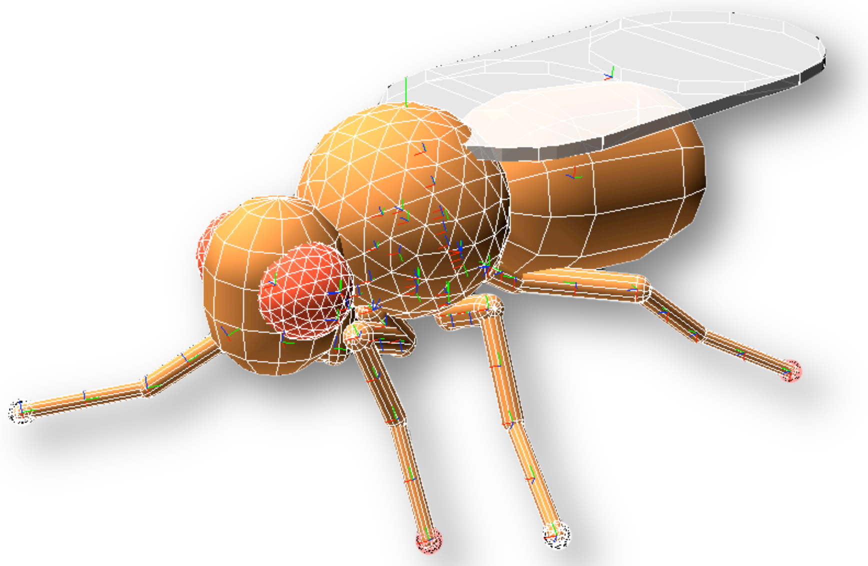


Building a Computational Fly

Semester Project

Raphaël Cherney

Microengineering Section

Spring 2012

Assistants:

Pavan Ramdya

Andrea Maesani

Laboratory of Intelligent Systems (LIS)

Prof. Dario Floreano

Swiss Federal Institute of Technology (EPFL)

School of Engineering (STI)

Institute of Microengineering (IMT)

EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE LAUSANNE
POLITECNICO FEDERALE DI LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY LAUSANNE

FACULTE SCIENCES ET TECHNIQUES DE L'INGENIEUR

LABORATORY OF INTELLIGENT SYSTEMS (LIS)
CH - 1015 LAUSANNE

SEMESTER PROJECT

Title: Build a Computational Fly.

Student(s): Raphael Cherney (MT)

Professor: Dario Floreano

Assistant 1: Pavan Ramdya **Assistant 2:** Andrea Maesani

Project description:

To understand the behavior of complex biological systems it is often useful to build a physically accurate simulation. Robotics has a history of using such computational tools and these can also be exploited to reverse engineer biological behaving systems. In return, lessons learned from these simulations may then be directly applied to the generation of advanced artificial intelligent systems.

We are studying the behavior of the hexapod insect, Drosophila melanogaster. Owing to hundreds of millions of years of insect evolution, its sensing and actuation mechanisms serve as useful guides for the development of sophisticated behaving robots.

For this project we will develop a 3D simulation environment for a morphologically and kinematically accurate computer generated fly. With this tool, we will test the means of achieving fly-like locomotion using dimension-reducing control strategies. Results from these experiments may suggest bio-inspired strategies for robotic locomotion.

Remarks:

You should present a research plan (Gantt chart) to your first assistant before the end of the second week of the project. An intermediate presentation of your project, containing 8 minutes of presentation and 7 minutes of discussion, will be held on March 30, 2012. The goal of this presentation is to briefly summarize the work done so far and discuss a precise plan for the remaining of the project. Your final report should start by the original project description (this page) followed by a one page summary of your work. This summary (single sided A4), should contain the date, laboratory name, project title and type (semester project or master project) followed by the description of the project and 1 or 2 representative figures. In the report, importance will be given to the description of the experiments and to the obtained results. A preliminary version of your report should be given to your first assistant at the latest 10 days before the final hand-in deadline. 2 copies of your final version, signed and dated, should be brought to your first assistant before noon June 6, 2012. A 20 minute project defense, including 5 minutes for discussion, will take place June 15, 2012. You will be graded based on your results, report, final defense and working style. All documents, including the report (source and pdf), summary page and presentations along with the source of your programs should be handed in on a CD on the day of the final defense at the latest.

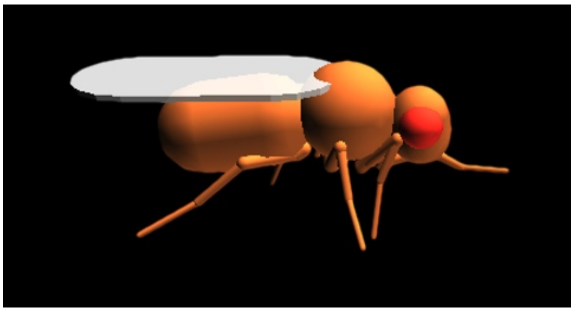
Responsible professor:	Responsible assistant:
Signature: Dario Floreano	Signature: Pavan Ramdya

Lausanne, 30 May 2012

Summary

This semester project, undertaken during the spring 2012 term at the Laboratory of Intelligent Systems (LIS) of the Swiss Federal Institute of Technology (EPFL), is about the development of a biologically-accurate 3D simulated *Drosophila melanogaster* model. We collected data on *Drosophila* morphology using image analysis and high-speed video. We then used this information to create a biologically-plausible fly model in Webots (Figure 1) and developed controllers to coordinate the 36 degrees of freedom. The control structure and model itself can be easily adapted to answer a variety of control-related questions related to biology and robotics.

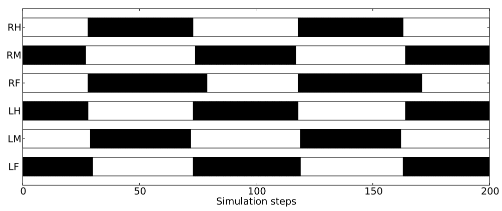
(a) High-speed video of live *Drosophila*



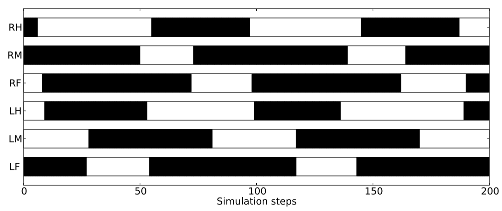
(b) Webots model with hand-tuned controller

Figure 1: Comparison between biology and model

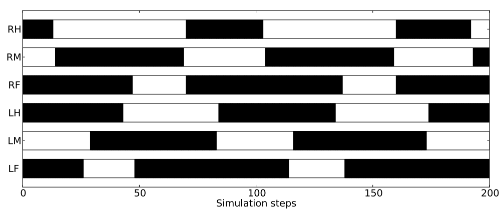
Using particle swarm optimization (PSO), we optimized the phase difference between independent, hand-tuned leg oscillators to maximize the speed of locomotion. Through this we found four emergent gaits, the fastest being a variant of a ripple gait (Figure 2).



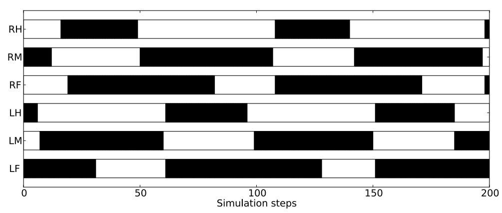
(a) Hand-tuned alternating tripod gait



(b) Speed-optimized ripple-like gait



(c) Emergent trot-like gait



(d) Evlolved alternating tripod gait

Figure 2: Emergent gaits from particle swarm optimization (swing phase in black)

Contents

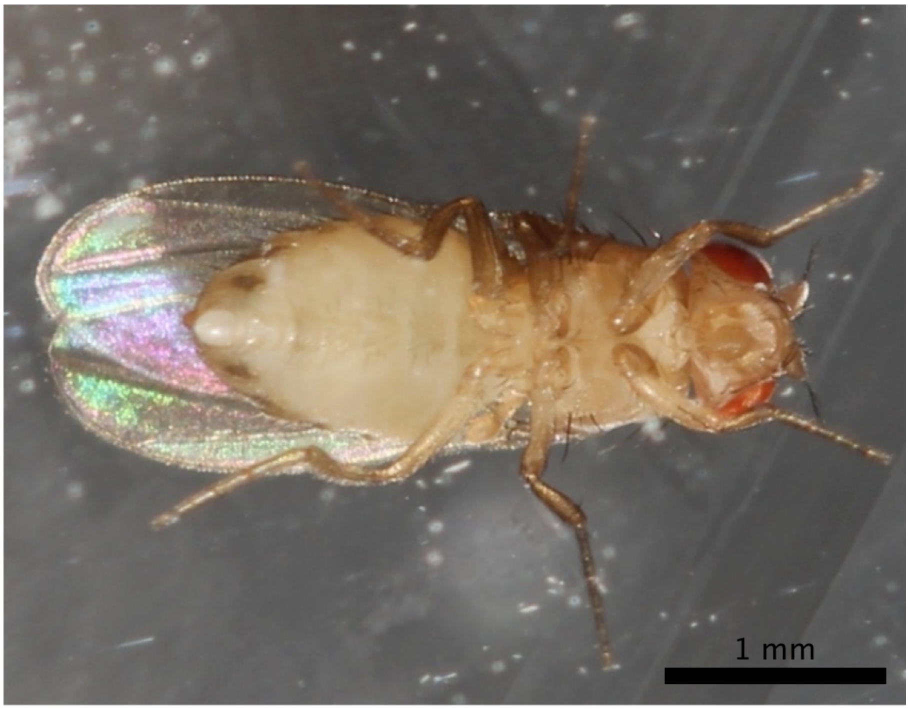
1 Introduction

1.1 Background

Locomotion is one of the most impressive and important skills that organisms have ever developed. It allows living things to experience different aspects of our world and exploit our environment in new and better ways. Insects (class Insecta) have a particularly well-developed and robust ability to move. Through millions of years of evolution, insects have gained the ability to fly and/or climb over obstacles. Using a collection of specialized sensory organs, insects are able to perceive and traverse very complex environments. They can coordinate many degrees of freedom in sophisticated and elegant ways. Despite this fact, insect nervous systems are relatively simple. This combination of robust, stable locomotion and simple control have made hexapod insects an interesting model in other fields such as robotics.

Drosophila melanogaster (Figure 3), more commonly known as the fruit fly, is a model organism in biology. Due to its small size, short generation time, ease of care, and large brood numbers, it has become one of the most intensely studied organisms in biology. In particular, we understand the *Drosophila* genome better than almost any other. This allows us to create and test a variety of genetic experiments that could not be done practically with any other organism. Through biological research and specialized experiments, we can better understand the mechanisms of insect locomotion. We often assume that because flies can fly, their walking behavior is somehow less interesting. However, flying insects spend a majority of the time attached to and traversing complex surfaces, and they have evolved a remarkable ability to traverse cluttered environments. By utilizing concepts from fly locomotion, we may be able to answer questions about advanced walking creatures and even use the ideas to create more robust walking machines.

(a) Side view



(b) Bottom view

Figure 3: Female *Drospophila melanogaster*

1.2 Project goals

The goal of this semester project was to develop a biologically-accurate, 3D model of *Drosophila melanogaster*. Using anatomical and behavioral data from live *Drosophila*, the model should be able to emulate the walking motion of flies. In particular, the model should have the following characteristics:

- Biologically accurate
- Capable of testing/optimizing control structures
- Easy to adjust for new experiments
- Easy to distribute

In order to accomplish these goals, we needed to select an appropriate modeling tool, collect biological data, build and verify the model, and design and implement a control structure.

2 State of the art

2.1 Insect models and control

Very little work has been done to understand and model *Drosophila* walking. Nevertheless, we can learn a lot by examining the research done on other walking insects. Almost all work to date has focused on stick insects or cockroaches. Stick insects have a simpler morphology and are therefore simpler to model while cockroaches are larger and easier to experiment with.

Insect walking

The most in-depth study of *Drosophila* walking is found in [50]. This paper outlines the basic coordination of limbs for walking at different speeds and turning (Figures 4 and 5). More general information on insect walking can be found in [52], and [39] describes the parameters specifically associated with stick insect walking.

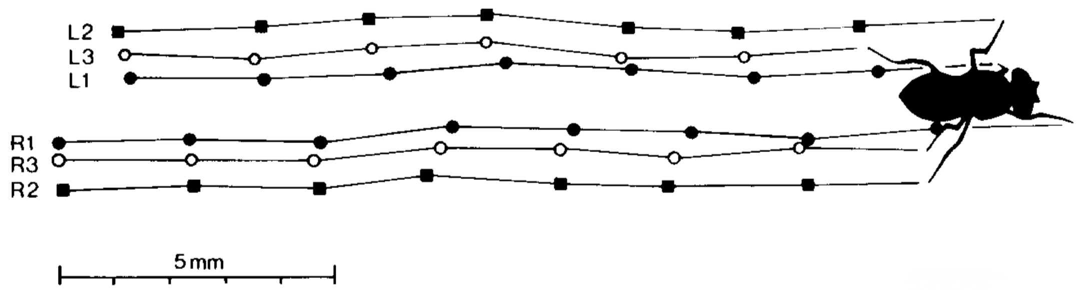


Figure 4: Patterns of footprints in *Drosophila* (from [50])

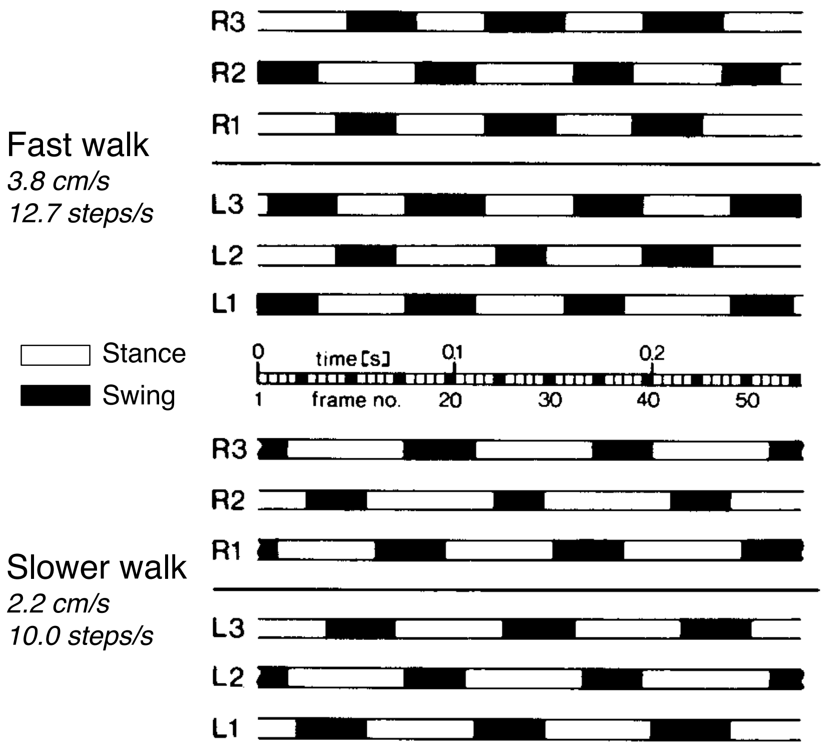


Figure 5: Temporal footfall patterns during walking (adapted from [50])

Central pattern generators (CPGs)

Legs tend to move in a consistent, coordinated, and rhythmic pattern. In many animals this neuronal coordination is organized in a hierarchical manner such that a simple motor command can generate a complex, rhythmic motion even without feedback from the peripheral nervous system. The oscillating network of neurons that create the signal is known as a central pattern generator (CPG) and has be shown to exist in a wide variety of animals from invertebrates to vertebrates. [37] provides a good overview of CPGs in both animals and robots while [11] describes the related biology. CPGs present a variety of interesting properties including distributed control, dealing with redundancies, and creating complex motion from simple control signals. CPGs can be modeled in many ways, but they are often simplified to a dynamical system of coupled oscillators. Several have suggested that insects use CPGs to control their walking, and there are several examples of CPGs used to control insect-inspired robots [1, 6, 38].

Series of reflexes

The work of Holk Cruse with the stick insects suggests that reflexive controllers (i.e. based on reflexes without CPGs) can also lead to robust insect locomotion [14, 17, 15, 16, 40, 45]. Based on biological data and tested through simulation, he condensed the reflexes into a set of rules that allow for insect-like walking (Figure 7).

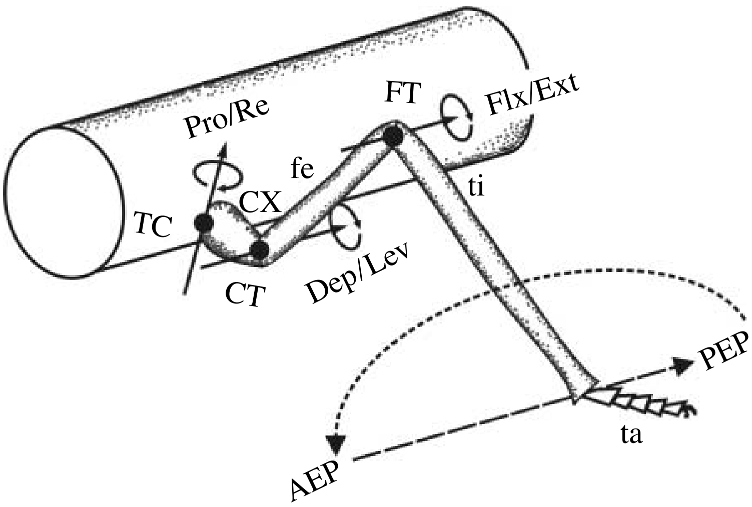
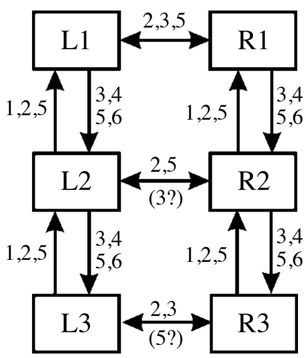


Figure 6: Cruise stick insect leg model (from [16])



rule	action	goal
1	suppress lift-off	avoid static instability
2	facilitate early protraction	favour temporal coherence
3	enforce late protraction	maintain temporal coherence
4	aim touch-down location	exploit prior foothold
5	distribute propulsive force	share load efficiently
6	enforce correction step	avoid stumbling

Figure 7: Leg coordination rules (from [16])

Hybrid approach

The truth probably lies somewhere between a centrally controlled coordination and pure reflex. Ekeberg investigates the interaction between these controls through a simulation in [26]. On the robotics side, [22] describes how to create robust, insect-inspired hexapod robots based on interactions between CPGs and sensory feedback from the moving legs.

2.2 Hexapod robots

Legged locomotion is the most common way for animals to move. It also allows for systems to traverse much more complicated terrain than traditional wheeled robots. Hexapods in particular have the advantage of being robust and statically stable during walking. Many bio-inspired hexapod robots have been designed and simulated over the years. Several, such as [1, 6, 38], use CPG-based controllers for control. Ferrell provides a good example that explores different gaits (Figure 8) and control structures in [28, 29]. [23] outlines the current knowledge of the physiological basis of insect walking along with its impact on robotics.

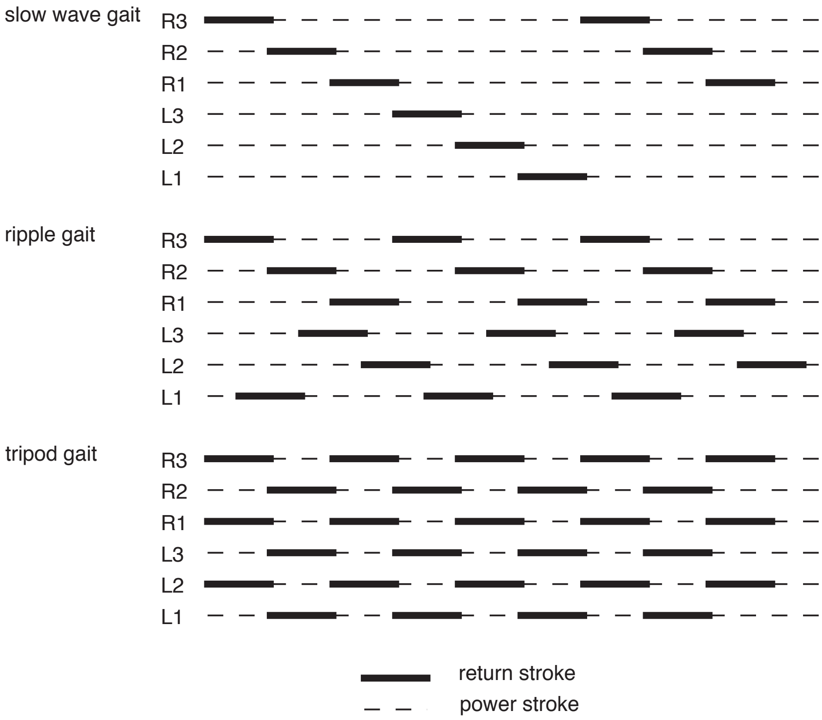


Figure 8: Commonly observed insect gaits used to develop robotic controllers (from [28])

3 *Drosophila melanogaster* morphology

3.1 Anatomy

The anatomy of *Drosophila* is similar to that of all insects. Flies have a three-segment body with a head, thorax, and abdomen. Connected to the thorax are two wings and three pairs of legs. Each leg has several joints and components in series. Figure 9 shows the major components of the fruit fly that should be included in an accurate model.

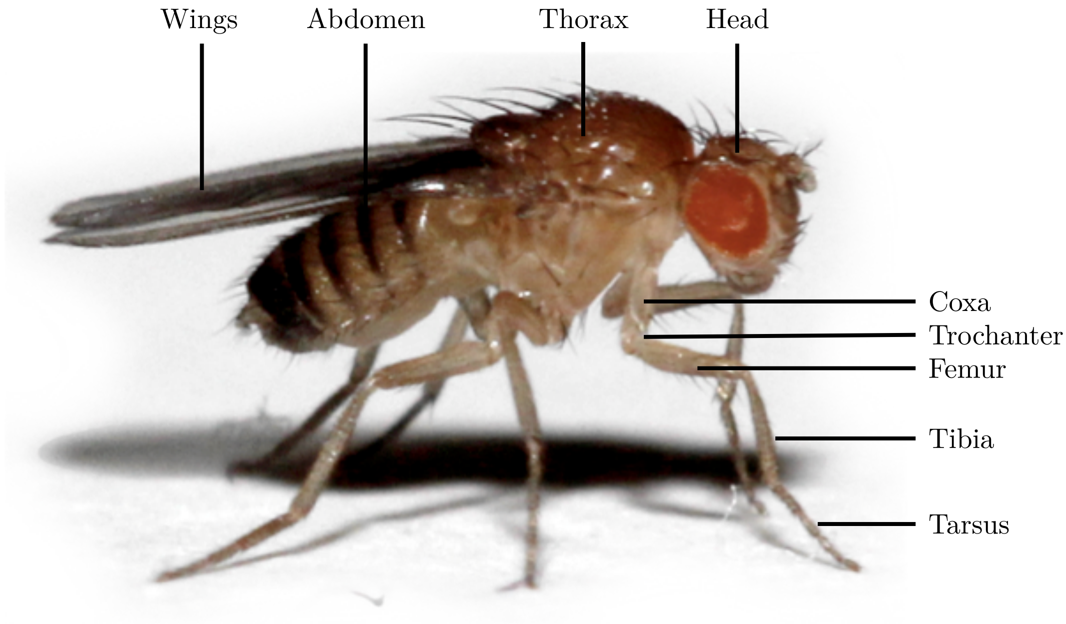


Figure 9: Major components of *Drosophila* anatomy

Figure 10 details the different elements of the *Drosophila* leg. The coxa is connected to the thorax and is the joint with the most degrees of freedom. The coxa is, in turn, connected to the trochanter. As with many insects, the trochanter and femur are fused together and can be viewed as a single component [47]. The femur is connected to the tibia, which is in turn connected to the tarsal segments, or tarsus. The tarsal segments have some flexibility and have been observed to bend during walking while lifting the forelegs.

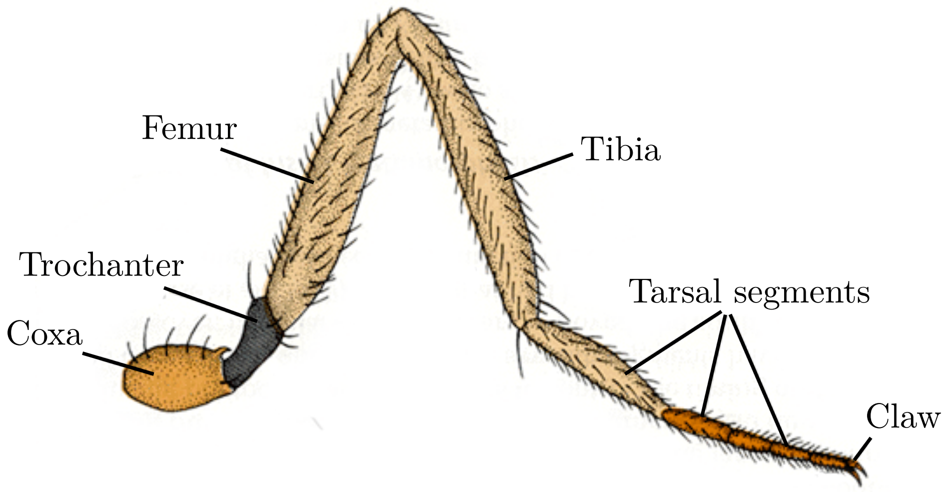
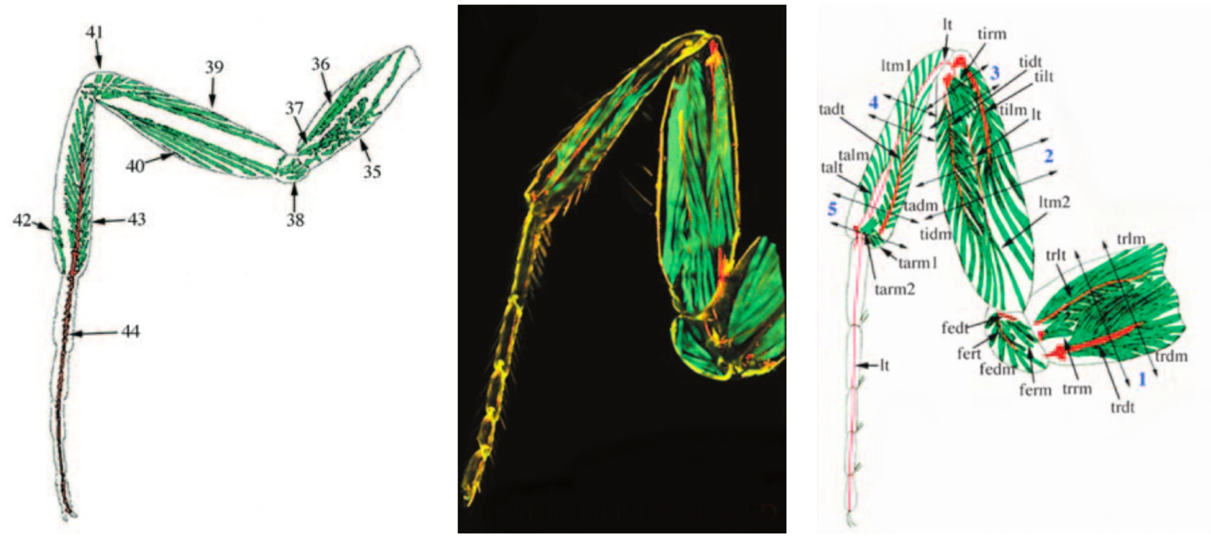


Figure 10: *Drosophila* leg anatomy (adapted from [31])

3.2 Leg degrees of freedom

The legs incorporate several different joints in series. These joints are held together by tendons and use antagonistic muscles for actuation. With the exception of the thorax-coxa joint, all of the joints are simple hinge-like elements that keep the leg mostly in plane. As with any system dependent on soft materials (i.e. tendons), there is a minute amount of flexibility in the non-actuated direction (much like a human knee joint), but the impact on overall motion of the leg can be considered negligible. In [48], Soler et al. discuss the muscles and tendons contained within the *Drosophila* leg. From this, we can begin to understand the degrees of freedom and relative strength of each joint. The detailed leg musculature can be seen in Figure 11. In short, the coxa-trochanter, femur-tibia, and tibia-tarsus joints are all in plane with one another and contain antagonistic muscle pairs (levators and depressors).



(a) Miller leg musculature [44] (b) MHC-GFP stained leg (c) Soler et al. revised musculature

Figure 11: Muscles and tendons in *Drosophila* leg (adapted from [48])

Sink provides a more in-depth look at muscle development in *Drosophila* in[47]. This work also addresses the thorax-coxa joint joint ignored in [48]. As shown in Figure 12, there are four major muscles which control the coxa position. I and K are able to “promote” the leg, while J and L “remote” the leg (move the limb forward vs. backward). Similarly, I and J will “abduct” the leg while K and L “adduct” the leg (lift vs. lower the limb). These muscles can also cause a slight rotation of the limb (“anterior/posterior rotation”), though this is used to a much smaller extent than promotion/remotion or abduction/adduction.

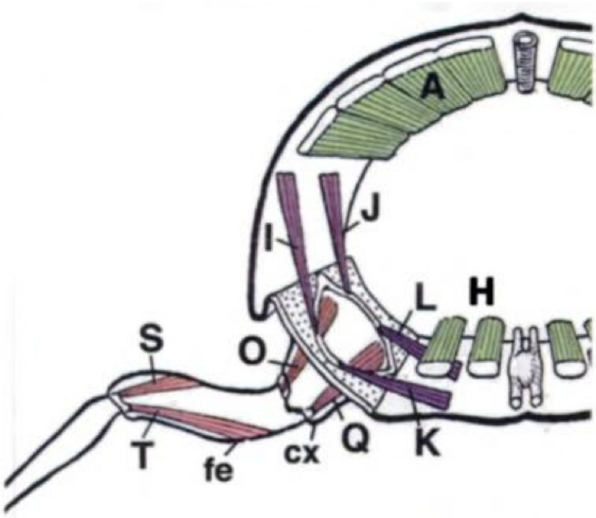


Figure 12: Thorax-coxa joint muscles (from [47])

Combining the understanding of anatomy with observations of live *Drosophila*, we can develop a good model of how the leg is constructed and the various degrees of freedom of each joint. Figure 13 shows the degrees of freedom associated with each joint. Each leg has 6 degrees of freedom: 3 at the thorax-coxa joint, 1 at the coxa-trochanter joint, 1 at the femur-tibia joint, and 1 at the tibia-tarsus joint. This creates a organism with 36 degrees of freedom within the legs. These internal degrees of freedom provide a lot of flexibility to the animal but require a non-trivial control system to coordinate all of the joints during walking. Fortunately, most of the motion is (approximately) contained within a single plane defined by the thorax-coxa joint, simplifying the control to a certain extent.

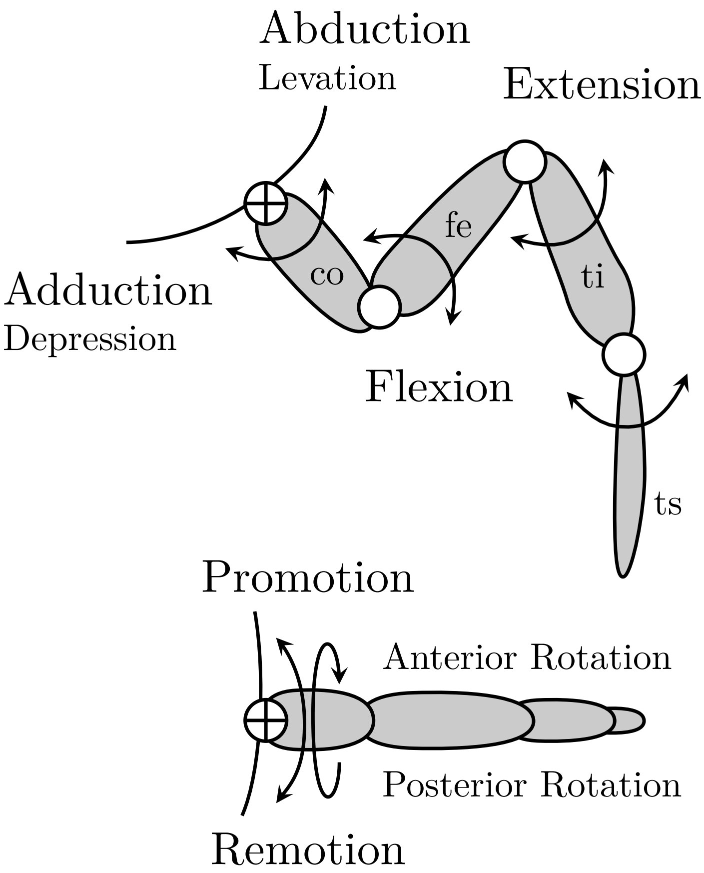


Figure 13: Leg degrees of freedom (adapted from [47])

3.3 Leg segment lengths

Given the morphology and degrees of freedom of the *Drosophila* leg, the next step before developing a model is to obtain biologically plausible measurements for the length of each segment. Unfortunately, there is no published data readily available on the lengths of the leg components. Nearly all size-related measurements in literature use either weight, wing length, thorax length, or total body length as a representative measurement. We are, however, instead interested in the sizes of the legs. To get a first estimate of this information, we took a series of pictures using a Canon EOS 600D camera with a macro lens. We took images of the flies inside of transparent containers and in the open. We attempted to get images of the flies in as many orientations as possible including the underside. Using these images, we can develop a reasonably accurate model of the leg. One of the few papers that addresses the size of particular body parts in *Drosophila* is [30]. We used the typical female mesothorax length of 1.08 mm described in this paper as a reference.

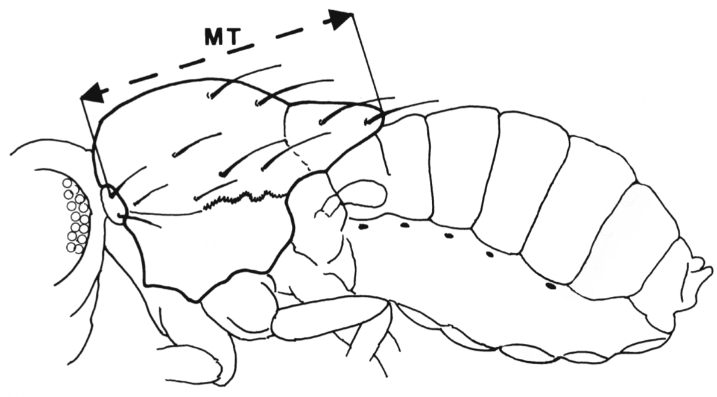
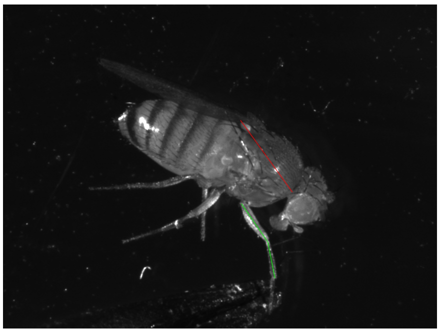


Figure 14: Reference measurement (adapted from [30])

The initial measurements suffered from several sources of error. Most notably, the measurements were rarely taken with the leg in plane with the image, leading to a projection with a shorter perceived length. Furthermore, because each image was taken at a different orientation and distance from the subject, it was difficult to accurately estimate the mesothorax length to scale each picture. In order to improve the quality of our measurements, we took additional images using a Leica optical microscope at the Université de Lausanne (UNIL). Average-sized female *Drosophila* were put on ice to immobilize them before placing them under the microscope. The flies were then manipulated to so that the particular legs could be well measured. The microscope images had a consistent magnification and focus (and therefore pixel size), allowing us to calculate the conversion factor to real units (the pixel size stored with the image metadata was incorrect). Using this improved method, we were able to significantly improve the quality of our length estimates. Table 1 shows the measurements taken on the images shown in Figure 15. All of these images are of a single, female *Drosophila*, chosen for its representative size. The measurements are in pixels with the conversion to millimeters applied at the end. The best guess measurement for each segment was based on the high end of the new measurements (due to the shortening effect of non-perpendicular projections) and previous knowledge about the approximate lengths. For example, the mid-leg femur estimate is longer than the single measurement because we knew the relative femur length from our earlier measurements (longer than tibia).

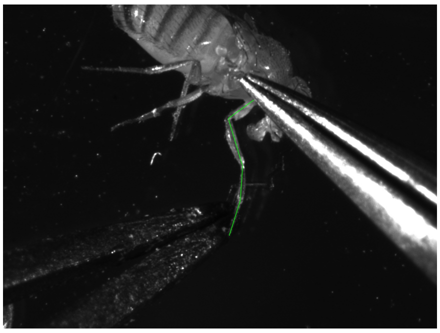
Image	**Foreleg Coxa**	**Foreleg Femur**	**Foreleg Tibia**	**Foreleg Tarsus**	**Mid-leg Coxa**	**Mid-leg Femur**	**Mid-leg Tibia**	**Mid-leg Tarsus**	**Hind-leg Coxa**	**Hind-leg Femur**	**Hind-leg Tibia**	**Hind-leg Tarsus**	**Mesothorax**
1	-	145	119	-	-	-	-	-	-	-	-	-	279
2	98	158	122	-	-	-	-	-	-	-	158	215	-
3	99	152	113	116	-	-	-	-	-	-	-	-	-
4	98	159	120	112	-	-	-	-	-	-	-	-	274
5	-	-	-	-	33	-	145	157	-	-	-	-	269
6	-	-	118	130	36	130	129	188	-	-	-	-	274
7	-	-	123	140	-	-	153	157	54	125	119	209	272
8	-	-	123	141	-	-	150	178	50	121	115	207	272
Best Guess	98	158	122	140	34	166	154	180	52	126	160	220	273
(in mm)	*0.39*	*0.63*	*0.48*	*0.55*	*0.13*	*0.66*	*0.61*	*0.71*	*0.21*	*0.50*	*0.63*	*0.87*	*1.08*

Table 1: Leg measurements

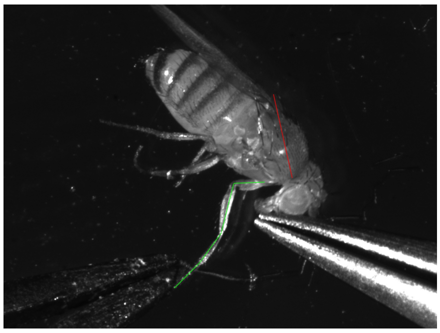


(1)

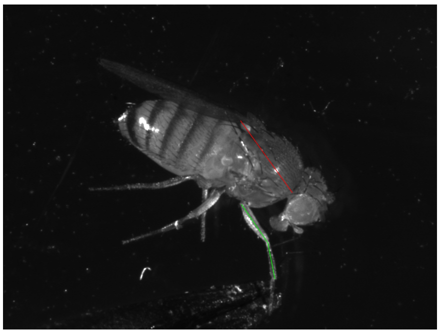
(2)



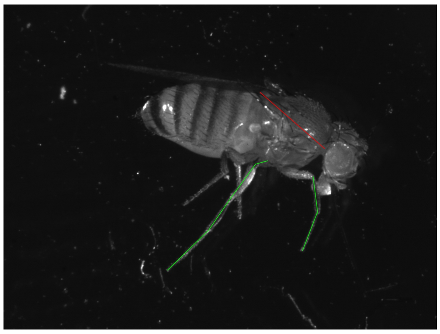
(3)



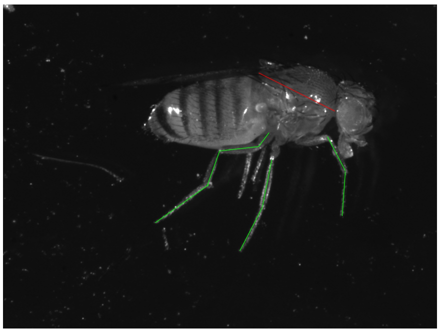
(4)



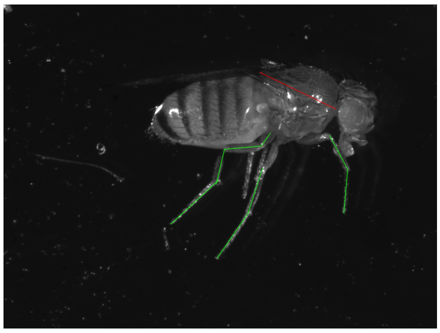
(5)



(6)



(7)



(8)

Figure 15: Images used for leg measurements

3.4 High-speed camera

In order to track and understand the coordination of leg movements, we used a high-speed video setup. We began by collecting videos using the high-speed camera and magnifying lens owned by the Laboratory of Intelligent Systems (LIS). Videos were taken at 250 and 500 frames per second. The higher frame rates allow for better tracking but require higher light levels. The camera was mounted and focused on a known point in a well-lit environment. The flies were then made to walk through this point of focus either by placing the fly on a paper which could be moved to the point of focus or by starting the fly on an elevated pathway such as a cable. The fly will naturally stay on the elevated surface, allowing us to predict (and guide) the trajectory of the fly. However, due to camera limitations, the quality of early high-speed camera footage was relatively poor. In particular, the data only had a 7-bit depth, meaning that the usable range of the camera was relatively low.

We collected significantly better data using a high-speed pco.edge 16-bit CMOS camera. The camera also features a higher resolution (5.5 megapixels, though only a subset were used). The setup is shown in Figure 17. The camera was mounted on a tripod and aimed at the sample. Light was provided using a specialized equipment to prevent the flies from being overheated by the bulbs. We placed average-sized female *Drosophila melanogastor* in a new, transparent fly food vial and focused the image on the transparent surface. Be moving the plane of focus, we can observe the motion of the flies from a variety of different angles. In particular, we wanted lateral and ventral views of the flies as shown in Figure 18. Note that videos showing the lateral view are taken sideways (i.e. the gravity vector points to the left side of the image). The videos captured with the pco.edge had significantly better noise dynamic range and noise performance in addition to being easier to use.

Figure 16: pco.edge sCMOS high-speed camera (©PCO AG)

(a) Tripod mount (b) Sample lighting

Figure 17: High-speed camera setup

(a) Lateral imaging

(b) Ventral imaging

Figure 18: Example high-speed images

By observing *Drosophila* while walking, we notice that not all of the degrees of freedom are used equally. There are also physical limits to the possible joint angles. We relied heavily on the high-speed video footage to configure our hand-tuned controller (see measurements in Appendix A). Furthermore, the video was used to further tune the morphology (particularly the leg measurements) from the results obtained through image analysis. We also noted a greater-than-expected amount of flexibility between tarsal segments of the forelegs during normal walking. Figure 19 shows the observed bending of the tarsus (in one direction). The significance of this fact remains unclear. It may be an inadvertent action, it may aid with releasing the grip/adhesion to the surface, or may serve some other purpose.

Figure 19: Flexibility of foreleg tarsus (3 ms between images)

4 Computational fly model

4.1 Webots

For our simulation we used the Webots 6 software from Cyberbotics Ltd.[1]. The software incorporates an integrated development environment (IDE), physics engine, and 3-dimentional graphics to create a relatively simple way of implementing complex motion simulations. Webots is designed as a robotics simulator and therefore has many actuation and sensing capabilities built in. The following list of features made Webots stand out as a simulation tool:

- Multi-platform (Linux, Windows, OS X)
- Open Dynamics Engine (ODE) for accurate physics simulation
- 3D visualization
- Sensor and actuator libraries to ease implementation
- Choice of programming languages (C, C++, Java, Python, MATLAB)
- EPFL knowledge base (BIOROB)
- Availability through EPFL license
- Expandable
- Existing documentation
- Easier conversion into hardware

In short, using Webots allowed this project and future projects to get up and running quickly. By simplifying distribution and easing implementation of new ideas, the model becomes significantly more valuable. A custom made solution may run more quickly,

[1] `http://www.cyberbotics.com/`

and may ultimately be a worthwhile investment, but a Webots-baed model can be easily distributed and used to prototype experiments. Furthermore, because it is a mature and commercially released piece of software, the number of bugs reduced and there is good support material to get started [19, 18].

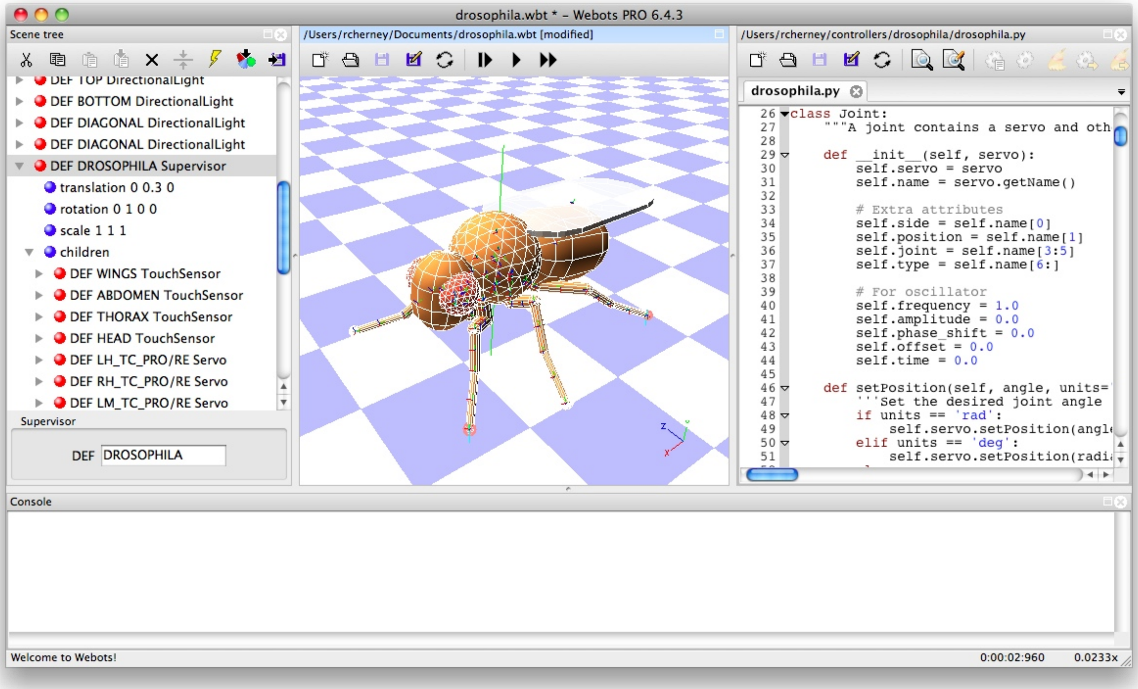


Figure 20: Webots graphical user interface (GUI)

4.2 Model

In Webots, the model and environment information is contained within a world (.wbt) file. Within this file, objects and relationships are represented by a tree of specialized Nodes such as Servos, TouchSensors, Transforms, etc. (Figure 21). The official Cyberbotics documentation in [18] is particularly useful for designing and using Webots models. The *Drosophila* model is constructed in a hierarchical manner. Figure 22 lists the most important subset of the Nodes and the hierarchical relationship between them. Children are denoted by indentation. Note that this list is not complete, but only lists the nodes the user will typically interact with.

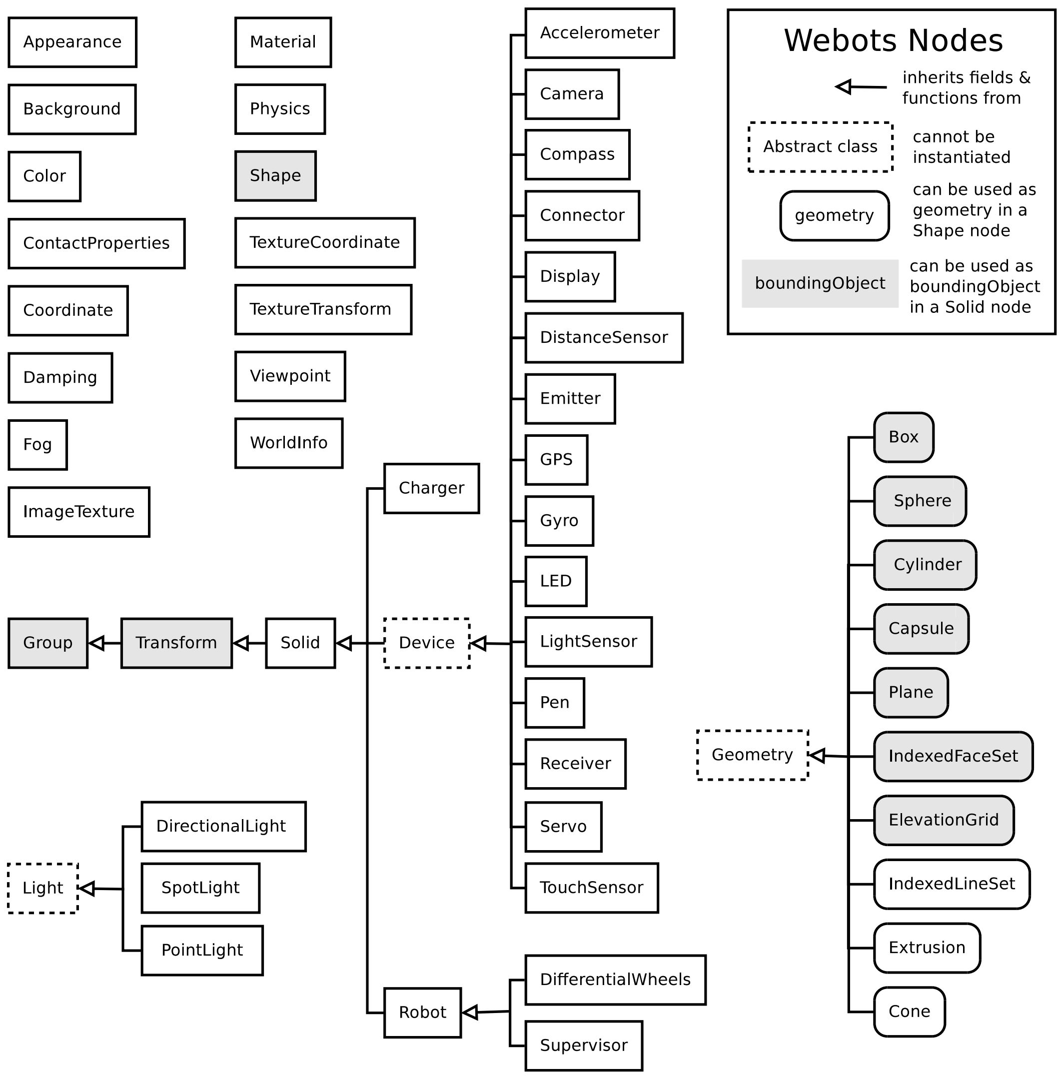


Figure 21: Webots Nodes chart (from [18])

```
DROSOPHILA Supervisor
    HEAD TouchSensor
    THORAX TouchSensor
    ABDOMEN TouchSensor
    WINGS TouchSensor
    LF_TC_PRO/RE Servo
        LF_TC_ABD/ADD Servo
            LF_TC_ROT Servo
                LF_COXA TouchSensor
                LF_CT_FLEX/EXT Servo
                    LF_FEMUR TouchSensor
                    LF_FT_FLEX/EXT Servo
                        LF_TIBIA TouchSensor
                        LF_TT_FLEX/EXT Servo
                            LF_TARSUS TouchSensor
                            LF_CLAW TouchSensor
    RF_TC_PRO/RE Servo
        RF_TC_ABD/ADD Servo
            RF_TC_ROT Servo
                RF_COXA TouchSensor
                RF_CT_FLEX/EXT Servo
                    RF_FEMUR TouchSensor
                    RF_FT_FLEX/EXT Servo
                        RF_TIBIA TouchSensor
                        RF_TT_FLEX/EXT Servo
                            RF_TARSUS TouchSensor
                            RF_CLAW TouchSensor
    LM_TC_PRO/RE Servo
        LM_TC_ABD/ADD Servo
            LM_TC_ROT Servo
                LM_COXA TouchSensor
                LM_CT_FLEX/EXT Servo
                    LM_FEMUR TouchSensor
                    LM_FT_FLEX/EXT Servo
                        LM_TIBIA TouchSensor
                        LM_TT_FLEX/EXT Servo
                            LM_TARSUS TouchSensor
                            LM_CLAW TouchSensor
    RM_TC_PRO/RE Servo
        RM_TC_ABD/ADD Servo
            RM_TC_ROT Servo
                RM_COXA TouchSensor
                RM_CT_FLEX/EXT Servo
                    RM_FEMUR TouchSensor
                    RM_FT_FLEX/EXT Servo
                        RM_TIBIA TouchSensor
                        RM_TT_FLEX/EXT Servo
                            RM_TARSUS TouchSensor
                            RM_CLAW TouchSensor
    LH_TC_PRO/RE Servo
        LH_TC_ABD/ADD Servo
            LH_TC_ROT Servo
                LH_COXA TouchSensor
                LH_CT_FLEX/EXT Servo
                    LH_FEMUR TouchSensor
                    LH_FT_FLEX/EXT Servo
                        LH_TIBIA TouchSensor
                        LH_TT_FLEX/EXT Servo
                            LH_TARSUS TouchSensor
                            LH_CLAW TouchSensor
    RH_TC_PRO/RE Servo
        RH_TC_ABD/ADD Servo
            RH_TC_ROT Servo
                RH_COXA TouchSensor
                RH_CT_FLEX/EXT Servo
                    RH_FEMUR TouchSensor
                    RH_FT_FLEX/EXT Servo
                        RH_TIBIA TouchSensor
                        RH_TT_FLEX/EXT Servo
                            RH_TARSUS TouchSensor
                            RH_CLAW TouchSensor
```

Figure 22: Structure of Webots model

The most important Node types to be familiar with are the TouchSensor and Servo nodes. In our model, TouchSensor nodes are used to represent the physical components of the body that interact with the environment. Each TouchSensor has a bounding box used by the physics engine. When enabled, collisions with the bounding box can be read out and used by the controller. The result can be either a binary "touched" (1.0) or "not touched" (0.0) or give the exact forces as calculated by the physics engine (type should be changed from "bumper" to "force" or "force-3d" for this). The Servo nodes act as the antagonistic muscles of the fly. The Servo naming convention is as follows:

F = foreleg
M = mid-leg
H = hind-leg

FLEX/EXT = exion/extension
PRO/RE = promotion/remotion
ABD/ADD = abduction/adduction
ROT = rotation

RF_TT_FLEX/EXT

L = left
R = right

TC = thorax-coxa joint
CT = coxa-trochanter joint
FT = femur-tibia joint
TT = tibia-tarsus joint

Note that using a servo motor to model a biological joint loses many of the biological details of the joint (muscle flexibility, mechanical advantage, etc.). Nevertheless, it a logical simplification. Servos have a simple control loop that try to get the angle to the desired position while following limitations on speed and force. These parameters can be adjusted by the user. The force of the Servo can even be set to zero to create a passive joint. Children of Servo nodes are connected and transformed as if they were physically connected. This is important for modeling the leg with a series of actuators. The model is designed so that, given position values of zero for each Servo, the the legs will extend straight down from the thorax. Angles are measured relative to the axis of each limb. Figure A shows how the Servo positions are calculated. Figures 24 and 25 show the 3-dimensional visualization of the model. The coordinate system of each Node is marked by a set of colored axes (x in red, y in green, and z in blue). Note that Webots also uses the y-direction as the vertical (with gravity).

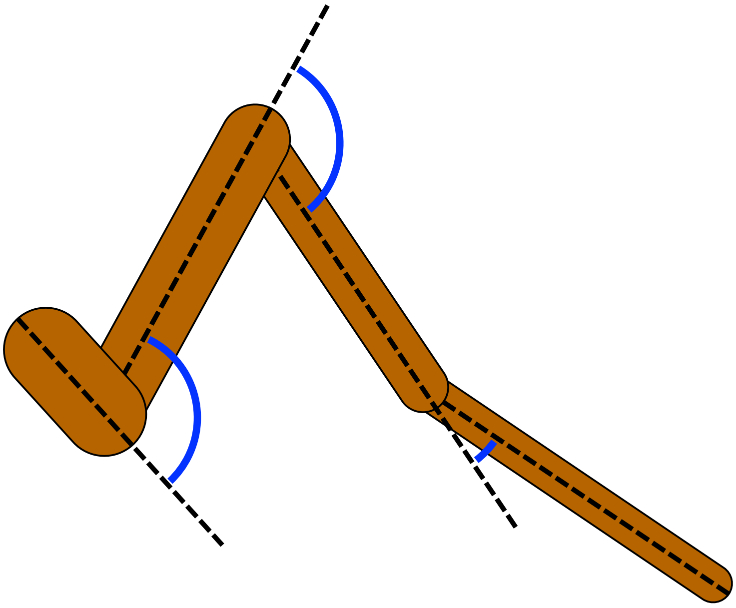


Figure 23: Angle measurement

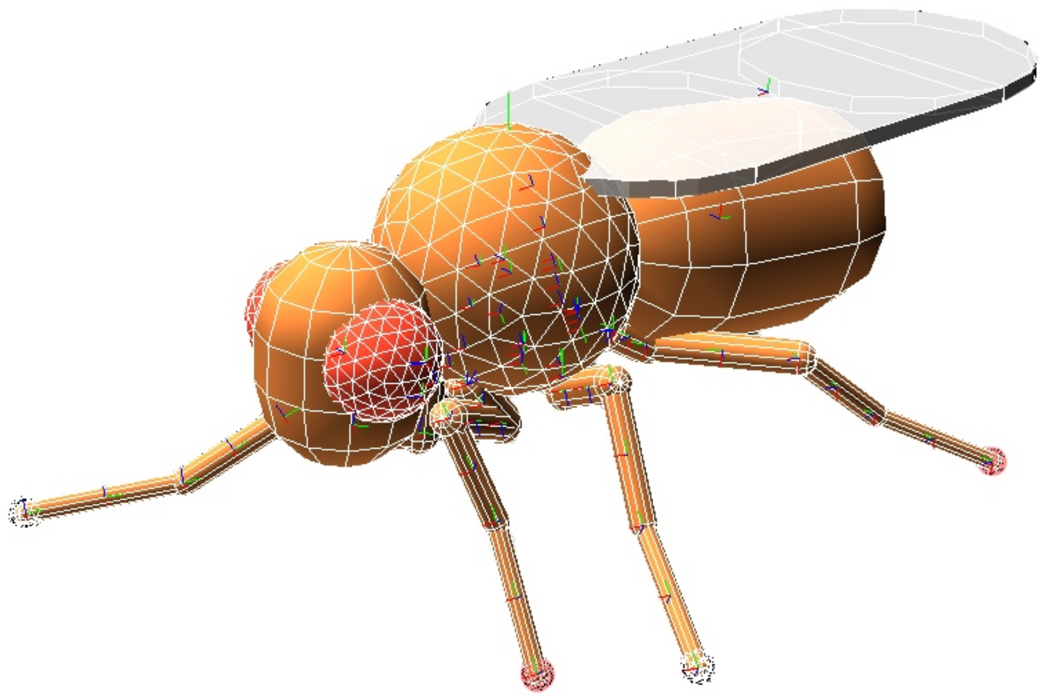
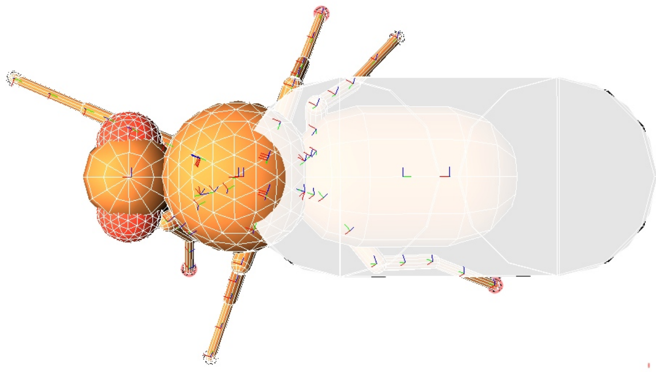
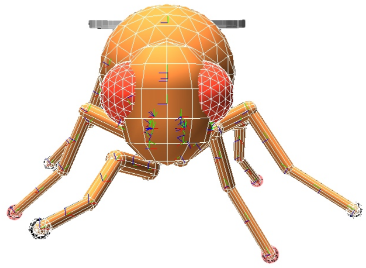


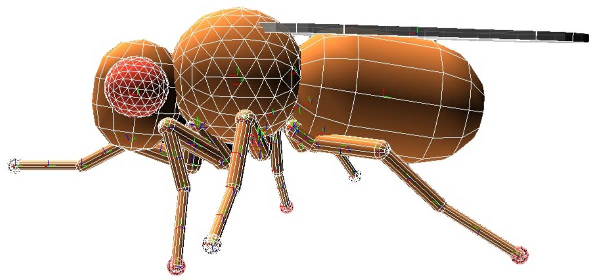
Figure 24: Webots *Drosophila* model



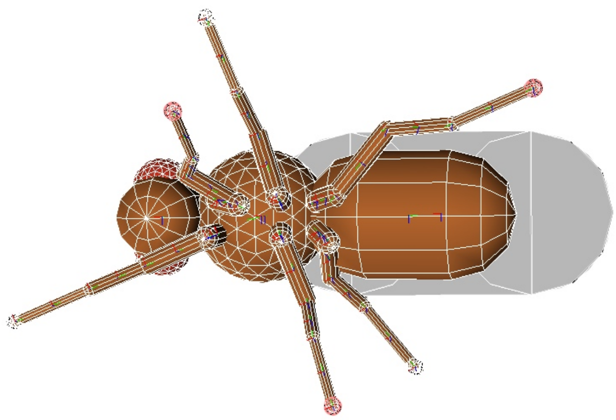
(a) Top view



(b) Front view



(c) Side view



(d) Bottom view

Figure 25: Webots model views

Webots is designed for robotics and therefore works better with objects of a certain size (size of typical robots and their environment ranges from in the centimeter-scale to the decimeter-scale). Therefore, the model is scaled by a factor of 100 in order to ease interaction with the model through the GUI. The mass of each component is calculated based on a constant density (which has not been tuned). The ABDOMEN has a density equal to half of the other nodes to improve balance. A current limitation of the Webots model is the simulation speed. *Drosophila* walk using a very fast pattern (upwards of 10 steps/s) while there are practical limits to the simulation time step within the Webots physics system (≈1 ms for the physics simulation). Currently the entire motion is run at a greatly reduced rate (one cycle every 1.4 seconds). The simulation could be improved by increasing the maximum velocity of the limbs and lowering the simulation time step.

Figure 26 shows the major components of the body. The head is modeled as a capsule and two spheres for eyes; the the thorax is a sphere; the abdomen is a capsule; the wings are a thin, capsule-shaped box; and the legs are made of a series of capsules with a sphere for the claw. The geometry is symmetric between the right and left sides of the body. Each leg has unique component identifiers (important for control) that begin with the two-letter leg identifier (shown in red in Figure 26). The first letter indicates the side and the second indicates the anterior/posterior position.

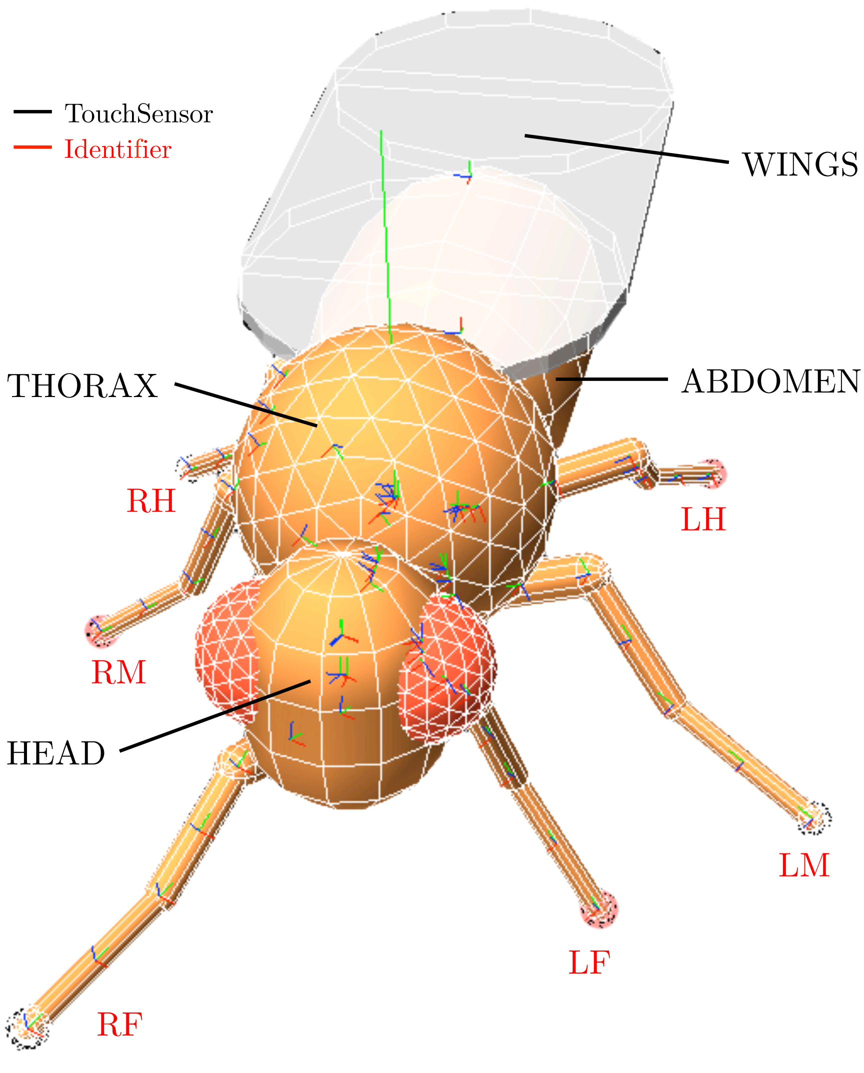


Figure 26: Annotated model

Figure 27 shows the annotated right foreleg as an example. The other legs are structured identically but have different identifiers and (possibly) lengths. The Servos (labeled in blue) follow the naming style described earlier.

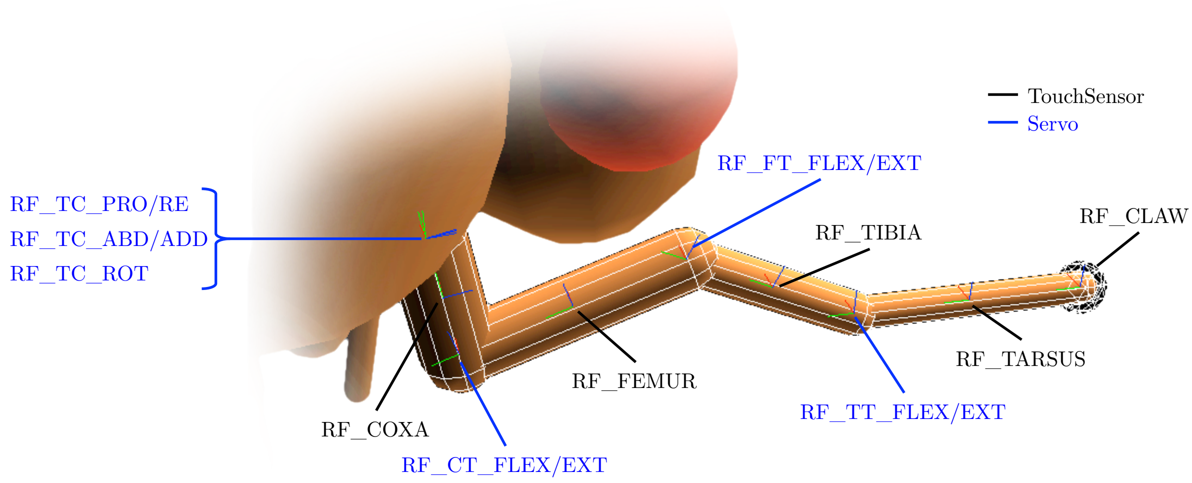


Figure 27: Annotated right foreleg

The similarity between the biological and simulated fly means that the simulation results can have both physical and biological significance. That is to say, our exploration of locomotion parameters in simulation can be used to create an effective controller for the physical robot and/or gain insight into the biological workings of living flies.

4.3 World Generator

In order to accommodate future experiments and changing morphological data, we created a world generator script in Python. Because many of the model parameters are interdependent, adjusting a single dimension of the model using the Scene tree quickly becomes a difficult process. The script allows the user to change dimensions associated with the fly morphology and automatically generate a new Webots world file. It was created by taking an existing world file and replacing key dimensions with variables that can be adjusted in a single location by the user. The script may also be useful for experiments regarding the co-evolution of morphology and control.

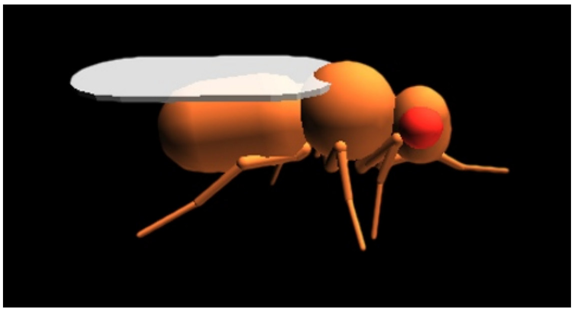
Because Webots also includes environmental information along with the model itself, the environment is also created using this script. Currently, the DROSOPHILA Supervisor is placed in a world with gravity and a single, flat "ground" plane on which to walk. Desired changes to the Webots world can be made to the generated world file itself, but all changes are lost once a new world is created. For this reason, it is best to incorporate any changes in the Webots world file into the Python script. This can be done by copying the (fortunately human-readable) changes into the script. Currently only the leg lengths are adjustable through this file.

4.4 Controller

Controllers can be written in any of Webot's supported languages (C, C++, Java, Python, Matlab). The API for each language is available in [18]. The current implementation is realized in Python. The main aspects of the controller are the initialization, in which the controller associates the Servos and TouchSensors associated with a model, and an update loop where joint positions are updated and the physics is calculated and applied.

Walking in *Drosophila melanogastor* is a cyclic and coordinated action. There are several ways to create this kind of motion. Quite possibly the simplest solution is to use a sine-based controller. Using this structure, each limb has a unique amplitude, offset, and phase lag relative to a global sine-wave. The rhythmic nature is guaranteed by the periodic nature of the sine wave, while coordination is achieved through the relative phase offsets between limbs. A sine-based controller can be thought of as an extremely simplistic CPG model in that it creates rhythmic motion without the need for feedback from peripheral senses. The parameters of the controller were tuned by hand to match the observed motion in the high-speed videos. In particular, we noted the maximum and minimum angles of each joint and the relative phase offsets during normal walking. Using this information we were able to create a realistic alternating tripod gait.

(a) High-speed video of live *Drosophila*



(b) Webots model with hand-tuned controller

Figure 28: Comparison between biology and model

In order to ease implementation, we created several new classes: a Joint class which contains the Servo and related oscillator settings; a Leg class which contains the associated joints; and a Drosophila class which is derived from the Webots-supplied Supervisor class which contains the six legs and runs the simulation.

5 Optimization

5.1 Structure

Problem

Fly walking is an extremely complex task. In addition to coordinating 36 degrees of freedom, the fly must make sense of its sensory information and react accordingly. There are many biological or control-related questions that can be tested with our model related to both locomotion and sensing. As a first experiment, we considered optimizing various control parameters and seeing how the results match biological data. We hope to ultimately test larger control questions about CPGs and reflex-based control in *Drosophila*, but for the purposes of this semester project, we limited the scope of our problem. We simply tested how well optimized the *Drosophila* gait is for speed. Unfortunately there are too many parameters to practically optimize each of the joint parameters independently. Even with a simple sine-based approach, each joint has 3 parameters, resulting in over 100 different parameters to optimize over the six legs. Therefore we limit the search space of our optimization question. Using our hand tuned parameters, we create six independent leg oscillators. The legs each have the same internal motion as in the biologically-modeled hand-tuned controller. The only parameters which are free to change are the relative phases between these leg oscillators. Since we have 6 legs and are only interested in the *relative* phase between the leg oscillators, we fix one of the phases (in our case the left foreleg). For an alternating tripod gait as we had in the hand-tuned controller, three of the phase offsets would be set to π and the others to zero.

Particle swarm optimization

We use the free, open-source inspyred[2] framework for our evolutionary computation. The inspyred package contains a variety of biologically-inspired optimization algorithms in Python including genetic algorithms (GA) and particle swarm optimization (PSO), among others. We use PSO for our optimization experiments due to its simplicity and quick convergence. Note that the particular optimization algorithm can be adjusted relatively easily within the framework.

Particle swarm optimization is an iterative computational method for solving optimization problems. Candidate solutions are initialized with a random position and velocity within the search space (in our case, a 5-dimensional hyperspace). The particles have a fitness value at their position (calculated with Webots) and a memory of the location and value of best fitness it has previously encountered. Each generation, the particle moves to a new position based on the previous velocity and attractors to the personal best solution and previous best in the neighborhood (associated particles which share their best results). In particular, each iteration the particles are adjusted according to equations 1 and 2.

$$\mathbf{v}_i^{t+1} = w\mathbf{v}_i^t + c_1 r_1 \left(\mathbf{p}_{b,i}^t - \mathbf{x}_i^t\right) + c_2 r_2 \left(\mathbf{p}_{n,i}^t - \mathbf{x}_i^t\right) \tag{1}$$

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^t \tag{2}$$

where $\mathbf{v}_i^{t+1}$ is the velocity at $t+1$ of particle i, $\mathbf{v}_i^t$ is the current velocity of the particle, $\mathbf{x}_i^t$ is the current position of the particle, $\mathbf{p}_{b,i}^t$ is the position of the personal best solution of particle i, $\mathbf{p}_{n,i}^t$ is the position of the neighborhood best solution, r_1 and r_2 are random numbers in the range [0,1], and the coefficients w (inertia weight), c_1 (cognitive rate), and c_2 (social rate) are fixed. The values of w, c_1, and c_2 are very important to the effectiveness of PSO. For our simulation, we use the values $w = 0.7$, $c_1 = 1.47$, and $c_2 = 1.47$ as suggested in [21, 12].

The approach as described here and implemented in inpyred is outlined in [21]. However, because our search space is actually periodic/circular ($0 = 2\pi$), we must implement a specialized bounding function and be sure to properly calculate the vector to the pervious bests. In order to do this, a custom PSO class is created which inherits the inspyred class, and replaces vector determination function to support periodic dimensions.

5.2 Results

Based on preliminary tests, we found that our PSO tended to converge within 50 generations. Knowing this, we ran 25 experiments with 30 particles each over 50 generations.

[2] `http://inspyred.github.com/`

The neighborhood was a ring topology with 11 neighbors (5 on each side) and the fitness function was simply the average speed over a 5 second simulation. We tracked the best, mean, and median fitness for each generation, in addition to the position of the best solution. The final results are shown in Table 2. A common problem with PSO is that particles often get stuck around local optima. We try to overcome this by running a large number of experiments and reinitializing the particles each time. Figure 29 shows the evolution of the best particle for each generation. We found that there were four optima that the PSO tended to find (colored in green, blue, yellow, and orange). These correspond to four different, yet locally optimized gaits. Figure 30 shows the result of our k-means clustering analysis graphed on three of the five dimensions. The relative number of occurrences of each optima can be seen in Figure 31. The primary/global optimum was found 60% of the time. Note that all of the optimized gaits had a better the fitness than the hand-tuned controller.

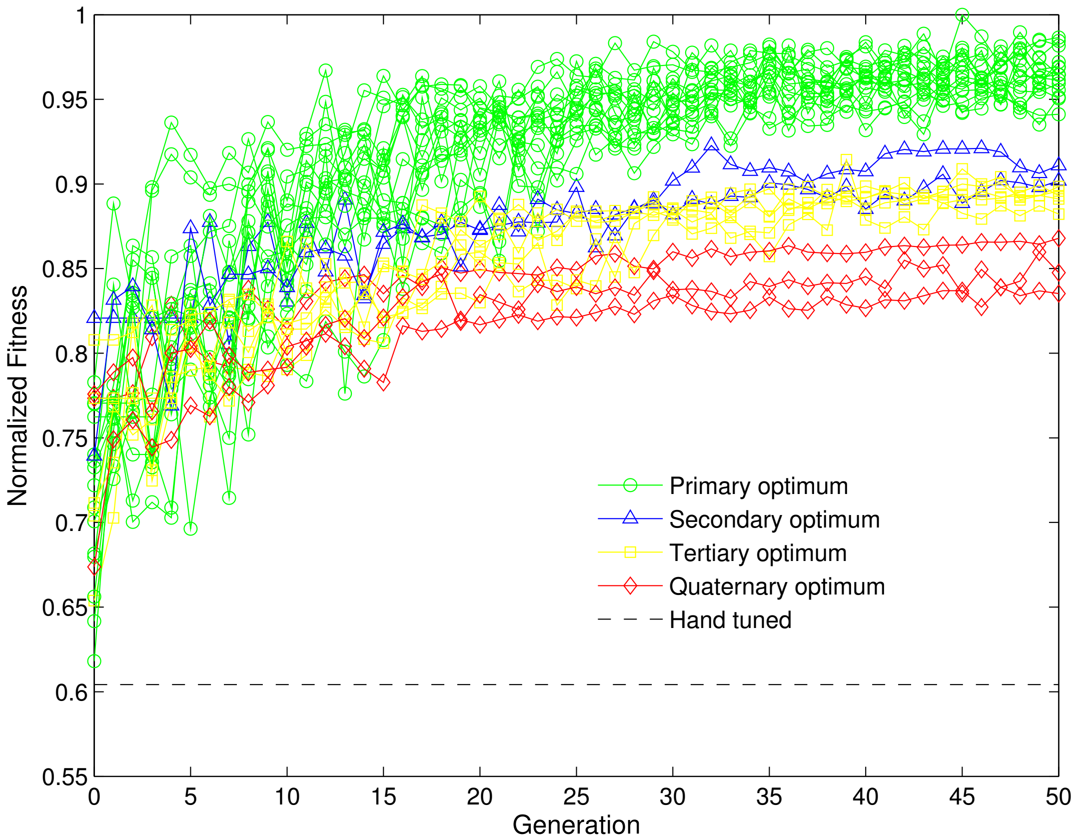


Figure 29: Evolution of best fitness categorized by final result

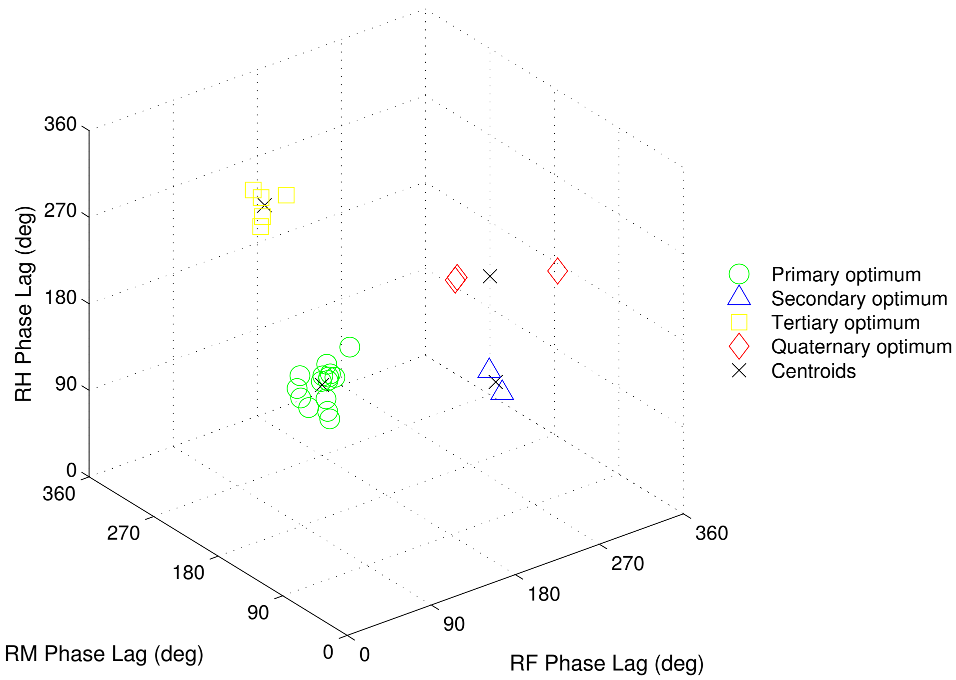


Figure 30: K-means cluster analysis (graphed on subset of dimensions)

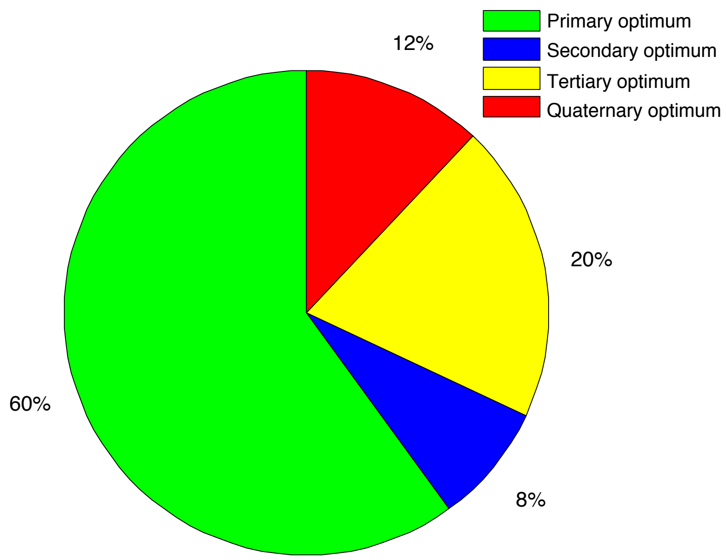


Figure 31: Occurrences of different optima

Normalized Fitness	LF Phase	RF Phase	LM Phase	RM Phase	LH Phase	RH Phase
			Primary optimum			
1.00	0.00	193.42	97.47	280.65	268.23	85.49
0.99	0.00	184.51	88.37	267.29	254.55	77.75
0.99	0.00	198.96	98.30	282.82	268.72	72.54
0.99	0.00	175.64	81.86	264.92	240.74	81.00
0.98	0.00	151.02	91.89	266.66	248.14	81.63
0.98	0.00	202.40	88.29	281.01	263.58	68.53
0.98	0.00	170.44	67.99	249.39	226.20	58.90
0.98	0.00	188.44	96.03	279.65	269.88	75.68
0.98	0.00	160.31	92.52	274.79	265.31	87.88
0.98	0.00	197.84	84.35	283.69	257.79	69.20
0.98	0.00	182.66	72.73	267.60	245.32	59.27
0.97	0.00	156.17	72.33	257.31	219.57	64.24
0.96	0.00	147.94	78.45	257.77	245.84	76.80
0.96	0.00	237.84	107.25	306.26	280.85	76.01
0.96	0.00	168.55	66.90	244.01	215.30	54.16
			Secondary optimum			
0.92	0.00	310.51	158.27	206.40	313.21	71.21
0.91	0.00	309.26	123.73	186.81	289.90	57.83
			Tertiary optimum			
0.91	0.00	79.92	257.98	189.19	134.97	343.05
0.91	0.00	45.12	236.55	176.97	115.10	338.47
0.90	0.00	57.18	246.65	194.64	132.49	346.13
0.90	0.00	39.14	235.55	171.78	109.14	332.66
0.90	0.00	52.75	248.51	199.85	135.56	353.04
			Quaternary optimum			
0.87	0.00	161.95	244.27	57.69	99.03	288.90
0.86	0.00	163.10	252.81	61.89	94.86	283.79
0.84	0.00	278.18	181.03	68.98	33.51	249.56

Table 2: Best solutions found through each particle swarm optimization

In order to show convergence of our particle swarm optimization, we can look at the results from the best of our runs. This experiment was actually run for 90 generations. Figure 32 shows the best fitness along with the mean and median fitness for each generation. First we note that the quality of solutions does not change significantly after 50 generations – showing that our previous cutoff of 50 generation is adequate. We also observe that the mean and median fitness improve quickly during the first 50 generations, indicating the the candidates are converging near the best result. This convergence signals a change from a global search to a more localized optimization. Figure 33 shows location the best particle location along all 5 dimensions for each generation of the best experiment (note that which particle is being graphed can vary from one generation to the next). Again, we note how the particles quickly converge to a solution. The fact that the solution converges so quickly means that there is not much time spent carrying out a global search of the hyperspace. Instead the swarm converges to the first good solution it can find – often leading to a convergence around a local optima. If we do not care about the local optima, the PSO parameters can be adjusted to help prevent premature convergence (using time-varying coefficients, for example).

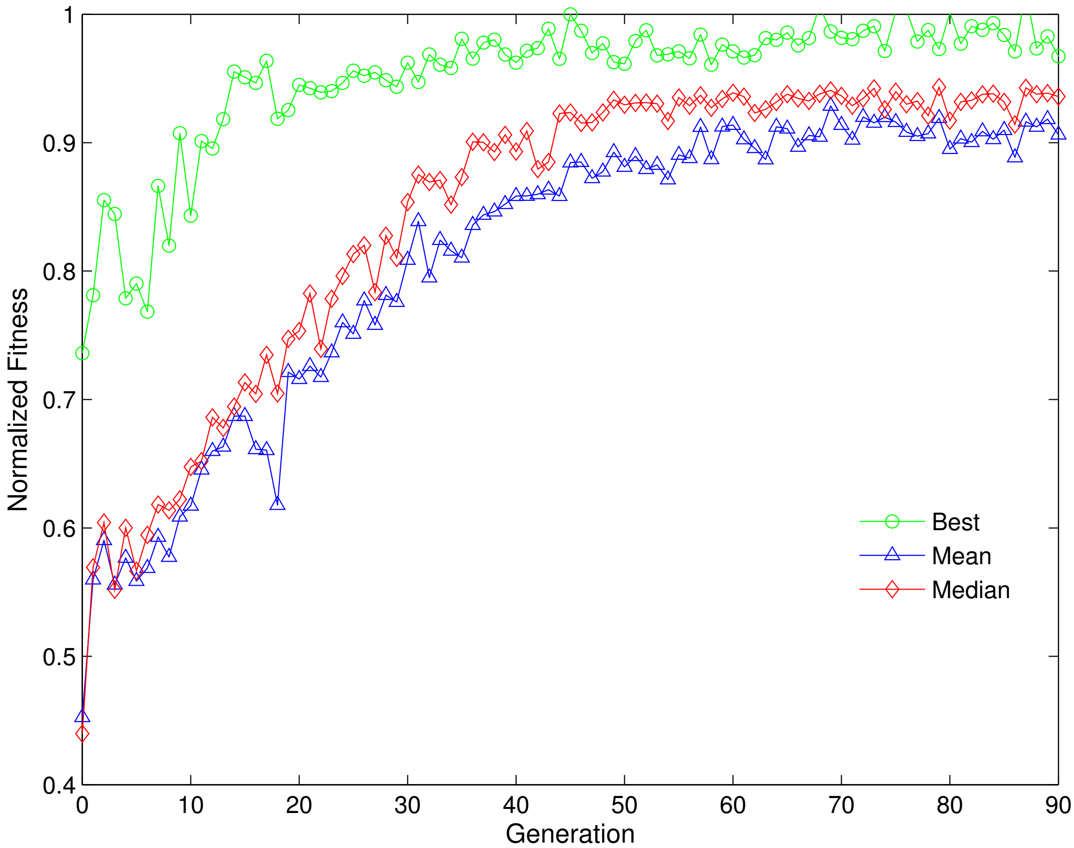


Figure 32: Best optimization results

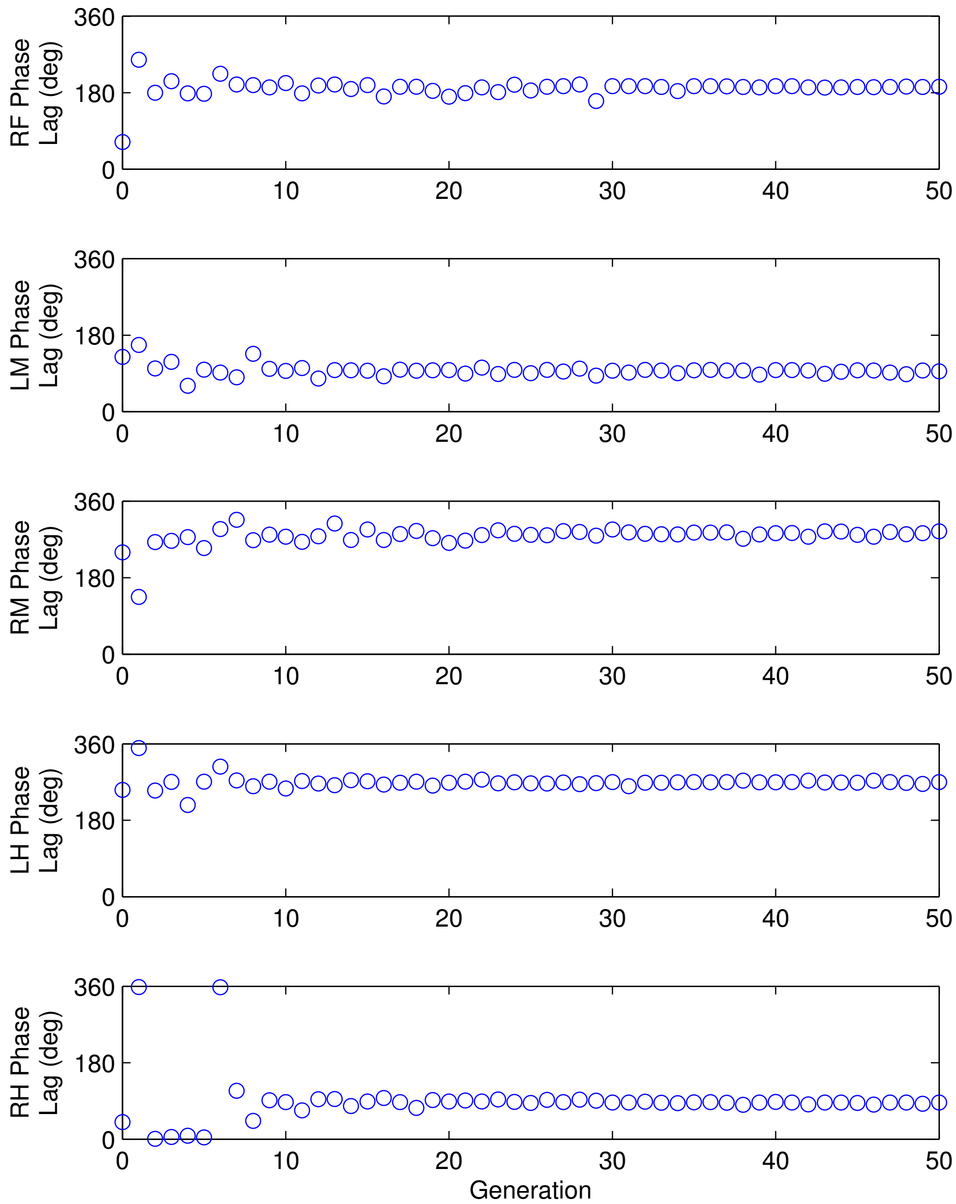


Figure 33: Convergence along dimensions

Given the four different optima, it is interesting to consider the gaits that these solutions correspond to. Figure 34 shows the gait diagrams of the hand-tuned and evolved gaits.

Hand-tuned

The hand-tuned gait is an alternating tripod gait. This is the gait typically associated with hexapods. It tends to have three legs touching the ground at the same time and is statically stable.

Primary optimum

The primary solution most closely resembles the ripple gait discussed in [28, 29] and observed in insects such as cockroaches. However, instead of having four legs on the ground at all time, the evolved gait only uses two or three. The gait is also similar to a quadruped walking trot with the foreleg stance phase distributed between the fore- and mid- legs (similar to how humans use different parts of their feet during during different phases of walking).

Secondary optimum

The secondary optimum has three, clear power phases. Two opposite legs are always pushing the fly forward together. The order is as follows: RF with LH, then RM with LM, and finally RH with LF before the whole cycle repeats.

Tertiary optimum

The tertiary optimum is nearly identical to the secondary optimum except that the order of the leg pairs is reversed: RH with LF, followed by RM and LM, and finally RF with LH.

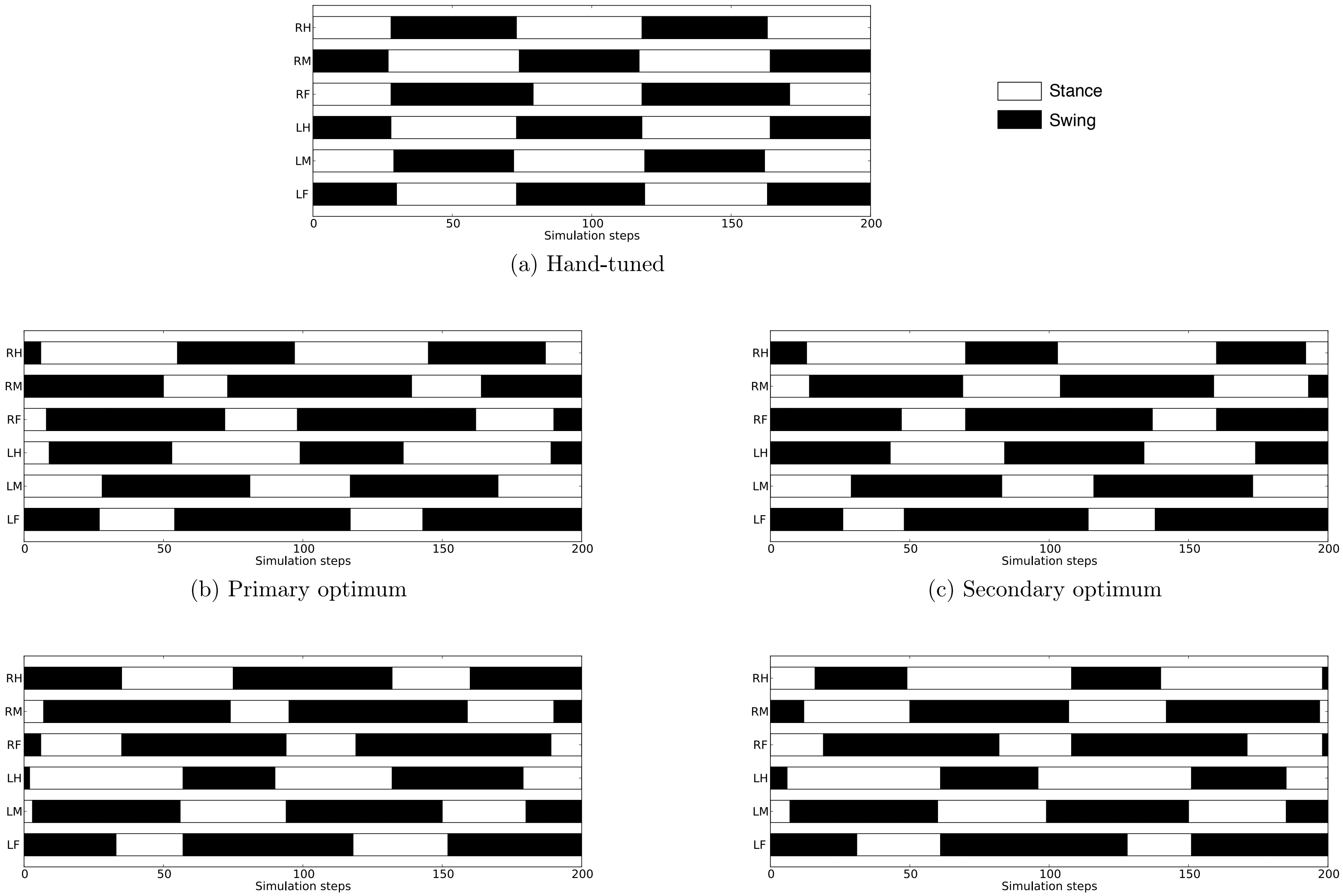
Quaternary optimum

The quaternaty solution is similar to the alternating tripod gait, but eliminates the speed losses caused by the forelegs touching the ground too early and preventing more forward motion.

5.3 Comparison to biology

The evolved gaits are similar to known biological gaits; however, most of the optimized solutions are not associated with *Drosophila* locomotion. Furthermore, a greater amount of time is spent in the swing phase than is observed in biology. In this way, the gaits are unique. The alternating tripod gait is often considered the hexapod-equivalent of the quadruped trot [4], but in our case, we found a gait which uses fewer legs touching the ground at a time. It is logical that such gaits would outperform the alternative tripod gait when only speed is considered. After all, if only two legs are used at a time, we can have three power phases for each oscillator period, while the alternating tripod gait is limited to two power strokes per cycle. This incongruity with nature comes from the limitations of our simulation and the fact that we were only optimizing for speed. Intuitively, we know that the *Drosophila* gait is optimized for more than just this. Energy consumption, stability, and maneuverability all come to mind as other aspects in a real-world fitness function.

Something which this simulation does not take into account was the fact that *Drosophila* (and other flying insects) usually walk on walls, ceilings, or other non-horizontal surfaces and need to actively adhere to the wall. With only two points of contact, the fly would be unstable and may fall off. For this reason, it appears that an important condition for *Drosophila* locomotion is that there always be **at least three legs touching the ground at a given point in time**. Because of the way that the simulation was constructed with gravity as the only attractive force, the fly is able to use gaits which are only dynamically stable – typically the realm of larger land animals. In many ways, we have found what gaits may have evolved if flies did not develop the ability to adhere to surfaces. A more accurate simulation would have to account for the grasping/adhesion of the claws to the surface. Practically, this could take several forms. As a simple



(a) Hand-tuned

(b) Primary optimum

(c) Secondary optimum

(d) Tertiary optimum

(e) Quaternary optimum

Figure 34: Gait comparison

implementation, claw grabbing or adhesion can included by dynamically changing the coefficient of friction between individual claw materials (which would need to be created) and the ground (every object that is included in the physics calculation has an associated material and coefficient of friction with other materials). Alternatively, the density/mass of the claw can be dynamically adjusted during the simulation by the controller in order to "grasp" or "not grasp" the surface. Note that the grasping should only be allowed once the claw is in contact with the surface. A more accurate, though difficult, solution is to adjust the physics engine to account for the attractive force of the claws. Additional research on the biological side will likely be necessary to understand the adhesion mechanism before implementing it in simulation.

6 Conclusion and future work

Understanding insect locomotion is an ambitious but ultimately achievable goal. *Drosophila melanogaster* are a particularly appealing insect to study due to its long history of biological and genetic studies. With *Drosophila*, we can engineer experiments which could not be done with any other organism. The outcome of such a pursuit could help inform our understanding of human locomotion and would allow us to tap the understanding of millions of years of evolution to create more robust and dynamic walking systems. The field of robotics is one which tries to leverage advances in all fields of science and technology, and this research lies at the exciting interface between biology and engineering. Ultimately, such work can help both both fields increase their understanding. Biological systems can inform how we design and program our robots, while simulation and robotic systems allow biologists to test hypotheses that might otherwise be impossible to test.

The model presented here will hopefully allow us to test some basic questions on *Drosophila* locomotion. We have created a biologically-plausible fly and have demonstrated its ability to be used in optimization experiments. There are currently several hypotheses on insect locomotion ranging from CPGs to a series of simple reflexes. Through a combination of biological and computational tests, we can hope to answer some of these questions. The model is also constructed in such a way to be easily adapted to new questions. For example, it may be interesting to co-evolve morphology and control as discussed in [42]. Furthermore, because the model is created in a robotics simulator, it also has the unique advantage of easily being able to be adapted to a real robot. New, physical bio-inspired *Drosophila* robots could be created and use the same controller structure that is being used to test biological questions.

The next immediate steps are to adapt the model to account of the adhesion of the claws to surfaces and test locomotion questions with this updated model. The gaits that we evolved, though interesting are not necessarily relevant to the *Drosophila* biology. They could be used to operate a hexapod robot with the *Drosophila* morphology in normal conditions, but consider the following: What if the robot could climb walls and ceilings? This is an attractive proposition. By understanding how flies are able to do this, we may be able to create robots that can work in the same way.

Acknowledgements

I would also like to express my gratitude to everyone in the Laboratory of Intelligent Systems for being so welcoming and supportive throughout the semester. A special thank you to Pavan Ramdya and Andrea Maesani for proposing and overseeing the project. They have been wonderful partners to work with and have been helpful at every step of the project. I would also like to acknowledge Felix Schill for his help with initial imaging of the flies. Finally, I would like to thank my parents for inspiring me and supporting me through my studies at the EPFL.

References

[1] Paolo Arena, Luigi Fortuna, Mattia Frasca, and Giovanni Sicurella. An adaptive, self-organizing dynamical system for hierarchical control of bio-inspired locomotion. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, 34(4):1823–37, August 2004.

[2] T.D. Barfoot, E.J.P. Earon, and G.M.T. D'Eleuterio. Experiments in learning distributed control for a hexapod robot. *Robotics and Autonomous Systems*, 54(10):864–72, October 2006.

[3] Ulrich Bässler. Functional Principles of Pattern Generation for Walking Movements of Stick Insect Forelegs: The Role of the Femoral Chordotonal Organ Afferences. 147:125–47, 1988.

[4] Ulrich Bässler and Ansgar Büschges. Pattern generation for stick insect walking movements–multisensory control of a locomotor program. *Brain research. Brain research reviews*, 27(1):65–88, June 1998.

[5] R. D. Beer and J. C. Gallagher. *Evolving Dynamical Neural Networks for Adaptive Behavior*, volume 1. June 1992.

[6] Randall D Beer, Hillel J Chiel, Roger D Quinn, and Kenneth S Espenschied. A Distributed Neural Network Architecture for Hexapod Robot Locomotion. *Engineering and Science*, 365(Graham 1985):356–65, 1992.

[7] John a Bender, Elaine M Simpson, Brian R Tietz, Kathryn a Daltorio, Roger D Quinn, and Roy E Ritzmann. Kinematic and behavioral evidence for a distinction between trotting and ambling gaits in the cockroach Blaberus discoidalis. *The Journal of experimental biology*, 214(Pt 12):2057–64, June 2011.

[8] Josh Bongard. Morphological change in machines accelerates the evolution of robust behavior. *Proceedings of the National Academy of Sciences of the United States of America*, 108(4):1234–9, January 2011.

[9] Josh Bongard, Victor Zykov, and Hod Lipson. Resilient machines through continuous self-modeling. *Science (New York, N.Y.)*, 314(5802):1118–21, November 2006.

[10] Rodney A. Brooks. A Robot that Walks; Emergent Behaviors from a Carefully Evolved Network. Technical report, 1989.

[11] D Bucher. Central Pattern Generators. In Larry R. Squire, editor, *Encyclopedia of Neuroscience*, pages 691–700. Academic Press, Oxford, 2009.

[12] Maurice Clerc. *Particle Swarm Optimization.* ISTE Ltd., 2006.

[13] J. J. Collins and Ian Stewart. Hexapodal gaits and coupled nonlinear oscillator models. *Biological Cybernetics*, 68:287–298, 1993.

[14] Holk Cruse. What mechanisms coordinate leg movement in walking. 1990.

[15] Holk Cruse. The functional sense of central oscillations in walking. *Biological Cybernetics*, 86(4):271–80, April 2002.

[16] Holk Cruse, Volker Dürr, and Josef Schmitz. Insect walking is based on a decentralized architecture revealing a simple and robust controller. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 365(1850):221–50, January 2007.

[17] Holk Cruse, Thomas Kindermann, Michael Schumm, Jeffrey Dean, and Josef Schmitz. Walknet-a biologically inspired network to control six-legged walking. *Neural networks : the official journal of the International Neural Network Society*, 11(7-8):1435–47, October 1998.

[18] Cyberbotics. Webots Reference Manual. Technical report, 2012.

[19] Cyberbotics. Webots User Guide. Technical report, 2012.

[20] Gennady S. Cymbalyuk, Roman M. Borisyuk, Uwe Müller-Wilm, and Holk Cruse. Oscillatory network controlling six-legged locomotion. Optimization of model parameters. *Neural networks : the official journal of the International Neural Network Society*, 11(7-8):1449–60, October 1998.

[21] Kalyanmoy Deb and Nikhil Padhye. Development of Efficient Particle Swarm Optimizers by Using Concepts from Evolutionary Algorithms. *Proceedings of the 12th annual conference on Genetic and evolutionary computation - GECCO '10*, page 55, 2010.

[22] Fred Delcomyn. Walking Robots and the Central and Peripheral Control of Locomotion in Insects. *Autonomous Robots*, 270:259–70, 1999.

[23] Fred Delcomyn. Insect walking and robotics. *Annual review of entomology*, 49:51–70, January 2004.

[24] Volker Dürr, Josef Schmitz, and Holk Cruse. Behaviour-based modelling of hexapod locomotion : linking biology and technical application. *Arthropod Structure & Development*, 33:237–50, 2004.

[25] Jacques Duysens and Henry W.a.a Van de Crommert. Neural control of locomotion; Part 1: The central pattern generator from cats to humans. *Gait & Posture*, 7(2):131–141, March 1998.

[26] Orjan Ekeberg, Marcus Blümel, and Ansgar Büschges. Dynamic simulation of insect walking. *Arthropod structure & development*, 33(3):287–300, July 2004.

[27] Kenneth S. Espenschied, Roger D. Quinn, Randall D. Beer, and Hillel J. Chiel. Biologically based distributed control and local reflexes improve rough terrain locomotion in a hexapod robot. *Robotics and Autonomous Systems*, 18:59–64, 1996.

[28] Cynthia Ferrell. *Robust Agent Control of an Autonomous Robot with Many Sensors and Actuators*. PhD thesis, 1993.

[29] Cynthia Ferrell. A comparison of three insect-inspired locomotion controllers. *Robotics and Autonomous Systems*, 16(2-4):135–59, December 1995.

[30] V French, M Feast, and L Partridge. Body size and cell size in Drosophila: the developmental response to temperature. *Journal of insect physiology*, 44(11):1081–9, November 1998.

[31] Scott F. Gilbert. *Developmental Biology*. Sinauer Associates, Inc., 9th edition, 2010.

[32] Karl Georg Göotz and Hans Wenking. Visual Control of Locomotion in the Walking Fruitfly Drosophila. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, 85(3):235–66, 1973.

[33] D Graham. Pattern and Control of Walking in Insects. *Advances*, page 1985, 1985.

[34] D M Guthrie. Multipolar Stretch Receptors and the Insect Leg Reflex. *Insect Physiology*, 13:1637–44, 1967.

[35] Cynthia L Hughes and John B Thomas. A sensory feedback circuit coordinates muscle activity in Drosophila. *Molecular and cellular neurosciences*, 35(2):383–96, June 2007.

[36] Stefan Hutter. *Co-evolution of Morphology and Controller of a Simulated Underactuated Quadruped Robot using Evolutionary Algorithms*. PhD thesis, 2009.

[37] Auke Jan Ijspeert. Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, 21(4):642–53, May 2008.

[38] Shinkichi Inagaki, Hideo Yuasa, and Tamio Arai. CPG model for autonomous decentralized multi-legged robot system–generation and transition of oscillation patterns and dynamics of oscillators. *Robotics and Autonomous Systems*, 44(3-4):171–9, September 2003.

[39] Thorsten Jeck and Holk Cruse. Walking in Aretaon asperrimus. *Journal of insect physiology*, 53(7):724–33, July 2007.

[40] T. Kindermann. Behavior and Adaptability of a Six-Legged Walking System with Highly Distributed Control. *Adaptive Behavior*, 9(1):16–41, March 2001.

[41] Bernhard Klaassen, Ralf Linnemann, Dirk Spenneberg, and Frank Kirchner. Biomimetic Walking Robot Scorpion : Control and Modeling. *Advanced Robotics*, 41(2-3):69–76, 2002.

[42] Ken Larpin. *Co-evolution of Morphology, Control and Behavior*. PhD thesis, 2011.

[43] H Lipson and J B Pollack. Automatic design and manufacture of robotic lifeforms. *Nature*, 406(6799):974–8, August 2000.

[44] Albert Miller. The Internal Anatomy and Histology of the Imago of Drosophila Melanogastor. In *Biology of Drosophila*, pages 420–531. John Wiley & Sons, New York, NY, 1950.

[45] J Schmitz, J Dean, T Kindermann, M Schumm, and H Cruse. A biologically inspired controller for hexapod walking: simple solutions by exploiting physical properties. *The Biological Bulletin*, 200(2):195–200, April 2001.

[46] Karl Sims. Evolving 3D Morphology and Behavior by Competition. *Artificial Life*, 1(4):353–72, January 1994.

[47] Helen Sink. *Muscle Development in Drosophila*. Springer, 2006.

[48] Cédric Soler, Malgorzata Daczewska, Jean Philippe Da Ponte, Bernard Dastugue, and Krzysztof Jagla. Coordinated development of muscles and tendons of the Drosophila leg. *Development (Cambridge, England)*, 131(24):6041–51, December 2004.

[49] Andrew J Spence, Shai Revzen, Justin Seipel, Chris Mullens, and Robert J Full. Insects running on elastic surfaces. *The Journal of experimental biology*, 213(11):1907–20, June 2010.

[50] R Strauss and M Heisenberg. Coordination of legs during straight walking and turning in Drosophila melanogaster. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, 167(3):403–12, August 1990.

[51] Roland Strauss. The central complex and the genetic dissection of locomotor behaviour. *Current Opinion in Neurobiology*, 12(6):633–38, December 2002.

[52] Donald M. Wilson. Insect Walking. *Annual Review of Entomology*, 11:103–22, 1965.

[53] Sasha Zill, Josef Schmitz, and Ansgar Büschges. Load sensing and control of posture and locomotion. *Arthropod structure & development*, 33(3):273–86, July 2004.

A Joint limits

Table 3 gives the observed joint angle limits (minimum and maximum) during normal walking. Refer to Figure 23 for a diagram of how angles are measured. All measurements are in degrees.

	Thorax-Coxa (Rotation)	**Thorax-Coxa (Lifting)**	**Coxa-Femur**	**Femur-Tibia**
Foreleg	70 to 80	-40 to 10	85 to 170	35 to 140
Mid-leg	-15 to 20	20	90 to 110	90 to 110
Hind-leg	-75 to -40	40 to 60	40 to 110	50 to 145

Table 3: Joint angle limits during normal walking

B Installation

Installing Python and packages

1. Get the latest version of Python (2.X) from `http://www.python.org/download/`
2. Install Python on the system:

```
$ tar xvjf Python-2.7.3.tar.bz2
$ cd Python-2.7.3.tar.bz2
$ ./configure
$ make
$ su
$ make install
$ exit
$ make clean
```

3. Get ez_setup.py from `http://peak.telecommunity.com/dist/ez_setup.py` (or download setuptools from `http://pypi.python.org/pypi/setuptools`)
4. Install easy_install:

```
$ python ez_setup.py
```

5. Install pip and use to get necessary (and useful) packages:

```
$ easy_install pip
$ pip install inspyred
```

```
$ pip install numpy
$ pip install scipy
$ pip install matplotlib
```

Running simulation

1. Open the `drosophila.wbt` file from inside the `worlds/` folder

Running optimization

1. Configure optimization parameters (coefficients, termination conditions, etc.) and ensure that the Webots path is correct within `optimize.py`

2. Run the `optimize.py`:

   ```
   $ python optimize.py
   ```

Note: multiple instances of Webots can run simultaneously. A simple way to run multiple experiments in parallel is to simply duplicate the `drosophila/` folder and run the script in the new folder as well.

C Code

All of the necessary files for the model are contained within a single folder (with the exception of the Python and Webots installations):

```
drosophila/
    controllers/
        drosophila/
            drosophila.py
            parameters.txt
            results.txt
    worlds/
        drosophila.wbt
    optimize.py
    world_generator.py
    log.csv
```

In reality, only the Python files are necessary, since `world_generator.py` can generate the Webots world file and most other files are only created once the optimization is run. The following pages contain the core Python code.