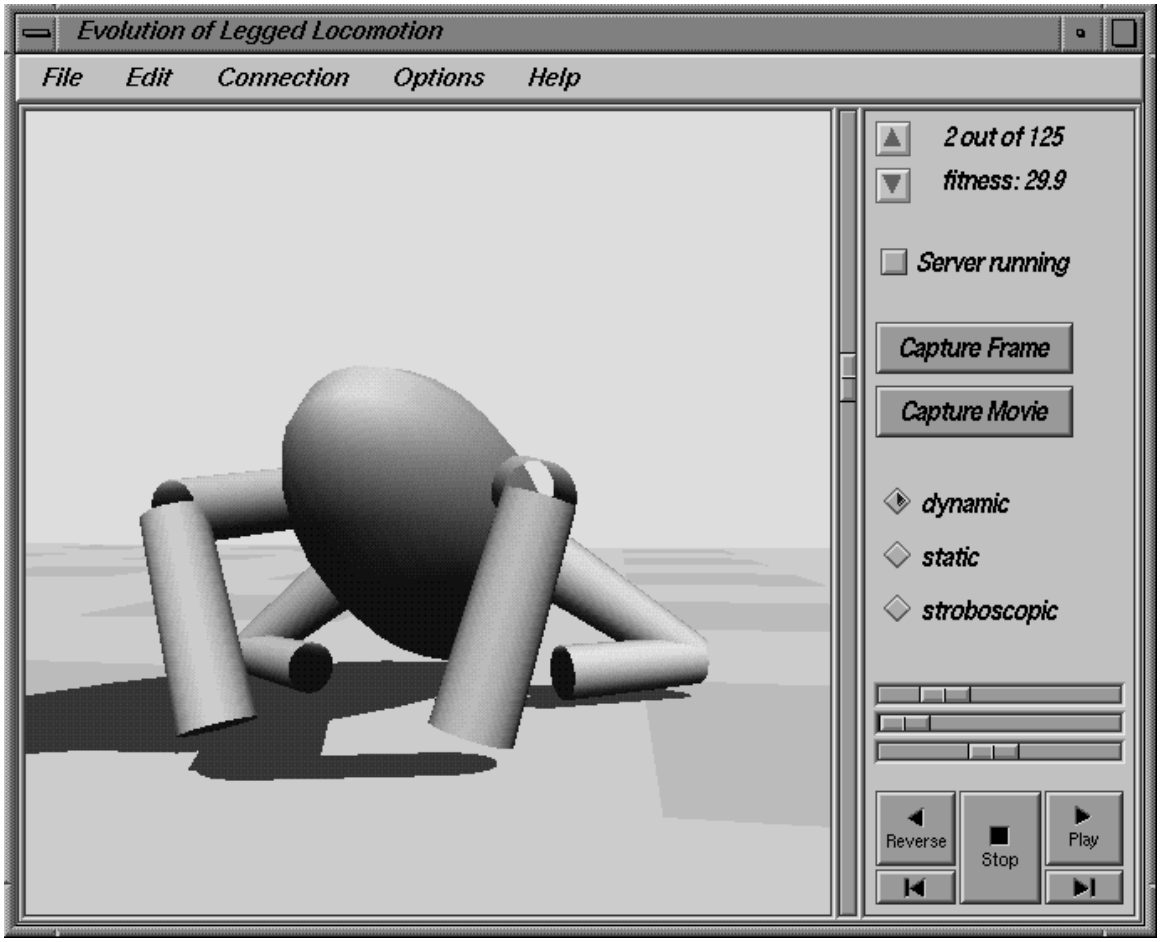

Figure A.1: *A screenshot of the client window in interactive mode.*

in time. The motion can be displayed with a static camera aiming at a particular point on the ground plane, with the camera dynamically following the moving creature, or in a stroboscopic rendering of a creature in various positions along its path. Furthermore, controls for repositioning the camera and a facility for automatically generating a movie file from a series of frames are provided.

The server side handles all of the physical simulation. A server process is designed as an endless loop forking off child processes whenever a connection is requested by a client. A child process receives a description of a creature and an initial state from the client and starts simulating the motion of the creature. If the connection was requested by a client running in batch mode, the server proceeds with the simulation until it either reaches the end of the specified simulation time interval or the connection is timed out by a signal sent

by the client. If the end of the simulation time interval has been reached, the server replies with the absolute distance traveled by the creature during the simulation. If the connection was requested by a client running in interactive mode, the simulation can be started or stopped at any point in time by signals from the client. Whenever sufficient information for computing the state of the creature during the next time frame has been generated, it is sent to the client.

Bibliography

[1] Bäck, T., *Evolutionary Algorithms in Theory and Practice*, (Oxford University Press, New York, 1996).

[2] Badler, N.I., B.A. Barsky, and D. Zeltzer (eds.), *Making Them Move*, (Morgan Kaufmann Publishers, San Mateo, CA, 1991).

[3] Baraff, D., "Coping with Friction for Non-Penetrating Rigid Body Simulation", *Computer Graphics* **24**(4), pp.31-40, (1991).

[4] Baraff, D., *Dynamic Simulation of Non-Penetrating Rigid Bodies*, Ph.D. thesis, Cornell University, Department of Computer Science, (1992).

[5] Barzel, R. and A.H. Barr, "A Modeling System Based On Dynamic Constraints", *Computer Graphics* **22**(4), pp.179-188, (1988).

[6] Bhatt, V. and J. Koechling, "Classifying Dynamic Behavior During Three Dimensional Frictional Rigid Body Impact", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp.2342-2348, (1994).

[7] Bhatt, V. and J. Koechling, "Partitioning the Parameter Space According to Different Behaviors During Three-Dimensional Impacts", *Transactions of the ASME* **62**(3), pp.740-746, (1995).

[8] Brandl, H., R. Johanni and M. Otter, "A Very Efficient Algorithm for the Simulation of Robots and Similar Multibody Systems Without Inversion of the Mass Matrix", *Proceedings of the IFAC/IFIP/IMACS International Symposium on the Theory of Robots*, pp.95-100, (1986).

[9] Brandl, H., R. Johanni, and M. Otter, "An Algorithm for the Simulation of Multibody Systems with Kinematic Loops", *Proceedings of the IFToMM Seventh World Congress on the Theory of Machines and Mechanisms*, pp.407-411, (1987).

[10] Bruderlin, A. and T.W. Calvert, "Goal-Directed, Dynamic Animation of Human Walking", *Computer Graphics* **23**(3), pp.233-242, (1989).

[11] Chang, C.-C. and R.L. Huston, "Computational Methods for Studying Impact in Multibody Systems", *Computers & Structures* **57**(3), pp.421-425, (1995).

[12] Cohen, M.F., "Interactive Spacetime Control for Animation", *Computer Graphics* **26**(2), pp.293-302, (1992).

[13] Craig, J.J., *Introduction to Robotics: Mechanics and Control* (2nd ed.), (Addison Wesley, Reading, MA, 1989).

[14] Dawkins, R., "The Evolution of Evolvability", in C.G. Langton (ed.), *Artificial Life*, pp.201-220, (Addison Wesley, Reading, MA, 1989).

[15] Featherstone, R., "The Calculation of Robot Dynamics Using Articulated-Body Inertias", *Robotics Research* **2**(1), pp.13-30, (1983).

[16] Featherstone, R., *Robot Dynamics Algorithms*, (Kluwer Academic Publishers, Norwell, MA, 1987).

[17] Goyal, S., E.N. Pinson, and F.W. Sinden, "Simulation of Dynamics of Interacting Rigid Bodies Including Friction I: General Problem and Contact Model", *Engineering with Computers* **10**, pp.162-174, (1994).

[18] Hahn, J.K., "Realistic Animation of Rigid Bodies", *Computer Graphics* **22**(4), pp.299-308, (1988).

[19] Hodgins, J.K., W.L.Wooten, D.C. Brogan, and J.F. O'Brien, "Animating Human Athletics", *Computer Graphics* **29**(4), pp.71-78, (1995).

[20] Isaacs, P.M. and M.F.Cohen, "Controlling Dynamic Simulation With Kinematic Constraints, Behavior Functions and Inverse Dynamics", *Computer Graphics* **21**(4), pp.215-224, (1987).

[21] Keller, J.B., "Impact with Friction", *Journal of Applied Mechanics* **53**(1), pp.1-4, (1986).

[22] Kilgard, M.J., "OpenGL and X, Part 1: An Introduction", *The X Journal*, Nov/Dec, (1993).

[23] KILGARD, M.J., "OpenGL and X, Part 2: Using OpenGL with Xlib", *The X Journal*, Jan/Feb (1994).

[24] KILGARD, M.J., "OpenGL and X, Part 3: Integrating OpenGL with Motif", *The X Journal*, Jul/Aug (1994).

[25] KOZA, J.R., *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, (MIT Press, Cambridge, MA, 1992).

[26] KOZA, J.R., *Genetic Programming II: Automatic Discovery of Reusable Programs*, (MIT Press, Cambridge, MA, 1994).

[27] LASSETER, J., "Principles of Traditional Animation Applied to 3D Computer Animation", *Computer Graphics* **21**(4), pp.35-44, (1987).

[28] LILLY, K.W., *Efficient Dynamic Simulation of Robotic Mechanisms*, (Kluwer Academic Publishers, Norwell, MA, 1993).

[29] MCKENNA, M. and D. ZELTZER, "Dynamic Simulation of Autonomous Legged Locomotion", *Computer Graphics* **24**(4), pp.29-38, (1990).

[30] MILLER, G.S.P., "The Motion Dynamics of Snakes and Worms", *Computer Graphics* **22**(4), pp.169-177, (1988).

[31] MIRTICH, B. and J. CANNY, "Impulse-based Dynamic Simulation", in K. GOLDBERG, D. HALPERIN, J.C. LATOMBE and R. WILSON (eds.), *The Algorithmic Foundations of Robotics*, (A.K. Peters, Boston, MA, 1995).

[32] MIRTICH, B. and J. CANNY, "Impulse-Based Simulation of Rigid Bodies", *Proceedings of the 1995 Symposium on Interactive 3D Graphics*, pp.181-188, (1995).

[33] MIRTICH, B., *Impulse-based Dynamic Simulation of Rigid Body Systems*, Ph.D. thesis, University of California at Berkeley, Department of Computer Science, (1996).

[34] MOORE, M. and J. WILHELMS, "Collision Detection and Response for Computer Animation", *Computer Graphics* **22**(4), pp.289-298, (1988).

[35] NGO, T. and J. MARKS, "Spacetime Constraints Revisited", *Computer Graphics* **27**(4), pp.343-350, (1993).

[36] Peterson, P.R., *A Genetic Engineering Approach to Texture Synthesis*, M.Sc. thesis, Simon Fraser University, School of Computing Science, (1997).

[37] Press, W.H., B. Flannery, S. Teukolsky, and W. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing* (2nd ed.), (Cambridge University Press, Camgridge, 1992).

[38] Raibert, M.H. and J.K. Hodgins, "Animation of Dynamic Legged Locomotion", *Computer Graphics* **25**(4), (1991).

[39] Rechenberg, I., *Evolutionsstrategie*, (Frommann, Stuttgart, 1973).

[40] Reynolds, C.W., "Competition, Coevolution and the Game of Tag", in R.A. Brooks and P. Maes (eds.), *Artificial Life IV*, pp.59-69, (MIT Press, Cambridge, MA, 1994).

[41] Roberson, R.E. and R. Schwertassek, *Dynamics of Multibody Systems*, (Springer Verlag, Berlin, 1987).

[42] Rudolph, G., *Globale Optimierung mit parallelen Evolutionsstrategien*, Diplomarbeit, Universität Dortmund, Fachbereich Informatik, (1990).

[43] Schwefel, H.-P., *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*, (Birkhäuser, Basel, 1977).

[44] Sims, K. "Evolving Virtual Creatures", *Computer Graphics* **28**(4), pp.15-22, (1994).

[45] Sims, K., "Evolving 3D Morphology and Behavior by Competition", in R.A. Brooks and P. Maes (eds.), *Artificial Life IV*, pp.28-39, (MIT Press, Cambridge, MA, 1994).

[46] Stronge, W.J., "Rigid Body Collisions With Friction", *Proceedings of the Royal Society London A* **431**, pp.169-181, (1990).

[47] Stronge, W.J., "Swerve During Three-Dimensional Impact of Rough Rigid Bodies", *Journal of Applied Mechanics* **61**(3), pp.605-611, (1994).

[48] Tu, X. and D. Terzopoulos, "Artificial Fishes: Physics, Locomotion, Perception, Behavior", *Computer Graphics* **28**(4), (1994).

[49] VAN DE PANNE, M. and E. FIUME, "Sensor-Actuator Networks", *Computer Graphics* **27**(4), pp.335-342, (1993).

[50] VENTRELLA, J., "Explorations in the Emergence of Morphology and Locomotion Behavior in Animated Characters", in R.A. BROOKS and P. MAES (eds.), *Artificial Life IV*, pp.436-441, (MIT Press, Cambridge, MA, 1994).

[51] WALKER, M.W. and D.E. ORIN, "Efficient Dynamic Computer Simulation of Robotic Mechanisms", *Journal of Dynamic Systems, Measurement, and Control* **104**, pp.205-211, (1982).

[52] WANG, Y. and M.T. MASON, "Two-Dimensional Rigid-Body Collisions With Friction", *Journal of Applied Mechanics* **59**(3), pp.635-642, (1992).

[53] WITKIN, A. and M. KASS, "Spacetime Constraints", *Computer Graphics* **22**(4), pp.159-168, (1988).