Biologically–inspired Control Framework for Insect Animation

SHIHUI GUO

A thesis submitted in partial fulfilment of the requirements of
Bournemouth University for the degree of Doctor of Philosophy

2nd September, 2015

Abstract

Insects are common in our world, such as ants, spiders, cockroaches etc. Virtual representations of them have wide applications in Virtual Reality (VR), video games and films. Compared with the large volume of works in biped animation, the problem of insect animation was less explored. Their small body parts, complex structures and high-speed movements challenge the standard techniques of motion synthesis.

This thesis addressed the aforementioned challenge by presenting a framework to efficiently automate the modelling and authoring of insect locomotion. This framework is inspired by two key observations of real insects: fixed gait pattern and distributed neural system. At the top level, a Triangle Placement Engine (TPE) is modelled based on the double-tripod gait pattern of insects, and determines the location and orientation of insect foot contacts, given various user inputs. At the low level, a Central Pattern Generator (CPG) controller actuates individual joints by mimicking the distributed neural system of insects. A Controller Look-Up Table (CLUT) translates the high-level commands from the TPE into the low-level control parameters of the CPG.

In addition, a novel strategy is introduced to determine when legs start to swing. During high-speed movements, the swing mode is triggered when the Centre of Mass (COM) steps outside the Supporting Triangle. However, this simplified mechanism is not sufficient to produce the gait variations when insects are moving at slow speed. The proposed strategy handles the case of slow speed by considering four independent factors, including the relative distance to the extreme poses, the stance period, the relative distance to the neighbouring legs, the load information etc. This strategy is able to avoid the issues of collisions between legs or over stretching of leg joints, which are produced by conventional methods.

The framework developed in this thesis allows sufficient control and seamlessly fits into the existing pipeline of animation production. With this framework, animators can model the motion of a single insect in an intuitive way by specifying the walking path, terrains, speed etc. The success of this framework proves that the introduction of biological components could synthesise the insect animation in a naturalness and interactive fashion.

List of Acronyms

AEP	Anterior Extreme Position
AI	Artificial Intelligence
BN	Bayesian Network
CMA	Covariance Matrix Adaptation
COM	Centre of Mass
COP	Centre of Pressure
CPG	Central Pattern Generator
CLUT	Controller Look-Up Table
DOF	Degree of Freedom
FD	Forward Dynamics
FK	Forward Kinematics
GPLVM	Gaussian Process Latent Variable Model
ID	Inverse Dynamics
IK	Inverse Kinematics
IPM	Inverted Pendulum Model
ODE	Ordinary Differential Equation
PCA	Principle Component Analysis
PEP	Posterior Extreme Position
PD	Proportional Derivative
ST	Supporting Triangle
TPE	Triangle Placement Engine
ZMP	Zero Momentum Point

Contents

List of Figures

List of Tables

Acknowledgements

Back to the point where starting this journey, I am like a newborn facing an unknown world (still feeling such way sometime). Were it not for the support and contributions from many people, this work cannot be completed.

First and foremost, I would like to thank my supervisors, Dr. Jian Chang and Prof. Jianjun Zhang. Their guidance and support, from research to life, both psychologically and financially, are worth my lifelong gratitudes. It is almost *mission impossible* to find such supervisors as them, who are patient to listen to my many *silly* ideas and insightful to guide me through unforeseen mistakes. This work is the result from our countless meetings, especially with Dr. Chang, whose inspirations and encouragements benefit me throughout this process.

Part of the inspirations leading to this work also arise from discussions with Dr. Richard Southern and Dr. Fangde Liu. Their critical comments are always there when I need them most.

Four years in the building of Tolpuddle Annex 2 is long enough to have some memorable friendships. Thanks to Dr. Xiaosong Yang, Dr. Lihua You, Dr. Zhidong Xiao, Dr. Hongchuan Yu for many interesting lunch breaks. Thanks to Wenxi Li, Mathieu Sanchez, Min Jiang, Shujie Deng, Wenshu Zhang for most leisure time.

This work can not happen if there were no financial supports from both Bournemouth University and Chinese Scholarship Council. This work can not either be completed without the help from Mrs. Jan Lewis, an amazing lady for her memory ability and considerate thoughts.

Finally, I am greatly indebted to my family, including my parents and parents-in-law, my brother, my wife and our son. For their love.

This thesis is dedicated to my father and son.

Declaration

This thesis has been created by myself and has not been submitted in any previous application for any degree. The work in this thesis has been undertaken by myself except where otherwise stated.

Chapter 1

Introduction

Motion synthesis, the process of generating realistic natural motion for a virtual character, is an area of growing research interest, as evidenced by the number of publications in prestigious computer graphics conferences and journals (Figure 1.1). The motion of these virtual characters is as important in determining the perceived quality of an application, such as video games and films, as the model fidelity or rendering quality. Various methods have been proposed for motion synthesis and they can be categorised into two broad groups: *example-based* methods (Pejsa and Pandzic, 2010) and *simulation-based* methods (Geijtenbeek *et al.*, 2011). Example-based methods synthesise motions by utilising existing motion databases; these methods are most widely used in entertainment applications such as video games due to the resultant advantages of naturalness and speed of the synthetic motion (Kovar *et al.*, 2008). Simulation-based methods generate motion by simulating the physical principles of both the virtual character and the environment. By following the rules designed by the controller, the character follows a strictly prescribed trajectory and is able to interact with the environment in a physically plausible fashion.

Compared with the large volume of published works on biped animation (Geijtenbeek *et al.*, 2011; Pejsa and Pandzic, 2010) and quadruped animation (Skrba *et al.*, 2008), to date there has been much less attention given to motion synthesis for *multi-legged insects*. Therefore, this will be the focus of the work presented in this thesis. Small-scale multi-legged insects, such as ants,

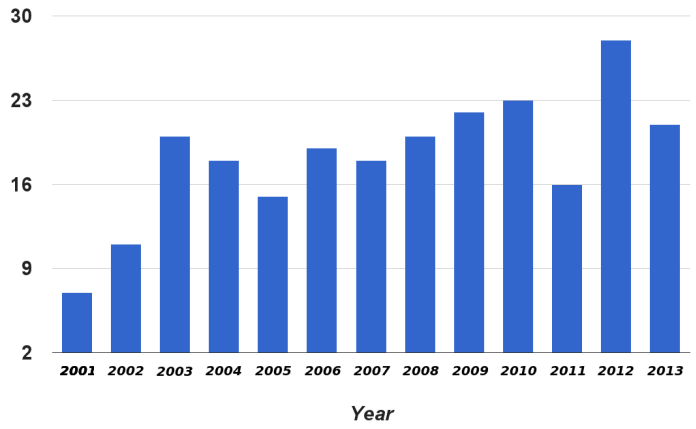


Figure 1.1: *Publication frequency of* motion synthesis *in* SIGGRAPH. *The data are collected from* Google Scholar *using keywords "Character Motion Synthesis" filtered with publications in* SIGGRAPH. *The growing number of publications shows a sharp increase in research interest in character motion synthesis in the past decade.*

crabs, spiders, centipedes and millipedes, are common in the natural world, and they are represented virtually in many applications such as Virtual Reality (VR), video games and films (for example, *The Ant Bully* (2006), *The Mist* (2007), *The Death of an Insect* (2010)).

Existing Challenges: Conventional techniques of motion synthesis, either example-based or simulation-based, face great challenges in animating multi-legged creatures.

The small size and rapid movement of insects make it difficult to capture their motion using a standard motion capture system (Gibson *et al.*, 2005) and their skeletal structure tends to be complex (centipedes typically have more than one hundred skeletal joints), making it challenging to build an accurate dynamic skeletal model (Abdul Karim *et al.*, 2012b; Cenydd and Teahan, 2013). This complex skeletal structure means that tremendous effort is required in order to manually animate the characters by key-framing.

In addition, real insects are able to survive in complex and unpredictable environments, often performing abrupt changes in their movement direction and speed as well as gait pattern in hazardous circumstances. Thus, modelling their delicate and agile movements is a complex task for both animators and

simulation-based methods.

1.1 Motivation

There are over a million known and characterised insects, making them one of the most diverse animal groups. Mobility, an important insect characteristic, is of crucial importance for their survival and reproduction (Chapman, 1998). Some important observations that differentiate the locomotion of insects from that of bipeds will now be described in the following paragraphs.

Fixed locomotion pattern A common gait pattern observed in six-legged insects, such as ants, is the *double-tripod* gait, whereby the left front, right middle and left hind legs form the left tripod (LT) while the other three legs form the right tripod (RT) (Zollikofer, 1994a,b,c). The tripod, formulated by three stance legs, is intuitively named as the *Supporting Triangle*. Research shows that the typical alternating tripod gait prevails when insects walk at moderate to high speed (Zollikofer, 1994c). This gait pattern is believed to be controlled by the Central Pattern Generator (CPG), a network of ganglia in the distributed nervous system which regulates the joint coordination (Zollikofer, 1994c).

Distributed nervous system The locomotion of an insect is controlled by the brain and ganglia (clusters of neurons in the trunk), with the brain determining the onset and velocity of motion and the ganglia coordinating the inter-leg and intra-leg movements (Chapman, 1998). In the field of computer animation, a user interface is said to be friendly if the function of the animator takes the role of the ”brain”, commanding only high-level behaviours without worrying about low-level joint coordination.

Specialised physiological properties Insects have evolved to possess special physiological characteristics, such as having an exoskeleton, having lightweight legs with single-rotational-axis joints connecting the leg segments (Chapman, 1998; Delcomyn, 1999). This allows one to create the virtual

insects with a simplified model, then facilitating controller design and improving computational performance.

These specialised insect features lead to a natural question regarding the design of a natural and stable controller for virtual insects: is it possible to design a suitable controller and replicate the capability and flexibility of real-world insects? Before investigating such a possibility, some research questions, proposed in the next section, must be addressed.

1.2 Research Questions

The work in this thesis aims to *replicate* the capability and flexibility of real-world insect locomotion in a virtual insect, while providing a user-friendly interface for industry practitioners. Such a *replication* should satisfy the following requirements from the perspective of character motion synthesis:

- Naturalness: the naturalness of the synthetic motion is important for determining the perceived quality of a given application, especially for interactive applications. Defects, such as foot-skating or ragdoll effects, would deliver negative impacts on user experiences.
- High level control: users should be able to manipulate the character with an intuitive control interface. The designed controller, instead of the user, should handle the task of computing joint rotations in example-based methods or forces and torques in simulation-based methods.
- Stability: the character should be able to resist against external perturbations within a range of magnitude. This feature is critical in game engines which involve frequent interactions between characters and environments.
- Ease of design: labour intensive processes, such as parameter tuning, should be automated as much as possible in order to facilitate controller design. This automatic process increases the efficiency of controller design, especially in the situation where each task requires a specialised controller.

Although the problem of motor control seems an easy task for real-world insects, it is challenging to replicate their capability in virtual insects. For example, one of the challenges is that the underlying machinery of how real insects move around is not fully understood, so that reasonable assumptions are needed to build a complete model. Although the Central Pattern Generator (CPG) has been widely accepted as the underlying mechanism for the production of the fixed gait pattern, the link between the CPG and rich motor skills is not fully understood (Ijspeert, 2008). Furthermore, different models have been proposed to describe the same structure or phenomenon. For example, the nervous system can be modelled either as a network of oscillators at a functional level, or as a population of spiking neuron cells at a structural level. Each particular case requires a careful selection of models to fit a set of specific purposes.

More specifically, the following questions must be addressed in order to develop a theoretic framework for modelling and describing the locomotion of virtual insects:

- How to model the control mechanism of a real insect given the aforementioned requirements of character motion synthesis? If a similar distributed system is adopted, how to construct the high-level components to allow intuitive control and the low-level components to ensure sufficient stability?
- How to communicate between the high-level user commands and the low-level control patterns? The map between these two spaces is not an easy task since the dimensions of user inputs are much fewer than the number of Degrees of Freedoms of joint rotations.
- How to model the delicate physical structure of a real insect in a biologically plausible and intuitive fashion? In other words, how far can one go in simplifying the dynamic model without sacrificing too much naturalness in the synthetic motion?
- Can this artificial system achieve the same level of performance as real insects in terms of naturalness, stability and adaptation? If not, which factors account for the defects and how should the research be properly

focussed to maximise the improvements?

The following hypotheses are now proposed to address each of these questions. Firstly, nonlinear oscillators with the property of convergence would be ideal to ensure the stability of the controller. Secondly, a precomputed control table arises as a promising trade-off solution for reducing the dimensions between extrinsic behaviour and intrinsic control parameters. Thirdly, the dynamic model needs to be simplified while keeping a selected number of key features.

1.3 Contributions

The work in this thesis presents a comprehensive framework to automate the modelling and authoring of insect locomotion. This work is, to the best of the author's knowledge, the first biologically-plausible framework that meets the demands from computer graphics. The success of this framework demonstrates that the introduction of biological components has the ability to improve the naturalness of synthetic motion as well as the stability of the controller. Furthermore, this framework provides an artist-friendly interface and fits seamlessly into the existing pipeline of animation production. This framework is able to model a wide repertoire of locomotion abilities for insects, including walking on both even and uneven grounds, recovery from perturbations, steering, load carrying, following curve paths and flexible runtime changes between different motion states and speeds etc. The detailed contributions of this work are as follows (Figure 1.2):

1. First, the Triangle Placement Engine (TPE), similar to the *brain* of an insect, is introduced to distribute a series of supporting triangles in a specific task. The TPE is built up by mathematically formulating experimental observations and brings a wide range of settings such as speed, load, path and terrain into consideration.

2. A network of nonlinear oscillators is then proposed in order to model the Central Pattern Generator (CPG) , which serves as a low-level controller to coordinate the angles of the leg joints. The introduction of

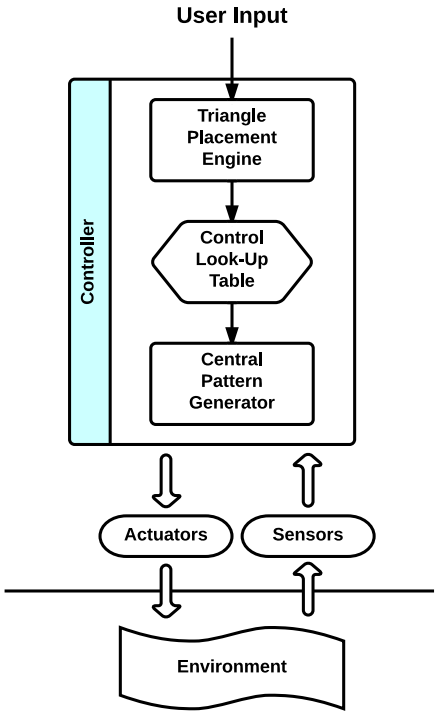


Figure 1.2: *Block diagram of the framework presented in this thesis. The* Triangle Placement Engine *takes various user settings as input and determines the size, location and orientation of the* Supporting Triangle*. The generated triangle profiles will be passed into the animation engine, which handles stance and swing legs separately. The* Controller Look-Up Table *maps the desired triangle profiles into a set of control parameters for the CPG. The actuators apply joint torques, which are converted into the interaction force on the foot of each leg and produce the desired movement. The information from the sensors includes the external force and joint angles etc, and will be passed into the physics engine for forward simulation.*

the *Hopf Bifurcation* comes with two significant merits. The first merit is that characters can start or stop moving (a common observation for insects) by toggling a single bifurcation parameter and the second merit is that the motion is able to converge to the periodic or discrete gait in the case of unexpected perturbations.

3. The Controller Look-Up Table (CLUT) relates the CPG to the triangle profiles by translating the high-level locomotion commands (mainly velocity manipulation) into fixed patterns in the low-level physics-driven controller. The CLUT is modelled by optimising the control parameters in the CPG to simulate the insect dynamics in a single stride under

an exhaustive set of settings.

4. A biologically-meaningful mechanism for insects to trigger the legs from stance to swing is also developed. Different strategies are proposed for insects moving at fast and slow speed. For the fast mode, the switch is triggered as soon as the Centre of Mass (COM) moves outside the Supporting Triangle, while for the slow mode, the switch is determined by a probability framework which takes into consideration various temporal and spatial factors. This mechanism provides an intuitive criterion for evaluating gait stability in general cases without further modification.

During the period of this research, I, together with my supervisors and colleagues, have published the following academic papers to disseminate the work related to this thesis:

- Shihui Guo, Jian Chang, Xiaosong Yang, Wencheng Wang, Jianjun Zhang, *Locomotion Skills for Insects with Sample-based Controller*. Computer Graphics Forum, Volume 33 - Issue 7, pp. 31-40 (Proceedings of Pacific Graphics 2014) (Link).

- Shihui Guo, Richard Southern, Jian Chang, David Greer and Jianjun Zhang, *Adaptive Motion Synthesis: A Survey*. The Visual Computer (2014): 1-16, Springer (Link).

- Shihui Guo, Jian Chang, Yang Cao, Jianjun Zhang, *A novel locomotion synthesis and optimisation framework for insects*, Computers & Graphics, Volume 38, February 2014, Pages 78-85 (Proceedings of CAD/CG 2013) (Link).

- Shihui Guo, Safa Tharib, Jian Chang, Jianjun Zhang, *Biologically-Inspired Motion Pattern Design of Multi-legged Creatures*, Springer Lecture Notes in Computer Science, Volume 7834, 2013, pp 145-156 (Link).

- Fangde Liu, Richard Southern, Shihui Guo, Xiaosong Yang and Jianjun Zhang, *Motion Adaptation With Motor Invariant Theory*, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, Volume: PP, Issue: 99, Page(s): 1-15, November 2012 (Link).

1.4 Structure of Thesis

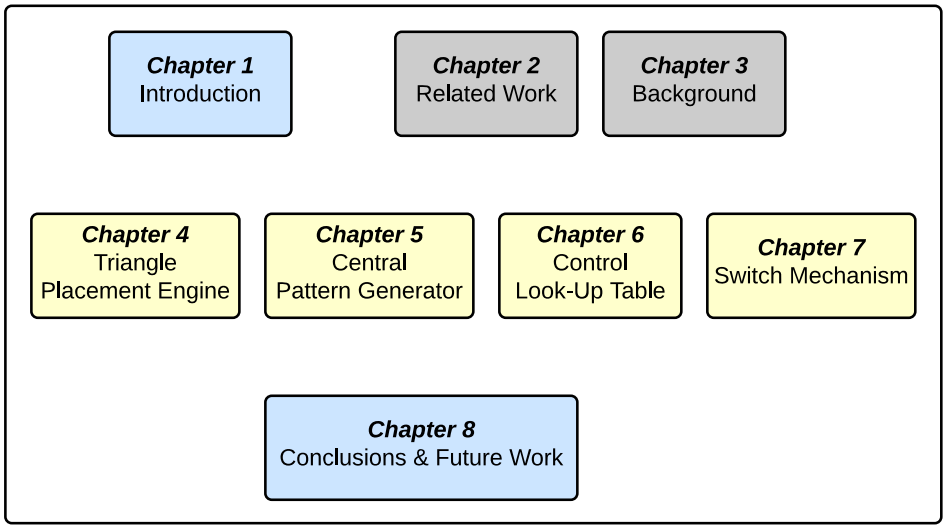


Figure 1.3: *Block diagram of the report structure.*

The thesis chapters give a thorough explanation on individual components formulating the proposed framework (Figure 1.3).

Chapter 2 gives a review of the existing works in modelling and synthesising character locomotion, citing a wide range of research from fields including biology, robotics and computer graphics.

Chapter 3 covers a selection of background knowledge about insect physiology. The inclusion of this chapter ensures that this thesis is self-contained and provides readers with sufficient background knowledge for latter chapters.

Chapter 4 moves to the topic of how animators can specify the locomotion behaviours using the typical gait patterns found in insects. This chapter explains the design of the Triangle Placement Engine (TPE) which takes user inputs and generates the triangle profiles.

Chapter 5 studies the design of the proposed CPG controller, which comprises a network of neuron-oscillators to actuate the stance legs and a procedural controller to animate the swing legs.

Chapter 6 presents the design of the CLUT, which serves as the middle-

ware between the low-level CPG controller and the high-level user commands.

Chapter 7 demonstrates how the controller determines the switch of leg states from stance to swing during both fast and slow movements. The switch mechanism introduced in this thesis is biologically meaningful and is demonstrated to be successful in a wide range of situations.

Chapter 8 concludes this thesis and suggests some future directions of the current work.

Chapter 2

Related Work

Motion synthesis is the process of artificially generating natural motion for a virtual character. The design of a satisfactory controller is challenging, given the demanding requirements, including stability against perturbations, naturalness of the synthetic motion, adaptation to the challenging environment and user intuition control. Existing research in computer animation could be categorised into two groups: example-based and simulation-based. Researchers also explored the hybrid way by combining both example-based and simulation-based methods. The following discussions will give a general introduction to the individual methods, before focussing on insect animation particularly.

The discussions on related work will move on to an emerging solution to build a locomotion controller using biologically-inspired methods. Locomotion is of crucial importance to the survival of insect. Despite their *unsophisticated* neural system compared with vertebrates, insects thrive in a wide range of environments and demonstrate a rich variety of complex behaviours, which outperform the best controller for both robots and virtual characters in most, if not all, aspects. The past two decades saw a surge, first in neuron-biomechanics simulation and then in actual robots, in experimenting with the design of a controller based on the underlying machinery of real insects. This section reviews existing work on biologically-inspired controllers, including robotics actuation and neuron-biomechanics simulation.

2.1 Example-based Methods for Character Motion Synthesis

Example-based methods synthesise motions by using existing databases and are widely used in entertainment applications such as video games due to their advantages of naturalness and speed (Kovar *et al.*, 2008). Existing motions are either resequenced and interpolated to synthesise new motion clips or transformed to other characters. This led to the development of *Motion Graph* (Kovar *et al.*, 2008) and *Motion Retargetting* (Gleicher, 1998) respectively.

In a *Motion Graph*, motion clips are represented as edges and connected by nodes, which are frames of similar character poses. By traversing from one clip to the next via a similar pose, novel motion clips can be synthesised. By building a large motion graph containing various motion performances, this method is able to react to different environments (Lau and Kuffner, 2006; Heck and Gleicher, 2007; Safonova and Hodgins, 2007; Beaudoin *et al.*, 2008; Kovar *et al.*, 2008)

Motion Retargetting adapts motion from a source character to a target one (Gleicher, 1998). This technique can be summarised as Inverse Kinematics (IK) solvers integrated with an optimisation for space-time constraints. Constraints are effectively the interactions between the characters and environments, which users intend to preserve during adaptation. The objective functions are designed to penalise unacceptable types of changes. The configuration of an articulated figure is denoted by $\mathbf{q}$, containing the root position of the hierarchy and the angles of its joints. The original motion and the retargeted motion are referred to as $\mathbf{q}_0(t)$ and $\mathbf{q}_1(t)$ respectively. The motion displacement $\mathbf{d}(t)$ represents the difference between two motions:

$$\mathbf{q}_1(t) = \mathbf{q}_0(t) + \mathbf{d}(t), \tag{2.1}$$

For retargetting, a simple objective is *to minimise the amount of noticeable changes*:

$$g(q) = \int_0^t (\mathbf{q}_1(t) - \mathbf{q}_0(t))^2 = \int_0^t \mathbf{d}(t)^2 \tag{2.2}$$

The above equation minimises the magnitude of motion differences over time.

In this optimisation framework, constraints can be either *equality constraints*, such as point-to-point attachment, or *inequality constraints*, such as joint limits:

$$\begin{aligned} \mathbf{C}(\mathbf{q}, t) &= 0 \\ \mathbf{C}(\mathbf{q}, t) &> 0, \end{aligned} \tag{2.3}$$

Instead of modifying the motion by constraining specific joints, another method is to construct statistical models of high-level behaviours based on a large motion database. This technique alone is not sufficient to adapt motions to either different characters or environments, but it is useful when the user intention is to generate motions with variations (Lau *et al.*, 2009). The most popular decision-making network in adapting character animation in various situations is the Bayesian Network (BN). A Bayesian network is a directed acyclic graph that represents a joint probability distribution over a set of random variables. Each node of the graph represents a random variable while edges represent the dependency relationship between these variables. The BN is widely used to synthesise motion for virtual characters (Rother, 2008; Qinxin, 2007) due to a number of appealing features including:

- The causal relationship between variables is intuitive to understand, given the graphical representation of the BN.
- This graphical representation also reduces memory overhead, especially when the dependencies in the joint distribution are sparse.
- The structure of a BN is based on existing data and can be updated dynamically.
- A BN handles missing observations by calculating marginal probabilities conditional on the observed data using Bayes' theorem.

A representative work (Lau *et al.*, 2009) learns a BN model from a few examples of a particular type of input motion. Here the dynamic BN defines

a joint probability distribution over X[0], ..., X[T]:

$$P(\mathbf{X}[0], \cdots, \mathbf{X}[T]) = \\ P_{G_{prior}}(\mathbf{X}[0], \mathbf{X}[1]) \cdot \prod_{t=0}^{T-2} P_{G_{trans}}(\mathbf{X}[t+2] \mid \mathbf{X}[t], \mathbf{X}[t+1]) \tag{2.4}$$

The prior network G_{prior} represents the joint distribution of the nodes in the first two frames, $\mathbf{X}[0]$ and $\mathbf{X}[1]$. The transition network G_{trans} (Figure 2.1) specifies the transition probability $P(\mathbf{X}[t+2] \mid \mathbf{X}[t], \mathbf{X}[t+1])$ for all t, assuming that a second-order Markov property applies to this network.

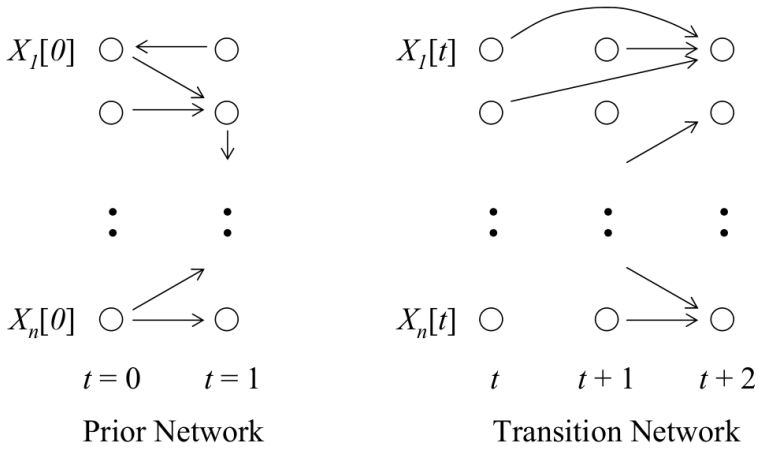


Figure 2.1: *Bayesian Network for variables $X_1, ..., X_n$. The prior network G_{prior} refers to the first two frames, and the transition network $P(\mathbf{X}[t+2]|\mathbf{X}[t], \mathbf{X}[t+1])$ models subsequent frames given the previous two frames, which assumes this BN as a second-order Markov Model. Image from Lau* et al. *(2009) used with permission.*

The most significant disadvantage of a BN-based approach is the fact that there is no universally accepted method for constructing a network from data. In addition, the acyclic property of the standard BN implies that feedback effects cannot be included in the network (Rother, 2008).

Insect Animation with Example-based Method The common disadvantage of example-based methods is the dependency on the original database and thus lack of adaptations to both different characters and environments (Guo *et al.*, 2014b). It is more challenging to apply this method to insects

because their high speed movement and small scale make it difficult to capture their motion data. A specialised system with high speed, synchronised cameras is set up to capture the insect motion so that novel animation sequences can be synthesised and applied to a swarm of insects (Gibson *et al.*, 2005, 2007). This state-of-the-art system is ideal for capturing and synthesising motions of small-scale insects, such as ants and spiders. However it is a non-trivial task to set-up this system and manual labelling is involved in the semi-automatic post-processing of the original footages. Since the motion is captured on a planar surface, additional efforts are needed to apply the synthetic motion to uneven surfaces.

Recently, online motion input from low-cost devices such as *Kinect* has been used to animate non-human characters, including spiders (Seol *et al.*, 2013). This method is composed of two phases: *design* and *puppetry*. First, during the *design* phase, direct feature mapping and motion classifier are trained using target animations for virtual characters and captured motion by performers with the intention to mimic the target animation. Direct feature mapping is used in cases where performers can easily mimic the input target animation, and motion classifier is introduced for more challenging tasks, such as performing the millipede walking. Second, during the *puppetry* phase, the motion features of target characters are computed by blending the results from both *direct feature mapping* and *motion classifier*. Compared with traditional puppetry (or *Motion Retargetting*) methods, this work has advanced the realistic animation of non-humanoid characters in real time. However, the significant differences of topological and geometrical structures between human and insects make this method difficult to achieve independent and accurate control of each Degree of Freedom in the target character. In addition, this work only considers the adaptation to different characters at a particular moment and ignores the adaptation to different environments.

Pros and Cons in a Nutshell Currently, example-based approaches are widely used in industry for their advantages in naturalness and real-time performance (Kovar *et al.*, 2008). Furthermore, the ability to generate a diverse range of motion styles by using statistical methods is of practical use to generate realistic crowd simulation (Lau *et al.*, 2009). However, it is difficult

to adjust the motion once it has been captured and it lacks the flexibility to transform to another character with topology differences (Yamane *et al.*, 2010; Seol *et al.*, 2013). Synthetic motion can interact with the environment with constraints, however not in a physically-plausible style. Due to the non-linearity of the statistical methods, it is difficult to apply accurate control over the character (Lau *et al.*, 2009). In addition to these common limitations, the capture of insect motions constrains the wide applications of example-based methods to animate virtual insects (Gibson *et al.*, 2005).

2.2 Simulation-based Methods for Character Motion Synthesis

In the past few years, there has been a surge in simulation-based methods (Liu and Popović, 2002; Yin *et al.*, 2007; de Lasa *et al.*, 2010; Coros *et al.*, 2011; Muico *et al.*, 2011; Wang *et al.*, 2012). This approach generates character animation by simulating the principles of physics in a virtual environment. Physical simulation is essential in motion synthesis because of the flexible level of control over virtual characters and real-time responses to environmental perturbations. According to Geijtenbeek *et al.* (2011), a simulation framework is composed of three parts: simulator, character and controller. Since the framework in this work is also simulation-based, fundamental knowledge on dynamics simulation is included in the Appendix A to help readers gain an in-depth understanding of this method.

Simulator Design The dynamics of a set of linked rigid bodies can be formulated as:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{h}(\mathbf{q}) = \mathbf{f}_{ext} \tag{2.5}$$

where $\mathbf{q}$, $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$ represent the joint angle poses, velocities, and accelerations respectively. Parameters $\mathbf{M}(\mathbf{q})$, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ and $\mathbf{h}(\mathbf{q})$ are the joint space inertia matrix, centrifugal/Coriolis and gravitational forces respectively. The controller applies the joint torque (or direct force), which is included in the term $\mathbf{f_{ext}}$, to generate the desired acceleration.

Basically, a physics simulator performs the following steps:

1. Collision detection: the simulator first investigates if intersections exist between different object geometries. If yes, a contact force will be applied between these two intersected geometries. Collision detection and the computation of contact force are regarded as the most difficult issues in designing a physics engine (Baraff, 2001).

2. Forward Dynamics (FD): the task of FD is to compute the accelerations of the inter-connected rigid bodies in response to the external forces and internal joint torques. Once the contact forces and other external forces are determined, individual rigid body will be considered as a separate unit.

3. Numerical integration: a numerical integrator updates the positions and velocities of the rigid bodies at $t + \delta t$ with their positions, velocities and accelerations at t. Two critical factors will affect, or even determine, the stability and performance of the numeric integration: one is the choice of integration technique, and the other is the size of time step (Geijtenbeek *et al.*, 2011). Implicit integration, opposite to the explicit one, would normally allow greater time step at the same level of accuracy (Baraff, 2001). Smaller time step will improve the accuracy and stability of the integration, at the increasing cost of computing time.

A standard approach in character animation is to use the readily available physics engine. Popular choices include *Open Dynamics Engine*, *Bullet*, *PhysX* and *Havok*. A performance comparison of these engines is presented by Boeing and Bräunl (2007). A recent work by Giovanni and Yin (2011) deployed a generalised physics-based locomotion control scheme to multiple simulation platforms, and studied how different platforms affect the performance of character locomotion.

Character Modelling The task of character modelling involves two parts: body and actuator modelling.

Body modelling in simulation-based methods mainly refers to designing a hierarchy of rigid bodies, each including geometry and inertial properties, the

connecting joint type (such as hinge joint) and joint properties (such as joint limits) (Geijtenbeek *et al.*, 2011). In most cases, this hierarchy is designed to follow the physiological properties of its equivalent in real world, such as bipeds or quadrupeds. In some cases, a simplified physics model (such as the Inverted Pendulum (Tsai *et al.*, 2010)) is intentionally selected to drive the motion of complex characters. The reason for this choice is:

- The abstraction of complex physical models to simplified ones often makes the same controller adaptable on different characters, even for those with significant topological differences (Liu and Popović, 2002; Coros *et al.*, 2010).
- The simplified models also reduce the computational burden and speed up the performance, which is of vital importance for real-time applications (Tsai *et al.*, 2010).

The design of abstracted models is determined by the *features* which the controller intends to manipulate. Features of character animation describe a set of high-level, physically-related properties of character state, such as trajectories of the Centre of Mass or end-effectors (de Lasa *et al.*, 2010). Specifying the variations in selected features provides an alternative tool to author the desired motion.

Actuator modelling defines how controllers apply forces and torques on the hierarchy of rigid bodies. As pointed out in Geijtenbeek *et al.* (2011), there are three types of actuators:

- *Muscle-based*: this type of actuator mimics the properties of a biological muscle. Muscles are activated by neural signals and produce force by contracting muscle proteins. A classical model is proposed by Hill (1938), who studied the two key relationships: between the contraction force and muscle length as well as between the contraction force and velocity. The non-linearity of these relationships is believed to account for the naturalness, stability and energy-saving of the movement of real animals (Loeb and Ghez, 2000). Wang *et al.* (2012) and Geijtenbeek *et al.* (2013) introduced non-linear muscle model to actuate various types of characters and produced better results than previous

work in terms of visual performance and stability.

- *Servo-based*: before the popularity of the non-linear muscle model, the community of character animation followed the standard approach from the robotics control: servo-based actuator. Servo-based actuators drive the joints by offsetting the differences between the current and desired trajectories, thus tracking the predefined motion. Yin *et al.* (2007) developed a simple control strategy to drive the physically-simulated characters with the Proportional Derivative (PD) servos as the actuator. Compared with the muscle model, this type of actuator enjoys the advantages in its simplicity of implementation, however is criticised for producing stiff movements.

- *Virtual force*: Similar to the abstraction of body model, researchers also introduced abstract actuators (such as *Virtual model control* (VMC) (Pratt *et al.*, 1997)). VMC transforms generalised virtual forces $\mathbf{f}$ into the actuator torques $\vec{\tau}$ (or forces) with a Jacobian matrix. By designing the virtual component, this technique is intuitive to compute the actuator torques required by high-level control tasks, especially in controlling the relative movement of end-effectors. More detailed examples in the field of character animation include balance control (Pratt *et al.*, 2001) and steering (Coros *et al.*, 2011).

Controller Design The representative work in simulation-based techniques is *Simbicon* (Yin *et al.*, 2007). The character is modelled as a typical hierarchy of linked rigid bodies and the actuator is modelled with the Proportional-Derivative (PD) servo to track target angles for individual Degree of Freedom. The whole motion cycle is divided into two or four states and target angles are specified for different states with the help of *Pose Graph*. Priori information from motion capture data is used to set the target value for individual PD servos. The resulting framework is able to respond to external perturbations and simple terrain variations.

The main contribution from *Simbicon* is the introduction of an additional balance controller which is simple and capable of adapting to a changing environment and external perturbations. The target angles in the PD servo

for both torso and swing hip are explicitly specified in the world coordinate, which ensures the character's balance by directly controlling the torso and hip. Assuming τ_{torso} and τ_B are the torques applied on torso and swing hip respectively (Figure 2.2), the torque on the stance hip can be derived from:

$$\tau_A = -\tau_{torso} - \tau_B \tag{2.6}$$

An additional component is a feedback strategy for swing foot placement. Assuming θ_d is the target angle for the PD controller at swing hip, θ_{d0} is the default value, the feedback law is designed as follows:

$$\theta_d = \theta_{d0} + c_d d + c_v v \tag{2.7}$$

where d is the horizontal distance from stance ankle to the COM and v is the velocity of the COM (Figure 2.2), and c_d, c_v are feedback gain parameters. The additional components feed the controller with the position and velocity of the COM, directly and effectively controlling the swing hip, and thus swing foot placement.

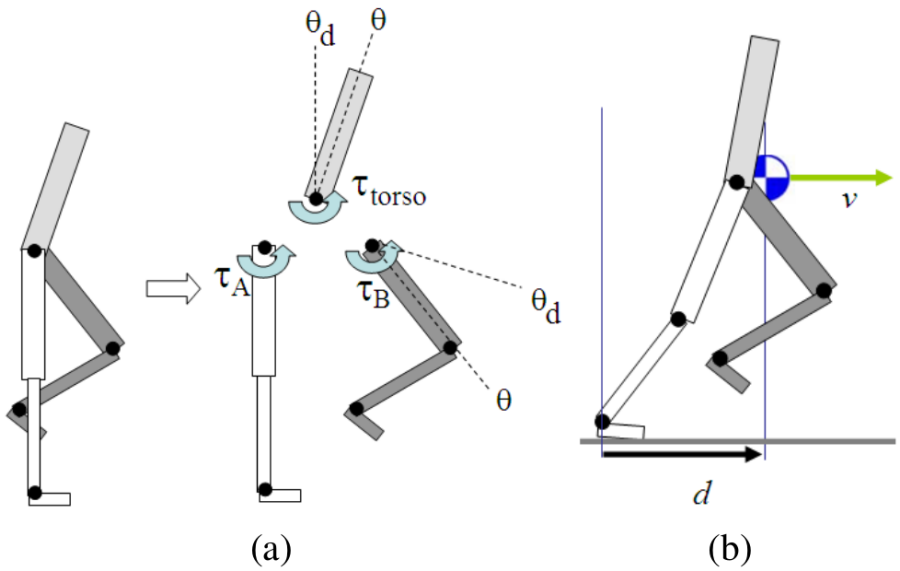


Figure 2.2: *Balance controller in* Simbicon *(Yin* et al.*, 2007). (a) The torso and swing hip are directly controlled by the PD servos with target angles with respect to the world coordinates. (b) An additional balance feedback strategy is introduced to control swing foot placement.*

Inspired by *Simbicon*, the following research pushes the adaptation boundary by employing various improvements. Changes include the introduction of four additional walking states in the Finite State Machine (FSM) to take the preparation of lifting and striking into consideration, thereby improving the naturalness (Wang *et al.*, 2012). Rather than manually setting the objective value for the PD servo, motion priori informs the controller design via the optimisation-based scheme. The implementation of *virtual model control* helps adapt to non-locomotive tasks, which can be used to adapt to simultaneous high-level control (Coros *et al.*, 2011).

A recent development replaces the PD servo with biologically-inspired muscle units to actuate joints (Wang *et al.*, 2012), improving the naturalness and stability. This implementation also provides a comparison against the performance driven by real muscles. While existing work only controls the lower body joints with these muscle units, the simulation of full body controlled with biologically-inspired principles is expected.

Insect Animation with Simulation-based Method Interestingly, there is few work on animating virtual insects with a pure simulation method. The reasons accounting for this scarcity may be that the complex physical structures of insects and their less important role compared with bipeds push researchers to find solutions to avoid building a sophisticated simulation framework. Therefore, a popular choice is the hybrid method, which combines example-based and simulation-based methods. This hybrid formulation produces the synthetic motion in a natural and fast way while maintaining the interaction with users and environments. How to animate virtual insects with hybrid methods will be discussed in later sections.

Abdul Karim *et al.* (2012a) proposed a procedural framework to animate multi-legged creatures in a dynamic environment in real time without using any motion data. The framework included:

- a *Character Controller* to manage the overall gait
- a *Gait/Tempo Manager* to regulate the temporal coordination of the legs
- a *3D Path Constructor* to generate the path in a 3D environment

- a *Footprints Planner* to determine the foot placement to adapt to different terrains

Collectively these four components adapt to different characters and environments. The framework is able to speed up by simulating the details at different levels, which is critical to simulate a swarm of insects. The main disadvantage of this framework is the necessity to pre-define the rules due to the lack of fully-feathered simulation.

Pros and Cons in a Nutshell Simulation-based methods are able to adapt to perturbations and terrains automatically. However a *pure* simulation-based method often produces rigid animation (Yin *et al.*, 2007), which explains the main differences between character motion synthesis in computer graphics and computer simulation in robotics. *Pure* simulation-based methods are also difficult to adapt to characters with significantly different geometries or topologies. Abstract physical models (such as the *Inverted Pendulum*) and abstract actuators (such as the *Virtual Model Controller*) have proved useful (Coros *et al.*, 2011). As there is no universal controller that could produce every motion, new controllers are needed in order to extend the skill repertoire (Muico *et al.*, 2011; Liu *et al.*, 2012). Animating virtual insects in a physically-plausible style is attractive but challenging due to the high dimensions resulting from their complex physical structures.

2.3 Hybrid Methods for Character Motion Synthesis

Researchers also explored the possibility of combining both example-based and simulation-based methods. This combination not only possesses the inherent feature of responsive adaptation to external environments from the simulation side, but also adapts to a broader range of situations by using constraints to produce additional high-level stylised motion, or using existing motion to inform the complex design of a simulation-based controller. With minor editing of the original motion data, this hybrid approach could

improve the naturalness of the synthesised motion, which has been one of the drawbacks of simulation-based methods. The specific method for this combination varies case by case, but in general there are three typical solutions which are discussed below.

Tracking the Reference Motion Trajectory tracking is a standard technique in robotics control. The success of *Simbicon* relies on specifying a set of key poses to provide input to the PD servos, however the scarcity of the key-poses requires the high frequency of simulation steps (Yin *et al.*, 2007). There are two significant advantages of using a large motion library instead of key-poses (Tsai *et al.*, 2010):

- allowing low frequency tracking with increasing number of sample points (as key-poses in Simbicon)
- enabling quick design of tracking controller in an automated method

Reference motion and online simulation may often stay out of phase, especially with different characters or environments. To adapt to these situations and maintain synchronisation, the controller needs to detect foot-strike events in physics simulation and compare against the reference trajectory. A solution is proposed from Lee *et al.* (2010): when the foot strike happens earlier in the simulation process than the reference trajectory $\mathbf{M}$, the rest of $\mathbf{M}$ is dumped and the current pose is smoothly blended towards the next reference trajectory; when the foot-strike happens later in the simulation, the current reference trajectory is extended by applying constant velocities at the end of $\mathbf{M}$.

By combining the simulation and the reference trajectory, the character is able to perform natural movements and recover from small perturbations. However the motion generated in this way stays close to the original motion, otherwise artefacts can be easily spotted. As the introduction of the reference trajectory requires a strict topological similarity between the animated character and the original actor performing the reference motion, these methods are difficult to adapt to a different topology. Recently, non-humanoid characters have been animated with human motion data (Yamane *et al.*, 2010; Seol *et al.*, 2013). This can be further combined with motion tracking to drive the

character in a simulation-based way (Tsai *et al.*, 2010).

Dynamic Selection of Motion Examples Trajectory tracking combines the motion example and simulation process in the same time window, while the second hybrid formulation switches between these two at different time windows (Zordan *et al.*, 2005, 2007). During undisturbed motion, the character will perform the example motion in a kinematic fashion, which is straightforward and ensures the naturalness of the motion. When perturbation or control is introduced, the simulation component will be activated and the character is able to react interactively.

As developed by Zordan *et al.* (2005), two key components are included in this system: a search engine to find the next motion clip after the simulation and a controller to drive the character state towards the target pose. Supervised learning was introduced in Zordan *et al.* (2005) to train the search engine to make an online prediction of the next motion. Blending can be performed to generate the transition clip, which can be tracked with the techniques described in the previous section.

This technique serves as an intelligent decision maker in emergency situations, and is able to adapt interactively to environmental perturbations. Since the dynamic selection of motion examples does not alter the existing motion data, this technique alone can not adapt motion to other situations, such as characters with different topologies. This high level adaptation could be further enhanced with additional constraints inside each motion clip, which may expand its adaptivity scope.

Physics Simulation with Statistics Models The third method is to combine the physics simulation with a statistical model. The former gives the user control over the virtual character, while the latter serves as the prediction mechanism and improves the naturalness of the synthetic motion. In the work of Wei *et al.* (2011), a non-linear probabilistic *force field function* is learned from pre-recorded motion data by the Gaussian Process and is combined with physical constraints in a probabilistic framework.

First, a *force field* prior is extracted from the motion capture data with

Newtonian dynamics:

$$\mathbf{u} = M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}}) + h(\mathbf{q}) \tag{2.8}$$

where the parameters notation follow the Equation 2.5.

The Gaussian process model learns this force field from a training database:

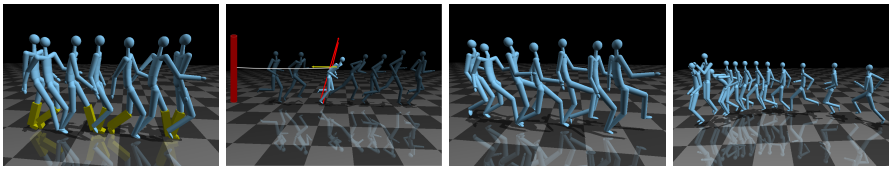
$$pr(\mathbf{u}|\mathbf{q}, \dot{\mathbf{q}}) = \Psi(\mu(\mathbf{q}, \dot{\mathbf{q}}), \sum(\mathbf{q}, \dot{\mathbf{q}})) \tag{2.9}$$

where both the mean and the covariance matrix are functions of kinematic states $[\mathbf{q}, \dot{\mathbf{q}}]$.

According to *Bayes's rule*, the probabilistic motion model $pr(\mathbf{x})$ is broken down into the following three terms:

$$pr(\mathbf{x}) = \underbrace{pr(\mathbf{q}, \dot{\mathbf{q}})}_{pr_{init}} \cdot \prod_t \underbrace{pr(\mathbf{u}^t|\mathbf{q}^t, \dot{\mathbf{q}}^t)}_{pr_{forcefield}} \cdot \underbrace{pr(\mathbf{q}^{t+1}, \dot{\mathbf{q}}^{t+1}|\mathbf{q}^t, \dot{\mathbf{q}}^t, \mathbf{u}^t)}_{pr_{physics}} \tag{2.10}$$

As shown in this chain-like equation, physical and statistical motion models are complementary. Statistical models ensure the naturalness of motion, while the physical components can equip the characters with the capability to react to external circumstances. Compared with previous example-based or simulation-based approaches, this hybrid formulation achieves wider adaptation in a single framework, including performing stylised walking conventionally generated with example-based methods (Figure 2.3a) and adapting to the external force originally from a simulation side (Figure 2.3b). More results are displayed in Figure 2.3.



(a) **(b)** **(c)** **(d)**

Figure 2.3: *Results demonstrated by the method in Wei* et al. *(2011): (a) walking with a heavy shoe; (b) resistance running; (c) stylised walking; (d) running→walking→jumping. Images from Wei* et al. *(2011) used with permission.*

Insect Animation with Hybrid Methods A popular method to animate virtual insects is to drive the character in a kinematic way for common cases and in a dynamic way for unexpected perturbations. In films or video games, insects are normally small-scale and act as the background characters, in which case the synthetic motion from example-based methods is sufficient. To improve the visual credibility, dynamics (often in a simplified version) are introduced to simulate the responses to unexpected situations.

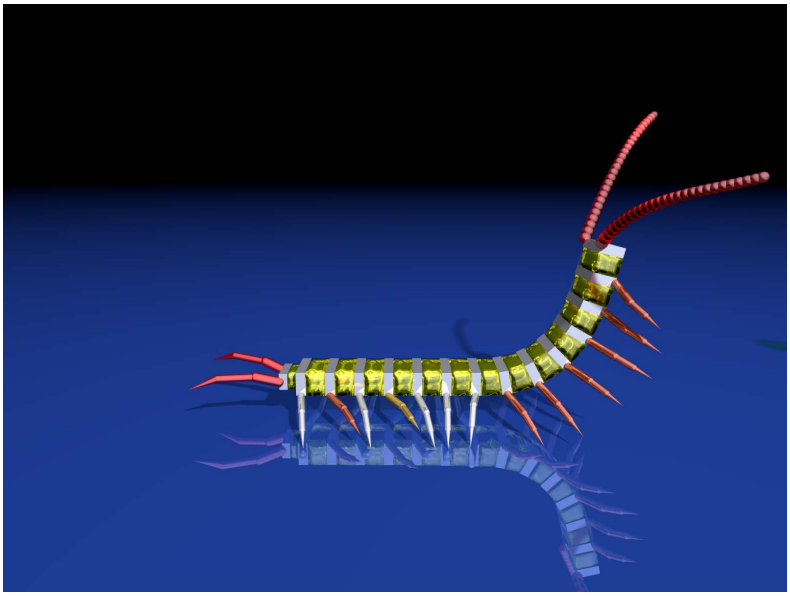


Figure 2.4: *Myriapod body structure in Fang* et al. *(2013). Their method includes dynamics simulation for rigid and deformable segments, and a procedural controller for legs. This hybrid formulation allows the natural synthesis of the motion, with realistic body deformation for the myriapoda's specialised body structure.*

This hybrid approach has been adopted by Fang *et al.* (2013) to simulate the locomotion of myriapoda by incorporating both kinematics and dynamics at different body parts. Firstly, a physical model was constructed with massless kinematic legs, rigid dynamic exoskeleton segments and passively dynamic elastically deformable segments. The deformable segments are used to connect the exoskeleton segments to simulate the soft body segments in myriapoda. The physics simulation controls the rigid parts of the main body and passively actuates the connecting segments. The kinematics computation is performed on the leg movement. This wave-like gait for individual legs is generated from a six-states machine with pre-defined target angles in a

kinematic fashion. A decentralised and distributed leg control system, similar to the *Central Pattern Generator* proposed in this thesis, was introduced to perform the natural wave-like ambulatory gait for myriapoda. However their controller may suffer from insufficient stability due to the lack of feedback control against perturbations, thus not able to recover from pushing.

Recently, Cenydd and Teahan (2013) dynamically simulated insect locomotion in an arbitrary environment. They found that physical simulation did not make any differences in environments which mainly consisted of flat terrain with small inclines. Supported by this argument, they approached the problem of insect locomotion with a hybrid framework combining both kinematic and dynamic control. For example, the gravity in their method is only activated when the character loses its balance.

Pros and Cons in a Nutshell Hybrid methods demonstrate their potential in tackling a wider range of problems than individual approaches, but still suffer from the inherited drawbacks from individual components (Wei *et al.*, 2011; Tsai *et al.*, 2010). For example, a statistical framework always suffers from non-determinism (Wei *et al.*, 2011; Seol *et al.*, 2013). In addition, there is still no satisfactory mechanism to design a physically-valid controller that provides sufficient naturalness. In the situation of insect animation, the solution of activating the simulation component when needed and deactivating when not proves effective (Abdul Karim *et al.*, 2012b; Cenydd and Teahan, 2013), but is far from perfect. A fully developed simulation framework is still worth exploring in order to take full advantage of physics simulation.

2.4 Bio-inspired Locomotion Controller

The underlying principle of character animation could find its similarities with actuating electronic servos to drive a robot and innervating the muscles to move an animal by itself. Such inspiration leads researchers to explore the possibility of designing a locomotion controller based on studies from computational neural science.

This section explains the related work in two main fields: *bio-robotics* and *neuron-biomechanics simulation*. It is worth noting that although these two fields, together with character animation, share fundamental knowledge, the requirements or goals are different for each specific field. For example, a controller to actuate a robot would always prioritise stability over other factors, however the naturalness of the synthetic motion and intuitive user control are often the major foci from the point of computer graphics applications.

Bio-robotics The field of bio-robotics is a fusion between the biology and robotics. It mimics the neural networks of real animals and applies the similar control strategy to drive a robot. The Central Pattern Generator (CPG) is believed to be the underlying mechanism for animal locomotion and has been proved by numerous researchers (Mellen *et al.*, 1995). To date, biologically-inspired controllers have been successfully applied to drive animal-like robots in the situations such as walking (Liu *et al.*, 2011), swimming and crawling fish (Crespi *et al.*, 2008), flying bird (Chung and Dorothy, 2010) etc. For a review on applications of the CPG for locomotion control in animals and robots see Ijspeert (2008).

Traditional control strategies, such as Zero Momentum Point (ZMP) (Vukobratović and Borovac, 2004), have been popular in the robotics community, producing impressive results, such as **ASIMO** (Hirai *et al.*, 1998). ZMP ensures the stability of the robot by constraining the Centre of Pressure (COP) within the stance foot region, which remains in flat contact with the ground at all times. The other body joints are actuated by stiff tracking pre-defined joint trajectories. However, this stability comes at the price of energy inefficiency, heavier computation than its biological counterparts and stiff motion performance (Collins *et al.*, 2005). In computer animation, ZMP was introduced later to control virtual characters as well (Wu and Popović, 2010).

In contrast, passive dynamic walking models (McGeer, 1990) are very energy efficient and require little or no computation to synthesise a simple downhill bipedal gait. However, the application of these approaches is limited as they generally suffer from poor state stability. Liu *et al.* (2012) introduced both global and local controllers to improve structural and state stability re-

spectively. Global control was achieved by coupling the dynamic system with a neural oscillator, which preserves the periodic structure of the motion primitive and ensures stability by entrainment. A group action derived from Lie group symmetry was introduced as a local control which transforms the underlying state space while preserving certain motor invariants.

Robotic researchers have proposed various controllers for multi-legged robots, especially hexapod (Pratt *et al.*, 1997; Saranli *et al.*, 2001). Hexapod, as a special category of robots, attracts much interest from researchers because of its stability (Altendorfer *et al.*, 2001). However, these robot-based controllers are designed to maximise the stability performance and are not entirely suitable for the computer graphics community.

A representative work on actuating a hexapod robot is *RHex* (Altendorfer *et al.*, 2001). *RHex* was able to outperform its predecessors in performing challenging tasks including walking at various speeds and traversing different height variations. The robot was designed to perform an alternating tripod gait by actuating the joints at each leg with periodic desired trajectories, enforced by the PD servos. However, there were only six motors, one for each leg, which significantly constrained the flexibility of the robot's movements, therefore special care was required in the turning of *RHex*. For example, during forward walking, the robot turns by adding differential perturbations to the controller parameters for contra-lateral legs. Nevertheless, this strategy may not be the best choice as it involves the manual tuning of the parameters. Although the control mechanism of *RHex* may provide further information to design a simplified version of character controller, the synthetic motion will be more likely to look rigid. Compared with the inherent stability of the limit cycle in the proposed Central Pattern Generator (CPG), the open loop control used in *RHex* also suffers from instability.

neuron-biomechanical simulation Computer simulation is a popular approach adopted by biologists to study the underlying mechanism of real organisms. Taga *et al.* (1991) first introduced this concept in the design of a neuron-musculo-skeletal system to execute the task of stepping in an unpredictable environment. One of the biggest challenges in designing such a

system is to set the large numbers of parameters by hand.

Due to its simplicity, the insect, particularly the hexapod, has been chosen by many researchers to build a sophisticated neuron-musculo-skeletal simulation framework. Ghigliazza and Holmes (2005) proposed a hybrid dynamic system incorporating rudimentary motor-neuron activation and agonist/antagonist Hill-type muscle pairs that drive a point mass body along a straight line. Later developments introduced more advanced activation rules and demonstrated stable gait in the events of perturbations (Kukillaya and Holmes, 2008, 2009). This series of work (Ghigliazza and Holmes, 2005; Kukillaya and Holmes, 2008, 2009) focused on the analysis of gait stability and made extreme simplification by ignoring the leg mass and treating the whole body as a point mass. The simulation is done in a 2D horizontal plane and excludes gravity. The distinct focus and model complexity differentiate their work from the one presented in this thesis.

Walknet is an artificial neural network which simulates the movements of walking stick insects (Cruse *et al.*, 1998). The earliest version of *Walknet* started with kinematic simulation (Cruse *et al.*, 1998), which was designed to mimic the joint rotations captured from real stick insects. This can be considered similar to the procedural modelling in computer animation (Abdul Karim *et al.*, 2012a). The controller was designed in a de-centralised architecture, similar to real insects and also to the method in this thesis. The controller managed eighteen leg joints (three for each leg) and was able to demonstrate impressive behaviours including turning and gait switching. Later various functional components were added to enhance the controller, such as searching for foot placements (Bläsing, 2006) and avoidance reflexes (Dürr, 2005). This control framework was also tested with dynamic simulation and six-legged robots (Dürr, 2005; Bläsing, 2006; Schilling *et al.*, 2012). For a review on a collection of work based on *Walknet* see Schilling *et al.* (2013).

Örjan Ekeberg (2004) constructed a reduced model of the nervous system by incorporating the activation mechanisms confirmed from biological experiments, and actuated a 3D biomechanical model using this nervous system. However the bistable circuits used in this system are only able to produce a constant activation value, rather than fully simulated spike signals of a neu-

ron. The muscle model follows the linear muscle model from Storrer (1976), ignoring some important features, such as the nonlinear relationship of force-length/force-velocity, from real muscles.

Szczecinski *et al.* (2014) proposed a neuron-mechanical simulation of the cockroach, which is currently the truest model compared with the individual and population behaviours of real neurons. However, the complexity of this framework limits its current applications to explore changes in locomotion when the animal transitions from straight walking to turning. There is still a significant gap between the skill repertoire of existing controllers and the demands of industry practitioners.

Pros and Cons in a Nutshell Existing work from bio-robotics and neuron-biomechanics simulation explores the alternative possibility of designing a locomotion controller. This bio-inspired solution has proved successful in terms of its naturalness and stability. However, state-of-the-art robotics are still far from their real equivalents in terms of naturalness while neuron-biomechanics research aims to *copy* the *real* controller as much as possible while currently ignoring the control inputs from users (Ijspeert, 2008). These artefacts constrain the direct application of existing work from both fields to animate virtual characters. The proposed framework in this thesis takes advantage of recent progress in neural science and bio-robotics, especially the inspirations in the design of the Central Pattern Generator (Ijspeert, 2008; Kukillaya and Holmes, 2008, 2009; Chung and Dorothy, 2010). Moreover, this work emphasises the intuitive and efficient control for animators and seamless integration with the traditional pipeline of animation production.

Chapter 3

Background

This chapter is an overview of insect physiology, in particular, how their physiological properties could affect, or even determine, their ability to move. Locomotion by itself is a complex interaction between the skeleton, muscle, nervous system and environment. The following sections will cover the fundamental knowledge on insect physiology from three aspects: skeletal structure, muscle actuator and nervous system. The purpose of including this chapter is to make this thesis as self-contained as possible, rather than to provide a thorough investigation. Therefore the presentation is highly selective. For an in-depth discussion, readers could refer to the additional literatures mentioned in the context or a textbook such as Chapman (1998).

The term *insect* used in this thesis refers to the class of *Insecta*, part of the phylum of *arthropoda*. The ant is selected as a representative insect for its typical physical structure and geographical diversity[1]. The morphology of ants varies across species and classes, even individuals. For example, considerably different from female and male ants, workers are wingless and are specialised for terrestrial locomotion. This thesis focuses on modelling the locomotion of workers, the majority of an ant colony.

[1] Ants are found on all continents except Antarctica and a few inhospitable islands, with an estimated 22,000 species (Jones, 2008)

3.1 Skeletal Structure

A typical insect body is composed of a body trunk and individual legs. The body trunk is divided into three parts as shown in Figure 3.1:

- **Head**, the front part, contains the brain, mouth, eyes, antenna etc.
- **Thorax**, the middle part, connects with six legs.
- **Abdomen**, the latter part, contains the heart and stomach etc.

The thorax consists of three segments known as the pro-, meso- and meta-thoracic segments, each of which possesses a pair of legs (Chapman, 1998). Each leg is typically divided into coxa, trochanter, femur, tibia, tarsus and pretarsus (Figure 3.1). The three main leg joints – the thorax-coxa (TC-) joint, the coxa-trochanter (CT-) joint and the femur-tibia (FT-) joint – are responsible for insect mobility (Büschges *et al.*, 1995). The structure of the tarsus and pretarsus allows strong attachment to surfaces, which also affects insect mobility (Ji *et al.*, 2011).

The following features are exploited to build a realistic, simplified insect model. Firstly, the femur is usually immovably attached to the trochanter (Büschges *et al.*, 1995; Chapman, 1998). Secondly, there is evidence showing that the pairs of coxa and trochanter, femur and tibia can only move relative to each other in the vertical plane (Büschges *et al.*, 1995; Chapman, 1998). Thirdly, although the joint between the thorax and coxa has more than one degree of freedom, most of the time it operates like a hinge joint (Blmel, 2011).

Unlike vertebrates with bones inside the body (endoskeleton), insects expose their skeletons outside (exoskeleton). This difference affects how the muscles connect and actuate the joint. The final torque applied on the joint is determined by both the contraction force and the moment arm. Each joint is connected by a pair of antagonist muscles – flexor and extensor (Figure 3.2a). As the name suggests, the contraction of extensor muscle increases the joint angle and extends the body part while the contraction of flexor performs the opposite function. The moment arm is the perpendicular distance from the rotating pivot (O in Figure 3.2b) to the force vector. The lengths, denoted

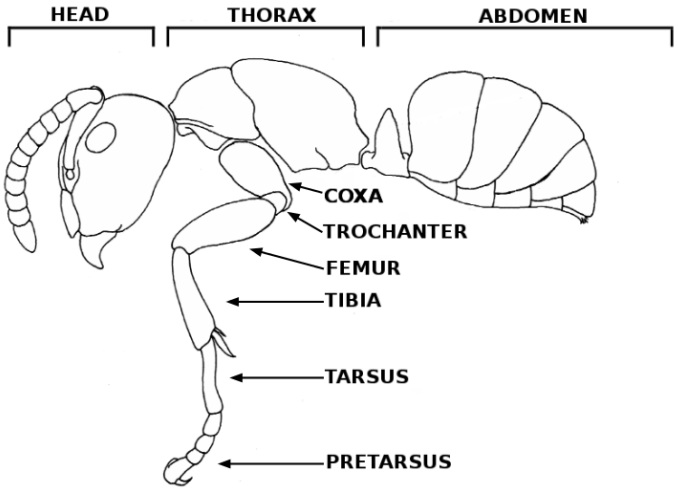


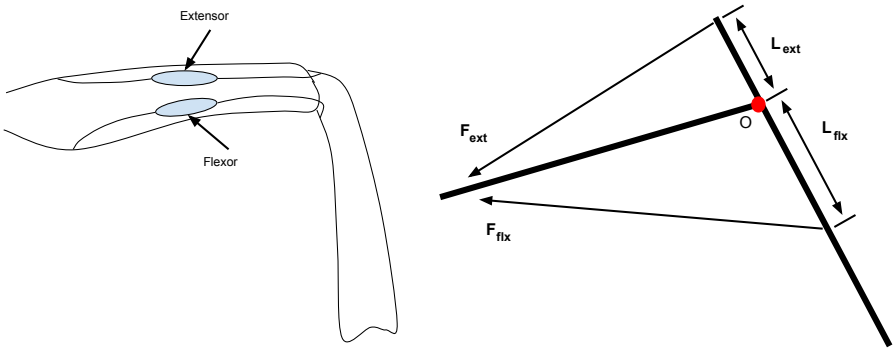
Figure 3.1: *Typical structure of an insect skeleton adapted from California Academy of Sciences online resources. The main body is divided into three parts: head, thorax and abdomen. The thorax is connected with six legs and each leg is typically divided into coxa, trochanter, femur, tibia, tarsus and pretarsus. For clarity only one leg is illustrated.*

as L_{ext} and L_{flx} in Figure 3.2b, represent the distances between the rotating pivot and attachment point respectively. In biological experiments and simulation, L_{ext} and L_{flx} are used as approximations of the moment arm of respective forces (Blmel, 2011).

Each joint has its rotational limits, therefore determining the furthest position a leg can reach. The Posterior Extreme Position (PEP) is the furthest point which the leg can reach backwards in relation to the COM in stance. The Anterior Extreme Position (AEP) is the furthest point which the leg can reach forwards in relation to the COM in stance. Research has suggested that a leg starts to swing when it approaches the AEP and PEP (Cruse, 1985). This observation is taken into consideration when designing the switch mechanism in Chapter 7.

3.2 Neural System

Compared with the vertebrate nervous system, the insect nervous system exhibits decentralisation, that is, instead of a definite central nervous system, the



(a) *Sketch of muscle attachment* **(b)** *Sketch of actuation mechanism*

Figure 3.2: *Sketch of a femur-tibia joint connection of an insect. (a) Sketch of the muscle attachment in the femur-tibia joint. Muscles are typically attached to either end of the skeleton, spanning across a joint so that contraction of the muscle moves one part of the skeleton relative to the other. (b) Sketch of the actuation mechanism in the femur-tibia joint.* F_{ext} *and* F_{flx} *denote the forces produced respectively by extensor and flexor muscles, and* L_{ext} *and* L_{flx} *denote the distances between the rotating pivot and attachment point respectively of extensor and flexor muscles.*

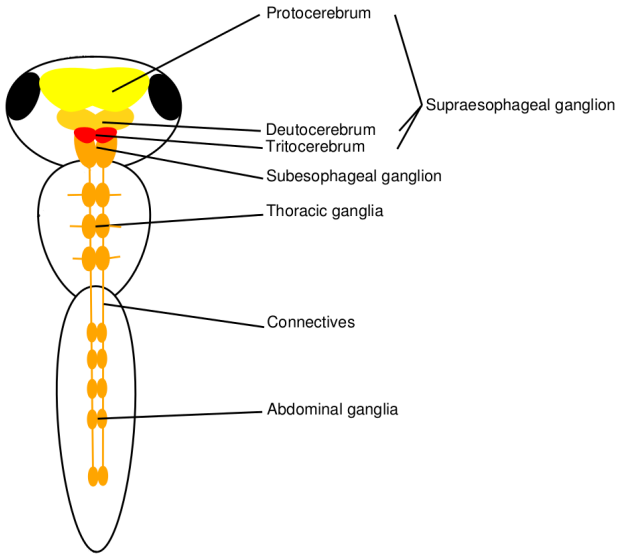


Figure 3.3: *Insect nervous system. The ganglia, each of which is a cluster of thousands of nerve cells, are distributed throughout the body, including the head, thorax and abdomen. The ganglia located in the brain send out the high-level commands while the ganglia located in the thorax coordinate local leg movements (Chapman, 1998).*

insect has numerous ganglia distributed throughout the body, though there is a major association centre in the head (Figure 3.3). Experiments demonstrate that each thoracic ganglion has its own walking pattern generating network (Bässler and Büschges, 1998). With regard to locomotion, the brain is in charge of the onset, direction and speed while the thoracic ganglia are responsible for the local control and coordination of legs. However the axons from the brain or subesophageal ganglion are still able to modulate the activity of thoracic ganglia, for example, to initiate, suppress, and maintain walking behaviour, or determine speed and direction of locomotory behaviour (Bässler and Büschges, 1998).

The ganglion is a cluster of thousands of nerve cells, or *neurons*, which are the basic units of the insect nervous system. There are two main types of neurons in the ganglia, motor neurons that control the muscle of the body, and inter-neurons that integrate information from sense organs. A neuron typically consists of a cell body (*soma*), a tree-like outreaching *dendrite* and a long cable-like *axon* (Figure 3.4). More specifically, the axon of a neuron is connected to the dendrite of the next neuron with a narrow distance, forming a structure called the *synapse*. When neural spikes (membrane potential peaks) arrive at the axon, chemical transmitters are released from the previous neuron and collected by the dendrite of the next neuron, and further processed by the soma in the next neuron, which fires its own series of spikes. This machinery forms the information processing of the insect nervous system (Gerstner and Kistler, 2002).

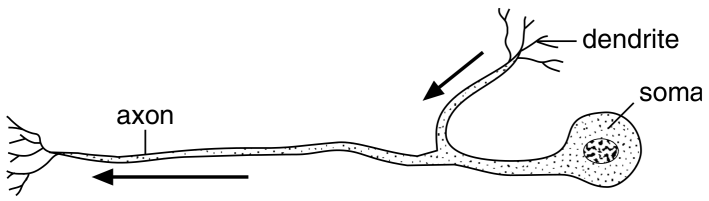


Figure 3.4: *The insect neuron as the basic unit of nervous system. Figure from Chapman (1998).*

The end of axon connects to either another neuron or muscle structure. For the latter case, it innervates the muscle. The innervation mechanism is complex and not yet fully understood. Most muscles are innervated by both

fast and slow axons and by inhibitory axons, but not all the fibres within a muscle are innervated by all three types of motor neurons (Chapman, 1998).

3.3 Muscle Property

Each muscle is made up of thousands of muscle fibers, each comprising many myofibrils in parallel (Figure 3.5). The motor neuron together with its innervated muscle fibre is referred to as the *Motor Unit* (Figure 3.6a). Individual myofibrils consist of longitudinally repeated cylindrical units, called sarcomeres. Each sarcomere contains contractile proteins, organised into a regular interdigitated matrix of thick and thin filaments, and is bounded by Z disks. The thick myosin filaments are composed of numerous myosin molecules, which are elongate structures with two globular *heads* at one end, while the thin actin filaments consist of two chains of actin molecules twisted round each other (Loeb and Ghez, 2000). The structure of insect skeletal muscle is detailed in Figure 3.5. The contraction force is produced by the relative sliding movement between the thick and thin filaments. The biochemical processes involved in muscle contraction are beyond the scope of this thesis. More detailed information could be found in Loeb and Ghez (2000).

The most widely used Hill-type muscle model (Zajac, 1988) is composed of three parts (Figure 3.6b):

- Contractile Element (CE) models the contraction machinery of muscle fibres and serves as the principle component of contraction forces when activated.
- Parallel Elastic (PE) spring models the resistive tension from muscle membranes when a muscle is passively stretched.
- Series Elastic (SE) spring models the effects of tendons that act like a spring to store elastic energy when an active muscle is stretched.

The analogue of tendon structure in insects is called the *apodeme*. When building an insect muscle model, it is appropriate to discount the *apodeme* or SE component in the Hill-type model because of the stiffness of *apodeme* (Holmes *et al.*, 2006).

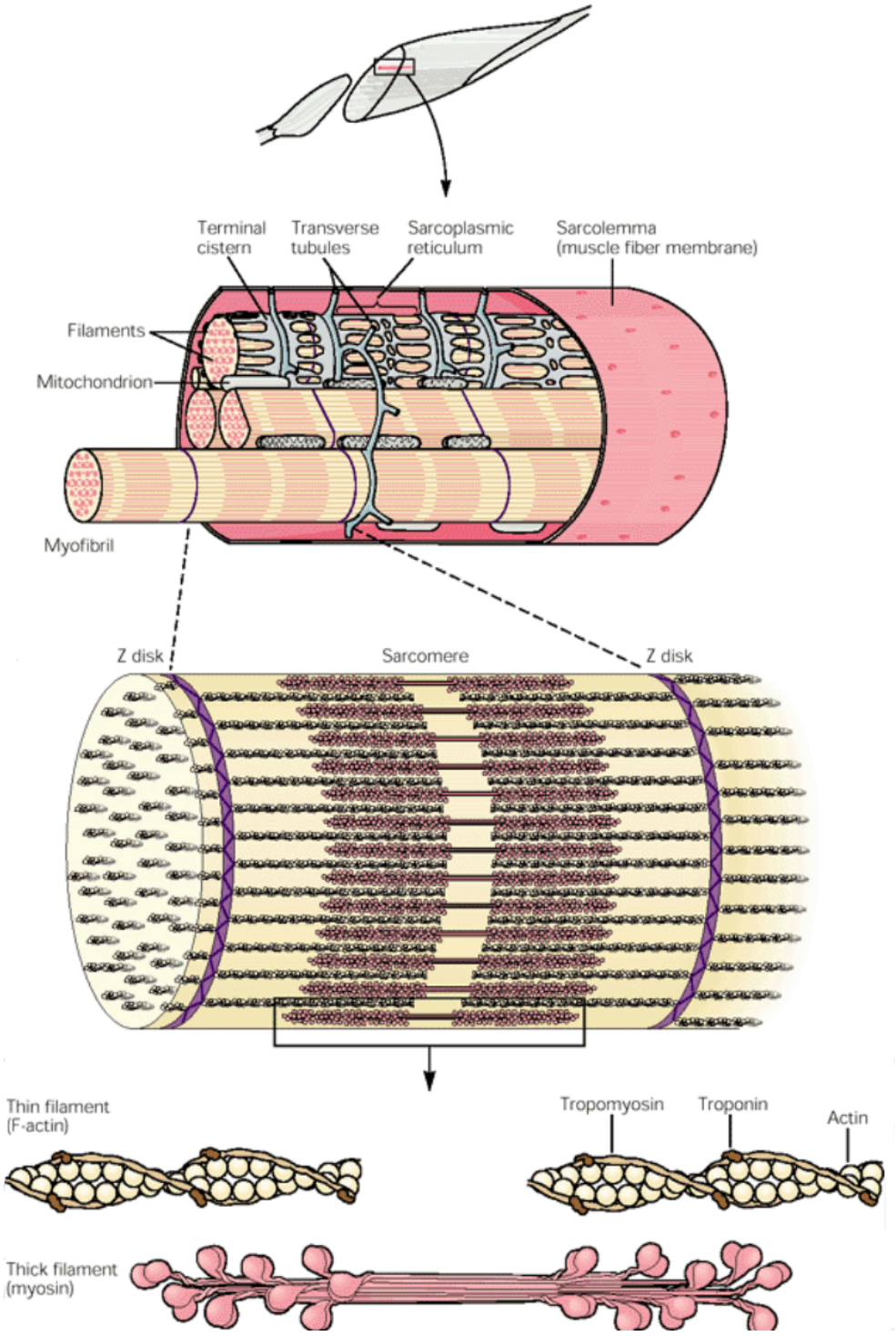
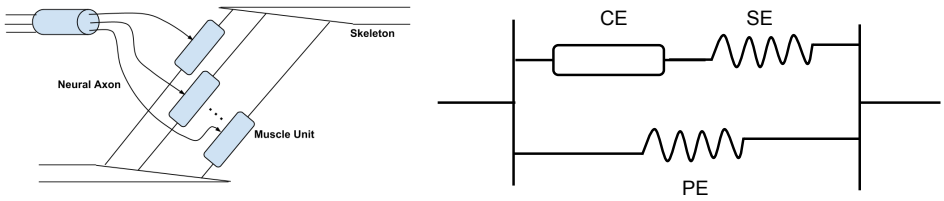


Figure 3.5: *Detailed structure of an insect muscle. From Loeb and Ghez (2000)*

Given these three components, the following equations hold:

$$\begin{aligned} F &= F_{pe} + F_{se} \\ F_{ce} &= F_{se} \\ L &= L_{pe} = L_{ce} + L_{se} \end{aligned} \tag{3.1}$$

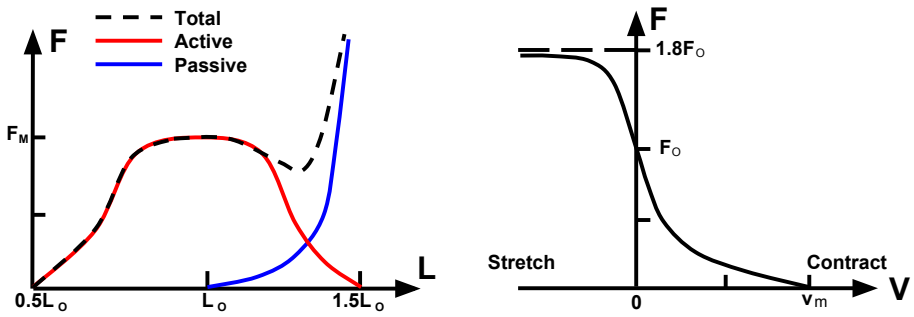
where variables of F, L indicate the measurements of force and length of each component respectively, and subscripts denote the corresponding components.



(a) **(b)**

Figure 3.6: *Components of an insect muscle. (a) Muscle structure corresponding with Figure 3.5. Thousands of muscle units, including the motor neuron and their innervated muscle fibres, constitute each muscle. (b) Muscle unit in correspondence with Equation 3.1. The component of Series Elastic (SE) is occasionally ignored due to its stiffness.*

The main features when building a Hill-type muscle model are the force-length and force-velocity relationships. The force-length (FL) component models the function of both active and passive forces in terms of the muscle length (Figure 3.7a). The data is normally collected in an isometric situation which means the muscle length is held constant. Experimental biologists measure the contraction force when muscles are stimulated (total force) and not stimulated (passive force). The active force can be derived by deducting the passive force from the total force (Zajac, 1988). The force-velocity (FV) component describes how the contraction force varies with respect to the muscle contraction velocity (Figure 3.7b). The data is collected by applying a constant force to pull at the end of the muscle, which contracts and eventually stops. The velocity is measured during this contraction process. Note that the FV relationship varies given different muscle lengths (Zajac, 1988).



(a) *Force-Muscle Length* **(b)** *Force-Muscle Contraction Velocity*

Figure 3.7: *Force-length and force-velocity relationships of insect muscle. This figure is reproduced based on the experimental result from Zajac (1988). (a) The passive force is generated when the muscle is passively stretched, thus creating resistive tension (F_{pe} in Equation 3.1). The active force is generated via the contraction of muscle fibres (F_{ce} in Equation 3.1). The total force is the sum of both active and passive forces. (b) The force indicates the total force. Muscle is stretched when the velocity is less than zero and contracts when the velocity is greater than zero.*

In comparison to the model of *Direct-Current* (DC) motors used in robotics, these springy muscles appear flawed for the following reasons:

- The direct output from the muscle is force and the final torque also depends on the moment arm. Different configurations of muscle length and velocity affect the outcome, giving rise to a more complex controller.
- Propagation of the control signal from the nervous system is slow and there is an additional delay when considering the activation and contraction dynamics inside the muscle.
- Fatigue effects can result from the nervous system failing to generate the signal or the motor unit failing to release Ca^+ to trigger the contraction dynamics.
- The single polarity means that the muscle can only pull and not push. Electronic servos, however, can alter the torque direction simply by alternating the direction of the electrical current.

The reason why these springy muscles have evolved in nature is not yet fully understood. It has been suggested that the springy effect saves energy dur-

ing transportation (Alexander, 1991, 2002). It has also been proposed that the springy feature increases joint stiffness, allowing automatic adaptation to external perturbations without control input (Loeb and Ghez, 2000).

Chapter 4

Triangle Placement Engine

4.1 Introduction

Insect locomotion demonstrates a strict alternating tripod pattern. Tripods formed by legs L1R2L3 and R1L2R3 switch alternatively from stance to swing in temporal sequence. *Supporting Triangles* are defined by foot prints of the tarsus of alternating legs L1R2L3 or R1L2R3 in a tripod gait. This work introduces a novel mathematical model, the Triangle Placement Engine (TPE), to author the characteristics of insect locomotion (for example speed and path) in a representation of the position and orientation of the supporting triangles.

The TPE is proposed to determine the distribution of supporting triangles along a given path while satisfying user-specified settings, such as locomotion speed, load condition, terrain and perturbations etc (Figure 4.1). The engine is built up by mathematically expressing the experimental observations on the existing patterns of supporting triangles made by the zoologist Dr. Zollikofer (Zollikofer, 1994a,b,c) and the following contributions from his peers (Jindrich and Full, 1999; Dürr and Ebeling, 2005; Seidl and Wehner, 2008). The results from Zollikofer (1994a,c) show that ants normally employ a double tripod gait when moving within a wide range of speeds. Zollikofer (1994c,b) reveals that the shape of the supporting triangle is independent of both speed and trajectory curvature (Figure 4.2), but is influenced by the carried load and

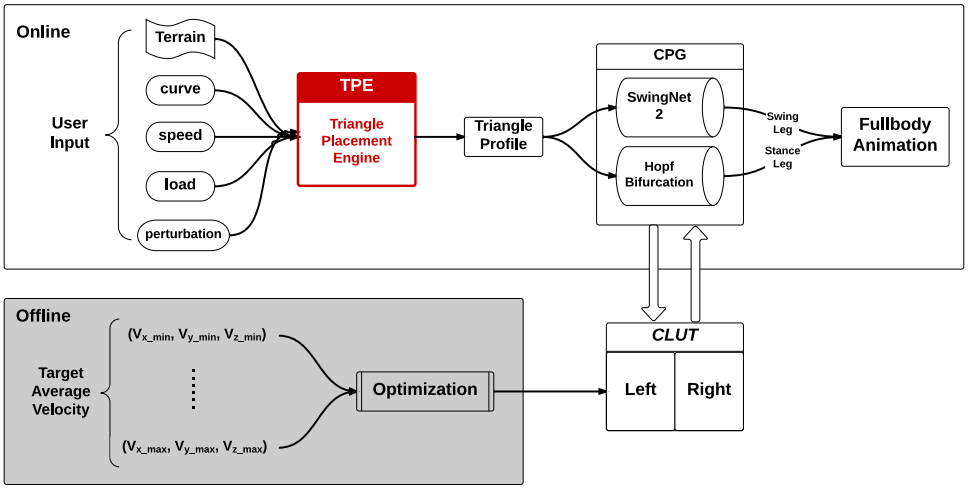


Figure 4.1: *Flowchart of the presented framework with the TPE highlighted. The TPE takes various user settings as input and determines the size, location and orientation of the* Supporting Triangle *along the path curve. The generated triangles are passed into the Central Pattern Generator (CPG), which selects the optimal control parameters from the Controller Look-Up Table (CLUT) in order to approach the desired trajectory.*

by external perturbations.

Animation authoring is an important aspect of character animation and recently it has attracted significant research interest. van Basten *et al.* (2010) proposed the concept of *step space* to generate character animations over a set of desired foot steps in real time. Their method is divided into three stages. First, the *step space* is constructed by extracting steps from existing motion capture data. A *step* is represented in a ten dimensional parameter space: six dimensions are used for the spatial parameters expressing the relative position of the swing foot in the local frame of the supporting foot at the start and the end of the swing phase, and four dimensions are used for the temporal parameters (stance durations of both placements, swing time and stance duration of the supporting foot). Given the initial foot placement, the weighted nearest-neighbour in the *step space* is then calculated to give the foot position for the next step. Spatial warping and time warping are then introduced in order to remove any artefacts. The concept of *Supporting Triangle* differentiates from the *Step Space* in its designed adaptation of the template triangle to various settings. This adaptation approach avoids the problem of lacking existing motion data, which is difficult to capture for small-scale insects.

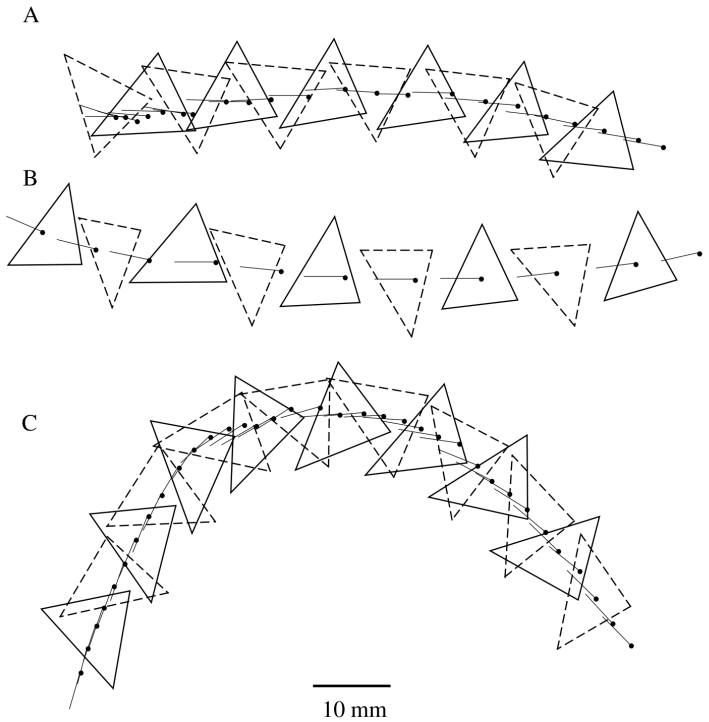


Figure 4.2: *Locomotion of Cataglyphis bombycina (body mass 17.9 mg) at a mean velocity of 17mm/s (A), 43 mm/s (B) and 15 mm/s (C); the mean radius of curvature in C is 30 mm. This figure shows that the shape of the supporting triangle remains unchanged under different settings of speed and trajectory curvature. Solid-line triangles are tripods R1L2R3; dashed-line triangles are tripods L1R2L3; the longitudinal axis of the body is indicated from the head position (dot) to the centre of mass (end of line); time intervals are 20 ms; walking direction is from left to right. This image is reproduced from Zollikofer (1994c).*

4.2 Triangle Placement Engine

The design of the Triangle Placement Engine (TPE) is composed of two parts: first, to find a triangle *template* for general cases and second, to adjust the orientation and shape of the *template* according to user-specified settings. The shape of the triangle is changed in the cases of carrying an object and walking against external forces, and the orientation of the triangle is influenced by the input curve path.

4.2.1 Triangle Template Design

A template for the Supporting Triangle is identified by computing the statistical mean of a collection of supporting triangles taken from a real ant walking along a straight line.

The shape of a triangle is defined as three sets of coordinates, each of which represents the location (x, y) of touchdown point in the local frame of the COM (the red dot in Figure 4.3). The static position of the COM is computed as the average of two threshold positions: one being the entrance point and the other being the exit point on the triangle edges (the blue dots in Figure 4.3). In total, 35 triangles were collected from three locomotion sequences taken from a real ant; the results are presented in Table 4.1.

	x	y
L1	2.625	0.973
L2	-0.105	2.714
L3	-2.520	1.746
R1	2.625	-0.973
R2	-0.105	-2.714
R3	-2.520	-1.746
Unit: $(x - mm)$; $(y - mm)$		

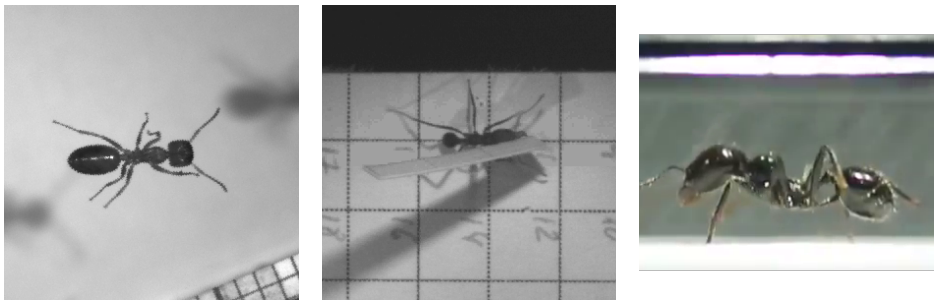
Table 4.1: *Coordinates of the vertices of the template triangle in the local frame of the COM. $+x$ aligns with the posterior-anterior direction. The symmetry of ant anatomy means that the tripods L1R2L3 and R1L2R3 have the same shape but are the reflection of each other along the anterior-posterior axis.*

Figure 4.3: *Extraction of the template triangle from video sequences of ant locomotion. The videos are provided by courtesy of researchers Holger Bohn from University of Freiburg, and Karin Moll and Walter Federle from University of Cambridge (Moll* et al.*, 2010). Blue dots indicate the positions where the COM enters and exits the triangle; the static position of the COM is the average of these two. The shape of the template triangle is represented as the relative positions of the three touchdown points of the standing legs in this local frame.*

Data Collection The foot contact positions for each leg are marked manually for every frame using the software *ImageJ* (Abràmoff *et al.*, 2004) and the plugin *MTrack* (Meijering *et al.*, 2012). The original videos are captured by researchers Holger Bohn from University of Freiburg, and Karin Moll and Walter Federle from University of Cambridge (Moll *et al.*, 2010). The pixel coordinates are then converted into real world coordinates (x, y). The videos are captured at a frame-rate of 30fps with a resolution of 640x480 pixels. The image resolution is sufficient to mark the touchdown position in common cases (Figure 4.4a). The *noise*, or the inaccuracy of the result, mainly results from two factors listed below:

- Visual occlusion occurs when the ant is carrying an object (a long paper flake in the captured videos). In Figure 4.4b, the right front and hind legs are occluded from the camera by the object. In this case, the position of the foot contact is inferred with the position where the foot re-enters the camera view. Though this issue inevitably introduces the noise to the collected data, the frames with such issue only occupy a small proportion of the samples (3 out of 57).
- Motion blur, a common problem in image processing, does not present great challenges in the case of determining the template triangle, because the foot stays static on the ground when forming the triangle

(Figure 4.4a and Figure 4.4b). However, this issue is critical in the case of acquiring the ground truth data of the foot trajectory during swing mode. For example in Figure 4.4c, it is hardly possible to determine the accurate position of the middle left leg ($L2$), which is swinging forward. This issue is mainly caused by the thinness of the legs combined with their fast movement for this frame-rate. For frames with such an issue, marker positions are determined by blending the marker coordinates in the preceding and following frames in which the positions can be accurately determined.



(a) **(b)** **(c)**

Figure 4.4: *Captured images of the locomotion of real ants. The images are provided by courtesy of researchers Holger Bohn from University of Freiburg, and Karin Moll and Walter Federle from University of Cambridge (Moll* et al.*, 2010). (a) Normal walking. In this case, the foot contact positions can be accurately marked with sufficient image resolution. (b) Walking while carrying an object. In some frames (3 out of 57), the foot contacts are occluded from the viewpoint by the object (for example, the right hind leg in this image). (c) Normal walking (side view). It is difficult to determine the accurate position of the foot during its swing mode (for example, the left middle leg in this image). Although this does not affect the design of the template triangle, the issue of motion blur introduces noise to the comparison between the simulated and real trajectory of foot.*

Adjustment of the Triangle Template Animators can adjust the shape and size of the triangle template via an IK handle on each leg, which connects from the coxa-femur joint to the tiba-tarsus joint (Figure 4.5). The final transformation of the IK handles will be saved and used as the template triangle. The manipulation of the IK handles is constrained by two limits. On one hand, the IK handles are constrained within the range specified by the Anterior Extreme Position (AEP) and the Posterior Extreme Position (PEP) of

each leg in the posterior-anterior direction (Figure 4.5). On the other hand, selected DOFs (three for each leg) are constrained within the range $[\bar{q}-g, \bar{q}+g]$. $\bar{q}, g$ are the mean value and the amplitude respectively of this rotational DOF. This constraint is in correspondence with the design of the CPG controller in Chapter 5.

The PEP is the furthest point which the leg can reach backward in relation to the COM in stance. Respectively, the AEP is the furthest point which the leg can reach forward in relation to the COM in stance. The distance between the AEP and PEP is equal to the maximum stride length $\bar{S}_{max}$, which can be computed by its relationship with respect to the walking velocity (Equation 4.4). Therefore, the positions of the AEP and PEP can be found as two end points of the segment, whose length is equal to $\bar{S}_{max}$ and middle point is the original position of the template triangle from the experimental result (Figure 4.5).

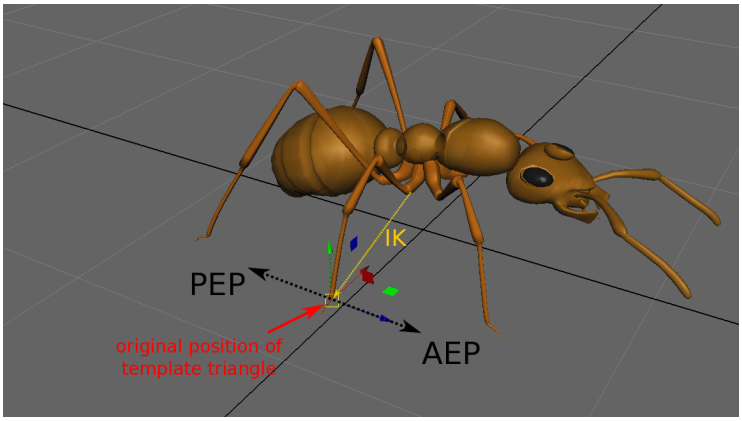


Figure 4.5: *Adjustment of the shape of the triangle template with an IK handle in the posterior-anterior direction. An IK solver connects from the coxa-femur joint to the end of the tarsus part. The adjustment is constrained by the Anterior Extreme Position (AEP) and Posterior Extreme Position (PEP) for each leg.*

4.2.2 Triangle Adaptation

Triangle Orientation–Curvature Ants are assumed to exert efforts to control their body in order to follow a curve path and, thus an intuitive way of controlling the placement of triangles along the curve path is developed. At

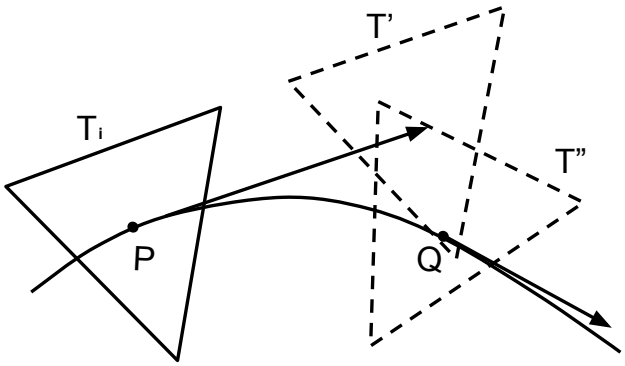


Figure 4.6: *In the case of walking along a curve, the triangle T_{i+2} is placed by interpolating the vertices of T' and T''.*

a certain point P of the curved path, if the triangle T_i on this point is known, it is necessary to put a triangle T_{i+2} of the next stride on the curve. A tangent line is first drawn at point P. A triangle T' is created by tiling along the tangent line for a distance of stride length $\bar{S}$ as if the ant moves along a straight path. Next a triangle T'' is created by drawing another tangent line at point Q as if the ant is moving along the tangent line at Q (Figure 4.6). The point Q is where the curve length $\hat{PQ}$ is equal to the stride length. The location of the next triangle T_{i+2} is computed as a weighted sum of the two congruent triangles T' and T'':

$$V_j^{T_{i+2}} = (1 - w_c)V_j^{T'} + w_c V_j^{T''} \tag{4.1}$$

where $V_j^{T'}$ denotes the location in fixed world coordinates of the j^{th} vertex of a certain associated triangle which is labelled by the superscript. The 1^{st}, 2^{nd} and 3^{rd} vertices are related to fore-, mid- and hind-legs respectively.

In classical mechanics, the centripetal force is to drive an object to move along a curve path. It is computed as (Takwale and Puranik, 1979):

$$F = mc\nu^2 \tag{4.2}$$

where m is the object mass, c is the curvature at the specific curve point and ν is the tangential speed along the path. The weight coefficient w_c is inspired

by the format of the centripetal force and takes the value of :

$$w_c = e^{\frac{-c\nu^2}{g}} \tag{4.3}$$

where g is the gravitational acceleration. When the curvature $c = 0$ (a straight line), the next triangle will be identical to T'', which locates exactly on the curve. The case of a large curvature indicates that the character must follow a sharp turn along the curve, which may lead to deviations of the simulated trajectory away from the predefined path. In such extreme cases, small velocities are required to achieve the sharp turning. In the example (Figure 4.7), the character walks along a curve at a speed $\nu = 0.015m/s$ and carries no object. The triangles are generated along the curve path according to Equation 4.1.

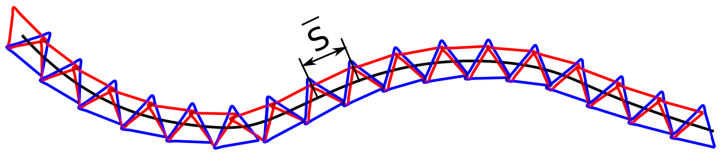


Figure 4.7: *The TPE generates the triangle profiles for characters walking along a curve path. In this case, the character is set to walk at $\nu = 0.015m/s$ without carrying an object.*

Triangle Distance–Speed This framework next considers how the speed of an ant affects the distance between two successive supporting triangles. Here the stride length $\bar{S}$ is defined as the distance between the barycentric points of two successive supporting triangles of same leg group, i.e. T_i and T_{i+2}. Results from experiments on real ants (Zollikofer, 1994c; Seidl and Wehner, 2008) demonstrate a linear dependency between the triangle distance $\bar{S}$ and speed $||\nu||$. During faster movement, insects reach their legs out further and this increases the distances between the foot contact positions of two successive strides:

$$\bar{S} = S_0 + a_s||\nu|| \tag{4.4}$$

The intercept $S_0 = 3mm$ and slope $a_s = 0.1s$ are constants. S_0 defines the minimum stride length. a_s defines the ratio of the amount of change in stride length to the amount of change in locomotion speed. The maximum velocity is $0.04m/s$. The reader should refer to Appendix D for a complete list of

constants used in the simulation. Figure 4.8 demonstrates a series of triangle profiles generated by the TPE. Compared with Figure 4.7, the velocity increases to $\nu = 0.02m/s$, which results in an increase in the distance between consecutive triangles.

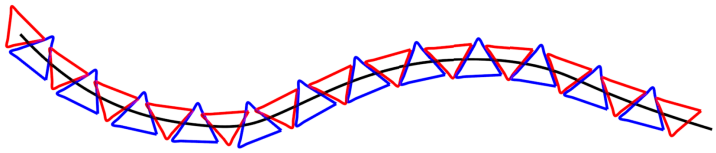


Figure 4.8: *The TPE generates the triangle profiles for characters walking with varying speeds. The distances between two consecutive triangles are increased compared to those seen in Figure 4.7. This is due to the increase in velocity (from* $0.015m/s$ *to* $0.02m/s$ *). Other settings remain the same.*

Triangle Shape–Load The shape of the *Supporting Triangle* is adjusted in the case of load-bearing characters. When carrying extra loads, ants actively lower their COM and increase the projection area of supporting triangles as a means of enhancing their locomotion stability. A linear model is introduced to describe the adjustments that are made to the supporting triangle, based on the experimental observations (Figure 1 in Zollikofer (1994b)). Although a nonlinear relationship may exist, there is insufficient experimental evidence and therefore, this work uses the linear model which is given as follows:

$$\begin{aligned} \Delta L_{COM} &= a_{com} m_w \\ \Delta x_i &= a_{xi} \Delta L_{COM} \\ \Delta y_i &= a_{yi} \Delta L_{COM} \end{aligned} \tag{4.5}$$

where m_w is the load weight, ΔL_{COM} is the shifting of the COM along the vertical axis and a_{com} is the slope ratio between these two variables. Δx_i and Δy_i are the anterior and lateral shifting in the touchdown position of the i^{th} leg measured in the local coordinate system. and a_{xi} and a_{yi} are coefficients specified for individual legs. Table 4.2 lists the values for a_x, a_y from linear regression on the experimental data (Figure 1 in Zollikofer (1994b)).

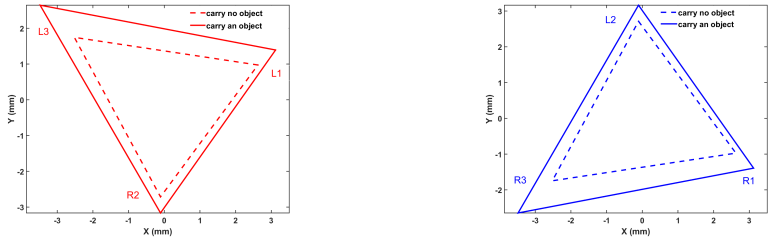
Figure 4.9 compares the shape of the supporting triangles in two different cases: when the character is carrying an object (the weight is $0.003kg$) and

	a_x	a_y
L1	0.33	0.28
L2	0.0	0.30
L3	-0.64	0.60
R1	0.33	-0.28
R2	0.0	-0.30
R3	-0.64	-0.60

a_x, a_y are ratios with no units

Table 4.2: *Coefficients for Equation 4.5.*

when the character is carrying no objects. When the character is carrying no objects, the shape of the supporting triangle is the same as the template triangle (Table 4.1). When the character is carrying an object, the shape of the supporting triangle is adjusted according to Equation 4.5. The result shows that in the case of carrying an object, the character stretches out its legs and enlarges the area of the supporting triangle.



(a) *Left Tripod (L1R2L3)* **(b)** *Right Tripod (R1L2R3)*

Figure 4.9: *Comparison of supporting triangles between the cases of carrying an object (the solid line) and carrying no objects (the dashed line). The object weight is* $0.003kg$.

Triangle Shape–Terrain Insects adjust the shape of their supporting triangles dynamically when walking up or down a gradient, which maximises the utilisation of gravitational potential and secures the stability. Biologists believe that these changes may be caused by the shifting of the COM, which is equivalent to load bearing (Seidl and Wehner, 2008). Here the effect of the slope rate on the shifting of the COM ΔL_{COM} is modelled as:

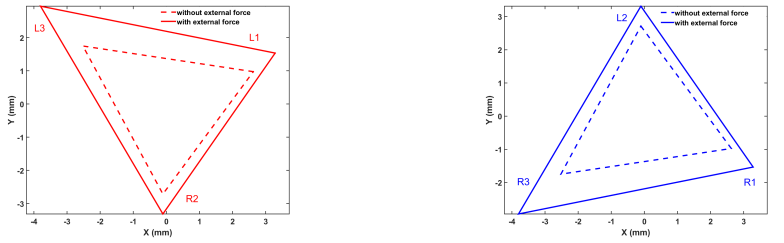
$$\Delta L_{COM} = -h\ sin(\theta) \tag{4.6}$$

where h is the average height of the COM during a single stride and θ is the angle of inclination of the gradient. θ is positive when the character walks upslope and negative when the character walks downslope. Given the variability of ΔL_{COM}, the changes to the supporting triangle are computed with Equation 4.5. When the terrain is uneven, the supporting triangle is first placed on the tangent plane and then the triangle vertices are projected to the nearest points on the terrain surface.

Triangle Shape–Perturbation If the character is subjected to external non-gravitational forces during locomotion, the forces can be modelled as perturbations similar to the case of load bearing. For the force F_{ext}, the equivalent distance of the COM shifting ΔL_{COM} is modelled as:

$$\Delta L_{COM} = \frac{-F_{ext}}{mg} h \tag{4.7}$$

where m is the mass of the ant, g is the gravitational acceleration and h is the height of the COM. Similar to the load-bearing scenario, insects lower their COM and spread out their legs in order to maximise the Supporting Triangle area, and thus the stability is improved. Following the previous discussions, the supporting triangles are adapted according to Equation 4.5. When walking against the external perturbations, the character increases the area of the supporting triangle (Figure 4.10).



(a) *Left Tripod (L1R2L3)* **(b)** *Right Tripod (R1L2R3)*

Figure 4.10: *Comparison of supporting triangles between the cases of walking against an external force (the solid line) and walking against no external force (the dashed line). The magnitude of the external force is* $0.01N$.

4.3 Results & Discussions

UI Interface A *Maya* plug-in is developed in order to allow animators to author insect behaviour in a user-friendly way (Figure 4.11). The interface is set up in correspondence with the design of the TPE. The ground surface and path curve are first constructed, using built-in functions of the *Maya*. Next, animators specify the velocity, load weight and external force for the locomotion task. Using customised Python commands, these variables can also be key-framed as attributes associated with individual characters. For example, characters are able to move along a continuous curved path with speed variations.

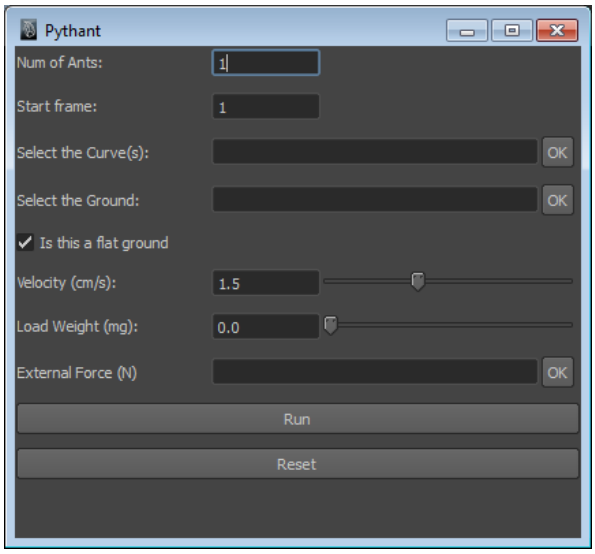


Figure 4.11: *UI interface written as a* Maya *plug-in. The framework is written as a* Maya *plug-in so that it can fit into the existing animation production pipeline. The user interface allows animators to specify the curve, ground, velocity, load and external force. Selecting the option of a flat ground speeds up the program by avoiding the costly computing of finding the closest point on the ground surface.*

Performance Table 4.3 presents a summary of computing time of generating the supporting triangles for some representative cases. The time cost is divided into two parts, $T1$ and $T2$ in Table 4.3. $T1$ refers to the time cost of the core algorithm of the TPE, including computing the shape, location and orientation of the supporting triangles. $T2$ refers to the time cost of other

auxiliary operations, such as rendering in the *Maya* viewport.

The time cost $T1$ for generating each supporting triangle is on average 1.1 milliseconds, with a standard deviation of 0.8 milliseconds. This performance outperforms the similar method in determining the foot-contacts of virtual characters (van Basten *et al.*, 2010). In comparison, their method requires on average 3 milliseconds, with a standard deviation of 1.9 milliseconds to determine the foot contact positions for the next step. The time cost $T2$ for other operations (rendering etc) on each supporting triangle is on average 13.8 milliseconds, with a standard deviation of 5.3 milliseconds. The result shows that the core algorithm of the TPE only occupies a small percentage (about 7%) of the total time ($T1 + T2$). Comparatively, the time cost $T2$ is most related with the functionality inside the *Maya*. This is part of the price to be paid, in exchange for the off-shelf tools from Maya and familiarity with its UI for animators. One of the method to reduce the $T2$ is to switch to C++ API of the *Maya*, rather than Python API used in the current implementation.

The time cost of determining the triangle location and orientation is much larger in the case of uneven terrain than other cases. This is due to the additional operations to find the closest point on the mesh for the triangle vertices, which is a time-consuming operation in the *Maya*. Replacing the existing *Maya* algorithm with faster alternatives could further speed up this operation.

For the case of swarm simulation, individuals from a group of 30 ants are instructed to follow 30 individual curves. By using the module of *multi-processing* (MP), the performance increases by approximately three folds when four processing cores are used in parallel (Table 4.3). Further code optimisation is expected to improve the performance.

Speed Locomotion speed is the key input from the animator and affects directly the distance between consecutive triangles. Animators are able to set the speed as key-frames while insects travel on the same curve. Given the strategy proposed in Equation 4.1, characters are constrained to walk at slow speed when following a curved path with a high curvature. Figure 4.12 compares three different paths with the same locomotion velocity ($0.04m/s$). In the case of straight path $\mathbf{A}$, the *Barycentric* centres of triangles are located

Cases	Curve Length	Ants	Triangles	T1	T2
U-turn	30.4	1	61	0.068	0.752
Perturbations	35.1	1	70	0.073	1.017
Uneven Terrain	32.0	1	64	1.028	2.350
Load Carrying	36.6	1	74	0.083	1.247
Swarm	765.23	30	1531	1.712	24.281
Swarm (MP)	765.23	30	1531	0.640	7.761

Unit: (Curve Length - cm); (T1 - second); (T2 - second)

T1: total time cost for computing the shape and location of all supporting triangles.
T2: total time cost for other operations, including rendering in Maya viewport.

Table 4.3: *Computing time of generating the supporting triangles for some representative settings. There is no significant influence by load/velocity/perturbations on the algorithm performance. The case of uneven terrain costs more time since it requires additional time to find the closest points on the mesh for each vertex of supporting triangles. Curve length is given in cm. For the case of swarm simulation, the curve length is the total length of curves for all individuals.*

exactly on the path. When the curvature of the input paths $\mathbf{B}, \mathbf{C}$ increases, the triangles start to deviate from the path. The deviation could be regarded as one of shortcomings of this framework since the triangles are not located exactly on the curve when the character travels along a curve path at a high speed. However, this shortcoming could also be regarded as another match with the experiences from real world: cars taking a sharp turning at high speeds are more likely to lose balance and deviate from their original path.

Carrying loads In the case of carrying an object (Figure 4.14a), insects normally slow down and shorten their stride length (in agreement with Equation 4.4). Both triangle expansion and distance shortening lead to an increasing overlapping area of two neighbouring triangles, T_i and T_{i+1}, which is believed to improve the stability of the ant in transition of stance legs (Zollikofer, 1994b). The maximum load for the simulated ant is determined by two limitations: the maximum leg stretch that still allows a full stride and the lowest COM position (assumed to be ground level). The maximum load is found to be $m_{Lmax} = 0.005kg$ (five times the ant's own weight).

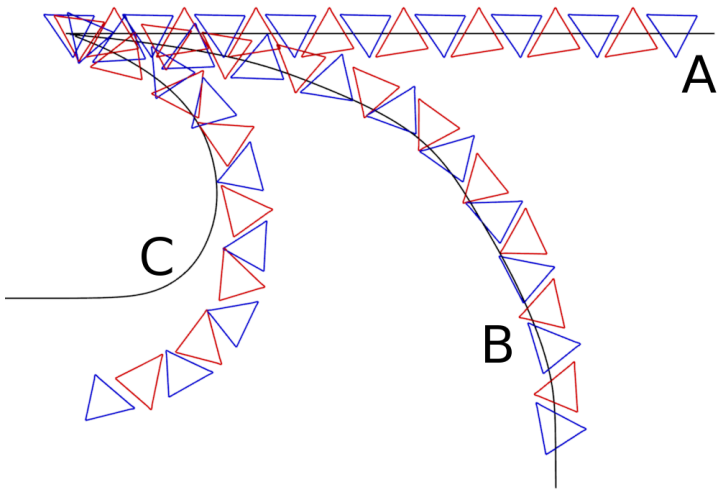


Figure 4.12: *Triangle profiles generated when the character is walking fast on three different curves. When the character is walking at fast speed while taking a sharp turning (in this case 0.04m/s), the generated triangle profiles will not strictly follow the desired trajectory (curve **B**), or even not converge to it (curve **C**).*

Curved path The proposed framework allows virtual characters to follow a predefined trajectory and achieve complex navigation, such as taking a moderate U turn (Figure 4.14b). However minor flaws are found for synthetic motion in cases such as extreme turning. This is partially caused by the issue of speed discussed above, and partially caused by joint limits, which prevent the swing leg from reaching the next supporting triangle.

Uneven terrain The TPE is capable of generating foot contact positions on an uneven terrain (Figure 4.14c). This allows the character to traverse the uneven terrain with the CPG controller proposed in the latter chapter. The terrain is input as a *Nurbs* surface in the *Maya*. The virtual ant is placed on a predefined uneven terrain, with gradients in the range [-10°, +10°]. The proposed controller adapts to the terrain changes by adjusting the COM height and the shape of Supporting Triangle, according to the rules defined in Equation 4.6. For the swing leg, the controller also anticipates the stepping location for the next stride and adjusts the $\mathbf{q}_{AEP}$ in Equation C.1.

Perturbations Since the controller is physics-based, the character can respond to external perturbations. Most existing works demonstrate the controller stability by applying unanticipated forces for short intervals. The controller presented here differentiates itself from previous works in its dynamic adjustments of the Supporting Triangles, which allows unanticipated pushes to be applied for a continuous period. Some typical circumstances that require a controller of this type include walking against a continuous wind. In the case of external perturbations, the character dynamically increases the area of the supporting triangle (Figure 4.13). The maximum force which insects can resist for at least 20 strides is $0.01N$.

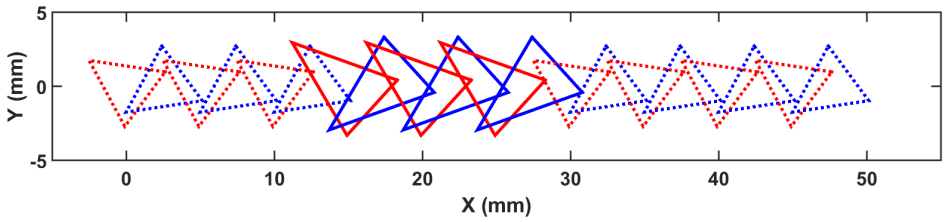


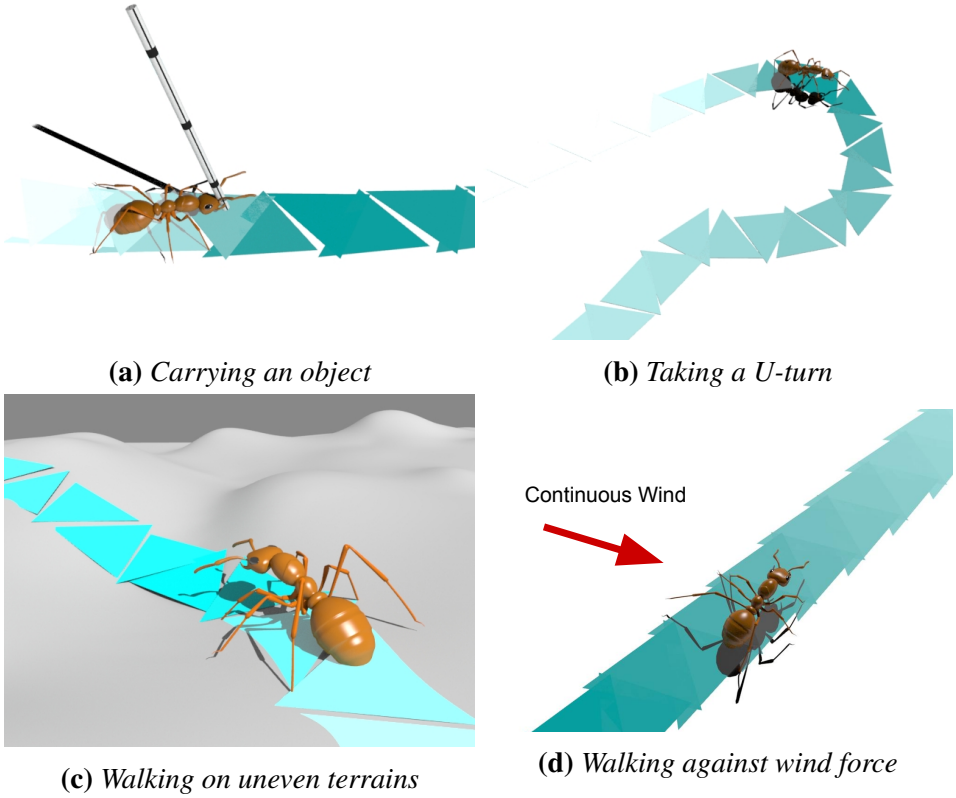
Figure 4.13: *Triangle profiles generated when an external force is introduced for a time interval. Triangles in solid lines are generated when the external force exists, and triangles in dashed lines are generated when the external force does not exist.*

4.3.1 User Study Experiment

Measuring the naturalness of synthetic motion is a difficult task. Currently there are two popular methods. One is to design a user study experiment, in which humans subjectively rate the naturalness based on their perceptions (Jansen and van Welbergen, 2009). Another is to compare against the motion capture data. This section presents the result of user study experiment, while the next section will discuss the comparison against the motion capture data.

Experiment Setup Two groups of participants take part in the experiment:

- Non-animators: 10 graduate students (5 male, 5 female) from the subject of computer science, with the age range between [24, 25].



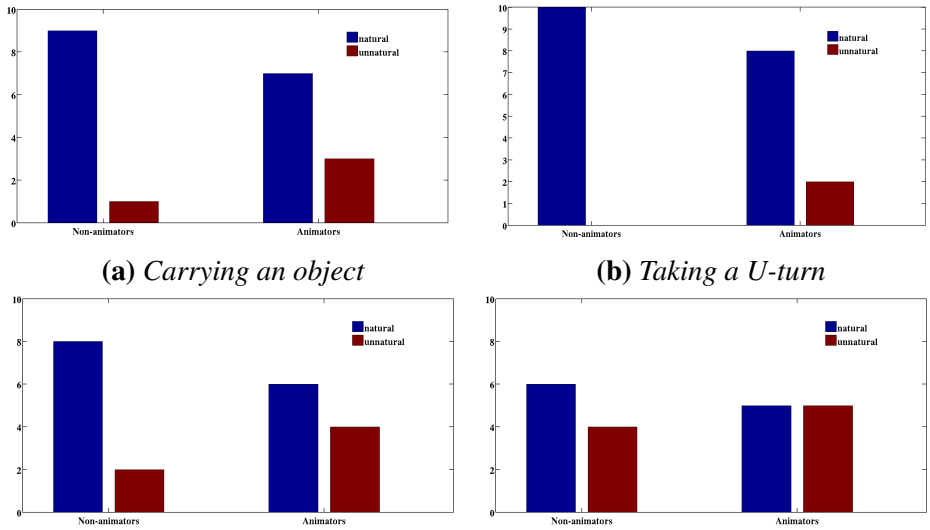
(a) *Carrying an object*

(b) *Taking a U-turn*

(c) *Walking on uneven terrains*

(d) *Walking against wind force*

Figure 4.14: *Screenshots of motion sequences generated from this framework. (a) carrying an object. (b) walking along a U-shape curve. (c)walking on uneven terrains. (d) walking against a continuous force from side wind.*



(a) *Carrying an object* **(b)** *Taking a U-turn*

(c) *Walking on uneven terrains* **(d)** *Walking against wind force*

Figure 4.15: *Result of user study experiments of four animation sequences in correspondence with Figure 4.14. The participants include both professional animators and non-professional students. The four sequences receive an average score of 73.75% (being rated as natural), which confirms the naturalness of the synthetic motion.*

- Animators: 10 professional animators (5 male, 5 female), with the age range between [24, 28] and the range of working experiences (in years) between [1, 5].

These two groups are intentionally selected to study whether there exist differences between the result of professionals and non-professionals.

The participants are asked to watch four different animation sequences generated by this framework and rate whether this animation looks natural or not. Screenshots of these four sequences are presented in Figure 4.14 (with supporting triangles being removed).

Result The result from the user experiment confirms the naturalness of the synthetic motion generated by the framework in this thesis (Figure 4.15). More than half of the participants rate the animation as natural in all cases. The average ratio is 73.75%; in other words, on average every three out of four animations are rated as natural by both professionals and non-professionals. This proves that this framework is able to generate natural insect animation.

Detailed analysis includes:

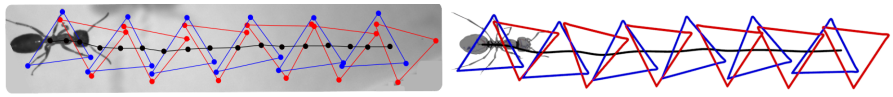
- Animators generally give lower scores than non-animators. This result comes as no surprise since the professionals should have a better capability in judging the animation quality. The major defects of the animation, suggested by the animators, are the rigid rotations of the body trunk. In other words, there is no relative rotation between the head, thorax and abdomen. Since the current framework considers the whole trunk as a single rigid body, secondary animation is needed to synthesise the relative rotations between the three body parts.
- The animation of walking against the wind force receives lower scores than the other three. This preference is consistent between the animators and non-animators. The reason for receiving low score on this case may be the lack of motion variability in this animation sequence. Van Welbergen *et al.* (2010) suggests that a performance looks unnatural if the same motion occurs repeatedly. In the case of walking against the wind, the adjustment of increasing the triangle area is not significant; therefore the character appears to repeat the same pattern along a straight line. In other cases, the character carries an object and takes a U-turn, which adds to the motion variability and increases the perceived naturalness of the motion.

4.3.2 Comparison with the ground truth

To quantitatively assess the synthetic motion, the simulated results are compared against the data collected from the ground truth.

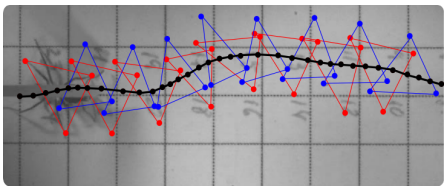
Walking along a curve path First, the triangles of real ants are extracted from captured videos of *Atta vollenweideri* (Figure 4.16(a) and 4.16(c)). Markers are placed at the positions of foot contacts when the legs stay on the ground, using the software *ImageJ* (Abràmoff *et al.*, 2004) and the plugin *MTrack* (Meijering *et al.*, 2012). The COM trajectory is drawn by connecting the position of the ant's trunk centre in each frame. In Figure 4.16, the triangles in red and blue are the left (L1R2L3) and right (R1L2R3) tripods

respectively, and the curve in black is the trajectory of the COM. Next, the COM trajectories of real ants are imported into the simulation framework. The supporting triangles for virtual ants are generated based on the input trajectories and the same settings (speed and load etc) extracted from the videos.

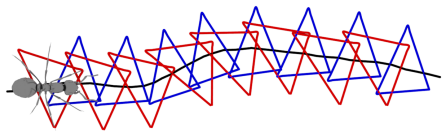


(a) *Real ant on a straight path*

(b) *Simulated ant on a straight path*



(c) *Real ant on a curve path*



(d) *Simulated ant on a curve path*

Figure 4.16: *Comparison between the Supporting Triangles generated by real and simulated ants. The images of (a) and (c) are provided by courtesy of researchers Holger Bohn from University of Freiburg, and Karin Moll and Walter Federle from University of Cambridge (Moll* et al.*, 2010). (a) a real ant walks along a straight path. The ant carries no object at an average speed of 15mm/s. (b) a simulated ant walks along a straight path. The character is set to carry no object and walks at a speed of 15mm/s. (c) a real ant walks along a curved path. The ant carries an object (the weight is 0.001kg) at an average speed of 20mm/s. (d) a simulated ant walks along a curve path. The character is set to carry an object (the weight is 0.001kg) and walks at a speed of 20mm/s.*

A good correspondence is achieved for the overall shape and placement of triangles. However, there still remain differences between the exact shape of the triangles, especially in the case of walking along a curved path (in Figures 4.16c and 4.16d). The reason for the deviations may be that the simulated location of the triangle is computed as a weighted sum of two triangles using Equation 4.1, instead of the one exactly on the trajectory.

Detailed data analysis of the comparison between the simulated and captured triangle profiles is given in Table 4.4.

- The average geometric shape difference A_{diff} is computed as:

$$A_{diff} = ||\frac{\sum A_{simu} - \sum A_{real}}{\sum A_{real}}|| \tag{4.8}$$

where $\sum A_{simu}, \sum A_{real}$ are the sum of areas of supporting triangles in simulated and real cases respectively.

- The deviation from the path D_{sum} is computed as:

$$D_{sum} = \sum_{T_i} d_{T_i},\ T_i \in \text{all simulated triangles} \tag{4.9}$$

where d is the perpendicular distance from the *barycentric* centre of the simulated triangle to the predefined locomotion trajectory.

The result shows that the shape of the simulated triangle is close to identical to the real ones when the character is following a straight path. However, the mismatch between the simulated and real triangles increases when the path is a curve one. It is worth pointing out that due to the data availability, the ground truth in Figure 4.16c is a case of an ant walking along a curve path while carrying an object (weight $0.001kg$). The effects of carrying an object on the triangle shape are removed by reversely applying Equation 4.5. However, this may be partially responsible for the deviation between the simulation and the ground truth presented in Table 4.4.

	Straight Path	Curve Path
Number of Triangles	12	16
Average Shape Difference	3.78	12.56
Average Deviation from Path	0.46	1.57
Unit: (Average Shape Difference - $\%$); (Average Deviation from Path - mm);		

Table 4.4: *Comparison between the simulated triangle profiles and the ground truth. The data show that in the case of straight walking, the simulated triangle profiles closely match the real ones. Larger differences exist, however, in the case of walking along a curved path. This is because the proposed adaptation of the triangle in this case is determined by both the curve trajectory and locomotion speed, which results in a* loose *tracking of the desired trajectory (Equation 4.1).*

Foot Trajectory Additional comparison is conducted about the trajectories of the left front foot $L1$ in both simulated and real cases (Figure 4.17). The foot trajectory of the real ant is obtained by manually marking the foot position in a sequence of video frames using the software *ImageJ* (Abràmoff

et al., 2004) and the plugin *MTrack* (Meijering *et al.*, 2012). The video is provided by courtesy of researcher Antoine (2014). The foot trajectory of the simulated ant is obtained by computing the foot positions after the whole body motion is synthesised. The comparison between the real and simulated trajectories reveals that:

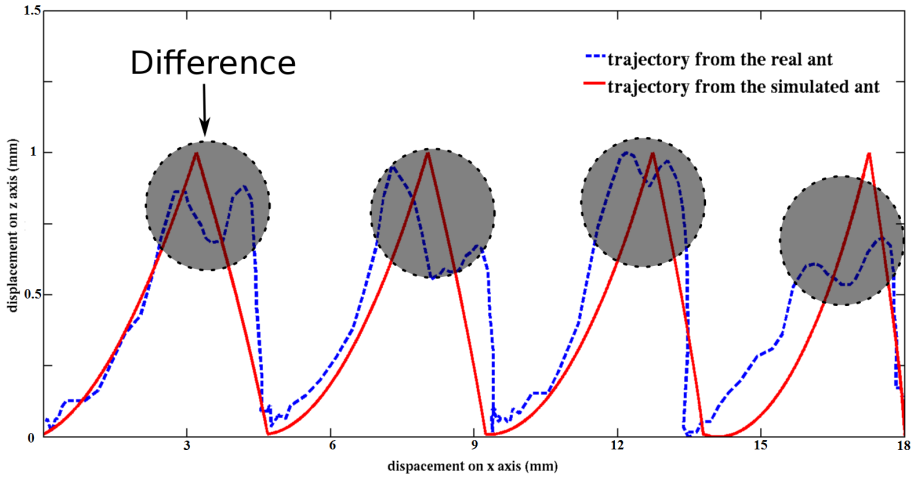
- The overall shapes of the two trajectories, including the timing of swing and stance phases, the peak value, are similar.
- However, there exists a significant difference in the middle of the swing phase (highlighted with shadowed circles in Figure 4.17). The simulated trajectory demonstrates a sharp turning before and after the peak value, while the real trajectory has a *valley* between two peaks. The reason accounting for this difference could be that each step of the simulated trajectory is only divided into stance and swing phases, while real ants may employ additional strategies to prepare for the event of ground contact during the second half of the swing phase.

4.4 Summary

This chapter explains the design of the Triangle Placement Engine (TPE). The TPE serves as the high-level interface communicating with the user and converts the user input into triangle profiles, which are then passed into the animation engine to generate the full-body animation. Based on the experimental data, this thesis investigated how ants adjust the shape and orientation of their Supporting Triangles under different conditions of speeds, curves, loads, terrains and perturbations. The controller is able to drive the character in various challenging situations, such as walking against a continuous wind which has not been possible in previous research.

Generating the whole body motion for virtual characters from their foot contacts or trajectory of the COM is not entirely new, and previous sections have described existing work in this field (Singh *et al.*, 2011; van Basten *et al.*, 2010; Lockwood and Singh, 2012). However the work presented in this thesis is the first to develop a comprehensive procedure where a wide

(a) *Captured image of ant locomotion (by courtesy of Antoine (2014)).*



(b) *Comparison of the foot trajectory of the left front leg ($L1$).*

Figure 4.17: *(a) Captured image of ant locomotion. This screenshot is taken from the video provided by courtesy of researcher Antoine (2014). (b) Comparison of the foot trajectory of the left front leg ($L1$) in both simulated and real cases. The dashed blue line is from the real ant and the solid red line is from the simulated ant. The foot trajectory from the real ant shows that the swing leg is lifted up again in the second half of the swing phase, which implies that additional strategies may be applied to prepare for the foot contact of the next step.*

range of settings are thoroughly considered and mathematically formulated using data from real insects.

Chapter 5

CPG–Based Controller Design

5.1 Introduction

The previous chapter discussed about how the shape and orientation of the Supporting Triangles adapt to the five high-level settings specified by users, including the path curvature, speed, load, terrain and perturbation. This chapter will cover the detailed design of the low-level component, the Central Pattern Generator (CPG). In recent years, biologists have confirmed that animals have evolved to possess a specialised neural structure, the CPG, to coordinate motor behaviour (Collins and Stewart, 1993; Mellen *et al.*, 1995; Golubitsky *et al.*, 1999; Katz and Hooper, 2007; Ijspeert, 2008). A CPG is a neural network that is able to produce endogenously (i.e., without rhythmic sensory or central input) coordinated patterns of rhythmic output (Chung and Dorothy, 2010).

This control principle has gained widespread acceptance in the robotics community and researchers have successfully designed CPG-inspired controllers to control robots in different situations, such as walking quadruped (Liu *et al.*, 2011), swimming fish (Crespi *et al.*, 2008) and flying bird (Chung and Dorothy, 2010) etc. Compared with other approaches based on finite-state machines, time-series tracking, or heuristic control law (e.g. Virtual Model), the CPG has the following advantages (Ijspeert, 2008):

- Robustness against perturbations of the state variables. Because a CPG

is designed to produce stable rhythmic patterns, the system is able to return automatically to its limit cycle on the phase plot.

- Reduced control parameters. With a hierarchy of modulated oscillators, a CPG normally has a small number of control parameters and this reduces dramatically the dimensionality of the control signal.
- Convenient integration of sensory feedback. A CPG is normally represented as a system of differential equations, which can be coupled with additional inputs from sensory feedback.

The objective of this chapter is to investigate the hypothesis that the adaptive control and synchronisation of coupled nonlinear oscillators, inspired by the CPG found in animal spinal cords, can generate realistic locomotion for virtual insects. Figure 5.1 presents an overview of the framework with the CPG highlighted. The design of the CPG includes separate treatments for legs in stance and swing modes. The main insect body is driven forward by legs in stance controlled by a network of nonlinear oscillators. Experimental results show that insect legs constitute only a small proportion of mass compared to that of the whole body (6% in cockroach) (Kram *et al.*, 1997); legs can therefore be treated as being massless and legs in swing are decoupled from the physics simulation. Legs in the swing mode are animated with the procedural controller *SwingNet2* proposed by Schumm and Cruse (2006). Readers should refer to Appendix C for detailed information on the implementation of this model.

5.2 CPG Model

5.2.1 Design Intuition

It is important to clarify, from the outset, that there is no *obvious* or *correct* representation of the CPG given the complexity of insect nervous system (an ant typically has 250,000 neurons) and behaviours (Ijspeert, 2008). Researchers have proposed several levels of CPG models to fit different purposes. Popular choices include (Ijspeert, 2008):

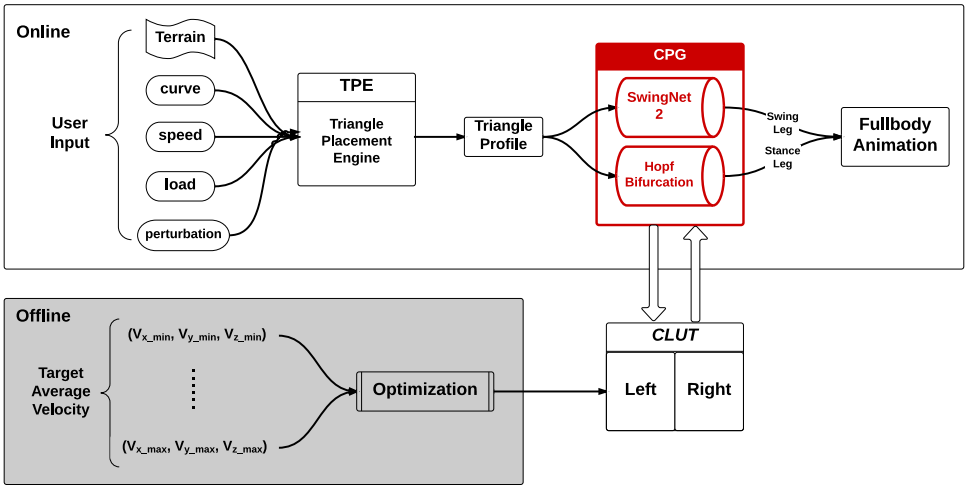


Figure 5.1: *Flowchart of the presented framework with the CPG highlighted. The CPG handles the stance and swing legs separately in order to simplify the control mechanism. Inputs of Supporting Triangles manipulate the behaviours of the CPG by selecting the optimal control parameters to follow the desired trajectory.*

- Detailed biophysical models that are constructed based on the *Hodgkin-Huxley* type of neuron models. This type of model computes how ion pumps and channels affect membrane potentials and the generation of action potentials (Hellgren *et al.*, 1992).
- Connectionist models that use simplified neuron models, such as the integrate-and-fire neurons, to analyse the overall network and inter-neuron connections (Ekeberg, 1993).
- Oscillator models that are represented as a network of nonlinear oscillators with the motivation to study the general topology of coupling (Collins and Stewart, 1993; Crespi *et al.*, 2008; Righetti and Ijspeert, 2006)

Even for the category of oscillator models, different researchers choose different oscillators (mathematically speaking Ordinary Differential Equation (ODE)) as the building blocks. One must choose an appropriate model for a particular purpose. The model selection depends on the researcher's own requirements since each model represents a different level of detail and has its own advantages and disadvantages. This work selects the third option (the oscillator model) mainly because:

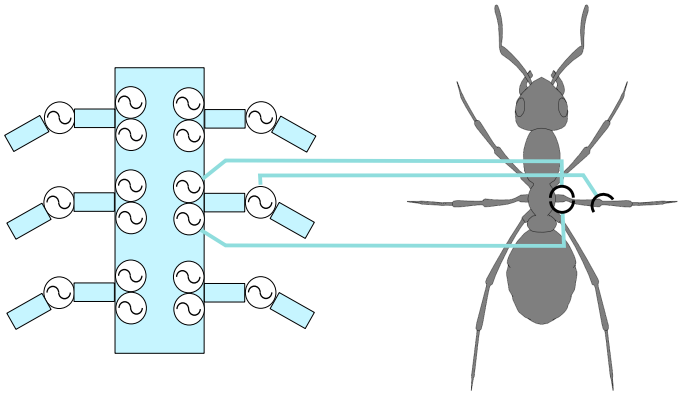


Figure 5.2: *The mapping between the oscillators in the CPG controller and specific joint DOFs. The unit oscillator is transformed by the operators into individual channels (DOFs). This distributed model, similar to the control machinery in real insects, has the potential to reduce the design complexity and computational demands.*

- the focus of this work is to design a locomotion controller, which is more closely related to the output from neuron control than to its biophysical structures.
- the exact neuron mechanism is not completely understood, and thus it is of limited value to attempt to model with accuracy.
- the inherent property of stability from the limit cycle enhances the stability of the final animation (Liu *et al.*, 2012)

In detail, the CPG in this work is formulated by two components: the *oscillator* and *operators*. In this framework, the CPG is composed of 18 coordinated *Hopf Bifurcation* oscillators. Each DOF is controlled by one oscillator instance. This setup is confirmed by biological experiments where each joint in stick insects is controlled by its own oscillator (Büschges *et al.*, 1995). Without sensory input, the activity of each joint is uncoordinated with that of other joints. The oscillator takes the current angle and velocity values of this DOF as input, and returns the corresponding values for the next simulation step. The operators transform the basic form of unit oscillator to produce a DOF-specific control signal and achieve the coordination between them. A block diagram of the CPG is illustrated in Figure 5.2. Details of the oscillator

and operators will be explained in the following paragraphs.

5.2.2 Oscillator Model

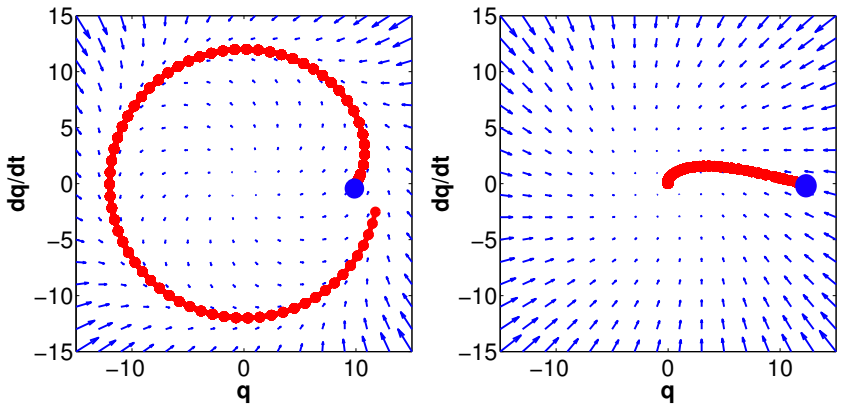
The first challenge in building the CPG is to select the appropriate neuron-oscillator. There are several popular oscillator models, for example Matsuoka (Matsuoka, 1985) and Van Der Pol (Dutra *et al.*, 2003). Here a dynamics system, mathematically defined as *Hopf bifurcation*, is selected (Chung and Dorothy, 2010). Its Ordinary Differential Equation (ODE) is given in Equation 5.1 and its phase plot is given in Figure 5.3:

$$\begin{pmatrix} \dot{\mathbf{q}} \\ \ddot{\mathbf{q}} \end{pmatrix} = \begin{bmatrix} -\lambda(\frac{\mathbf{q}^2+\dot{\mathbf{q}}^2}{\rho^2} - \sigma) & -\omega(t) \\ \omega(t) & -\lambda(\frac{\mathbf{q}^2+\dot{\mathbf{q}}^2}{\rho^2} - \sigma) \end{bmatrix} \begin{pmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{pmatrix} \tag{5.1}$$

$$Equivalently, \dot{\mathbf{Q}} = \mathbf{f}(\mathbf{Q}; \rho; \omega; \sigma), with \ \mathbf{Q} = (\mathbf{q}, \dot{\mathbf{q}})^T$$

Equation 5.1 defines a limit cycle of the *Hopf oscillator* with a radius $\rho > 0$. $\mathbf{q}$ denotes the vector containing the joint angles for all DOFs. In the mathematical theory of bifurcations, a *Hopf Bifurcation* (also sometimes called Poincaré-Andronov-Hopf bifurcation) refers to the local birth of a periodic solution (self-excited oscillation) from a equilibrium point when a parameter (here referred to as σ) passes a critical value (Marsden and McCracken, 1976). The bifurcation parameter σ in Equation 5.1 can switch from 1 to -1 so that the stable limit cycle is converted into a globally stable equilibrium point (Steven, 1994) (Figure. 5.3). The parameter ω determines the oscillation frequency of the limit cycle. The parameter λ denotes the convergence rate to the circular limit cycle or an attraction point.

The nonlinear oscillator presented in Equation 5.1 has also been successfully applied in actuating a flying robot (Chung and Dorothy, 2010). Although the concept of the *Hopf bifurcation* is similar, the sensory information is passed into their controller as a separate component while the controller proposed here reacts to the sensory feedback by adjusting the internal parameters. Compared with other oscillator choices, it is chosen because of its following merits:



(a) *Limit Cycle* **(b)** *Attraction Point*

Figure 5.3: *The phase plot of Equation 5.1 can switch between the limit cycle ($\sigma = 1$) and attraction point ($\sigma = -1$). $\mathbf{q}$ denotes the vector containing the joint angles for all DOFs. (a) Starting from a random position, the output $\mathbf{q}$ is able to converge to a limit cycle, thus ensuring that the character can recover from external perturbations and continue stable locomotion. (b) When σ switches to -1, the phase plot converts itself into an attraction point, thus allowing characters to maintain a static pose or perform discrete movements.*

- Symmetry

  Unlike other popular oscillator models the phase plot of this nonlinear oscillator is symmetric (Figure 5.3), indicating that the phase coordination can be represented as a *phase shifting* operator (this will be explained later).

- Controllable Switch Between Periodic and Discrete Motion

  With the assistance of the Hopf bifurcation, a unified representation of periodic and discrete motion can be given, and this further simplifies the control framework.

- Fast Inhibition of Oscillation by Hopf Bifurcation

  The rapid convergence of this oscillator ensures a stable response against external perturbations and ensures a smooth transition between periodic and discrete motions.

A limitation of this oscillator is the simplicity of the generated signal; however, since the control parameters in Equation 5.1 are constantly updated during walking, the output profiles of joint angles demonstrate a high level of complexity. The same oscillator is also used in producing and controlling

stable flapping flight patterns, which require fast adaptation and agility in order to meet more challenging aerodynamics compared with terrestrial locomotion (Chung and Dorothy, 2010). Published research also shows that modulation of sinusoidal patterns can be implemented to successfully drive humanoid robots (Morimoto *et al.*, 2008). The final animation demonstrates that the modulation of this oscillator is sufficient for the less-developed locomotion pattern of multi-legged insects.

5.2.3 Operators

One of the main contributions of the proposed controller is the design of operators which transform the basic form of unit oscillator into a DOF-specific control signal. The operator corresponds to an individual DOF and maps the basic control signal to each DOF. Alexander (1984) proposed the *Dynamic Similarity Hypothesis*, stating that two systems of moving bodies are described as dynamically similar if the motion of one can be made identical to that of the other by 'multiplying all time intervals by another constant factor, all linear dimensions by some constant factor, and all forces by a third constant factor '. Despite the differences in body form between insects and other animals (bipeds, quadrupeds etc), researchers have shown that they share similarities in force production, stride frequency and mechanical energy production (Full and Tu, 1990).

The choices of operators are vital to the final result and determines the usability of the proposed controller. Inspired by observations of data obtained from human motion (Liu *et al.*, 2012), the following operators are proposed:

Phase Shifting An advantage of choosing the nonlinear system in Equation 5.1 is its symmetry when the phase plot is a limit cycle. Because of this symmetry, this dynamics system can introduce a rotational matrix $R(\Delta) = \begin{pmatrix} cos(\Delta) & -sin(\Delta) \\ sin(\Delta) & cos(\Delta) \end{pmatrix}$ to project the motion of different joints onto the different positions on a same phase plot:

$$\mathbf{f}(\mathbf{R}(\Delta)\mathbf{Q}; \rho; \omega; \sigma) = \mathbf{R}(\Delta)\mathbf{f}(\mathbf{Q}; \rho; \omega; \sigma) \tag{5.2}$$

Δ denotes the difference of two joint angles on the phase plot. For example, the two tripod groups (L1R2L3 and R1L2R3) demonstrate a phase lag of π at any given moment during a walking cycle. By computing the joint values on one tripod and converting it to the values on another tripod with the *Phase Shifting* operator, the computations for the nonlinear oscillator reduce by half.

Energy Scaling This operator can adjust the amplitude of individual channels separately.

$$\mathbf{f}(g\mathbf{Q};\rho;\omega;\sigma) = g\mathbf{f}(\mathbf{Q};\rho/g;\omega;\sigma) \tag{5.3}$$

g denotes the ratio between the amplitude of the standard oscillator and the transformed one. It is straightforward to shift the energy of the whole system by changing the amplitude of the unit oscillator. However, in some cases, it is necessary to adjust the amplitude of the selected DOFs only. In this situation, DOFs can be separated into different groups and their amplitude can be independently adjusted by applying the energy scaling operator on individual groups of DOFs.

Offsetting The unit oscillator produces a signal with the average value of zero. This operator can shift the average of the oscillator signal from zero to a target value:

$$\mathbf{f}(\mathbf{Q}-\bar{\mathbf{q}};\rho;\omega;\sigma) = \mathbf{f}(\mathbf{Q};\rho;\omega;\sigma) - \bar{\mathbf{q}} \tag{5.4}$$

$\bar{\mathbf{q}}$ denotes the offsetting value between average of the standard oscillator and the target one. For periodic motion (normal walking pattern), the oscillator is under the mode of *limit cycle* and the offset value denotes the average value of individual DOFs. For discrete motion (reaching for objects), the oscillator is under the mode of *attraction point* and the offset value denotes the target value of individual DOFs.

Although the CPG is known to behave in a self-sustained fashion, sensory feedback is also crucial in altering motion patterns when adapting to the environment or perturbations. The operators introduced here fulfil this adaptation. The values of the parameters contained in these operators will be adjusted by

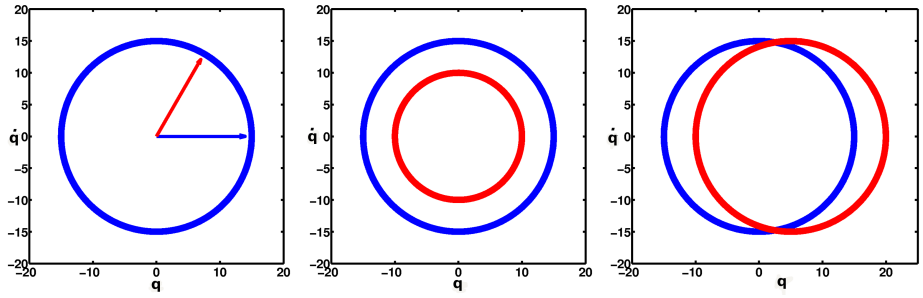


Figure 5.4: *Three types of operators introduced in this work: (a) Phase Shifting (b) Energy Scaling (c) Offsetting. The original phase space (in blue) is transformed into the new phase space (in red) by applying the respective operators. The original phase space is generated by integrating Equation 5.1 and plotting the output* $(q_o, \dot{q}_o)$*. The transformed phase space is generated by individually applying the operators to the initial condition* $(q_i, \dot{q}_i)$*, integrating Equation 5.1 and plotting the transformed output* $(q_o, \dot{q}_o)$*.*

the CLUT (discussed in Chapter 6), according to the input from the high-level commands from the TPE.

5.2.4 Sensors and Actuators

Sensors The physics model in this work has two types of sensor: positional and force sensors.

Positional sensors are located at each joint to measure the rotational angles and velocities. The angle values are computed with an analytical Inverse Kinematics solver (Tolani *et al.*, 2000). The angular velocity for each DOF is computed as the difference of the rotational angles with respect to the time step:

$$\dot{\mathbf{q}} = \frac{\mathbf{q}_{n+1} - \mathbf{q}_n}{\Delta t} \tag{5.5}$$

The collected data are fed into the CPG controller (Equation 5.1) in order to produce the angles and velocities $(\mathbf{q}, \dot{\mathbf{q}})$ for the next simulation step.

Force sensors measure the external forces applied to the character at the positions of end-effectors (foot) and the COM. The ground reaction force $\mathbf{F}_i$ on the i^{th} foot is measured as the opposite to the active force $\mathbf{F}'_i$ from the character (*Newton's Third Law*). The variable $i = 1 \cdots 6$ refers to the leg

$L1, L2, L3, R1, R2, R3$ respectively. The active force $\mathbf{F}'_i$ acting against the ground by end–effectors is computed in a representation of joint torques by *Virtual Control Model* (Pratt *et al.*, 1997):

$$\mathbf{F}_i = -\mathbf{F}'_i = -(\mathbf{J}^T)^{-1} \begin{bmatrix} \tau_\alpha \\ \tau_\beta \\ \tau_\gamma \end{bmatrix} \tag{5.6}$$

$\tau_{\alpha,\beta,\gamma}$ is computed using Equation 5.7. The Jacobian $\mathbf{J}$ can be derived from the relationship between joint velocity and end effector velocity (Craig, 2004). The measured forces $\mathbf{F}_i$ are passed as the external force $\mathbf{F}$ in Equation B.1 in the Appendix B during the process of physics simulation. The readers could refer to the Appendix B for a detailed design of the physics simulation.

Figure 5.5 shows the ground contact force applied on the left middle leg $L2$ when the character is travelling on an uneven terrain. When the leg is in the swing mode, the contact force is zero. When the leg is in the stance mode, the contact force behaves as a bell-shaped function. This bell-shaped function means that the contact force starts from zero when the foot contact is initialised, increases to the peak value in the middle of the stance phase and decreases to zero by the end of the stance phase. This fact is in accordance with the strategy proposed in Chapter 7. The proposed strategy triggers the swing mode of a leg if the force decreases to zero, which means that this leg reaches the end of the stance phase.

Actuators Once the joint angle and angular velocity for the next frame are obtained, each joint is propelled by a Proportional Derivative (PD) controller (Yin *et al.*, 2007):

$$\tau = k_p(\mathbf{q}_{n+1} - \mathbf{q}_n) + k_d\dot{\mathbf{q}}_n \tag{5.7}$$

where $\mathbf{q}, \dot{\mathbf{q}}$ are the angle value and velocity, respectively, for the corresponding DOF with their subscripts denoting the corresponding simulation step. k_p, k_d are the gain and damping coefficients of this controller, respectively.

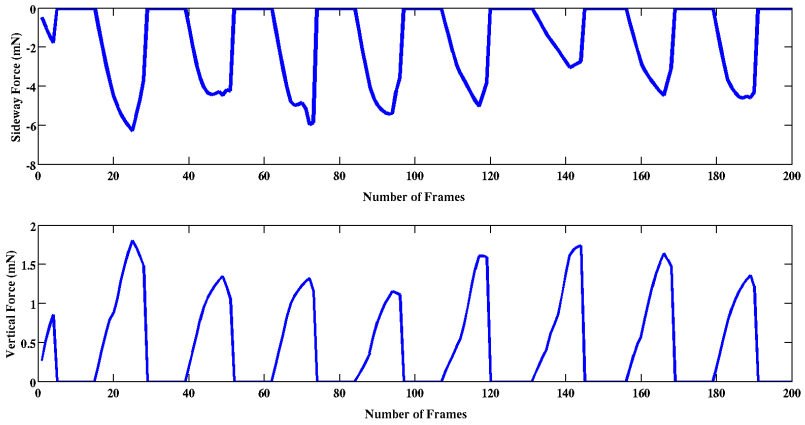


Figure 5.5: *Ground contact force applied on the left middle leg ($L2$) when the character is travelling on an uneven terrain. The top figure is the force applied in the sideways direction ($-y$), while the bottom figure is the force applied in the vertical direction ($+z$). The contact forces behave as a bell-shaped function in both directions when the leg is in stance. The contact forces are zero when the leg is swinging.*

5.3 CMA Optimisation Strategy

A major challenge in designing a controller is how to determine the optimal control parameters. Manual tuning is only possible for systems with a limited number of parameters (Yin *et al.*, 2007). The parameters of the proposed CPG controller include operator parameters ($\Delta, g, \bar{\mathbf{q}}$) and PD controller parameters (k_p, k_d) for each DOF. Assuming the parameters on different sides of the body to be symmetrical (g, k_p, k_d are the same while $\Delta, \bar{\mathbf{q}}$ are opposite values), the total number of parameters is 45.

There is not, to date, a well-established method of designing a CPG controller, and different approaches including hand-coding, learning and optimisation algorithm, *guessing* from dynamical system theory (Buchli *et al.*, 2006; Ijspeert, 2008) have been explored. Optimisation has been demonstrated as a useful technique in designing motion controllers for virtual characters. Given the fact that objective functions are generally non-convex and high-dimensional for the problem of character animation, designing an appropriate objective function and choosing a robust optimisation strategy are both crucial (Wang *et al.*, 2009).

Here the Covariance Matrix Adaptation (CMA) is employed to set the pa-

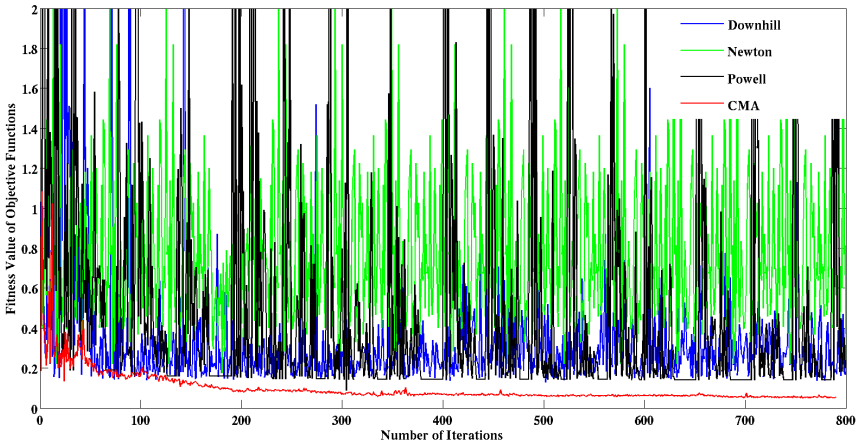


Figure 5.6: *Comparison of four different optimisation algorithms in setting the value of the CPG control parameters. The vertical axis is the value of the weighted-sum fitness function (Equation 5.22). Standard methods, including a downhill simplex method (Wright, 1996), a truncated Newton method (Nash, 1984), a modified Powell method (Powell, 1964), are only able to find a local minimum or fail to converge. In comparison, the CMA method used in this thesis produces better results in successful convergence and smaller fitness value of the objective function.*

rameters in the CPG automatically. CMA is an advanced evolutionary algorithm especially suited for non-linear non-convex optimisation problems, and it has been applied successfully to design controllers in character animation (Wampler and Popović, 2009; Tan *et al.*, 2011; Wang *et al.*, 2012). Standard gradient methods, such as Newton-based methods, will generally fall into local minima and fail to reach global optimal solutions. Figure 5.6 presents a comparison of four different optimisation algorithms in minimising the same weighted-sum fitness function (Equation 5.22). The demonstrated methods include a downhill simplex method (Wright, 1996), a truncated Newton method (Nash, 1984), a modified Powell method (Powell, 1964) and the CMA method. The result shows that the CMA method outperforms other options in successful convergence and smaller fitness value of the objective function. A brief introduction on the CMA is now given in the following section and readers can refer to Hansen and Kern (2004) for a complete tutorial on the CMA.

5.3.1 Introduction to the CMA

The CMA (Hansen, 2006) handles general constraints of space-time optimisation on non-differential variables with high dimensions. The optimisation process is initialised with the mean vector $\mathbf{m}_0$ and the standard deviation vector σ_{CMA}. New search points are sampled with a normal distribution:

$$\mathbf{x} \sim \mathbf{m} + \sigma N_i(0, \mathbf{C}), for\ i = 1, ..., \lambda \tag{5.8}$$

The *mean* vector $\mathbf{m} \in R^n$ represents the favourite solution, and $\sigma \in R^+$ controls the step length. The covariance matrix $\mathbf{C} \in R^{n \times n}$ determines the shape of the distribution ellipsoid. This covariance matrix is updated by the principle that this adaptation increases the likelihood of successful steps to appear again.

λ_{CMA} offspring are sampled from this distribution and are evaluated individually by a fitness function f. By sorting the entire population in order of the lowest associated values of f, the lowest μ samples, defined as *elites*, are selected:

$$f(\mathbf{x}_1) \leqslant \cdots \leqslant f(\mathbf{x}_\mu) \leqslant \cdots \leqslant f(\mathbf{x}_{\lambda_{CMA}}) \tag{5.9}$$

Therefore, the new mean for the next generation is updated from a weighted sum of these μ elites. The routine for defining weights ω_i is adopted from Wampler and Popović (2009):

$$\begin{aligned} \mathbf{m}_{i+1} &= \sum_{i=1}^{\mu} \omega_i \mathbf{x}_i \\ \omega_i &= \frac{ln(\mu + 1) - ln(i)}{(\mu + 1) - \sum_{k=1}^{\mu} ln(k)} \end{aligned} \tag{5.10}$$

The covariance matrix is updated as a weighted sum of the previous Covariance Matrix, Evolution Path and stepsize control:

$$\mathbf{C}^{g+1} = (1 - c_1 - c_\mu)\mathbf{C}^g + c_1\mathbf{C}_1 + c_\mu\mathbf{C}_\mu \tag{5.11}$$

$\mathbf{C}^g$ refers to the covariance matrix for g^{th} generation. $\mathbf{C}_1, \mathbf{C}_\mu$ are the Rank-One and Rank-μ update respectively. This update strategy is able to reuse

prior information and takes control of step size for the next generation.

The Rank-One update $\mathbf{C}_1$ uses the evolution path $\mathbf{p}$ and reduces the number of function evaluations to adapt to a straight ridge from $O(n^2)$ to $O(n)$ (Hansen *et al.*, 2003).

$$\mathbf{p}_c^{g+1} = (1 - C_c)\mathbf{p}_c^g + \sqrt{C_c(2 - C_c)\mu}\frac{\mathbf{m}^{g+1} - \mathbf{m}^g}{\sigma^g} \tag{5.12}$$

$$\mathbf{C}_1 = \mathbf{p}_c^{g+1}(\mathbf{p}_c^{g+1})^T \tag{5.13}$$

The constant C_c is chosen so that $\mathbf{p}_c^{g+1}$ fits the normal distribution $N_i(0, \mathbf{C})$.

The Rank-μ update $\mathbf{C}_\mu$ extends the update rule for large population sizes λ using $\mu > 1$ vectors to update $\mathbf{C}$ at each generation. The matrix

$$\mathbf{C}_\mu = \sum_{i=1}^{\mu} \omega_i \mathbf{y}_{i:\lambda_{CMA}}^{g+1}(\mathbf{y}_{i:\lambda_{CMA}}^{g+1})^T \tag{5.14}$$

$$\mathbf{y}_{i:\lambda_{CMA}}^{g+1} = \frac{\mathbf{x}_{i:\lambda}^g - \mathbf{m}^g}{\sigma^g} \tag{5.15}$$

computes a weighted mean of the outer products of the best μ steps and increases the possible learning rate in large populations. It can reduce the number of necessary generations roughly from $O(n^2)$ to $O(n)$ (Hansen *et al.*, 2003).

New individuals for the next generation are re-sampled with updated mean and covariance matrix. By repeating this process, individuals are expected to converge to the optimal value. There are two difficulties in employing such an optimisation strategy, one is to set the initial value, and the other is to design the fitness functions.

5.3.2 Setting Up Initial Values

The initial guess of oscillator parameters ($\Delta, g, \bar{\mathbf{q}}$) is taken from animation cycles created by animators which track the motion of real ants. For example, because the leg groups (L1R2L3 and R1L2R3) alternate in a temporal sequence, the phase shifting operator Δ is set to zero for the left tripod L1R2L3 and π for the right tripod R1L2R3. The values of PD controller parameters

(k_p, k_d) are initialised by following design guidelines on reaction force $\mathbf{F}$ on individual foot:

$$F_{iz} = mg/3 \tag{5.16}$$

$$F_{1y} = F_{3y} = -\frac{F_{5y}}{2} \tag{5.17}$$

$$F_{4y} = F_{6y} = -\frac{F_{2y}}{2} \tag{5.18}$$

The first subscript $i = (1 \cdots 6)$ refers to the leg $(L1, L2, L3, R1, R2, R3)$ respectively. The second subscript (x, y, z) denotes the force direction. Here the same proportion of gravity is spread across the three standing legs based on experimental observations (Kram *et al.*, 1997). The horizontal forces are designed to minimise the movement on the frontal plane.

5.3.3 Objective Function Design

The optimal solution is defined as the one which simultaneously minimises the following objectives:

Metabolic Consumption To quantify the energy efficiency of a specific gait, the energy cost corresponding to one unit of character mass travelling one unit of distance is calculated as:

$$E_m = \frac{1}{ms} \sum_i \sum_j ||\tau_{ij} \delta\theta_{ij}||^2 \tag{5.19}$$

where m is the mass of the insect, s is the travelling distance, $\tau_{ij}, \delta\theta_{ij}$ are respectively the torque and angle differences for j^{th} DOF on i^{th} frame.

Torque Limit Soft constraints on joint torque are set up to minimise the implausible actuation.

$$E_t = \sum_i \sum_j ||\frac{\tau_{ij} - \tau_j^*}{\tau_j^*}||^2, \text{if } ||\tau_{ij}|| \geqslant \tau_j^* \tag{5.20}$$

where τ_j^* is the torque threshold for j^{th} DOF on a single leg. The same threshold is applied for DOFs at the same location across different legs. Here the term $||\frac{\tau_{ij}-\tau_j^*}{\tau_j^*}||$ is used instead of $||\tau_{ij} - \tau_j^*||$, in order to normalise the magnitude differences of different DOFs. The same principle applies when computing the position and direction deviations in the following paragraph.

Position and Direction Deviations Soft constraints are also set up to reduce the deviations in the sagittal and coronal planes. Large deviations in these two planes are assumed to not contribute to the overall movement of the insects and to have increased the energy cost. Due to the periodic torque applied by supporting legs, the insect's main body rotates at a certain frequency. A penalty is introduced to maintain the head rotation to the heading direction. Large head deviations have the potential to interfere with the visual senses that are necessary for an insect to move through a challenging environment. This objective function is defined as:

$$E_d = \sum_i (||\frac{y_i - y^*}{y^*}||^2 + ||\frac{z_i - z^*}{z^*}||^2 + ||\frac{\phi_i - \phi_i^*}{\phi_i^*}||^2), \tag{5.21}$$

where y_i, z_i are current y/z coordinates of the COM and y^*, z^* are the y/z coordinates of the COM when insects remain in a static pose. ϕ_i is the current angle between the heading direction and $+x$ axis, ϕ_i^* is the desired heading direction for i^{th} frame. Each component is normalised by the thresholds y^*, z^*, ϕ_i^* respectively.

However, the aforementioned objective functions may conflict with each other and there is no unique solution which minimises all of the objectives simultaneously. The weighted sum method transforms the multi-objective optimisation problem into a single-objective optimisation problem:

$$E_{total} = \omega_m E_m + \omega_t E_t + \omega_d E_d \tag{5.22}$$

The formulation of the weighted sum introduces an additional problem: how to properly set the weight values? The magnitude of each objective function differentiates from each other, thus making it difficult for users to choose the weight values. Therefore, the normalisation of the objective functions is

critical in removing the effects of different magnitudes.

Some of the normalisation methods include (Grodzevich and Romanko, 2006):

- *normalise by the magnitude of the objective function at the initial point*
- *normalise by the minimum of the objective functions*
- *normalise by the differences of optimal function values at the* Nadir *and* Utopia *points that give the length of the intervals where the optimal objective functions vary within the* Pareto *optimal set*

The first two methods, either by initial or optimal points, can often lead to distorted scaling since each of them only represents a single point in the whole optimisation landscape. This work chooses the third method as the normalisation approach of the objective function. The justification and implementation are presented in the following paragraphs.

Pareto Optimality A multi-objective optimisation problem can be defined in a general form as follows:

$$\begin{aligned} &\min \{f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_k(\mathbf{x})\} \\ &s.t.\ \mathbf{x} \in \Omega \\ &Z = \{z \in R^k : z = (f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_k(\mathbf{x}))\} \end{aligned} \tag{5.23}$$

where Ω is the feasible space of $\mathbf{x}$. A solution $\mathbf{x}^*$ is *Pareto optimal* if there exists no another $\mathbf{x} \in \Omega$ such that $f_i(\mathbf{x}) \leq f_i(\mathbf{x}^*)\ \forall i = 1 \cdots k$ and $f_j(\mathbf{x}) < f_j(\mathbf{x}^*)$ for at least one index j (Grodzevich and Romanko, 2006). In other words, there exists no other feasible solution that improves the value of at least one objective function without deteriorating any other objectives. The set of Pareto optimal solutions forms a Pareto optimal set.

Utopia Point The *Utopia* point z^U defines the lower bound of the Pareto optimal set by:

$$z_i^U = f_i(\mathbf{x}^{[i]}) \text{ where } z_i^U = \text{argmin}_{\mathbf{x}}\{f_i(\mathbf{x}) : \mathbf{x} \in \Omega\} \tag{5.24}$$

This is to find the solution $\mathbf{x}^{[i]}$ with the minimum fitness value for each objective function in an independent fashion. However, the *Utopia* point is generally not feasible because the objective functions may conflict with each other and the solutions $\mathbf{x}^{[i]}$ are different for each objective. This conflict is evidenced in Table 5.1. The three objective functions E_m, E_t, E_d require three different solutions $\mathbf{x}$ to achieve the lowest value respectively.

	Utopia Point $\mathbf{x}^{[i]} = (\Delta, \bar{\mathbf{q}}, g, k_p, k_d)_j, j = 1 \cdots 9$
Metabolic Consumption E_m	(-0.902, -0.790, 1.301, 0.812, -0.935, $\cdots$)
Torque Limit E_t	(-0.852, -0.722, 1.165, 0.538, -0.653, $\cdots$)
Deviation Error E_d	(-0.848, -0.691, 1.190, 0.421, -0.967, $\cdots$)
Unit: $(\Delta, \bar{\mathbf{q}} - rad); (k_p - Nm/rad); (k_d - Nms/rad)$, ($g$ is a ratio with no units);)	

Table 5.1: *Utopia points for different objective functions. Utopia points are different for different objective functions, which means there exists conflict between these objective functions and the utopia points cannot be satisfied simultaneously.*

Nadir Point The Nadir point z^N defines the upper bound of the Pareto optimal set by:

$$z_i^N = \max_{1 \leq j \leq k} (f_i(\mathbf{x}^{[j]})), \ \forall i = 1, \cdots k \tag{5.25}$$

This is to select the solution with the maximum fitness value for each objective function from the Pareto optimal set. In practice, the *Nadir* objective vector can only be approximated as, typically, the whole Pareto optimal set is unknown (Grodzevich and Romanko, 2006; Kim and De Weck, 2006). The *Nadir* point can be approximated by iterating all the optimal solutions $\mathbf{x}^{[i]}$ for each objective function (Marler and Arora, 2004; Kim and De Weck, 2006). Table 5.2 lists the *Nadir* points for the three objective functions in this work. The result shows that the the objective functions E_t, E_d share the same solution of the *Nadir* point and differentiate from E_m.

Table 5.3 lists the fitness values of the objective functions at both *Nadir* and *Utopia* points. To bring all the objective functions to a consistent magnitude, each of them will be bounded by

$$0 \leq \frac{f_i - z_i^U}{z_i^N - z_i^U} \leq 1 \tag{5.26}$$

	Nadir Point $\mathbf{x}^{[i]} = (\Delta, \bar{\mathbf{q}}, g, k_p, k_d)_j, j = 1 \cdots 9$
Metabolic Consumption E_m	(-0.848, -0.691, 1.190, 0.421, -0.967, $\cdots$)
Torque Limit E_t	(-0.902, -0.790, 1.301, 0.812, -0.935, $\cdots$)
Deviation Error E_d	(-0.902, -0.790, 1.301, 0.812, -0.935, $\cdots$)
Unit: $(\Delta, \bar{\mathbf{q}} - rad)$; $(k_p - Nm/rad)$; $(k_d - Nms/rad)$, (g is a ratio with no units);)	

Table 5.2: *Nadir points for different objective functions. The Nadir points are approximated by iterating all the optimal solutions* $\mathbf{x}^{[i]}$ *for each objective function (Grodzevich and Romanko, 2006; Kim and De Weck, 2006).*

Figure 5.7 presents the comparison between the non-normalised and normalised fitness value of the objective functions. Without the normalisation, three objective functions demonstrate significant differences in the magnitude (Figure 5.7a). After normalising each objective function, the fitness value falls between the range of [0, 1] except for a few outliers at the very beginning of the optimisation (Figure 5.7b). The fitness values at these few solutions are greater than one after normalisation. This is due to the fact that the *Nadir* points are the approximate, rather than the accurate, upper bounds of the *Pareto* optimal set.

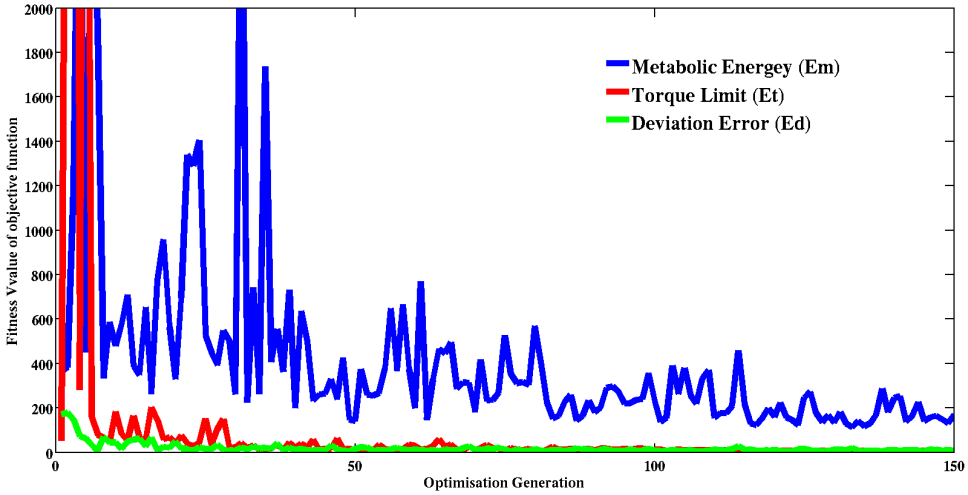
	Nadir Point	Utopia Point
Metabolic Consumption E_m	3040.093	81.562
Torque Limit E_t	4524.422	1.673
Deviation Error E_d	392.434	3.454
Unit: (E_m - $J/(kg \cdot s)$); (E_t, E_d are ratios with no units)		

Table 5.3: *Fitness value of the objective function at both* Nadir *and* Utopia *points.*

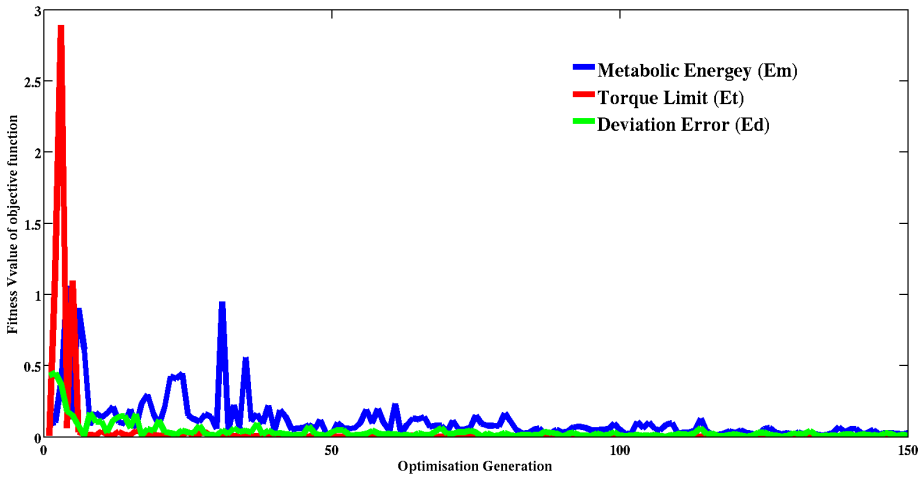
With the consistent magnitude for all objective functions E_m, E_t, E_d , the weight values $\omega_m, \omega_t, \omega_d$ for each objective function solely reflect their importance in determining the final result of the optimisation problem. In the method of the weighted sum optimisation, it is a common strategy to choose the sum of the weights to be one:

$$\omega_m + \omega_t + \omega_d = 1 \tag{5.27}$$

that degenerates the dimension of the weight space to two: ω_m, ω_t are chosen to be independent variables while ω_d is computed as $\omega_d = 1 - (\omega_m + \omega_t)$. To



(a) *Non-normalised fitness value of the objective functions.*



(b) *Normalised fitness value of the objective functions.*

Figure 5.7: *Comparison between non-normalised and normalised fitness value of the objective functions. After normalising each objective function with* Nadir *and* Utopia *points, the fitness values of three functions fall within the range of [0, 1]. There are still a few solutions whose fitness values are greater than one. This is caused by the fact that the* Nadir *points used here are not the accurate maximum values for the* Pareto *set.*

choose the appropriate weight values, ω_m, ω_t are linearly sampled between the range of [0, 1] while satisfying the constraint that $\omega_m + \omega_t \leq 1$ (Figure 5.8).

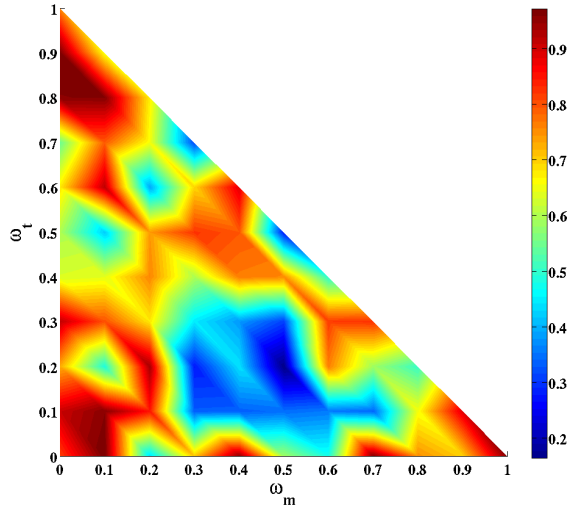


Figure 5.8: *The best optimal value of the weighted sum objective function, with respect to different weight choices. This map is drawn by linearly sampling the weights ω_m, ω_t and calculating the optimal value at each sample. The weight values $\omega_m, \omega_t, \omega_d$ are selected at the position of the best optimal value of the objective function.*

Figure 5.8 shows that the best case of the objective function occurs at the position: $\omega_m = 0.5, \omega_t = 0.2, \omega_d = 0.3$. ω_m is set to be larger than the other two because of its significance in determining the outcome of this optimisation problem. The main priority of the locomotion controller is for the insect to travel as far as possible at the price of least energy, which is equivalent to the goal of maximising s and minimising E_m (Equation 5.19). For the other two components E_t, E_d, the weights are set to improve visual performance by reducing the unnecessary deviations in the coronal plane.

5.4 Results & Discussions

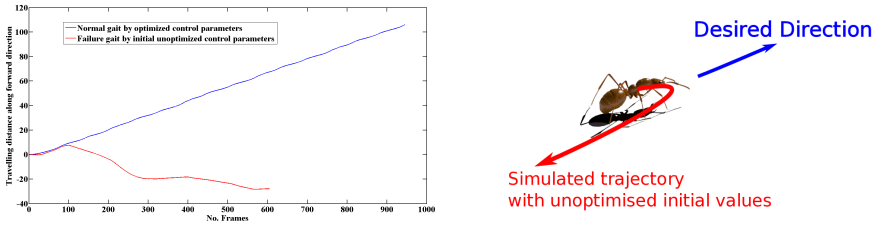
5.4.1 Implementation

The articulated character (ant) is comprised of 13 links (one for main body and two for each leg) and 18 internal DOFs. There is no check for collisions between links during simulation. A list of parameters of the ant model is attached in the Appendix D. The simulation loop is solved by ODE45 with an integration step of 0.001s. It is found that the time step is crucial for the proposed implementation. Larger time step will result in failure of the oscillator convergence. Concerning the computational performance, one walking cycle takes about 500 simulation steps (or 0.5s).

5.4.2 Simulated Results

Normal Gait The control parameters of the CPG are initialised with the basic rules in Section 5.3.2. However, the rules are designed to provide an approximate value, instead of an accurate one. For example, the gravitational force is equally distributed across three legs in the same tripod group. However, this force distribution should be constantly changing in both real insects (Full and Tu, 1991) and simulated characters (Figure 5.5). Therefore, the initialised parameters may not produce the locomotion in a stable and desired way, but could serve as the initial values for the CMA optimisation. Figure 5.9 plots the comparison of the produced motion between optimised control parameters and unoptimised initial control parameters. With the initial values, the character is able to move forward for a short interval and then move backward in an unpredictable fashion until its COM touches ground, terminating the simulation. With the optimised control parameters, the character performs the stable gait of forward walking.

The optimisation strategy, CMA, is introduced to find the optimal control solution. Important parameters used in this implementation include 100 individuals per generation, 30 are selected for re-sampling in the next generation. It is found that increasing the number of individuals is able to avoid the local minima and, after testing, a generation of 100 individuals is chosen. The



(a) *Travelling Distance in x axis* **(b)** *Sketch of the failure case*

Figure 5.9: *The comparison of the produced motion between optimised control parameters and unoptimised initial control parameters. (a) Travelling distance in x axis. $+x$ is the desired locomotion direction. (b) Sketch of the comparison between the stable locomotion and the failure one. With optimised CPG parameters, the character is able to travel forward at a stable velocity (solid line in blue). In comparison, the character moves backward, as opposite to the desired direction (Figure (b)), with unoptimised initialised parameters (solid line in red). At the end of the simulation in failure, the COM touches the ground and the simulation terminates.*

fitness function (Equation 5.22) is evaluated every time the insects take 100 steps. The optimisation is set to terminate after 1000 generations, which takes approximately 30 hours on a six-core PC. Parallel computing is implemented for each individual standing leg. In practice, the optimisation takes around 200 generations to converge (Figure 5.10).

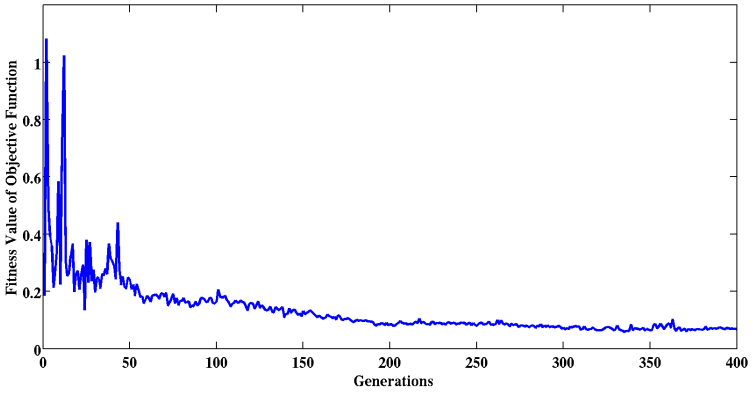


Figure 5.10: *The result of optimising the parameters in CPG controller. The vertical axis represents the best fitness function value for each generation.*

Stability The *stability* of a physics controller refers to its capability to recover from external perturbations. Figure 5.11 plots the angle value of the coxa-femur joint in the left middle leg in the case of applying external forces.

The character demonstrates a perturbed gait after applying the force, but eventually converges back to its original stable gait. This recovery is accomplished with the inherent property of convergence of the *Hopf Bifurcation* in the mode of limit cycle. The CPG parameters remain unchanged.

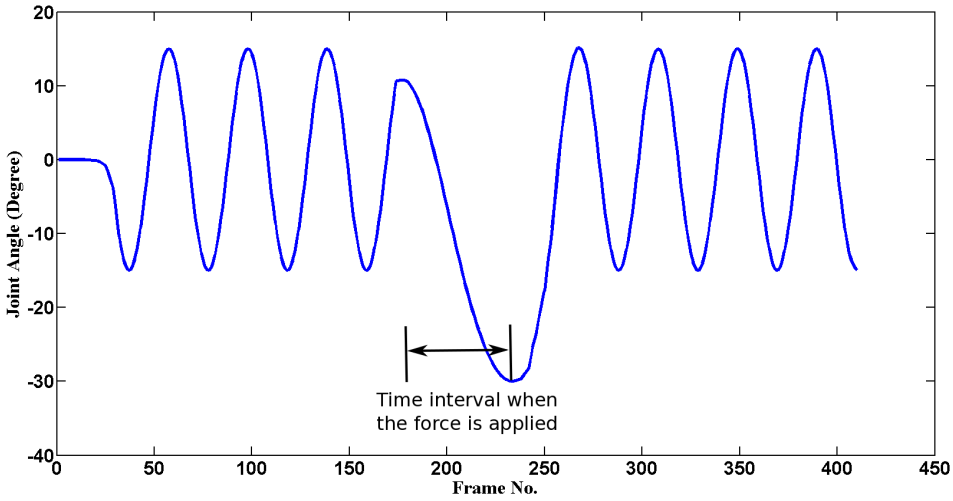


Figure 5.11: *The angle value of coxa-femur joint in the left middle leg when applying external forces. It shows that the character is capable of maintaining its stable gait after applying the perturbations, which confirms the stability of the CPG controller. This feature of stability is achieved with the convergence property of the* Hopf Bifurcation *and requires no further strategic adjustment from the controller.*

This chapter is concerned with the stability of the CPG controller only, with no additional assistance from other components of the framework, such as the TPE or the CLUT. When the character encounters the perturbations, additional strategies can be introduced. For example, the character increases the area of the supporting triangle in the case of a continuous external force (Figure 4.13). To effectively and objectively assess the stability of the CPG controller, these auxiliary strategies are not employed here.

Given a fixed time interval for which the force is applied, the magnitude of a force which a character can resist is determined by several factors. More specifically, a character is said to successfully resist a force if the character is able to maintain the specified locomotion speed after dismissing the external force.

The scientific research method, *control variable* (Salkind, 2010), is used

here to independently assess the relationship between the force magnitude and individual determinants. When studying the effects of a specific determinant, other determinants remain constant. The determinants include:

- The force direction. The force direction is measured as the angle of counter-clockwise rotation about $+z$ axis, starting from zero ($+x$ axis) and with a range of [0, 360°] (Figure 5.12).
- The initial timing. The timing when the external force is initialised is within [0, T], where T is the walking period. The force is applied for half of the walking period.
- The locomotion speed. The increase in speed leads to an increase in stride length and the distance between two supporting triangles. When transitioning between two supporting triangles during high speed movement, the character will be supported by no legs (*flying* in the air), decreasing the locomotion stability.

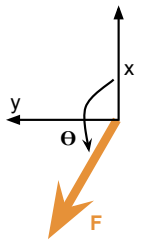


Figure 5.12: *Sketch of the force direction θ*

Table 5.4 gives the best and worst cases of the maximum force that a character is able to resist from different force directions. The forces are applied between the same time range $[\frac{T}{4}, \frac{3T}{4}]$ and the character is walking at the same speed $2cm/s$. This result shows that the character is able to resist larger perturbations from the opposite direction of the locomotion velocity, than from the lateral direction of the body. This is because the force from the lateral direction is more likely to push the COM outside the current supporting triangle.

Table 5.5 gives the best and worst cases of the maximum force that a character is able to resist at different timings. The forces are applied from the same direction and the character is walking at the same speed $2cm/s$. The

	Force Direction	Initial Timing	Speed	Maximum Force
best case	180	$T/4$	2	9.38 x 10^{-3}
worst case	90	$T/4$	2	3.56 x 10^{-3}
Unit: Direction - $^{\circ}$; Initial Timing - $second$; Speed - cm/s; Maximum Force - N				

Table 5.4: *The maximum force that a character is able to resist from different directions. The data of best and worst cases are obtained by linearly sampling the force direction; for each force direction, the force magnitude starts from zero and increases by* $10^{-5}N$ *until the simulation runs into failure.*

result shows that the character is able to resist larger perturbations when its COM is in the middle of a supporting triangle, than during the transition between two supporting triangles. This is because the COM is prevented from entering the next triangle when the force is introduced during the transition between two triangles.

	Force Direction	Initial Timing	Speed	Maximum Force
best case	180	$T/4, 3T/4$	2	6.18 x 10^{-3}
worst case	180	$0, T/2, T$	2	2.05 x 10^{-3}
Unit: Direction - $^{\circ}$; Initial Timing - $second$; Speed - cm/s; Maximum Force - N				

Table 5.5: *The maximum force that a character is able to resist at different timings. The data of the best and worst cases are obtained by choosing different initial timings from a uniform distribution of [0, T]; for each case of initial timings, the force magnitude starts from zero and increases by* $10^{-5}N$ *until the simulation runs into failure.*

Table 5.6 gives the best and worst cases of the maximum force that a character is able to resist at different speeds. The forces are applied from the same direction and within the same time range $[\frac{T}{4}, \frac{3T}{4}]$. The result is in agreement with the daily experiences. Characters are capable of absorbing larger perturbations during slow locomotion, compared with fast locomotion. The best case occurs when the character is at the static position (its speed is zero). In this case, the *Hopf Bifurcation* switches to the mode of *attraction point*, instead of the *limit cycle*.

Uneven Terrain The proposed method is sufficiently robust to adapt to uneven terrain including varying gradients. The force term $\mathbf{F}$ in Equation B1 (in

	Force Direction	Initial Timing	Speed	Maximum Force
best case	90	$T/4$	0	12.46 x 10^{-3}
worst case	90	$T/4$	4.0	2.55 x 10^{-3}
Unit: Direction - $^{\circ}$; Initial Timing - $second$; Speed - cm/s; Maximum Force - N				

Table 5.6: *The maximum force that a character is able to resist at different speeds. The data of the best and worst cases are obtained by linearly sampling the speed; for each speed, the force magnitude starts from zero and increases by* $10^{-5}N$ *until the simulation runs into failure.*

the Appendix B) is replaced as follows:

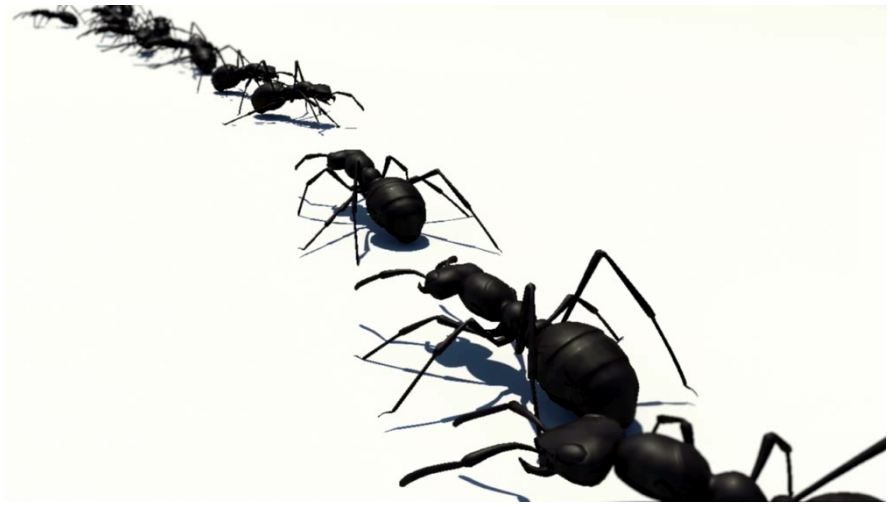
$$\mathbf{F} = \sum_{i=1}^{3} \mathbf{F}_i - \begin{bmatrix} mgsin(\xi) \\ 0 \\ mgcos(\xi) \end{bmatrix} \tag{5.28}$$

where ξ is the gradient angle in radians. By splitting gravitational force into directions parallel and vertical to the gradient plane, ants demonstrate different speeds when walking upslope and downslope. It is found, however, that the adaptation generated by this way appears rigid due to the lack of actuation adaptation of control mechanism.

Frequent Motion Transition The feature of collective social behaviour exhibited by ants results in a high possibility of an ant encountering other individuals, which requires a sudden change in gait. Such a frequent transition between walking and stopping can be achieved by toggling the bifurcation parameter σ in the proposed CPG controller from 1 to -1. This provides flexible high level control by key-framing a single parameter, thus fitting into traditional animation pipeline. The following example of ant *foraging* behaviour demonstrates the advantage of this distinctive feature.

Forage Behaviour Modelling A specific experiment to model ant foraging behaviour is set up to further validate that the introduction of *Hopf bifurcation* enables characters to perform frequent motion transitions. In nature, foraging ants lay chemical trails, known as the *pheromone*, between their nest and food sources, which other ants follow and repeat laying more pheromone to

(a) *Viewed from above.*



(b) *Close-up view.*

Figure 5.13: *Animation screenshot of ant foraging along a predefined chemical trail. Viewed from above, the nest is located at the left end while the food is at the right end.*

strengthen the trail. In so doing, ants converge to a collective route connecting the nest and food.

To model this natural phenomenon, the scene is set up as a narrow strip ($x = [0, 1], y = [-0.01, 0.01]$, unit: m) in the x/y plane. The ant nest is at $[0, 0]$ while the food source is placed at $[1, 0]$. The position ($p = [x, y]$) and orientation (ϕ) of a character are initialised as:

$$p = \begin{bmatrix} x_{min} + (x_{max} - x_{min}) * rand() \\ y_{min} + (y_{max} - y_{min}) * rand() \end{bmatrix} \tag{5.29}$$

$$\phi = 2\pi * rand() \tag{5.30}$$

$$Goal = \begin{cases} nest, \text{ if } rand() \geq 0.5 \\ food, \text{ if } rand() < 0.5 \end{cases} \tag{5.31}$$

where the value *rand()* indicates the pseudo-random values drawn from uniform distributions between 0 and 1. Note that only horizontal movements are considered here. Initial velocities of ants are all set to 0. The *Goal* prop-

erty ensures that ants move towards either their nest or the food source, and loosely constrains the group of ants to move within this narrow strip.

Each ant is able to detect other ants within a sensory radius of $r = 0.02m$ and the interaction between two ants is determined by the following rules:

- When two ants have opposite *Goal* directions, they stop, communicate and then resume moving. This can be simulated by switching the bifurcation parameter $\sigma : 1 \rightarrow -1 \rightarrow 1$.
- When two ants have the same *Goal* directions, they deviate from each other to avoid further collision.

Figure 5.14 plots the coxa-femur joint angle when the character status of the character changes between motion and stop. The transition is controlled purely by manipulating the bifurcation parameter σ. With this particular feature, the sudden changes in motion status can be easily authored by animators.

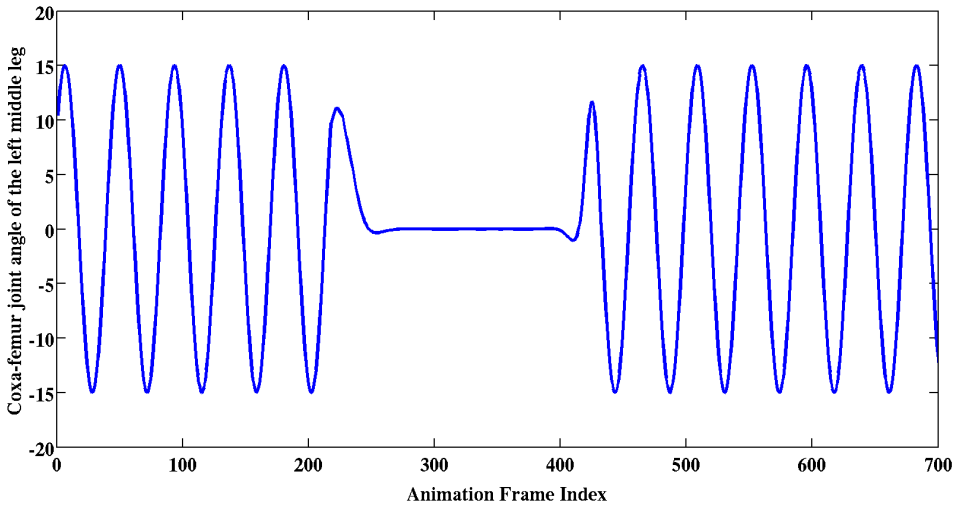


Figure 5.14: *The coxa-femur joint angle of the left middle leg when the character switches between motion and stop. When the bifurcation parameter σ switches to -1, the oscillator will converge to an attraction point and the character will stop. When σ is switched back to 1, the oscillator will converge to the limit cycle and the character will resume walking.*

Discussions The simulation results have demonstrated that the proposed CPG controller is able to generate a stable and natural gait. This work differs

from previous work in a number of aspects:

- Compared with example-based methods (Gibson *et al.*, 2005; Seol *et al.*, 2013), the proposed method does not require setting up a motion capture system and post-processing of captured sequences. Furthermore, contrary to example-based methods, the proposed controller can control individual joints with accuracy and is responsive to external perturbations and changing environments.
- Compared with simulation-based (Abdul Karim *et al.*, 2012a,b) or hybrid methods (Cenydd and Teahan, 2013; Fang *et al.*, 2013), the proposed controller is biologically meaningful and is able to flexibly change motion states (onset, termination, directions and speed) of a character, which is more specifically tailored for small scale insects.
- Compared with biologically inspired methods (McKenna and Zeltzer, 1990), the convergence characteristics of the *Hopf bifurcation* ensures that the motion is able to recover from unexpected situations while the method by McKenna and Zeltzer (1990) uses only oscillatory signals, which lacks stability. Besides, abstract actuator modelling (*Virtual Model Control*) and automatic parameter tuning allow the same framework to be applied to other characters with similar locomotive pattern.

5.5 Summary

This chapter presents the design of the CPG-based controller. This controller is composed of two parts: the *oscillator* and *operators*. The oscillators are modelled as instances of ODEs, with joint angles and angular velocity as inputs. Each oscillator controls an individual joint DOF. The operators transform the properties of a standard oscillator and achieve the coordination between different DOFs and legs. The parameters of the CPG controller are optimised with the Covariance Matrix Adaptation, and the optimal result will serve as the initial guess when building the CLUT in Chapter 6.

Limit cycle proves to be an effective solution to improve locomotion stability (Laszlo *et al.*, 1996). This work chooses a nonlinear oscillator with the

property of the *Hopf bifurcation*, which is able to switch between limit cycle and attraction point. This design not only ensures the locomotion stability but also allows intuitive switching between periodic and discrete motions. However, the CPG controller alone is only capable of producing the fixed locomotion pattern of walking forward with a double-tripod gait. In Chapter 6, the controller look-up table adjusts the parameters of the CPG controller, according to the information embedded in the supporting triangles. By doing this, the character is capable of performing more challenging tasks, such as following curve paths, which cannot be achieved with the CPG controller alone.

Chapter 6

Controller Look-up Table

6.1 Introduction

Insect neural control involves the brain, which determines the onset, direction and speed of walking and the ganglia, which control the local coordination between leg joints (Delcomyn, 1999). The designed controller in this thesis is to mimic this hierarchy: the user input serves as the brain and is in charge of the overall behaviour of the character, while the Central Pattern Generator (CPG) is responsible for producing the coherent joint movement. Figure 6.1 presents an overview of this framework with the Controller Look-Up Table (CLUT) highlighted. CLUT serves as a black box and maps the character behaviours, represented by profiles of supporting triangles, to the control parameters in the CPG: $\xi = [\Delta_1, g_1, \bar{\mathbf{q}}_1 \cdots \Delta_k, g_k, \bar{\mathbf{q}}_k]$ (where k is the number of joints). Biological experiments confirm that insects are unlikely to adjust their control on the fly given their limited computing power (Kukillaya and Holmes, 2008). The design of the CLUT separates the high level task commands specified by animators from the low level control parameters, and allows animators to efficiently specify insect locomotion behaviours. The moderate size of the CLUT is also convenient for record and inquiry.

The idea of the CLUT is similar to the biped control strategy outlined by Coros *et al.* (2008), in which researchers aim to dynamically navigate physically-driven characters with stepping constraints. First, control solu-

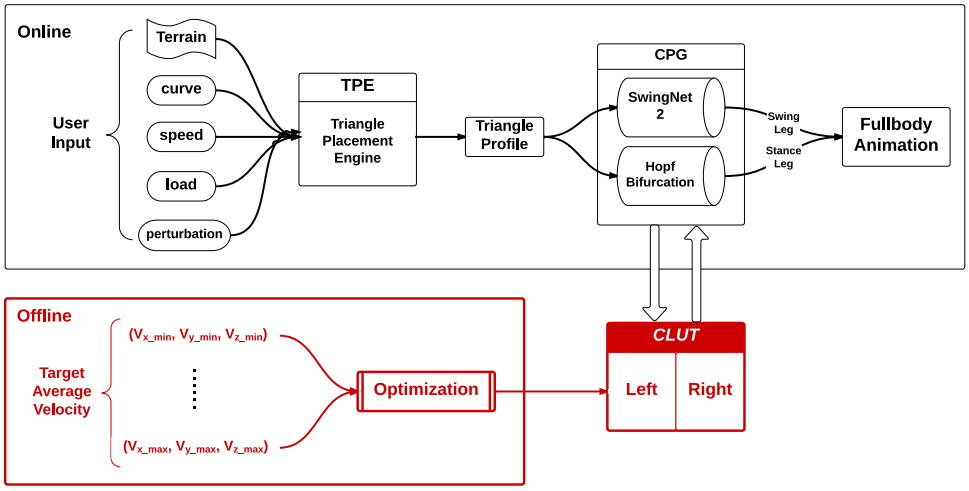


Figure 6.1: *Flowchart of this framework with the CLUT highlighted. CLUT converts the high-level commands, represented as a series of Supporting Triangles, into low-level control parameters of the CPG. The whole process includes offline optimisation to construct the table and online query.*

tions are computed for randomly generated example problems with an offline optimisation. The example problem is to find a control sequence that results in the character stepping precisely at the desired target foot locations (Figure 6.2). Second, the example motions and their control patterns are analysed to build a low-dimensional step-to-step model. The *step-to-step dynamics model (SSDM)* is built using *k-Nearest-Neighbors (kNN)* on both low-dimensional state space (manually selected) and action space (reduced by *Principle Component Analysis (PCA)*). During online simulation, the planner finds an optimal solution to solve the new instances of the task at interactive rates. Different from the work by Coros *et al.* (2008), the work presented in this thesis focuses on insect animation with emphasis on the intuitive and efficient control for animators. Biped characters are different from insects in terms of physiological structure and locomotion patterns, which makes the direct transfer of existing work on simulation of biped locomotion to insects over complicated and infeasible.

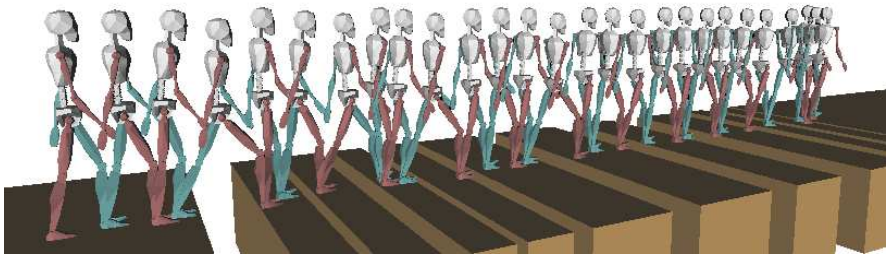


Figure 6.2: *An example problem in Coros* et al. *(2008). This work aims to dynamically navigate physically-driven characters with stepping constraints.*

6.2 Precomputing the CLUT

The CLUT is precomputed by solving the example problems under different settings with offline optimisation. The input problem is defined as: given the fixed amount of time (half period), which control parameters can successfully drive the insect COM to reach the target location? Or in other words, which control parameters are able to maintain the average speed $\nu = \frac{\mathbf{P}_{target}}{T/2}$ in a fixed time step?

The input vector includes:

- χ: which leg group (L1R2L3 or R1L2R3) forms the current supporting triangle.
- $\nu = (\nu_x, \nu_y, \nu_z)$: the average velocity for the current stride.

There are two separate tables because the stance legs are formulated by different leg groups (either L1R2L3 or R1L2R3). ν is represented in the local coordinates of insect body frame.

Firstly, velocity ranges for three directions are defined and subdivided into fixed steps. Table 6.1 shows the settings used in the current application. It should be noted that the $x/y/z$ coordinates are in the character's local frame with $+x$ as the posterior-anterior direction. The step width (SW) is set to $0.001m/s$ for all three directions while the velocity ranges are differentiated, with the consideration that the body movement along the vertical direction is smaller compared with movements on the horizontal plane. The varied settings for three dimensions saves computing time of offline optimisation (discussed in later sections), without sacrificing much utility. Section 6.4

presents discussions on the experiments of different values of SW and their effects on the overall performance and accuracy of the controller. For the settings shown in Table 6.1, $40 \times 40 \times 20 = 32000$ different velocities are obtained: $\nu = [\nu_x, \nu_y, \nu_z]$. The storage size for each table is 6.3 megabytes on the disk. For each case, the optimisation is performed to ensure that the target velocity is successfully satisfied. Table 6.2 shows a snippet of the CLUT.

	max	min	steps	step width (SW)
ν_x	0.02	-0.02	40	0.001
ν_y	0.02	-0.02	40	0.001
ν_z	0.01	-0.01	20	0.001
Unit: (max - m/s); (min - m/s); (SW - m/s)				

Table 6.1: *Settings for precomputing the CLUT. Unit of speed and step width: m/s. The planar and vertical movements are differentiated in order to balance between performance and accuracy.*

linear sampling of 3 directions

v_x	v_y	v_z	Δ_1	$\bar{q}_1$	g_1	.....
-0.020	-0.020	-0.010	0.00868	-0.02678	0.01181	.....
-0.020	-0.019	-0.010	0.00876	-0.02686	0.01181	.....
-0.020	-0.018	-0.010	0.00884	-0.02695	0.01183	.....
⋮	⋮	⋮	⋮	⋮	⋮	⋮
0.020	0.020	0.010	0.02553	0.039921	0.85052	.....

27 parameters of CPG

Unit: (v_x, v_y, v_z - m/s); ($\Delta, \bar{\mathbf{q}}$ - $radian$); (g - ratio with no unit)

Table 6.2: *Snippet of the CLUT. The CLUT gathers the optimal control parameters $\xi = [\Delta_1, \bar{\mathbf{q}}_1, g_1, \cdots \Delta_k, \bar{\mathbf{q}}_k, g_k]$ ($k = 9$ here) for individual cases by linearly sampling the velocity in three directions.*

6.2.1 Objective Function Design

The design of objective functions is critical to the final performance of the controller. It is worth pointing out that both objective functions and optimisation algorithms applied to construct the CLUT are different from those to

build the CPG controller in Chapter 5. Objective functions to build the standard CPG are based on fundamental principles of locomotion, for example minimum metabolic energy consumption, while this chapter focuses on designing a high-level interface for animators, and thus the criteria are designed to evaluate how well the simulated result matches the desired trajectory. In fact, the parameters optimised by the objective functions in Chapter 5 serve as the initial values in precomputing the CLUT in this chapter.

The top priority is to ensure that the direction and magnitude of average velocity in this stride meet the target. In other words, the aim is to minimise the differences between the target position and the simulated result:

$$E_p = \sum (\frac{\mathbf{P}_{target}^{COM} - \mathbf{P}_{simulation}^{COM}}{\Delta \mathbf{P}})^2 \tag{6.1}$$

where $\mathbf{P}_{target}^{COM}$ is the target position of the COM of the virtual character while $\mathbf{P}_{simulation}^{COM}$ is the simulated position of the COM of the virtual character. $\Delta \mathbf{P}$ is the distance between the Barycentric centres of two consecutive triangles. $\mathbf{P}_{target}^{COM}$ is computed as:

$$\mathbf{P}_{target}^{COM} = \mathbf{P}_{initial}^{COM} + \Delta \mathbf{P} \tag{6.2}$$

$\mathbf{P}_{initial}^{COM}$ is the position of the COM of the virtual character when the current triangle is initialised.

Additionally, the movements on the lateral and vertical directions (along y and z axis) are constrained in order to improve the energy efficiency:

$$E_d = \sum_i (||\frac{y_i - y^*}{y^*}||^2 + ||\frac{z_i - z^*}{z^*}||^2) \tag{6.3}$$

where y_i, z_i are current y/z coordinates of the COM in the i^{th} frame and y^*, z^* are the y/z coordinates of the COM when insects stay at static pose.

Finally, the velocity direction should always be consistent with the vector pointing towards the target position. For example, if the target position is

ahead of the initial position, only forward velocity would be expected.

$$E_v = \sum_i ||\frac{\nu_i}{\nu_{target}}||^2, if\ \nu_i \cdot \nu_{target} < 0 \tag{6.4}$$

where ν_i is the simulated velocity in the i^{th} frame.

The final value of the objective functions is a weighted sum of all these three components:

$$E_{sum} = \omega_p E_p + \omega_d E_d + \omega_v E_v \tag{6.5}$$

Similar to the problem of optimising the CPG controller in Section 5.3.3, the weight values are determined by first normalising each objective function, and then selecting the set of weight values which produces the best fitness value of the weighted-sum objective function (Equation 6.5).

First, the *Utopia* points are computed for each objective function E_p, E_d, E_v by iterating all the sample points. The *Utopia* point for each objective function is the solution which produces the lowest fitness value of the corresponding objective function. The *Nadir* point for each objective function is approximated by selecting the solution with the largest fitness value among all the *Utopia* points. The fitness values of the objective functions at both *Utopia* and *Nadir* points are presented in Table 6.3. It is worth noting that values of all *Utopia* points are close to zero, especially for the term E_v. The zero-value of the objective function E_v means that this requirement is *perfectly* satisfied. In other words, the velocity direction always points towards the target position.

	Nadir Point	Utopia Point
E_p	1.583	0.000095
E_d	5.576	0.00010
E_v	27.720	0.00000

E_p, E_d, E_v are ratios with no units

Table 6.3: *Fitness values of the objective function at both* Nadir *and* Utopia *points.*

Next, the objective functions E_p, E_d, E_v are normalised with the fitness values of their *Nadir* and *Utopia* points respectively, before computing the

final fitness value E_{sum} in Equation 6.5. Figure 6.3 presents the comparison between the non-normalised and normalised fitness value of the objective functions. After the normalisation, the fitness value is bound between [0, 1] for most of the time. The reason for the cases where values are greater than one is that the *Nadir* points can only be approximated (Grodzevich and Romanko, 2006; Kim and De Weck, 2006). Figure 6.3 shows that the normalised and non-normalised function shapes of E_p, E_d, E_v are identical. This is because the fitness values at *Utopia* points z_i^U are close to zero, and Equation 5.26 is transformed as:

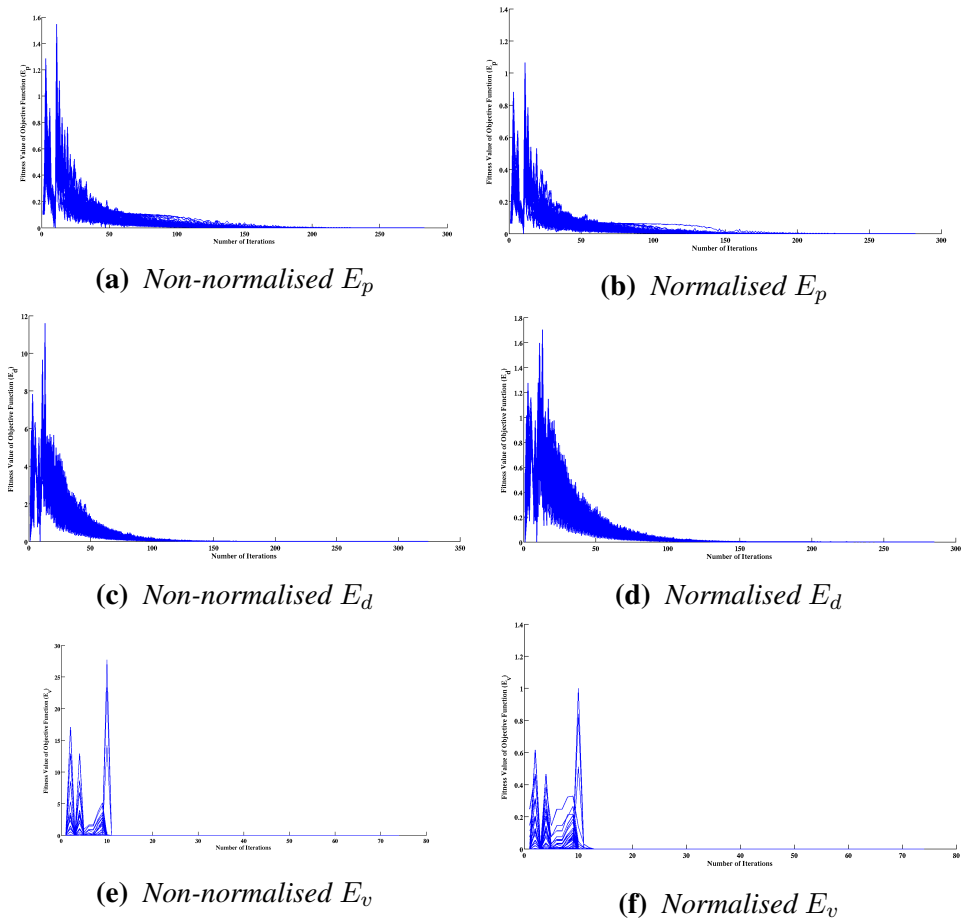
$$\frac{f_i - z_i^U}{z_i^N - z_i^U} \Rightarrow \frac{f_i}{z_i^N} \tag{6.6}$$

The above equation shows that the fitness value of each objective function is scaled by the value of the *Nadir* point only. In this case, the functions shapes of normalised and non-normalised look identical to each other.

Finally, the weights $\omega_p, \omega_d, \omega_v$ are determined with the similar approach in Chapter 5. Assume that $\omega_p + \omega_d + \omega_v = 1$. By linearly sampling the range between [0, 1] for ω_p, ω_d, it is possible to acquire a list of weight choices $(\omega_p, \omega_d, \omega_v)$. After iterating all the choices in the list, the lowest fitness function value of Equation 6.5 could be determined, thus finding the best choice for $\omega_p, \omega_d, \omega_v$. Figure 6.4 plots the result of the fitness values, with respect to different combinations of (ω_p, ω_d). The value of ω_v is determined by $1 - \omega_p - \omega_d$. Eventually, the weights are set as: $\omega_p = 0.6, \omega_d = 0.2, \omega_v = 0.2$.

6.3 Online Lookup

When running the online simulation, the CLUT is loaded into the memory and ready to be inquired. In theory, the program could look up the desired CPG parameters in the table at any time during the simulation. The higher the look-up frequency is, the better the simulation accuracy will be. Section 6.4 discusses the performance and effects of different look-up frequencies. In most cases, the control parameters for the CPG are updated once a new Supporting Triangle is initialised.



(a) *Non-normalised* E_p **(b)** *Normalised* E_p

(c) *Non-normalised* E_d **(d)** *Normalised* E_d

(e) *Non-normalised* E_v **(f)** *Normalised* E_v

Figure 6.3: *Comparison between non-normalised and normalised fitness value of the objective functions. The result shows that the function shapes of non-normalised and normalised cases are almost identical. The reason is that the fitness values at the* Utopia *points are close to zero, which makes the effects of normalisation similar to scaling by a scalar (the fitness value at* Nadir *points). This is proved in Equation 6.6.*

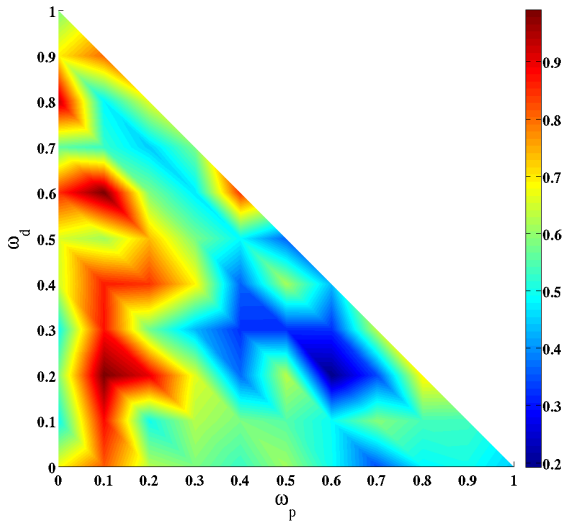


Figure 6.4: *The best optimal value of the weighted sum objective function (Equation 6.5), with respect to different weight choices. This map is drawn by linearly sampling the weights ω_p, ω_d and calculating the optimal value at each sample. The weight values $\omega_p, \omega_d, \omega_v$ are selected at the position of the best optimal value of the objective function.*

During the precomputing process, the initial velocity is assumed to be zero, which is not true for most cases during simulation. The effect of the initial velocity ν is eliminated by deducting its corresponding contribution to the overall trajectory:

$$\nu_{lookup} = \frac{\Delta P_n^{n+1} - \int_0^{t_f} \nu_i dt}{t_f} \tag{6.7}$$

where ΔP_n^{n+1} is the distance between the barycentric position of $n+1^{th}$ and n^{th} supporting triangle. t_f is the expected time to cover this stride (normally half period).

During the online simulation, the value of the velocity may not exactly match the samples in the look-up table. For example, the velocity $[0.0005, 0, 0]$ falls into the range between two samples $[0, 0, 0]$ and $[0.001, 0, 0]$. In this case, the desired CPG parameters are computed as a weighted sum of the parameters of the neighbouring samples points, with the method of *Inverse Distance*

Weighting (Shepard, 1968):

$$\Xi = \sum_{i=1}^{n} \omega_i \xi_i, \qquad \xi_i \in \Omega \tag{6.8}$$

$$\omega_i = \frac{d_i^{-p}}{\sum_{j=1}^{n} d_j^{-p}} \qquad p = 2, n = 8$$

Ω is the set containing all the neighbouring points, d_i is the *Euclidean* distance between two vectors $\|\xi_i - \xi_j\|$. The weighted-sum vector Ξ is selected to update the parameters of the CPG controller in order to achieve the desired velocity.

6.4 Results & Discussions

Performance The offline process to generate the CLUT requires a significant amount of optimisation. The optimisation is performed with a standard method, which accepts an unconstrained multivariate scalar function with the algorithm of Newton-Conjuagete-Gradient from the scientific computing libraries *Scipy*. To speed up the process of optimisation, the result from the last optimisation is selected as the initial value for the next iteration. On average, it takes around 300 iterations (Figure 6.5) and one second for the optimisation algorithm to process one sample point, with the fitness value of the objective function reaching the value below 10^{-4}. The total time to complete the optimisation for the whole CLUT (32000 samples) is around ten hours on a standard PC (CentOS with 4GB memory, Intel Core Quad CPU @2.83GHz). This could be further reduced to less than three hours (on four threads) with parallel processing. Though the performance is acceptable for an offline optimisation, it can be improved with more advanced hardware specifications (Wang *et al.*, 2012) and using the optimisation library written in a low-level language such as C.

Using the optimal solution from the last sample point as the initial value for the next sample point is an effective strategy to speed up the optimisation process. The value of the objective function for the initial iteration is as high as 13.472 (for the purpose of image scale, this is omitted from Figure 6.5).

The initial value of the objective function for the rest of sample points starts from less than one. Additionally, the optimisation for the first sample point takes more than 300 iterations to converge, while the rest take about 200-300 iterations (Figure 6.5).

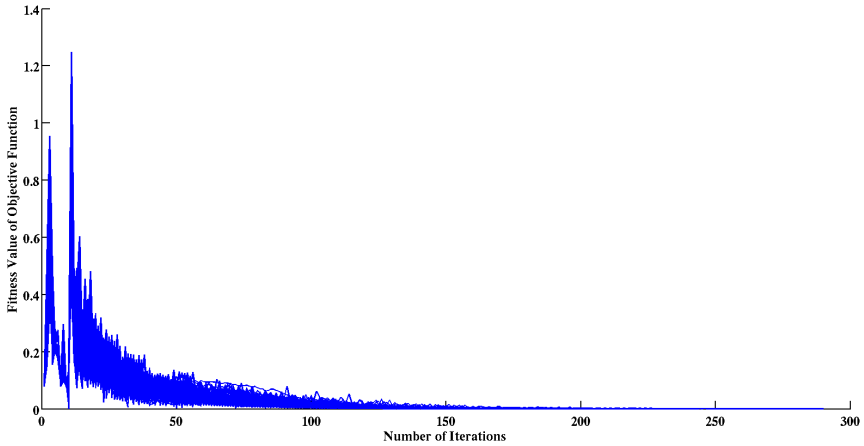


Figure 6.5: *The result of the optimisation process when precomputing the CLUT.*

Sampling Rate There are two types of sampling rates which affect the final outcome of this method. The first is the sampling rate when computing the CLUT and the second is the sampling rate (or look-up rate) when running the simulation during runtime.

The selection of the suitable step width is a trade-off between the time to construct the look-up table and the deviation error which may occur during online simulation. On one hand, choosing a smaller step width increases the number of samples, thus increasing the time cost of optimisation. On the other hand, deviation errors decrease when the accuracy of the control parameters increases due to a smaller step width. Table 6.4 presents a comparison of time cost, storage size and deviation error for different sampling rates.

- Step Width (SW): the distance between two nearby sampling points in $x/y/z$ directions (unit: m/s). Note the velocity ranges remain the same as Table 6.1.
- Time Cost: the total time cost for the optimisation process (unit: $hour$). This data are collected after speeding up the performance with parallel

processing with four threads. Figure 6.5 shows that each sample takes around 300 iterations to converge. The average computing time for each sampling point is 1.13 seconds with a standard deviation of 0.29 seconds. Therefore, the total time cost grows approximately in a linear fashion with the number of samples.

- File Size: the storage size on the disk (unit: megabytes). Each sample point is a fixed-length vector including an input velocity vector of 3 dimensions and a control parameter vector of 27 dimensions (Table 6.2). Therefore, the file size of the CLUT grows in a linear fashion with respect to the number of sampling points.
- Deviation Error: the summed differences between the target trajectory and simulation result (unit: $metre$). The simulation setting is a straight curve with its length $L = 0.5m$, walking velocity $\nu = 0.015m/s$. The look-up frequency is twice per walking cycle.

Step Width (SW)	Number of Points	Time Cost	File Size	Deviation Error
0.0005	256,000	24	52.4	0.008
0.0010	32,000	3	6.3	0.026
0.0020	4,000	0.5	0.80	0.069

Unit: (SW - m/s); (Time Cost - $hour$); (File Size - $megabytes$); (Deviation Error - $metre$)

Table 6.4: *The data of total time cost are collected using a multi-processing technique (unit: hour). As the number of samples grows, the time cost increases in proportion while the deviation error decreases.*

Table 6.4 does not take into consideration the time cost for online lookup because the whole CLUT is loaded into the memory and the memory-reading action is fast enough to be ignored (in units of nanoseconds).

On the other hand, if the look-up frequency is increased during the runtime simulation, the control parameters of the CPG will be updated at a higher frequency and the controller is able to react with the environment or follow the user-defined trajectory with more accurate adaptation. This also leads to a reduced error between the simulation result and the target trajectory. The simulation setting is a straight curve with its length $L = 0.5m$ and walking velocity $\nu = 0.015m/s$. The CLUT is generated with a step width $SW = 0.001m/s$. The criteria used in the table are:

- Look-up frequency: the relative frequency of look-up action compared with the walking frequency. When it is set to 2, the control parameters are updated when each triangle is initialised.
- Number of look-ups: the total counts of look-up actions during the simulation.
- Time: the time cost for the simulation (unit: second).
- Deviation Error: the summed differences between the target trajectory and simulation result (unit: $metre$).

Look-up Frequency	Number of look-ups	Time	Deviation Error
2	67	2.50	0.026
4	123	2.57	0.014
8	269	2.62	0.007

Unit: (Look-up Frequency - Hz); (Time - $second$); (Deviation Error - $metre$)

Table 6.5: *Comparison of the time cost and deviation error for different look-up frequencies. The result shows that different look-up frequencies have little effects on the time cost. Meanwhile the deviation error could be significantly reduced by increasing the look-up frequency. It is worth noting that by increasing the look-up frequency to 8, the deviation error reduces to a similar level as decreasing the step width to half of the standard setting.*

Discussions on scalability The data shown in Tables 6.4 and 6.5 reveal the need to handle the settings of these two rates with care, in order to find a balance between accuracy and performance. The number of sampling points is calculated as $S_x \times S_y \times S_z$ (each indicates the number of samples along $x/y/z$). The heavy computation, especially for offline optimisation, constrains the scalability of the method if the precision of the CLUT needs to be adjusted. A solution is to design a look-up strategy with an adapted frequency. When the character is walking in an unchallenged mode such as along a straight path on a flat ground, a lower look-up frequency could be selected. While in more complex situations such as sudden turning, a higher look-up frequency could be introduced.

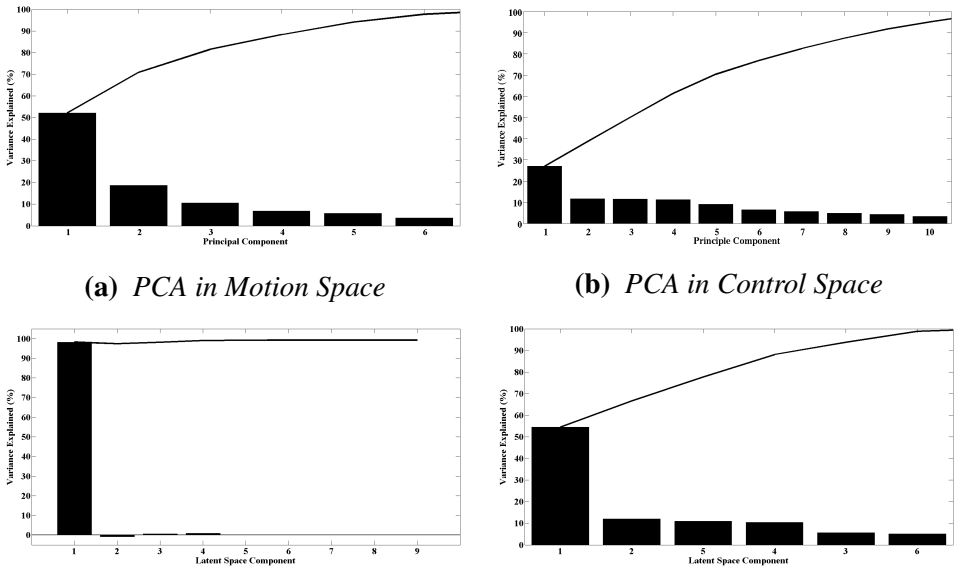
Another possible solution is to construct a low-dimensional space. Potential dimension-reduction methods include the linear one, such as Principle

Component Analysis (PCA) (Coros *et al.*, 2008), or the nonlinear one, such as Gaussian Process Latent Variable Model (GPLVM) (Levine *et al.*, 2012). Both methods are tested in the motion space of joint angles and the parameter space of the controller. The results are shown in Figure 6.6. Some observations can be drawn from the results:

- The result of applying PCA to the joint motion space is shown in Figure 6.6a. The vertical axis is the percentage of the total variance explained by each principal component. The joint motion space originally has 18 dimensions, while using PCA, more than 95% of the total variance is explained by the first six principle components. The first principle component accounts for about 50% of the variability in the data. In comparison, the result of applying PCA to the control space is shown in Figure 6.6b. The control space originally has 27 dimensions, while using PCA, the first ten principle components are needed to reach the threshold of 95%. Figure 6.7 demonstrates the coefficients of first and second principle components (PC) at different velocities along x axis. The velocity along x axis varies between [-20, 20]mm/s, and the velocities on the y and z axes are zero. The result shows a high level of non-linearity, which confirms the complexity of the control look-up table. The variance of the first principle component reduces to about 30%, lower than the case of joint motion space. This comparison reveals that the CLUT demonstrates a higher level of non-linearity compared with the motion space.

- Figure 6.6c shows an extreme correlation between the 18 DOFs of the joint motion, which are dramatically reduced to one single dimension by GPLVM. This result may appear surprising at the first place, but in fact is in accordance with the design of the CPG controller. The CPG controller is constructed in such a way that DOF-specific operators transform the signal from a standard oscillator. Note that the input motion data is from a normal walking cycle, in which no perturbation, turning, noise is added. In this simplified case, the operators (amplitude, phase shifting, offsetting) remain constant for individual DOFs. GPLVM is able to extract the information from the standard oscillator, while removing the information contained in the unchanged operators.

When applied to the control space (Figure 6.6d), GPLVM still outperforms PCA. This advantage is verified by the larger variance explained by the first principle component, and less dimensions needed to reach the threshold of 95%. This comparison demonstrates the advantage of GPLVM over PCA in processing non-linear data.

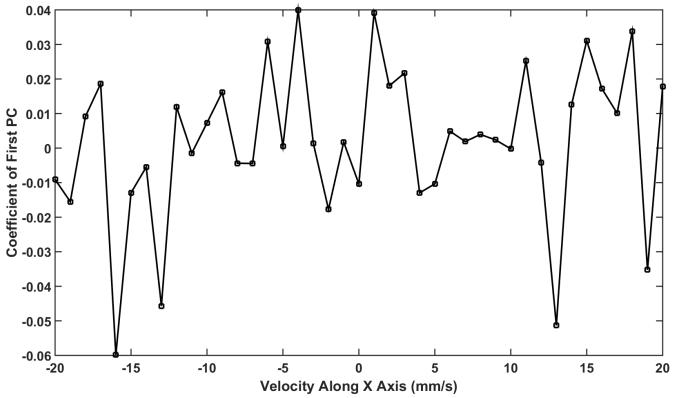


(a) *PCA in Motion Space* **(b)** *PCA in Control Space*

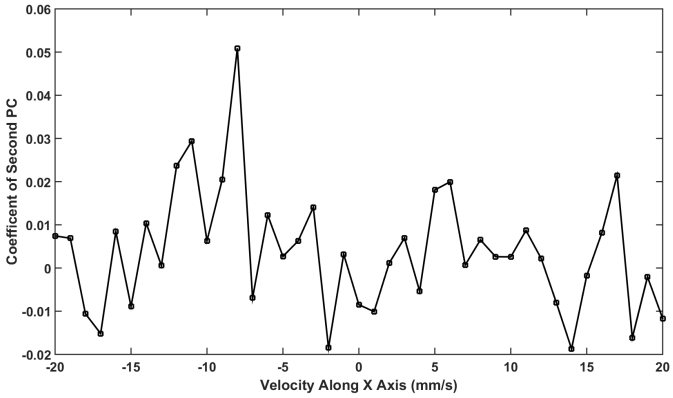
(c) *GPLVM in Motion Space* **(d)** *GPLVM in Control Space*

Figure 6.6: *Applying PCA and GPLVM to both the motion space and control space. Please note that the input motion data is from a normal walking cycle, in which no perturbation, turning, noise is added. The comparisons of (a)-(b), (c)-(d) show that the control space contains greater variability compared with the motion space. The comparisons of (a)-(c), (b)-(d) demonstrate the advantage of GPLVM over PCA in reducing the dimensions of both joint and control spaces.*

Methods such as PCA or GPLVM transform the discrete representation of the parametrised control space into a continuous one. This transformation has pros and cons. On one hand, the elegant low-dimensional space not only reduces the computation but also filters possible noises. On the other hand, it introduces an additional layer, adding up the complexity. In the case of an unexpected simulation result, it requires additional efforts to identify the cause, whether it is rooted in the original sample space, or results from the regression error between the discrete and continuous spaces.



(a) *Coefficients of first principle component (PC) at different velocities along x axis. The velocities on the y and z axes are zero.*



(b) *Coefficients of second principle component (PC) at different velocities along x axis. The velocities on the y and z axes are zero.*

Figure 6.7: *Coefficients of first and second principle components (PC) at different velocities along x axis. The result shows high non-linearity of the control parameters among different samples of velocities. This could be caused by the optimisation strategy when constructing the look-up table. Given the high dimensions of the control parameters, there is a high probability that the optimisation falls into different local minima for different velocities.*

6.5 Summary

This chapter presents the design and application of the CLUT. As one of the most popular techniques used in robotics control, this method proves to be effective in authoring the locomotion behaviours of insects. The key idea behind the CLUT is not constrained to the implementation of the current controller, or any type of design of the CPG. Therefore, it is possible to transfer this to design an efficient control strategy in robotics. The balance between performance and accuracy is worth pointing out. Currently the look-up for control solutions is implemented per step. Velocities would certainly change during the course of one step, which demands control solutions different from the one initialised at the beginning of the current step. The deviation errors could be reduced by increasing the look-up frequency.

Geijtenbeek *et al.* (2013) pointed out that the dependence on optimisation strategy introduces additional complexity when designing a controller, which applies to this case. This adds to the confusion when the simulation does not produce the desired result. Furthermore, since the control parameters in the CLUT are responsible only for a single stride, it is difficult to justify that the final outcome from the optimisation is the *optimal* solution in a biological sense.

Chapter 7

Switch Mechanism

7.1 Introduction

The motion cycle is divided into two phases for each leg: stance and swing. To determine when legs switch from stance to swing or vice versa, traditional approaches use time-series signals (e.g. switch after a fixed time period) (McKenna and Zeltzer, 1990; Yin *et al.*, 2007) and foot-strike events (Cenydd and Teahan, 2013). Some use the spatial information, to initiate the leg swing when the horizontal distance between the COM and ground contacts exceeds a threshold (Wang *et al.*, 2012; Geijtenbeek *et al.*, 2013) (Figure 7.1).

These strategies are not applicable to multi-legged insects where interactions between neighbouring legs play an important factor. In addition, they do not take into consideration the case when insects are carrying an object and making an effort to maintain their stability. Cruse (1985) conducted the experiment in which a stick insect (*Carausius morosus*) walked on a tread-wheel and studied the relationship between the triggering of a swing phase and the load and position of a leg. The results showed that although the leg position is an important parameter, load, instead of position, is possibly the true determinant for the decision to terminate the stance phase and initiate the swing phase.

While experimental biologists are still looking for the determinant which initiates the swing phase, this study aims to build a capable and intuitive

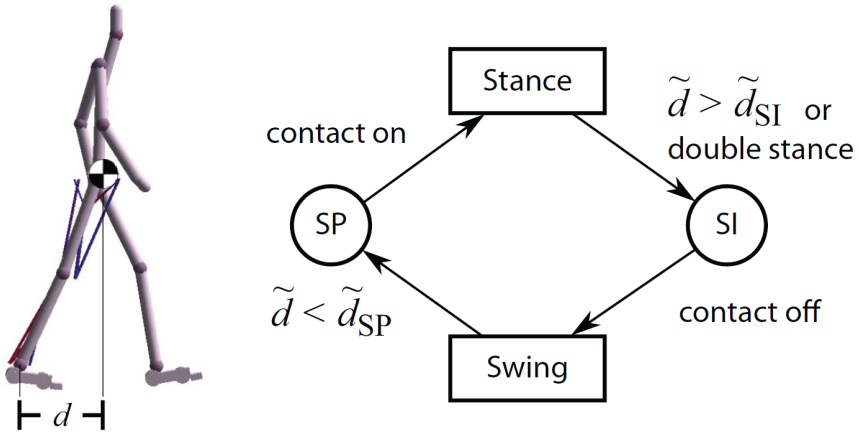


Figure 7.1: *The switch mechanism for each leg in Wang* et al. *(2012); Geijtenbeek* et al. *(2013). The variable $\tilde{d}$ is computed as the normalised signed horizontal distance (d) between the COM and the stance ankle (normalised by leg length). $\tilde{d}$ is compared against two threshold parameters ($\tilde{d}_{SI}, \tilde{d}_{SP}$), to decide whether or not the Finite State Machine should switch to the states of swing initialisation (SI) and stance preparation (SP) respectively.*

framework for applications in animation and film production. Given the special pattern of tripod gait in the hexapod, existing work suggests switching to the swing mode when the COM reaches the edge of the current supporting triangle (Ting *et al.*, 1994; Guo *et al.*, 2014a). This is true when the insect is moving fast and allows no extra time for complicated decision making. However, when insects are moving at slow speeds, rigid animation is produced with this triangle approach and is more easily noticed than high speed movement.

Here the *Froude* number is the criterion chosen to select the proper rule for stance-swing switch. *Froude* number is a general indicator which reflects gait patterns among animals with different sizes and masses (Alexander, 1984).

$$Fr = \frac{\|\nu\|^2}{gh} \tag{7.1}$$

where ν is the average speed in a stride, h is the average height of the COM above the ground and g is the gravitational acceleration.

- If $Fr >= Fr_{threshold}$, the ant is walking fast or running. In this case, the leg transition between stance and swing is triggered when the COM steps outside the Supporting Triangle (Ting *et al.*, 1994; Guo *et al.*,

2014a).

- If $Fr < Fr_{threshold}$, the ant is walking slowly. In this case, a novel probability framework is proposed to replace the tripod gaits and determines the switch for individual legs based on four independent conditions.

Equation 7.1 states that the threshold of Fr is determined by the threshold velocity, given the gravitational acceleration $g = 10m/s^2$ and standard COM height $h = 0.002m$. The threshold velocity of transitioning between walking and running is set to $0.02m/s$, the median value of the speed range [0, $0.04m/s$]. Therefore, $Fr_{threshold}$ is set to 0.02.

7.2 High Speed

The *Supporting Triangle* is used as the switch mechanism when the *Froude* number is larger than $Fr_{threshold}$. The state of the individual legs is toggled between stance and swing whenever the COM steps out of the current supporting triangle, resulting in the periodic alternation of the left and right tripod.

This pattern is observed in the captured walking sequences of an ant. Figure 7.2 provides a side-by-side comparison of the supporting triangles between real and simulated ants. In addition to its underlying biological intuition, *Supporting Triangle* is independent of gait and can be applied to evaluate gait stability in general cases, such as in turning (Figure 7.2(c)). When the COM steps outside the triangle via other edges, the same switch mechanism is triggered and the new triangle is placed according to the current body orientation and position.

Figure 7.3 illustrates the algorithm to check whether the COM is inside or outside the supporting triangle. The point in red denotes the COM projected onto the plane of the Supporting Triangle. The point $\mathbf{O}$ in black denotes the average of the three vertices:

$$P_{\mathbf{O}} = \frac{\mathbf{V}_1 + \mathbf{V}_2 + \mathbf{V}_3}{3} \tag{7.2}$$

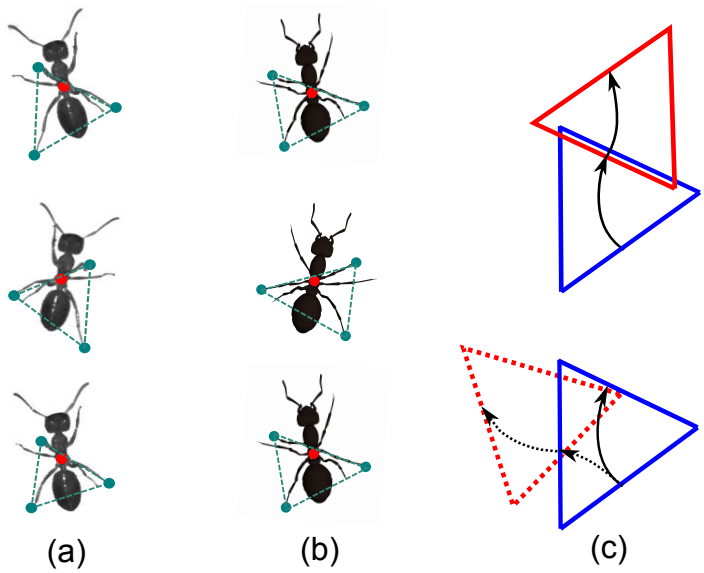


Figure 7.2: *Using Supporting Triangle to determine the switch from stance to swing in the case of high speed. (a) Video screenshots of a* real *ant with two consecutive steps. The image sequences in (a) are provided by courtesy of researchers Holger Bohn from University of Freiburg, and Karin Moll and Walter Federle from University of Cambridge (Moll* et al.*, 2010). (b) Animation screenshots of a* simulated *ant with two consecutive steps. The red dots in both figures indicate the position of the COM. (c) State transition between two supporting triangles during normal walking (top) and turning (bottom). The solid curve indicates the trajectory of the COM when the character is walking straight forward, and the dashed curve indicates the actual trajectory when the character is turning left.*

Note that **O** will always lie within the triangle. The question to check whether the COM is inside the triangle is then converted into: whether the COM falls on the same side with **O** for each of the triangle edges? $\vec{\mathbf{r}}_i^{COM}, i \in (1,2,3)$ denotes the vector connecting from the COM to the i^{th} vertex of Supporting Triangle. $\vec{\mathbf{r}}_i^{O}, i \in (1,2,3)$ denotes the vector connecting from **O** to the i^{th} vertex of Supporting Triangle. Next the cross products of these vectors are computed respectively and the results are compared against the other:

$$Array_{COM} = [\vec{\mathbf{r}}_1^{COM} \times \vec{\mathbf{r}}_2^{COM}, \vec{\mathbf{r}}_2^{COM} \times \vec{\mathbf{r}}_3^{COM}, \vec{\mathbf{r}}_3^{COM} \times \vec{\mathbf{r}}_1^{COM}] \tag{7.3}$$

$$Array_{O} = [\vec{\mathbf{r}}_1^{\mathbf{O}} \times \vec{\mathbf{r}}_2^{\mathbf{O}}, \vec{\mathbf{r}}_2^{\mathbf{O}} \times \vec{\mathbf{r}}_3^{\mathbf{O}}, \vec{\mathbf{r}}_3^{\mathbf{O}} \times \vec{\mathbf{r}}_1^{\mathbf{O}}] \tag{7.4}$$

if each corresponding element in $Array_O$ and $Array_{COM}$ has the same positive or negative signs respectively, the COM falls within the current Supporting Triangle.

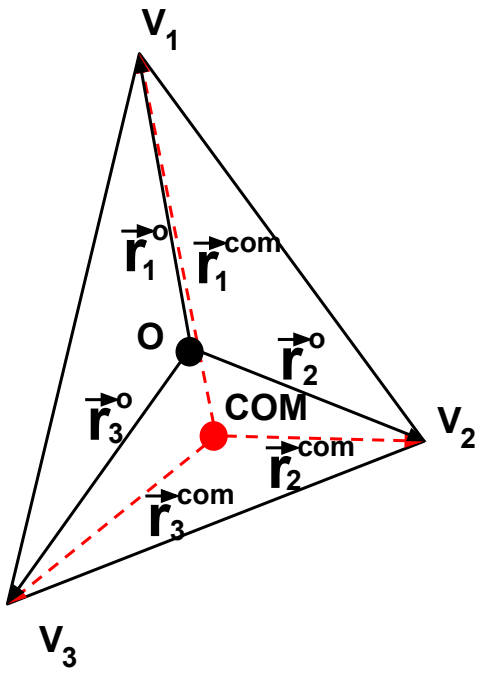


Figure 7.3: *The algorithm used to check whether the COM falls within the Supporting Triangle (ST).*

7.3 Low Speed

In the mode of low speed, an insect adopts the mechanism of metachronal coordination by taking into consideration four different factors. A *metachronal* gait is a stepping sequence in which leg movements propagate from the back of the insect to the front on each side of the body (Beer *et al.*, 1997).

Probability of extreme position Cruse (1985) suggests that there exists a relationship between the possibility of switch occurrence and the relative position of a leg to its Anterior Extreme Position (AEP) and Posterior Extreme Position (PEP). AEP and PEP are the furthest points which a leg can reach forward and backward in relation to the COM of an insect. Therefore, the possibility of a leg swing can be defined as a function of the relative position to its extreme pose. The variable ΔD_{com} is the travelling distance of the COM since the foot contact is initialised on the ground. The further the COM moves relative to each stance leg, the greater the possibility of state switching is. This probability is modelled as:

$$P_e = 1 - e^{-a_e \frac{\Delta D_{com}}{S}} \tag{7.5}$$

where S is the stride length (average value is $5mm$) and the constant $a_e = 5$.

Probability of phase The switch of a leg will be triggered once this leg stays in stance mode after a sufficiently long time. This is the most common situation. Because the whole walking cycle is divided into two phases - stance and swing, a leg is expected to start swinging after standing on the ground for half of the motion period. $\phi \in [0, 2\pi]$ is the phase variable indicating the relative temporal position in a walking cycle. This probability is modelled as:

$$P_t = 1 - e^{-\frac{\phi}{\phi_c}} \tag{7.6}$$

where $\phi_c = \frac{\pi}{4}$.

Probability of Neighbouring leg Another factor is the relative position of a neighbouring leg. In the case of forward walking, the possibility for the i^{th} leg to switch from stance to swing increases, if the $(i+1)^{th}$ leg on the same side moves closer. This probability is modelled as:

$$P_d = e^{-\frac{\Delta d}{d_c}} \tag{7.7}$$

where Δd is the relative distance between the end points of two neighbouring legs and $d_c = 0.5mm$.

Probability of Load The probability of a leg to start swinging is also determined by the load on the leg, that is the force applied on the leg in the vertical direction:

$$P_l = e^{-\frac{F_L}{F_{Lc}}} \tag{7.8}$$

During the stance phase, the leg is loaded in order to support and propel its body. Once it reaches the PEP, the leg is unloaded and enters the swing phase (Zill *et al.*, 2004; Szczecinski *et al.*, 2014). F_{lc} is set to $0.005N$ (half of the ant's own gravitational force) through simulation trials.

Figure 7.4 plots the probability of a leg switching from stance to swing, with respect to the aforementioned four factors. The final probability is calculated as follows:

$$P = 1 - (1 - P_e)(1 - P_t)(1 - P_d)(1 - P_l) \tag{7.9}$$

This concatenation form of probability allows the switch to take place as long as at least one of the four aforementioned conditions is satisfied. Different thresholds are set for three legs in the same group: $P_{threshold} = [0.6, 0.7, 0.8]$. If $P > P_{threshold}$ for a specific leg, the leg is switched to the swing mode, otherwise the leg remains in stance.

It is worth noting that this probability module is likely to result in the situation where legs belonging to different groups are visually in the same state (either stance or swing), but the group which they belong to is actually unchanged. In other words, each leg is hand-coded into two fixed groups, either L1R2L3 or R1L2R3.

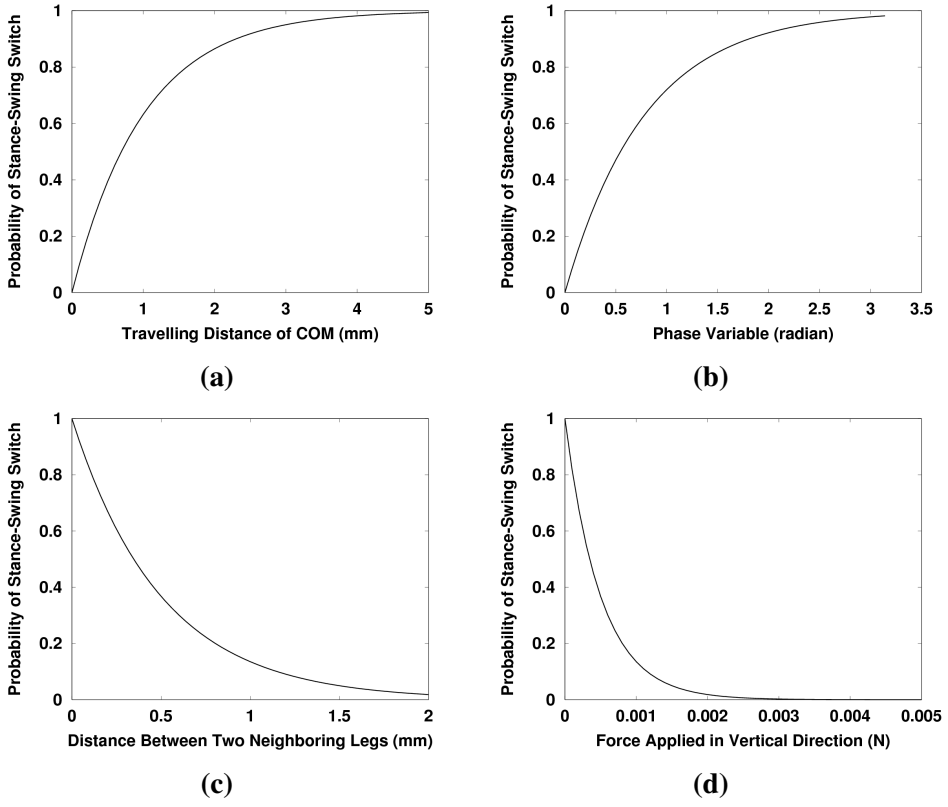


Figure 7.4: *Four probability components introduced in this switch mechanism. Assume P indicates the probability of a stance leg to start swinging. (a)P_e: the relationship between the traveling distance of COM and P (Equation 7.5). (b)P_t: the relationship between the phase variable and P (Equation 7.6). (c)P_d: the relationship between the distance of neighbouring legs and P (Equation 7.7). (d)P_l: the relationship between the applied load weight and P (Equation 7.8)*

7.4 Results & Discussions

Phase Plot Figure 7.5 shows the phase plot for six legs of insects, walking at a comparatively slow mode and carrying no objects. According to Equation 7.1 and given that $\nu = 1.5cm/s, h = 0.2cm$, the *Froude* number $Fr = 0.01125$. This confirms that the character is currently moving slowly and will adopt the metachronal coordination. The phase plot demonstrates a greater versatility of timing for legs switching from stance to swing, instead of a strict double-tripod pattern. The phase plot in Figure 7.5 also reflects the overlap of two successive triangles, which is meant to enhance the stability.

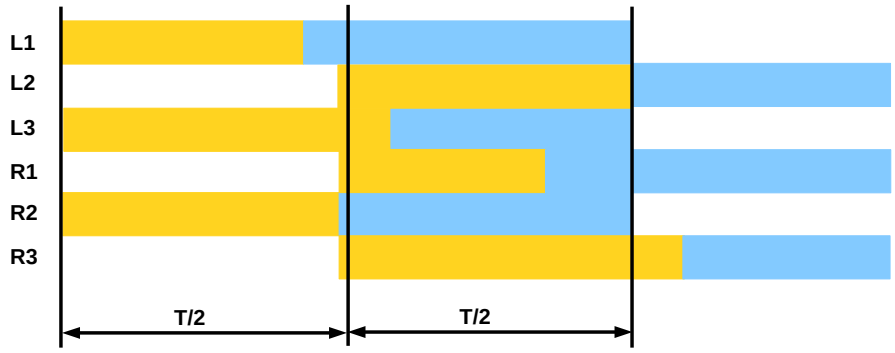


Figure 7.5: *Phase plot of six legs when speed $\nu = 1.5cm/s$ (slow mode) and the load weight $m_L = 0kg$. Yellow denotes that the leg is in stance mode while blue is for swing mode. This graph reveals that legs in the same group are triggered to swing at different time points. The coordination varies when the different thresholds $P_{threshold}$ are chosen.*

The influence of distance of neighbouring legs The synchronised movements of legs along the longitude axis produce the metachronal wave. When insects move fast, the neighbouring legs on the same side tend to collide with each other if the anterior leg does not start to swing earlier than normal. In Figure 7.6, the probability of distance of neighbouring legs is disabled in the case of fast movement, resulting in the overlapping of the front and middle legs on the same side.

The influence of extreme positions The stride length increases when the locomotion speed of a character increases. The area of the supporting triangle

Figure 7.6: *Comparison between two simulated results to demonstrate the defect of front leg's over-crossing. (L): The influence of the distance of neighbouring legs is taken into consideration. (R): The influence of the distance of neighbouring legs is ignored. When ants move fast ($\nu = 3cm/s$), the distance between Supporting Triangles increases and neighbouring legs on the same side are more likely to overlap each other. In the right picture, the front and middle legs on the left side overlap.*

increases when the character is carrying an object or resisting against perturbations. In these two situations, the AEP and PEP will reach further in the forward and backward direction respectively. This could lead to the defect of breaking the joint limit and over-stretching or over-bending a specific joint. In Figure 7.7, the influence of extreme positions is ignored, resulting in the over-stretching of the left hind leg.

Figure 7.7: *Comparison between two simulated results to demonstrate the defect of hind leg's over-stretching. (L): The influence of the extreme positions is ignored. (R): The influence of the extreme positions is taken into consideration. When an ant carries an object, the area of the Supporting Triangles increases and so does the Posterior Extreme Position (PEP). When the influence of PEP is ignored in triggering the leg swing, the leg will be over-stretched, as is the case for the left-hind leg on the left side of the image.*

Limitations However, the method proposed in this work does not consider the collision between the legs and main body. This type of collision happens

frequently when the character is walking sideways, which is commonly observed in the case of collective transport (Figure 7.8). This failure is partially caused by the fact that the current probability framework only determines when a leg switches from stance to swing. Additional strategy to switch from swing to stance is needed to terminate the swing phase and initialise the stance phase in time. One of the solutions is to terminate the swing phase ahead of time when the leg approaches the extreme poses in the lateral direction. This is similar to the effects of the extreme poses in the posterior and anterior directions in Equation 7.5.

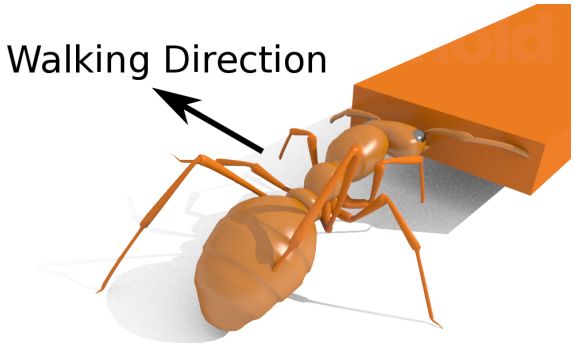


Figure 7.8: *Collision between the right hind leg and main body. The right hind leg collides with the abdomen when the character is walking sideways. This is because the framework only determines when a leg switches from stance to swing, and does not address the* reverse *problem of switching a swing leg to the mode of stance.*

7.5 Summary

This chapter presents a novel strategy to trigger the swing of a stance leg with fast and slow movements treated separately. For fast movements, legs in stance are triggered to swing as soon as the COM steps outside the current Supporting Triangle. However when insects move at slow speeds, they demonstrate remarkable variations of leg coordination, which cannot be produced with the same strategy for high speed. The strategy designed for slow movements takes various factors into consideration, including the relative dis-

tance to the extreme positions, the relative distance between neighbouring legs, the current phase in whole motion cycle and the load on the legs. This strategy is able to cope with complex tasks such as collective transport, which is challenging for previous controllers.

Chapter 8

Conclusions & Future Work

8.1 Conclusions

Natural-looking insect animation is difficult to simulate. On one hand, it is difficult to capture the motion data of real insects because of their fast movement and small scale, therefore conventional example-based methods are not applicable here. On the other hand, simulation-based methods require modelling their delicate and complex physiological structure, which makes it challenging to design an appropriate controller to actuate the joints.

This thesis addressed this challenge by presenting a control framework of insect animation, based on two key observations of real insects: fixed gait pattern and distributed neural system. These observations lead to the development of the two components in this framework: the Triangle Placement Engine (TPE) and the Central Pattern Generator (CPG). On one hand, the TPE addresses the limitations of example-based methods, by utilising the information embedded in the foot contact of real insects; therefore, the framework in this thesis is able to generate whole body animation of insect, without the access to the joint motion data. On the other hand, the CPG addresses the challenges with simulation-based methods, by constructing the controller as a network of oscillators, each of which separately actuates one DOF; this design mimics the distributed neural system in real insects and provides an alternative solution to controller design in simulation-based methods.

The framework is able to generate a wide range of locomotion skills for virtual insects, including walking along straight or curve paths, traversing uneven terrains, recovering from emergent or continuous perturbations, carrying objects individually or cooperatively etc. The following paragraphs will first discuss the findings which arise in this exploration, and second systematically evaluate this framework using the four criteria in the introduction chapter.

Findings

Firstly, this work demonstrates that a biologically-inspired framework is able to successfully synthesise insect animation. This framework is tailored to address the problem of insect animation in the following manners:

- The TPE is constructed as a novel mathematical model, extracting the ant's stepping pattern under different locomotion speed, path curvature and load from experimental results by Zollikofer (1994a,b,c). Different from existing works in using step patterns to generate whole body animation (van Basten *et al.*, 2010), the TPE specifically takes advantage of the fixed double-tripod gait, which is widely observed in insects (Chapman, 1998). This component also avoids the problem of lacking motion data in joint space for example-based methods.
- The CPG is modelled as a distributed network, where a standard oscillator signal is transformed by DOF-specific operators. This hierarchy mimics the specialised structure of insects, whose joints are independently controlled by individual body ganglia. The CPG controller demonstrates its capability in resisting the external perturbations (Figure 5.11). This feature of stability is benefited from the convergence property of the oscillator.
- The CLUT connects the high level commands from the TPE with the low level control pattern of the CPG. This is accomplished by precomputing the map between the desired velocity and the corresponding control parameters in the CPG. The advantage of the look-up table is that it is conceptually simple and relatively easy to implement.

The performance of this framework, including comparisons with example-based and simulation-based methods, will be evaluated with four criteria - naturalness, high level control, stability and ease to design - in the following section *Evaluation*.

Secondly, this work proves that a complete framework (TPE+CLUT+CPG) achieves more flexibility and stability than an isolated CPG controller. Current research in neuron-biomechanics simulation focuses on modelling the CPG only (Örjan Ekeberg, 2004; Kukillaya and Holmes, 2008; Szczecinski *et al.*, 2014). In comparison, this framework connects the CPG with the CLUT and TPE, improving the performance in the following two aspects:

- Stability. Although the CPG controller is able to respond to perturbations by itself, the magnitude of the force it can resist is limited (on an average numeric scale of $10^{-3}N$ for a period of one stride in Table 5.4, 5.5, 5.6). Connecting the CPG with the TPE proves to be useful in increasing the stability of this control framework. This is evidenced in the case of walking against the continuous wind force in Figure 4.14d. The simulation of walking against sideways wind shows that by increasing the area of the supporting triangle in the case of perturbations, the character is able to resist a larger force ($0.01N$) for a longer period (20 strides).

- Flexibility. The capability of generating normal locomotion with the CPG alone is confirmed by the work presented in Chapter 5. Furthermore, the whole framework presented here takes full advantage of the CPG's inherent stability, and successfully addresses its limitations in flexibility. The TPE adjusts the control parameters of the CPG according to external environment or user commands. This allows high-level user control over the locomotion trajectory, velocity etc of the virtual characters (Figure 4.14), which cannot be accomplished by the CPG controller alone.

Thirdly, this work shows that without the data of joint angles, the foot contact information of insects alone can be successfully used to generate the whole body animation. This finding is important, because it provides an alternative solution for researchers with the goal to synthesise insect animation.

The naturalness of the synthetic motion is confirmed by a good match between the simulated result and ground truth in terms of supporting triangles (Figure 4.16) and foot trajectory (Figure 4.17). However, the differences between the simulated result and ground truth deserve extra attention, because these differences may be the causes for the gap between the current controller and real insects. For example, currently the simulation divides the motion cycle into two phases: swing and stance, while the foot trajectory from real insects suggests that it may be better to divide the swing phase into two sub-phases. The evidence supporting this hypothesis could be found in Figure 4.17, where insects clearly lift up the swing foot again during the second half of the swing period. This implies that the insect may employ different strategies in the first and second halves of the swing period.

Evaluation

Naturalness Naturalness is defined as the perceived realism of the character animation (Van Welbergen *et al.*, 2010). Measuring the naturalness of character animation is difficult because it is, for most of the time, subjectively evaluated by human observations, and determined by physics plausibility, style and variability etc of the synthetic motion (Van Welbergen *et al.*, 2010). Van Welbergen *et al.* (2010) summarises that currently there are two types of methods to assess the naturalness of the synthetic motion: comparison with captured motion data and user study evaluation. The synthetic motion produced by this framework is evaluated by these two methods respectively:

- Comparison with captured motion data. The naturalness of the synthetic motion is first quantitatively evaluated by comparing against the ground truth data from different aspects. The comparison shows that there exists a good match between the synthetic motion and the ground truth, in terms of the shape and location of supporting triangles (Figure 4.16 and Table 4.4), foot trajectory of a swing leg (Figure 4.17) and the switch mechanism during high speed motion (Figure 7.2). These results confirm the naturalness of the synthetic motion.

- User study evaluation. The synthetic animations are rated by both professional animators and non-professional students. The result (Figure 4.15) shows that the animations are rated as natural by an average score of 73.75%; in other words, almost three out of four trials are rated as natural by participants. This further confirms that this framework is able to synthesise natural insect animation. The suggestions from animators, such as the relative rotation between the head, thorax and abdomen, will be taken into consideration in future studies.

High level control Compared with simulation-based methods (Yin *et al.*, 2007; Wang *et al.*, 2009, 2012), this framework frees animators from calculating joint torques or adjusting control parameters, but still produces physically-plausible animation with the assistance from control patterns of the CPG controller. This is key to industry applications of simulation-based methods. With this framework, animators are allowed to directly manipulate the settings, such as path, speed, terrain etc because the design of the CLUT separates low-level joint actuations from high-level user commands.

This framework also provides high level control with another particular feature of the proposed CPG controller — *bifurcation*. The scenario of *forage* in Section 5.4 simulates a swarm of ants, which involves frequent interactions between individuals. In this case, the insect locomotion is characterised by sudden initialisation and termination. The result (Figure 5.14) shows that animators can easily simulate this sudden change in motion status by toggling the *bifurcation* parameter σ between -1 and 1.

Stability The stability of a controller refers to its capability of resisting external perturbations and recovering to normal locomotion states. The result in Section 5.4 shows that the character is able to resist various perturbations from different force directions (Table 5.4), at different timings (Table 5.5), and at different locomotion speeds (Table 5.6). This feature of stability requires no additional adaptations of the controller, and is an inherent benefit from the stable convergence of *Hopf Bifurcation* to either the *limit cycle* or the *attraction point*.

This framework also shows that compared with the case of the CPG alone, the complete controller (TPE+CLUT+CPG) demonstrates greater stability. This improvement is a result of the dynamic adaptation of the controller to the external environment. When a character walks against the continuous wind force in Figure 4.14d, the TPE increases the area of the supporting triangle and the character lowers its COM. Both strategies allow the character to resist larger perturbations for a longer period ($0.01N$ for 20 strides). In comparison, the CPG controller alone is only able to resist a smaller force for a shorter interval ($<0.01N$ for one stride in Table 5.4, 5.5, 5.6). This is discussed as one of the findings in this work as well.

Ease to design The criterion of *ease to design* concerns about the amount of labouring work, which is needed in the design process. Controller design, especially the process of parameter tuning, requires a significant amount of time and efforts in previous works (Yin *et al.*, 2007). Optimisation emerged as a successful solution in designing the locomotion controller (Wang *et al.*, 2009; Coros *et al.*, 2011; Wang *et al.*, 2012). In this thesis, the parameters of the CPG controller and the control look-up table are automatically determined with the optimisation strategy, which minimises the need for manual intervention.

However, character locomotion is normally a multi-objective problem, since the optimisation needs to consider various factors, including to minimise the energy consumption, to minimise the deviation from target positions or velocities, to minimise the head rotation etc. It is a common strategy to use a weighted-sum form to convert the multi-objective function into a single scalar objective function (Wang *et al.*, 2009; Coros *et al.*, 2011; Wang *et al.*, 2012). Existing works still determine the weight values for individual objectives by manual trial and error. This work first normalises each objective functions to a consistent magnitude [0, 1], and automatically selects the weights, which produce the smallest fitness value of the weighted-sum function (Figure 5.8 and 6.4). This further automates the process of controller design, proving that the framework in this thesis is ease of use to design a locomotion controller.

The current method is not without its limitations. Compared with body

movements of real insects, the synthetic motions of animated insects are of somewhat lesser fidelity. This can be explained, in part, by the absence of a fully actuated model. A better result is expected when a more faithful dynamics model is introduced that uses a physics engine such as Bullet, ODE or OpenSim. The current implementation uses the PD servo as the actuator to connect the controller and simulator. In several recent works on biped animation, non-linear muscles were demonstrated as a successful alternative to PD servo with enhanced naturalness and stability (Wang *et al.*, 2012; Geijtenbeek *et al.*, 2013). The integration of non-linear muscle into the current framework is a future direction. Another limitation is the scalability when building the CLUT. The performance complexity is $O(n^3)$ where n is the number of sample points for one dimension. A means of reducing the time cost to pre-compute the look-up table is to maintain low sampling rates while increasing the look-up frequency. Other methods include constructing a latent space based on current parametrisation, using either a linear one (PCA in Coros *et al.* (2008)), or a nonlinear one (GPLVM in Levine *et al.* (2012)).

This work bridges between biology and computer animation by selectively applying what is known about locomotion control in insects to design a locomotion controller for virtual insects. Although the neural control inside the insect is not completely understood, the few principles that are already known are rarely used to design a stable and natural controller in the area of character animation. This may partially be due to the lack of communication between these two fields. This thesis could be regarded as a moderate attempt to explore this less exploited area. The future research directions which can be derived from this thesis will be discussed shortly.

8.2 Future Research Directions

This work brings together a wide range of inter-disciplinary research in a novel context. It would be regarded as a starting point for my continuing research career. Based on the current implementation, there are three general directions that are worth exploring in the future.

One of the directions is to improve the proposed system by applying more

sophisticated models. Although the current implementation has taken many realistic features of insect locomotion into consideration, the model is still far from closely representing the locomotion of real insects. There are two major differences between the proposed model and a realistic insect: the actuator and controller. On one hand, the actuator in current implementation is PD servo, which is criticised for producing stiff movements. The use of non-linear muscles is expected to improve the naturalness of the synthetic motion and this has been demonstrated in recent success of biped animation (Wang *et al.*, 2012; Geijtenbeek *et al.*, 2013). One would also expect similar improvements after introducing a non-linear muscle model into the proposed framework. On the other hand, the current controller is modelled as a network of non-linear oscillators while the controller in real insects (the brain and ganglia) is composed of thousands of neurons, which send out pulse-like spiking signals. Various models have been proposed by experimental biologists to describe the behaviours of both a single neuron and large populations of neurons (Gerstner and Kistler, 2002). It is an intriguing problem to simulate large scale neuronal groups and to design their spiking patterns in order to activate joint muscles. Simple models, such as those proposed by Izhikevich *et al.* (2003), serve well as a starting point. However, the introduction of neurons at such a level presents further challenges in maintaining the stability of the controller and designing an intuitive interface for animators.

Insects posses a rich skill repertoire, which is well beyond the capability of the current implementation. It would be of practical application in film and animation productions to build a diversified controller library. Even more interesting is to connect locomotion skills to behaviour. There has been a recent surge in research that models the interaction of multiple characters (Ho *et al.*, 2010; Wampler *et al.*, 2010; Shum *et al.*, 2012; Hwang *et al.*, 2014); however, most are limited to biped animation. Insects demonstrate some interesting and specialised behaviours, from which the problem of foraging is among the most interesting. Still use ants as the exemplar of the insect group. Ant forage is an intelligent colony-level behaviour which emerges out of interactions among large numbers of individuals with limited competence. An interesting feature in their forage behaviour is group retrieval, whereby a group of ants collectively transports a large prey; an activity that would be impossible

for a single ant. This cooperation is special because there is no direct communications between individuals and no hierarchical coordination inside this swarm (Sudd, 1965; Berman *et al.*, 2010). Surprisingly, a group of ants is still able to achieve a highly-coordinated result without any sophisticated machinery. Most existing works on character animation simulate crowd behaviour by designing rules for a single agent while ignoring collaboration between agents (Peters and Ennis, 2009; Qiu and Hu, 2010). Such rules are not sufficient in the case of group transport of insects, which requires coordination of force and involves interactions with prey and the environment. To achieve the comparable result of the real equivalents, a solution should have following features:

- Scalable to simulate large swarm behaviour
- Flexible to author different shapes or changing positions of food supply
- Sufficiently stable to adapt to a complex environment, such as obstacles

The existing CPG implementation is modelled as a network of non-linear oscillators. A further extension to synchronise oscillators of different individuals in order to achieve group coordination is worth exploring (Cross *et al.*, 2004).

The work presented in this thesis draws inspiration from biology and endeavours to replicate the capability and flexibility of real insects. The encouraging results from this work could be regarded as a success in the field of bionics. Meanwhile, the question of *reverse engineering*, using the proposed simulation to answer the unattended questions in biology, is worth exploring. There is already a large collection of works in building neuron-biomechanics models to study the locomotion of insects, as reviewed in Chapter 2. Some questions, however, remain unaddressed, such as how can simulations be utilised to demonstrate the merits of non-linear muscles. For the classical *Hill*-type muscle model (Hill, 1938), the force produced by a muscle depends on its length and contraction velocity, which in turn are constantly changing while joints are rotating. This seems at first glance an *unfortunate* complication, while Loeb and Ghez (2000) proposed that the brain takes advantage of this property in order to create the joint stiffness and enhance gait stability.

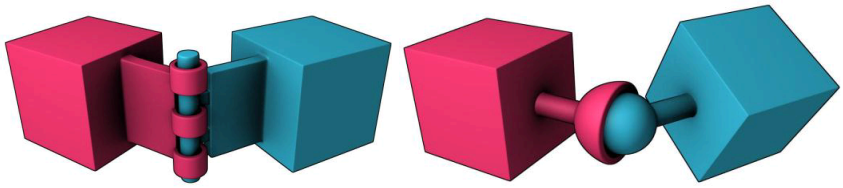
The reproduction of this feature in the proposed framework would provide further support to this claim. The other is to use the proposed framework to study the long-term interaction between species and environment. *Evolutionary theory* claims that large environmental differences lead to natural selection and thus individual variation. Locomotion, a vital skill for insect survival, plays a significant role in the evolution of their physical characteristics and structure. However, how the external environment affects their locomotion and the development of their internal organism is poorly understood. The optimisation strategy used in this thesis (Covariance Matrix Adaptation (CMA)), as one type of *Evolutionary Algorithm*, reflects the similarity between the methodology of optimisation and the process of evolution. From this perspective, the proposed simulation framework provides a potential solution to this kind of problem.

Appendices

Appendix A

Physics Simulation

A character is typically modelled as a hierarchy of interconnected rigid bodies assigned with kinematics (for example, length and radius) and dynamics (for example, mass and inertia) properties. Rigid bodies are connected to each other by constraints. Compared with the delicate structures of real animal joints, constraints are virtually modeled in a more simplified style, for example hinge or ballpoint constraints (Fig A.1).



(a) *Hinge Constraint* **(b)** *Point Constraint*

Figure A.1: *Hinge and point constraints in various physics engine, including* Open Dynamics Engine*(ODE) and* Bullet*. Different types of constraints determine how two connected rigid bodies could move relative to each other. For example, a hinge constraint has only one Degree of Freedom (DOF) which means two objects can only rotate around one axis while a point constraint allows two objects to rotate around three axes. But both constraints do not allow translational movements.*

The state of a character can be expressed using the global positions $\vec{X}_{com} = [x_{com}\ y_{com}\ z_{com}]$ and orientations $\vec{\Theta}_{com} = [\theta_x\ \theta_y\ \theta_z]$ of the Centre of Mass

(COM) and all joint angles (body configurations) $\vec{\Theta} = [\theta_1 \ \cdots \ \theta_n]$:

$$\vec{q} = \begin{bmatrix} x_{com} \\ y_{com} \\ z_{com} \\ \theta_x \\ \theta_y \\ \theta_z \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \tag{A.1}$$

In general, the problem of physics simulation can be divided into four sub-problems: Forward Kinematics (FK), Inverse Kinematics (IK), Forward Dynamics (FD), Inverse Dynamics (ID). Following paragraphs will explain these sub-problems shortly.

Forward Kinematics and Inverse Kinematics

Forward Kinematics When dealing with the environment (for example collision detection), it is often required to compute the position and orientation of a specific body part. This problem is defined as Forward Kinematics (FK), which aims to express the position (x, y, z) of the Centre of Mass (COM) of a rigid body and orientation (ϕ_x, ϕ_y, ϕ_z) of the body frame in the world space as a function of the joint angles:

$$\begin{bmatrix} x \\ y \\ z \\ \phi_x \\ \phi_y \\ \phi_z \end{bmatrix} = \begin{bmatrix} h_x(\theta_1, \cdots, \theta_n) \\ h_y(\theta_1, \cdots, \theta_n) \\ h_z(\theta_1, \cdots, \theta_n) \\ h_{\phi_x}(\theta_1, \cdots, \theta_n) \\ h_{\phi_y}(\theta_1, \cdots, \theta_n) \\ h_{\phi_z}(\theta_1, \cdots, \theta_n) \end{bmatrix} \tag{A.2}$$

Note here the global translation and rotation of the root are not considered. When needed, this global component can be added in a linear fashion.

Inverse Kinematics There exists an opposite problem to FK — Inverse Kinematics (IK), which is to compute the body configuration given the relative position of two body parts. This problem is of more interest when it comes to character control and trajectory generation. The solutions of IK can be categorized into two different ways: the analytical solution (Gan *et al.*, 2005) or the iterative solution (Buss, 2004).

Forward Dynamics and Inverse Dynamics

The ultimate problem in simulating character animation is to compute the force and torque applied at each joint. When it comes to dynamics, there are two equations that lay the foundation for the rigid body dynamics — the *Newton* equation:

$$\vec{F} = m\dot{\vec{\nu}} \tag{A.3}$$

and the *Euler* equation:

$$\vec{\tau} = \mathbf{I}\dot{\vec{\omega}} + \vec{\omega} \times \mathbf{I}\vec{\omega} \tag{A.4}$$

here $m, \mathbf{I}$ indicate the mass and inertia tensor of this rigid body, $\vec{\nu}, \vec{\omega}$ are its linear and rotational velocity.

Forward Dynamics The problem of Forward Dynamics (FD) is a fundamental component when designing a physics simulation engine. The challenges of simulating the trajectory of linked rigid bodies mainly lie in two aspects: collision detection and contact forces (Baraff, 2001) since the body parts are regarded as solid and do not allow inter-penetration. As long as the contact forces are computed correctly, the forces are applied on individual rigid bodies as though they are unconstrained.

Once the accelerations of the rigid bodies are calculated from the process of FD, the velocities and positions are updated by numeric integration. Two critical factors will affect, or even determine, the stability and performance

of the numeric integration: one is the choice of integration technique, and the other is size of time step (Geijtenbeek *et al.*, 2011). Implicit integration, opposite to the explicit integration, would normally allow greater time step at the same level of accuracy (Baraff, 2001). Smaller time step will improve the accuracy and stability of the integration, at the increasing cost of computing time.

Inverse Dynamics Out of the four sub-problems, Inverse Dynamics (ID) is most relevant to character control. In this case, an in-depth explanation is presented on a technical level to give readers a clear picture. The problem of ID is to calculate the joint torques when given the motion trajectory, and this problem is also called *Motion Tracking* or *Trajectory Tracking*. A typical formulation is *iterative Newton-Euler Dynamics* (Craig, 2004).

First, an outward process is carried out to compute the linear and rotational accelerations for individual rigid bodies composing the virtual character. It normally starts from the COM and propagates towards the end–effector using the following equations:

$$\begin{aligned} {}^{i+1}\omega_{i+1} &= {}_{i}^{i+1}R\,{}^{i}\omega_i + \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} \\ {}^{i+1}\nu_{i+1} &= {}_{i}^{i+1}R({}^{i}\nu_i + {}^{i}\omega_i \times {}^{i}P_{i+1}) \end{aligned} \tag{A.5}$$

${}_{i}^{i+1}R, {}^{i}P_{i+1}$ indicates the rotational matrix, linear translation of $i+1^{th}$ joint relative to i^{th} joint. ${}^{i+1}\hat{Z}_{i+1}$ assumes that the rotational axis in the body frame is the Z axis.

By applying the differential operators, we can derive the linear and rotational accelerations:

$$\begin{aligned} {}^{i+1}\dot{\omega}_{i+1} &= {}_{i}^{i+1}R\,{}^{i}\dot{\omega}_i + {}_{i}^{i+1}R\,{}^{i}\dot{\omega}_i \times \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} + \ddot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} \\ {}^{i+1}\dot{\nu}_{i+1} &= {}_{i}^{i+1}R({}^{i}\dot{\nu}_i + {}^{i}\omega_i \times {}^{i}P_{i+1} + {}^{i}\omega_i \times ({}^{i}\omega_i \times {}^{i}P_{i+1})) \\ {}^{i+1}\dot{\nu}_{Ci+1} &= {}^{i}\dot{\nu}_i + {}^{i}\omega_i \times {}^{i}P_{Ci+1} + {}^{i}\omega_i \times ({}^{i}\omega_i \times {}^{i}P_{Ci+1}) \end{aligned} \tag{A.6}$$

${}^{i+1}\dot{\nu}_{Ci+1}$ denotes the linear velocity of COM of the $i+1^{th}$ body in $i+1^{th}$ joint frame. ${}^{i}P_{Ci+1}$ denotes the translation of COM of the $i+1^{th}$ body in i^{th} joint frame.

By substituting the Equation A.6 into Equation A.3 and A.4, we can calculate the force and torque applied at the Centre of Mass (COM) of each rigid body:

$$\begin{aligned} {}^{i+1}\vec{F}_{i+1} &= m_{i+1}\,{}^{i+1}\dot{\nu}_{Ci+1} \\ {}^{i+1}\vec{\tau}_{i+1} &= {}^{i+1}I_{i+1}\,{}^{i+1}\dot{\omega}_{i+1} + {}^{i+1}\omega_{i+1} \times {}^{i+1}I_{i+1}\,{}^{i+1}\omega_{i+1} \end{aligned} \tag{A.7}$$

The next step is to convert the force and torque applied at each rigid body into the original torque from each joint. This process is an inward iteration starting from the end–effector towards the COM. We define:

f_i = force applied on rigid body i by rigid body i-1

τ_i = torque applied on rigid body i by rigid body i-1

By analyzing the force–balance and torque–balance of each rigid body, we can derive the iterative equations for computing the joint torques:

$$\begin{aligned} {}^{i}f_i &= {}^{i}_{i+1}R\,{}^{i+1}f_{i+1} + {}^{i}F_i \\ {}^{i}\tau_i &= {}^{i}_{i+1}R\,{}^{i+1}\tau_{i+1} + {}^{i}P_{Ci} \times {}^{i}F_i + {}^{i}P_{i+1} \times {}^{i}_{i+1}R\,{}^{i+1}f_{i+1} + {}^{i}\tau_i \end{aligned} \tag{A.8}$$

The torque for this joint is simply the $\hat{Z}$ component of the ${}^{i}\tau_i$.

The final representation of the torque in terms of the joint angles and character–specific properties can often be rewritten in this fashion:

$$\vec{\tau} = \mathbf{M}(\vec{\Theta})\ddot{\vec{\Theta}} + C(\vec{\Theta}, \dot{\vec{\Theta}}) + G(\vec{\Theta}) \tag{A.9}$$

Parameters $M(\vec{\Theta})$, $C(\vec{\Theta}, \dot{\vec{\Theta}})$ and $G(\vec{\Theta})$ are the joint space inertia matrix, centrifugal/Coriolis and gravitational forces respectively.

Appendix B

Physics Model

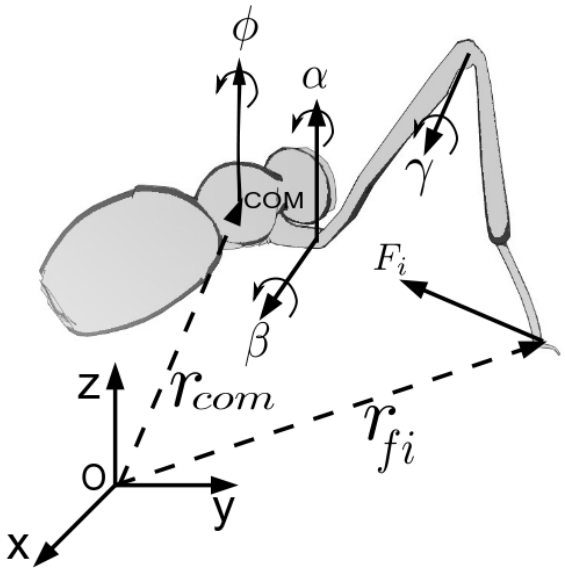


Figure B.1: *A standing leg of the character in this implementation. Each leg has three DOFs:* α, β, γ *(shown in this figure).* $\mathbf{r}_{COM}$ *denotes the world coordinate of the Centre of Mass (COM) and* $\mathbf{r}_{fi}$ *denotes the world coordinate of* i^{th} *foot.* $\mathbf{F}_i$ *denotes the reaction force with the ground on* i^{th} *foot.*

The dynamics complexity is simplified by making two assumptions: a) the main body is regarded as a single rigid body; b) the legs are massless. These assumptions are based on the experimental result that the legs of insects only possess a small proportion of mass compared with whole body (6% in cockroach (Kram *et al.*, 1997)).

With the simplified model, the state of the system is reduced as $\vec{q} =$

$[x, y, z, \phi]$, where x, y, z are the world coordinates of the COM and ϕ denotes the angle between heading direction and $+x$ axis. Each leg has three DOFs: $\vec{\Theta} = [\alpha, \beta, \gamma]$, two on the hip joint, the third on the knee joint. α corresponds to the thorax-coxa (ThC) joint, which is responsible for the protraction and retraction of the leg. β corresponds to the coxa-trochanter (CTr) joint, which is responsible for the levitation and depression of the leg. γ corresponds to the femur-tibia (FTi) joint, which is responsible for the extension and flexion of the leg.

Readers could refer to Figure B.1 for visual notation. The forward dynamics is governed by the following equations:

$$\begin{bmatrix} \mathbf{M} & 0 \\ 0 & I \end{bmatrix} \ddot{\mathbf{q}} = \begin{bmatrix} \mathbf{F} \\ \tau \end{bmatrix} \tag{B.1}$$

$$\mathbf{F} = \sum_{i=1}^{3} \mathbf{F}_i - \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix}$$

$$\tau = \sum_{i=1}^{3} (\mathbf{r_{fi}} - \mathbf{r_{com}}) \times \mathbf{F}_i$$

where M is the mass matrix and I is the angular inertial matrix, the gravitational constant $g = 9.8kg{\cdot}m/s^2$ (please also refer to Figure B.1 for visual notation of other variables).

Instead of using *Coulomb's Law* to model friction forces, these frictional forces are explicitly equating to the horizontal components of ground reaction force. To compute the ground reaction force $\mathbf{F}_i$, the forces acting against the ground by end–effectors are computed in terms of joint torques by *Virtual Control Model* (Pratt *et al.*, 1997):

$$\mathbf{F'}_i = (\mathbf{J}^T)^{-1} \begin{bmatrix} \tau_\alpha \\ \tau_\beta \\ \tau_\gamma \end{bmatrix} \tag{B.2}$$

The Jacobian $\mathbf{J}$ can be derived from the relationship between joint velocity and end effector velocity (Craig, 2004). Then the ground reaction force $\mathbf{F}_i$ on

the character is opposite to the active force $\mathbf{F}'_i$ from the character (*Newton's Third Law*). The foot contact is maintained by an analytic *Inverse Kinematics* solution (Tolani *et al.*, 2000).

The Jacobian $\mathbf{J}$ matrix of each leg takes the following form:

$$\mathbf{J} = \begin{bmatrix} (L_{tibia}S_{\beta+\gamma} + L_{femur}S_{\beta})C_{\alpha} & (L_{tibia}C_{\beta+\gamma} + L_{femur}C_{\beta})S_{\alpha} & L_{tibia}C_{\beta+\gamma}S_{\alpha} \\ (L_{tibia}S_{\beta+\gamma} + L_{femur}S_{\beta})S_{\alpha} & -(L_{tibia}C_{\beta+\gamma} + L_{femur}C_{\beta})C_{\alpha} & -L_{tibia}C_{\beta+\gamma}C_{\alpha} \\ 0 & -(L_{tibia}S_{\beta+\gamma} + L_{femur}S_{\beta}) & -L_{tibia}S_{\beta+\gamma} \end{bmatrix} \tag{B.3}$$

$$\begin{aligned} S_{\alpha} &= sin(\alpha) \\ C_{\alpha} &= cos(\alpha) \\ S_{\beta} &= sin(\beta) \\ C_{\beta} &= cos(\beta) \\ S_{\beta+\gamma} &= sin(\beta+\gamma) \\ C_{\beta+\gamma} &= cos(\beta+\gamma) \end{aligned}$$

L_{femur} and L_{tibia} are the lengths of the femur and tibia respectively.

Appendix C

SwingNet 2

In the locomotion of biped characters, such as human beings, their legs take a big portion of the total body mass, and their movement and rotation have to be simulated dynamically to generate good quality animation (Ha *et al.*, 2012; Muico *et al.*, 2009). Insects are different. The total weight of six legs of a ant is about $1/20$ of its body weight, therefore, its contribution to the body dynamics is trivial in the simulation and can be ignored without causing degeneration to the results.

Based on previous observations, an artificial neural network model, *Swingnet 2* (Schumm and Cruse, 2006), is introduced to simulate the swing of insect leg. The *Swingnet 2* model calculates the velocity of the rotation angle $\mathbf{q} = [\alpha, \beta, \gamma]$ explicitly as,

$$\frac{d\mathbf{q}}{dt} = A\mathbf{q} - B\mathbf{q}_{AEP} + C(t) \tag{C.1}$$

$$A = \begin{bmatrix} \frac{3.4}{T_s} & 0 & 0 \\ 0 & \frac{6}{T_s} & 0 \\ 0 & 0 & \frac{3}{T_s} \end{bmatrix}, B = \begin{bmatrix} \frac{3}{T_s} & 0 & 0 \\ 0 & \frac{6}{T_s} & 0 \\ 0 & 0 & \frac{3}{T_s} \end{bmatrix}, C = \begin{bmatrix} 0 \\ \frac{-t}{5T_s} \\ 0 \end{bmatrix}$$

$\mathbf{q}_{AEP}$ is the joint angle values at the Anterior Extreme Position.

Appendix D

Table of Coefficients

Variable	Symbol	Value	Unit
Mass of ant	m	0.001	kg
Moment of Inertia	I	0.0000001	$kg \cdot m^2$
Oscillator frequency at normal state	ω	10	Hz
Oscillator bifurcation parameter	σ	1	
Oscillator convergence parameter	λ	10	
Gravity acceleration	g	9.8	m/s^2
Average height of COM	h	2	mm
Intercept of Equation 4.4	S_0	3	mm
Slope of Equation 4.4	a_s	0.1	s
Slope coefficients of ΔL_{COM} and m_w	a_{com}	0.1	mm/kg
Length of Trunk	L_{trunk}	5	mm
Length of Coxa	L_{coxa}	0.3	mm
Length of Femur (Front/Middle/Back)	L_{femur}	1.3775/1.3848/2.2705	mm
Length of Tibia (Front/Middle/Back)	L_{tibia}	1.0120/1.1580/1.9163	mm
Length of Tarsus (Front/Middle/Back)	L_{tarsus}	1.2310/1.3562/2.1075	mm

Table D.1: *Constants used in the simulation. As the only data we can access are for Cataglyphis fortis, other ant species may take different coefficients.*

References

Abdul Karim A., Gaudin T., Meyer A., Buendia A. and Bouakaz S., July 2012a. Procedural Locomotion of Multi-Legged Characters in Dynamic Environments. *Computer Animation and Virtual Worlds*.

Abdul Karim A., Meyer A., Gaudin T., Buendia A. and Bouakaz S., December 2012b. Generic spine model with simple physics for life-like quadrupeds and reptiles. In *VRIPHYS 2012: 9th Workshop on Virtual Reality Interaction and Physical Simulation*, 97–106.

Abràmoff M. D., Magalhães P. J. and Ram S. J., 2004. Image processing with imagej. *Biophotonics international*, **11**(7), 36–42.

Alexander R. M., 1984. The gaits of bipedal and quadrupedal animals. *The International Journal of Robotics Research*, **3**(2), 49–59.

Alexander R. M., 1991. Energy-saving mechanisms in walking and running. *Journal of Experimental Biology*, **160**(1), 55–69.

Alexander R. M., 2002. Tendon elasticity and muscle function. *Comparative Biochemistry and Physiology Part A: Molecular & Integrative Physiology*, **133**(4), 1001–1011.

Altendorfer R., Moore N., Komsuoglu H., Buehler M., Brown H., McMordie D., Saranli U., Full R. and Koditschek D., 2001. Rhex: A biologically inspired hexapod runner. *Autonomous Robots*, **11**, 207–213.

Antoine F., 2014. Walking ant in slow-motion. Accessed: 22 July 2015.

Baraff D., 2001. Physically based modeling: Rigid body simulation. *SIGGRAPH Course Notes, ACM SIGGRAPH*, **2**(1), 2–1.

Bässler U. and Büschges A., 1998. Pattern generation for stick insect walk-

ing movements — multisensory control of a locomotor program. *Brain Research Reviews*, **27**(1), 65–88.

Beaudoin P., Coros S., van de Panne M. and Poulin P., 2008. Motion-motif graphs. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 117–126.

Beer R. D., Quinn R. D., Chiel H. J. and Ritzmann R. E., 1997. Biologically inspired approaches to robotics: What can we learn from insects? *Communications of the ACM*, **40**(3), 30–38.

Berman S., Lindsey Q., Sakar M. S., Kumar V. and Pratt S., 2010. Study of group food retrieval by ants as a model for multi-robot collective transport strategies. In *Robotics: Science and Systems*. Citeseer.

Bläsing B., 2006. Crossing large gaps: a simulation study of stick insect behavior. *Adaptive Behavior*, **14**(3), 265–285.

Blmel M. *Locomotor system simulations and muscle modelling in stick insects (Carausius morosus)*. PhD thesis, Universitt zu Köln, 2011.

Boeing A. and Bräunl T., 2007. Evaluation of real-time physics simulation systems. In *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*. ACM, 281–288.

Buchli J., Righetti L. and Ijspeert A. J., 2006. Engineering entrainment and adaptation in limit cycle systems. *Biological Cybernetics*, **95**(6), 645–664.

Büschges A., Schmitz J. and Bässler U., 1995. Rhythmic patterns in the thoracic nerve cord of the stick insect induced by pilocarpine. *The Journal of Experimental Biology*, **198**(2), 435–56.

Buss S. R., 2004. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. *IEEE Journal of Robotics and Automation*, **17**.

Cenydd L. a. and Teahan B., 2013. An embodied approach to arthropod animation. *Computer Animation and Virtual Worlds*, **24**(1), 65–83.

Chapman R. F., 1998. *The insects: structure and function*. Cambridge university press.

Chung S.-J. and Dorothy M., 2010. Neurobiologically inspired control of engineered flapping flight. *Journal of guidance, control, and dynamics*, **33** (2), 440–453.

Collins J. J. and Stewart I. N., 1993. Coupled nonlinear oscillators and the symmetries of animal gaits. *Journal of Nonlinear Science*, **3**(1), 349–392.

Collins S., Ruina A., Tedrake R. and Wisse M., 2005. Efficient bipedal robots based on passive-dynamic walkers. *Science*, **307**(5712), 1082–1085.

Coros S., Beaudoin P. and van de Panne M., 2010. Generalized biped walking control. *ACM Transactions on Graphics (TOG)*, **29**(4), 130.

Coros S., Beaudoin P., Yin K. K. and van de Pann M., December 2008. Synthesis of constrained walking skills. *ACM Trans. Graph.*, **27**(5), 113:1–113:9.

Coros S., Karpathy A., Jones B., Reveret L. and van de Panne M., August 2011. Locomotion skills for simulated quadrupeds. *ACM Trans. Graph.*, **30**, 59:1–59:12.

Craig J. J., August 2004. *Introduction to Robotics: Mechanics and Control (3rd Edition)*. Prentice Hall, 3 edition.

Crespi A., Lachat D., Pasquier A. and Ijspeert A. J., 2008. Controlling swimming and crawling in a fish robot using a central pattern generator. *Autonomous Robots*, **25**(1-2), 3–13.

Cross M., Zumdieck A., Lifshitz R. and Rogers J., 2004. Synchronization by nonlinear frequency pulling. *Physical review letters*, **93**(22), 224101.

Cruse H., 1985. Which parameters control the leg movement of a walking insect?: Ii. the start of the swing phase. *Journal of Experimental Biology*, **116**(1), 357–362.

Cruse H., Kindermann T., Schumm M., Dean J. and Schmitz J., 1998. Walkneta biologically inspired network to control six-legged walking. *Neural networks*, **11**(7), 1435–1447.

de Lasa M., Mordatch I. and Hertzmann A., July 2010. Feature-based locomotion controllers. *ACM Trans. Graph.*, **29**(4), 131:1–131:10.

Delcomyn F., 1999. Walking robots and the central and peripheral control of locomotion in insects. *Autonomous Robots*, **7**(3), 259–270.

Dürr V., 2005. Context-dependent changes in strength and efficacy of leg coordination mechanisms. *Journal of experimental biology*, **208**(12), 2253–2267.

Dürr V. and Ebeling W., 2005. The behavioural transition from straight to curve walking: kinetics of leg movement parameters and the initiation of turning. *Journal of experimental biology*, **208**(12), 2237–2252.

Dutra M., de Pina Filho A. and Romano V., 2003. Modeling of a bipedal locomotor using coupled nonlinear oscillators of van der pol. *Biological Cybernetics*, **88**(4), 286–292.

Ekeberg Ö., 1993. A combined neuronal and mechanical model of fish swimming. *Biological cybernetics*, **69**(5-6), 363–374.

Fang J., Jiang C. and Terzopoulos D., 2013. Modeling and animating myriapoda: a real-time kinematic/dynamic approach. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '13, New York, NY, USA. ACM, 203–212.

Franks N. R., May 1986. Teams in social insects: group retrieval of prey by army ants (eciton burchelli, hymenoptera: Formicidae). *Behavioral Ecology and Sociobiology*, **18**(6), 425–429.

Full R. J. and Tu M. S., 1991. Mechanics of a rapid running insect: two-, four- and six-legged locomotion. *Journal of Experimental Biology*, **156** (1), 215–231.

Full R. J. and Tu M. S., 1990. Mechanics of six-legged runners. *Journal of Experimental Biology*, **148**(1), 129–146.

Gan J. Q., Oyama E., Rosales E. M. and Hu H., 2005. A complete analytical solution to the inverse kinematics of the pioneer 2 robotic arm. *Robotica*, **23**(1), 123–129.

Geijtenbeek T., Pronost N., Egges A. and Overmars M. H., 2011. Interactive character animation using simulated physics. *Eurographics - State of the Art Reports*.

Geijtenbeek T., van de Panne M. and van der Stappen A. F., 2013. Flexible muscle-based locomotion for bipedal creatures. *ACM Transactions on Graphics (TOG)*, **32**(6), 206.

Gerstner W. and Kistler W. M., 2002. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press.

Ghigliazza R. and Holmes P., 2005. Towards a neuromechanical model for insect locomotion: Hybrid dynamical systems. *Regular and Chaotic Dynamics*, **10**(2), 193–225.

Gibson D. P., Oziem D. J., Dalton C. J. and Campbell N. W., 2005. Capture and synthesis of insect motion. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '05, New York, NY, USA. ACM, 39–48.

Gibson D. P., Oziem D., Dalton C. J. and Campbell N. W., 2007. A system for the capture and synthesis of insect motion. *Graphical Models*, **69**(5), 231–245.

Giovanni S. and Yin K. 2011. 227–241. Locotest: deploying and evaluating physics-based locomotion on multiple simulation platforms. In *Motion in Games*, Springer.

Gleicher M., 1998. Retargetting motion to new characters. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, 42.

Golubitsky M., Stewart I., Buono P.-L. and Collins J., 1999. Symmetry in locomotor central pattern generators and animal gaits. *Nature*, **401**(6754), 693–695.

Grodzevich O. and Romanko O., 2006. Normalization and other topics in multi-objective optimization. *Proceedings of the FieldsMITACS Industrial Problems Workshop*.

Guo S., Chang J., Cao Y. and Zhang J., 2014a. A novel locomotion synthesis

and optimisation framework for insects. *Computers & Graphics*, **38**, 78 – 85.

Guo S., Southern R., Chang J., Greer D. and Zhang J. J., 2014b. Adaptive motion synthesis for virtual characters: a survey. *The Visual Computer*, 1–16.

Ha S., Ye Y. and Liu C. K., November 2012. Falling and landing motion control for character animation. *ACM Trans. Graph.*, **31**(6), 155:1–155:9.

Hansen N., 2006. The cma evolution strategy: A comparing review. In *Towards a New Evolutionary Computation*, volume 192 of *Studies in Fuzziness and Soft Computing*. Springer Berlin / Heidelberg, 75–102.

Hansen N. and Kern S., 2004. Evaluating the cma evolution strategy on multimodal test functions. *Parallel Problem Solving from Nature - PPSN VIII*, **3242**, 282–291.

Hansen N., Müller S. D. and Koumoutsakos P., March 2003. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evol. Comput.*, **11**(1), 1–18.

Heck R. and Gleicher M., 2007. Parametric motion graphs. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*. ACM, 129–136.

Hellgren J., Grillner S. and Lansner A., 1992. Computer simulation of the segmental neural network generating locomotion in lamprey by using populations of network interneurons. *Biological cybernetics*, **68**(1), 1–13.

Hill A., 1938. The heat of shortening and the dynamic constants of muscle. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 136–195.

Hirai K., Hirose M., Haikawa Y. and Takenaka T., 1998. The development of honda humanoid robot. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 2. IEEE, 1321–1326.

Ho E. S., Komura T. and Tai C.-L., 2010. Spatial relationship preserving character motion adaptation. In *ACM Transactions on Graphics (TOG)*, volume 29. ACM, 33.

Holmes P., Full R. J., Koditschek D. and Guckenheimer J., 2006. The dynamics of legged locomotion: Models, analyses, and challenges. *Siam Review*, **48**(2), 207–304.

Hwang J., Suh I. and Kwon T., 2014. Editing and synthesizing two-character motions using a coupled inverted pendulum model. In *Computer Graphics Forum*, volume 33. Wiley Online Library, 21–30.

Ijspeert A., May 2008. Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, **21**(4), 642–653.

Izhikevich E. M. and others , 2003. Simple model of spiking neurons. *IEEE Transactions on neural networks*, **14**(6), 1569–1572.

Jansen S. E. and van Welbergen H., 2009. Methodologies for the user evaluation of the motion of virtual humans. In *Intelligent Virtual Agents*. Springer, 125–131.

Ji A., Han L. and Dai Z., 2011. Adhesive contact in animal: morphology, mechanism and bio-inspired application. *Journal of Bionic Engineering*, **8** (4), 345–356.

Jindrich D. L. and Full R. J., 1999. Many-legged maneuverability: dynamics of turning in hexapods. *Journal of experimental biology*, **202**(12), 1603–1623.

Jones A. S., July 2008. Fantastic ants did you know? National Geographic Magazine.

Katz P. S. and Hooper S. L., 2007. Invertebrate central pattern generators. *Cold Spring Harbor Monograph Archive*, **49**, 251–279.

Kim I. Y. and De Weck O., 2006. Adaptive weighted sum method for multiobjective optimization: a new method for pareto front generation. *Structural and Multidisciplinary Optimization*, **31**(2), 105–116.

Kovar L., Gleicher M. and Pighin F., 2008. Motion graphs. In *ACM SIGGRAPH 2008 classes*, SIGGRAPH '08, New York, NY, USA. ACM, 51:1–51:10.

Kram R., Wong B. and Full R. J., July 1997. Three-dimensional kinematics

and limb kinetic energy of running cockroaches. *J Exp Biol*, **200**(13), 1919–1929.

Kukillaya R. P. and Holmes P., 2009. A model for insect locomotion in the horizontal plane: feedforward activation of fast muscles, stability, and robustness. *Journal of theoretical biology*, **261**(2), 210–226.

Kukillaya R. P. and Holmes P. J., March 2008. A hexapedal jointed-leg model for insect locomotion in the horizontal plane. *Biol. Cybern.*, **97**(5), 379–395.

Laszlo J., van de Panne M. and Eugene F., 1996. Limit cycle control and its application to the animation of balancing and walking. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, New York, NY, USA. ACM, 155–162.

Lau M., Bar-Joseph Z. and Kuffner J., 2009. Modeling spatial and temporal variation in motion data. *ACM Transactions on Graphics (TOG)*, **28**(5), 1–10.

Lau M. and Kuffner J. J., 2006. Precomputed search trees: planning for interactive goal-driven animation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, 299–308.

Lee Y., Kim S. and Lee J., July 2010. Data-driven biped control. *ACM Trans. Graph.*, **29**, 129:1–129:8.

Levine S., Wang J. M., Haraux A., Popović Z. and Koltun V., 2012. Continuous character control with low-dimensional embeddings. *ACM Transactions on Graphics (TOG)*, **31**(4), 28.

Liu C. K. and Popović Z., 2002. Synthesis of complex dynamic character motion from simple animations. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques - SIGGRAPH '02*, San Antonio, Texas. 408.

Liu C., Chen Q. and Wang D., 2011. Cpg-inspired workspace trajectory generation and adaptive locomotion control for quadruped robots. *Systems,*

Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, **41**(3), 867–880.

Liu F., Southern R., Guo S., Yang X. and Zhang J., 2012. Motion adaptation with motor invariant theory. *Cybernetics, IEEE Transactions on*, **PP**(99), 1–15.

Lockwood N. and Singh K., 2012. Fingerwalking: motion editing with contact-based hand performance. In *Proceedings of the 11th ACM SIGGRAPH/Eurographics conference on Computer Animation*. Eurographics Association, 43–52.

Loeb G. E. and Ghez C., 2000. *The Motor Unit and Muscle Action. In: Principles of neural science*, volume 4. McGraw-Hill New York.

Marler R. T. and Arora J. S., 2004. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, **26** (6), 369–395.

Marsden J. E. and McCracken M., 1976. *The Hopf Bifurcation and Its Applications*. Springer-Verlag, New York, NY.

Matsuoka K., 1985. Sustained oscillations generated by mutually inhibiting neurons with adaptation. *Biological Cybernetics*, **52**(1), 345–353.

McGeer T., 1990. Passive dynamic walking. *the international journal of robotics research*, **9**(2), 62–82.

McKenna M. and Zeltzer D., September 1990. Dynamic simulation of autonomous legged locomotion. *SIGGRAPH Comput. Graph.*, **24**(4), 29–38.

Meijering E., Dzyubachyk O., Smal I. and others , 2012. Methods for cell and particle tracking. *Methods Enzymol*, **504**(9), 183–200.

Mellen N., Kiemel T. and Cohen A. H., 1995. Correlational analysis of fictive swimming in the lamprey reveals strong functional intersegmental coupling. *Journal of Neurophysiology*, **73**(3), 1020–1030.

Moll K., Roces F. and Federle W., 2010. Foraging grass-cutting ants (atta vollenweideri) maintain stability by balancing their loads with controlled head movements. *Journal of Comparative Physiology A*, **196**(7), 471–480.

Morimoto J., Endo G., Nakanishi J. and Cheng G., feb. 2008. A biologically inspired biped locomotion strategy for humanoid robots: Modulation of sinusoidal patterns by a coupled oscillator model. *Robotics, IEEE Transactions on*, **24**(1), 185 –191.

Muico U., Lee Y., Popović J. and Popović Z., 2009. Contact-aware nonlinear control of dynamic characters. In *ACM Transactions on Graphics (TOG)*, volume 28. ACM, 81.

Muico U., Popović J. and Popović Z., May 2011. Composite control of physically simulated characters. *ACM Trans. Graph.*, **30**, 16:1–16:11.

Nash S. G., 1984. Newton-type minimization via the lanczos method. *SIAM Journal on Numerical Analysis*, **21**(4), 770–788.

Örjan Ekeberg A. B., Marcus Blümel, 2004. Dynamic simulation of insect walking. *Arthropod Structure & Development*, **33**(3), 287 – 300.

Pejsa T. and Pandzic I., 2010. State of the art in example-based motion synthesis for virtual characters in interactive applications. *Computer Graphics Forum*, **29**(1), 202–226.

Peters C. and Ennis C., 2009. Modeling groups of plausible virtual pedestrians. *IEEE Computer Graphics and Applications*, **29**(4), 54–63.

Powell M. J., 1964. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The computer journal*, **7**(2), 155–162.

Pratt J., Chew C., Torres A., Dilworth P. and Pratt G., 2001. Virtual model control: An intuitive approach for bipedal locomotion. *The International Journal of Robotics Research*, **20**(2), 129.

Pratt J., Dilworth P. and Pratt G., 1997. Virtual model control of a bipedal walking robot. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 1, 193–198 vol.1.

Qinxin Y. *A decision network framework for the behavioral animation of virtual humans*. PhD thesis, University of Toronto, Ontario, Toronto, Canada, 2007.

Qiu F. and Hu X., 2010. Modeling group structures in pedestrian crowd simulation. *Simulation Modelling Practice and Theory*, **18**(2), 190–205.

Righetti L. and Ijspeert A. J., 2006. Programmable Central Pattern Generators: an application to biped locomotion control. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*.

Rother D. D. *Bayesian network applications in molecular biology, computer graphics and computer vision*. PhD thesis, University of Minnesota, Minneapolis, MN, USA, 2008.

Safonova A. and Hodgins J. K., 2007. Construction and optimal search of interpolated motion graphs. In *ACM Transactions on Graphics (TOG)*, volume 26. ACM, 106.

Salkind N. J., 2010. *Encyclopedia of research design*, volume 1. Sage.

Saranli U., Buehler M. and Koditschek D. E., 2001. Rhex: A simple and highly mobile hexapod robot. *International Journal of Robotics Research*, **20**, 616–631.

Schilling M., Hoinville T., Schmitz J. and Cruse H., 2013. Walknet, a bio-inspired controller for hexapod walking. *Biological cybernetics*, **107**(4), 397–419.

Schilling M., Paskarbeit J., Schmitz J., Schneider A. and Cruse H., 2012. Grounding an internal body model of a hexapod walker control of curve walking in a biologically inspired robot. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2762–2768.

Schumm M. and Cruse H., 2006. Control of swing movement: influences of differently shaped substrate. *Journal of Comparative Physiology A*, **192** (10), 1147–1164.

Seidl T. and Wehner R., 2008. Walking on inclines: how do desert ants monitor slope and step length. *Frontiers in zoology*, **5**(8).

Seol Y., O'Sullivan C. and Lee J., 2013. Creature features: online motion puppetry for non-human characters. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '13, New York, NY, USA. ACM, 213–221.

Shepard D., 1968. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*. ACM, 517–524.

Shum H. P. H., Komura T. and Yamazaki S., 2012. Simulating multiple character interactions with collaborative and adversarial goals. *Visualization and Computer Graphics, IEEE Transactions on*, **18**(5), 741–752.

Singh S., Kapadia M., Reinman G. and Faloutsos P., 2011. Footstep navigation for dynamic crowds. *Computer Animation and Virtual Worlds*, **22** (2-3), 151–158.

Skrba L., Revéret L., Hétroy F., Cani M.-P. and O'sullivan C., 2008. Quadruped animation. In *Eurographics '08, State-of-the-Art Report, EG-STAR, April, 2008*, Crete, Grèce. Eurographics Association, Eurographics Association, 7–23.

Steven S., 1994. *Nonlinear dynamics and chaos with applications to physics, biology, chemistry and engineering*. Westview Press.

Storrer J. *Systemanalytische Untersuchungen am" Kniesehnenreflex" der Stabheuschrecke Carausius morosus Br.(Orthoptera).* PhD thesis, Universität of Kaiserslautern, 1976.

Sudd J. H., 1965. The transport of prey by ants. *Behaviour*, 234–271.

Szczecinski N. S., Brown A. E., Bender J. A., Quinn R. D. and Ritzmann R. E., 2014. A neuromechanical simulation of insect walking and transition to turning of the cockroach blaberus discoidalis. *Biological cybernetics*, **108**(1), 1–21.

Taga G., Yamaguchi Y. and Shimizu H., 1991. Self-organized control of bipedal locomotion by neural oscillators in unpredictable environment. *Biological Cybernetics*, **65**, 147–159.

Takwale R. and Puranik P., 1979. *Introduction to classical mechanics*. Tata McGraw-Hill Education.

Tan J., Gu Y., Turk G. and Liu C. K., August 2011. Articulated swimming creatures. *ACM Trans. Graph.*, **30**, 58:1–58:12.

Ting L., Blickhan R. and Full R., 1994. Dynamic and static stability in hexapedal runners. *Journal of Experimental Biology*, **197**(1), 251–269.

Tolani D., Goswami A. and Badler N. I., 2000. Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical models*, **62**(5), 353–388.

Tsai Y.-Y., Lin W.-C., Cheng K., Lee J. and Lee T.-Y., march-april 2010. Real-time physics-based 3d biped character animation using an inverted pendulum model. *Visualization and Computer Graphics, IEEE Transactions on*, **16**(2), 325 –337.

van Basten B. J., Peeters P. and Egges A., 2010. The step space: example-based footprint-driven motion synthesis. *Computer Animation and Virtual Worlds*, **21**(3-4), 433–441.

Van Welbergen H., Van Basten B. J., Egges A., Ruttkay Z. M. and Overmars M. H., 2010. Real time animation of virtual humans: A trade-off between naturalness and control. In *Computer Graphics Forum*, volume 29. Wiley Online Library, 2530–2554.

Vukobratović M. and Borovac B., 2004. Zero-moment pointthirty five years of its life. *International Journal of Humanoid Robotics*, **1**(01), 157–173.

Wampler K. and Popović Z., 2009. Optimal gait and form for animal locomotion. *ACM Transactions on Graphics (TOG)*, **28**(3), 1–8.

Wampler K., Andersen E., Herbst E., Lee Y. and Popović Z., 2010. Character animation in two-player adversarial games. *ACM Transactions on Graphics (TOG)*, **29**(3), 26.

Wang J. M., Fleet D. J. and Hertzmann A., December 2009. Optimizing walking controllers. *ACM Trans. Graph.*, **28**(5), 168:1–168:8.

Wang J. M., Hamner S. R., Delp S. L. and Koltun V., July 2012. Optimizing locomotion controllers using biologically-based actuators and objectives. *ACM Trans. Graph.*, **31**(4), 25:1–25:11.

Wei X., Min J. and Chai J., 2011. Physically valid statistical models for human motion generation. *ACM Transactions on Graphics (TOG)*, **30**(3), 19.

Wright M. H., 1996. Direct search methods: Once scorned, now respectable. *Pitman Research Notes in Mathematics Series*, 191–208.

Wu J.-c. and Popović Z., July 2010. Terrain-adaptive bipedal locomotion control. *ACM Trans. Graph.*, **29**, 72:1–72:10.

Yamane K., Ariki Y. and Hodgins J., 2010. Animating non-humanoid characters with human motion data. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 169–178.

Yin K., Loken K. and van de Panne M., July 2007. Simbicon: simple biped locomotion control. *ACM Trans. Graph.*, **26**(3).

Zajac F. E., 1988. Muscle and tendon: properties, models, scaling, and application to biomechanics and motor control. *Critical reviews in biomedical engineering*, **17**(4), 359–411.

Zill S., Schmitz J. and Büschges A., 2004. Load sensing and control of posture and locomotion. *Arthropod structure & development*, **33**(3), 273–286.

Zollikofer C., 1994a. Stepping patterns in ants-influence of body morphology. *Journal of Experimental Biology*, **192**(1), 107–118.

Zollikofer C., 1994b. Stepping patterns in ants-influence of load. *Journal of Experimental Biology*, **192**(1), 119–127.

Zollikofer C., 1994c. Stepping patterns in ants-influence of speed and curvature. *Journal of experimental biology*, **192**(1), 95–106.

Zordan V., Macchietto A., Medina J., Soriano M. and Wu C.-C., 2007. Interactive dynamic response for games. In *Proceedings of the 2007 ACM SIGGRAPH Symposium on Video Games*, Sandbox '07, New York, NY, USA. ACM, 9–14.

Zordan V. B., Majkowska A., Chiu B. and Fast M., July 2005. Dynamic response for motion capture animation. *ACM Trans. Graph.*, **24**, 697–701.