Automated gait synthesis and path planning for legged underwater vehicles

Andrew German

Technical Report CSE-2008-03

July 10, 2008

Department of Computer Science and Engineering

4700 Keele Street Toronto, Ontario M3J 1P3 Canada

Abstract

Legged mobile robots move by executing patterns of leg-joint angles called *gaits*. Synthesizing gaits by hand is a complex and time-consuming task that is even more challenging when the vehicle operates in the underwater domain. When the vehicle is suspended in a fluid any motion of the limbs, no matter how small, causes drag which applies forces to the vehicle. Underwater gaits must therefore be constructed to mitigate these unwanted forces so that joint-angle changes apply a net force in the desired direction. This thesis presents an automatic gait synthesis system for underwater legged vehicles. The process has two stages: First, a simulated annealing engine is used to build an alphabet of short duration gaits that can be used to reposition and reorient the robot in the six-degrees-of-freedom space. Second, an obstacle-aware path planner combines members of the gait alphabet to form more complex motions. The technique is applied to synthesize gaits for a simulated version of the AQUA amphibious hexapod although it is general enough to be applied to other legged vehicles.

Table of Contents

Abstract **i**

Table of Contents **ii**

1 Introduction **1**

1.1 Problem statement 6

1.2 Structure of this work 9

2 Previous work **11**

2.1 Locomotion 11

2.1.1 Wheeled and tracked locomotion 12

2.1.2 Body undulation locomotion 15

2.1.3 Propeller-driven locomotion 18

2.1.4 Legged locomotion 25

2.2 Gaits 28

2.2.1 Underwater gaits 31

2.3 Gait synthesis 33
2.3.1 Neural nets 35
2.3.2 Genetic algorithms 37
2.3.3 Simulated annealing 40
2.3.4 Gait transition 43
2.4 AI search techniques 45
2.5 Summary 48

3 Motion synthesis **51**
3.1 Gait synthesis 52
3.1.1 Seed gait 54
3.1.2 Simulated annealing engine 56
3.1.3 Evaluation of the final state 59
3.1.4 Sample synthesis 60
3.2 Summary 70

4 An alphabet of gaits **73**
4.1 The gaits 73
4.1.1 Yaw Left 74
4.1.2 Yaw Right 78
4.1.3 Surge 82

4.1.4 Pitch Up 86
4.1.5 Pitch Down 90
4.1.6 Roll Left 94
4.1.7 Roll Right 98
4.1.8 Heave Up 102
4.1.9 Heave Down 106
4.1.10 Accelerate 110
4.1.11 Decelerate 114
4.2 Summary 118

5 Path planning with the gait alphabet **123**
5.1 Application of A* to motion synthesis 125
5.2 Linking gaits into complex maneuvers 128
5.3 Path-planning example 129
5.4 Dynamics constraints and considerations 134
5.5 Summary 136

6 Experimental validation **140**
6.1 Experiment 1 140
6.2 Experiment 2 143
6.3 Summary 144

7 Summary and future work **147**

A Simulator details **152**

Bibliography **156**

Chapter 1 Introduction

Is there a moment mid-stride when horses have all four legs off the ground? This popular question spurred lively debate in the mid-1800s. Some insisted they could discern the horse's ballistic flight while others countered that without the support of at least one leg, the horse would collapse. In 1872, former California Governor, railroad baron and believer of the "unsupported transit" theory Leland Stanford sought out photographic artist Eadweard Muybridge to collect the photographic evidence necessary to settle the debate [25]. In 1878 their partnership produced its first unequivocal results: A series of still photographs captured as a horse drawn cart individually triggered 12 (then) state-of-the-art cameras aimed at the track, all in less than half a second (samples of Muybridge's photographs can be seen in Figures 1.1(a) and 1.1(b)). The caption in Figure 1.1(a) reads:

> "Abe Edgington," owned by Leland Stanford; driven by C. Marvin, trotting at a 2:24 gait over the Palo Alto track, 15th June 1878. The negatives of these photographs were made at intervals of about the twenty-fifth part of a second of time and twenty-one inches of distance; the exposure of each was about the two-thousandth part of a second, and illustrate one single stride of the horse. The vertical lines were placed twenty-one inches apart; the lowest horizontal line represents the level of the track, the other elevations of four, eight and twelve inches respectively. The negatives are entirely "untouched."

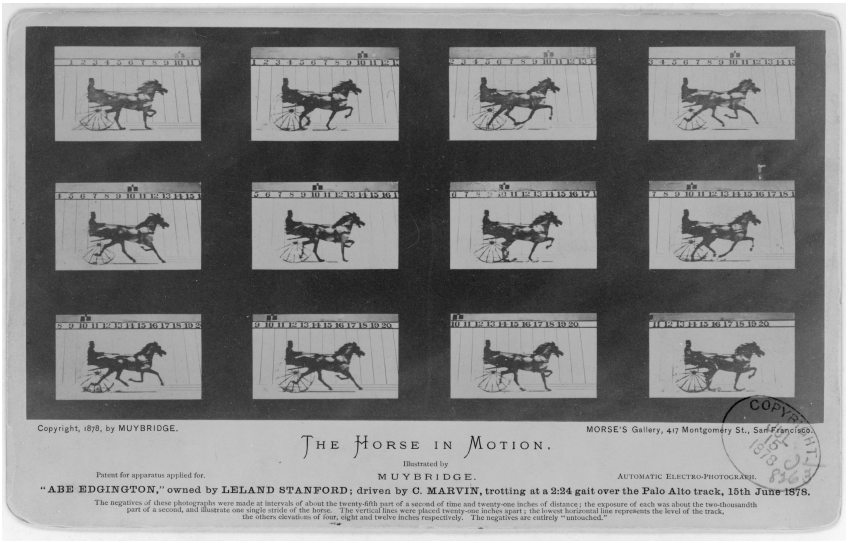
The caption for Figure 1.1(b) reads:

> "SALLIE GARDNER," owned by LELAND STANFORD; running at a 1.40 gait over the Palo Alto track, 19th June, 1878. The negatives of these photographs were made at intervals of twenty-seven inches of distance, and about the twenty-fifth part of a second of time; they illustrate consecutive positions assumed in each twenty-seven inches of progress during a single stride of the mare. The vertical lines were twenty-seven inches apart; the horizontal lines represent elevations of four inches each. The exposure of each negative was less than the two-thousandth part of a second.

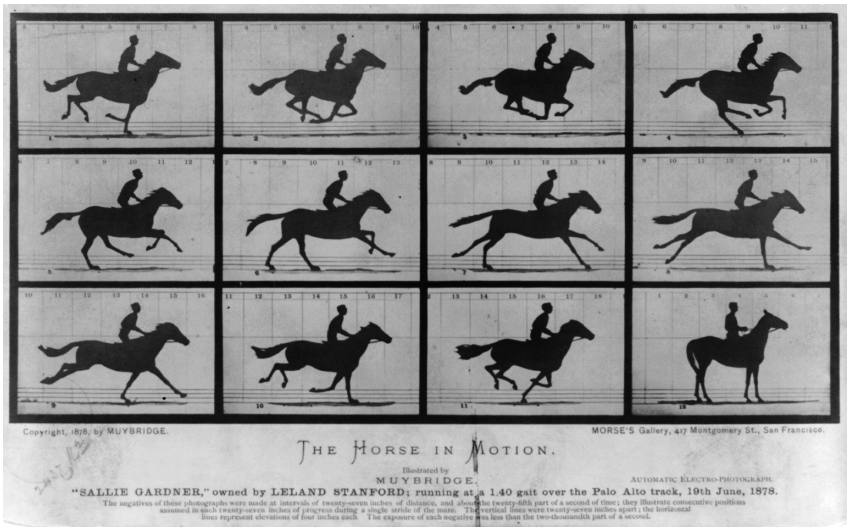
As is clearly evident from the images, horses do indeed have short moments of ballistic flight in mid-gallop.

The work of Stanford and Muybridge was important for reasons beyond settling the popular debate: First, the pictures helped artists and sculptors tune their own equine representations to look more realistic. (Until that point galloping horses were usually represented in the hobbyhorse pose with all four legs extended.) Second, the photographic techniques developed continue to inspire motion picture creators to this day. The movie "The Matrix," for example, contains 'Bullet-Time' sequences shot using techniques initially developed by Muybridge. Third, the photos taken of the galloping horse were some of the first forays into scientific gait research.

Gait refers to the patterns of leg placements throughout the locomotion cycle of a legged system. For a given legged system many different gaits are possible. Human gaits include shuffling, walking, running, and hopping. Horse gaits include walking, trotting, cantering, and galloping. Legged robots like the ones depicted in Figure 1.2 also move by executing patterns of leg motions. In terms of animals or machines with



(a)

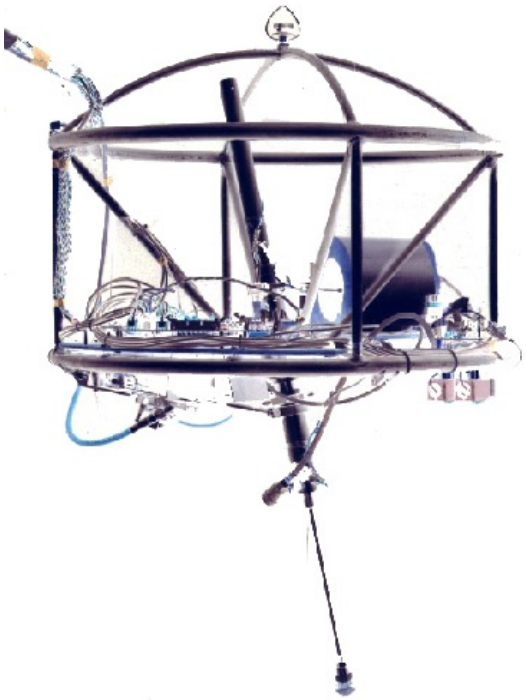


(b)

Figure 1.1: Photographs of “The Horse in Motion” by Eadweard Muybridge [30].

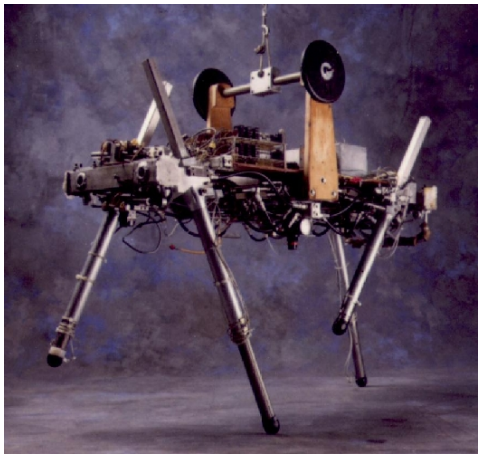
rigid articulated legs the gait can be described in terms of the pattern of joint angles as a function of time. Figure 1.3 illustrates the leg angles involved in the surge gait for the AQUA hexapod shown in Figure 1.2(d). This hand crafted gait for the hexapod was developed for underwater swimming. It involves having each leg oscillate sinusoidally $\pm 22.5^{\circ}$ from the plane of the vehicle. A phase offset is added to the angles of the two middle legs which was found to provide added vehicle stability. This gait moves the vehicle forward along its longitudinal axis.

The task of generating gaits for legged robots is complex. Given a start position (state) and an end position (state) for the robot, the task is to generate patterns of joint angles as a function of time that move the robot between those two configurations. Automatic gait generation for legged robots is a relatively new research area. Most commonly, legged robot gaits are designed and tuned by hand. This approach, while quick to implement, has several drawbacks: First, moving the robot between two positions involves carefully hand-tuning the pattern of individual leg joint angles to generate the intended motion. This can be very time consuming. Second, hand tuning gaits does not scale well with the complexity of the vehicle (its degrees-of-freedom – DOF) or with increases in environmental complexity. For example, in a six degree-of-freedom (6DOF) environment such as underwater or outer space, motion not only involves the x and y position and the yaw of the vehicle, but the z position and the roll and pitch as well.

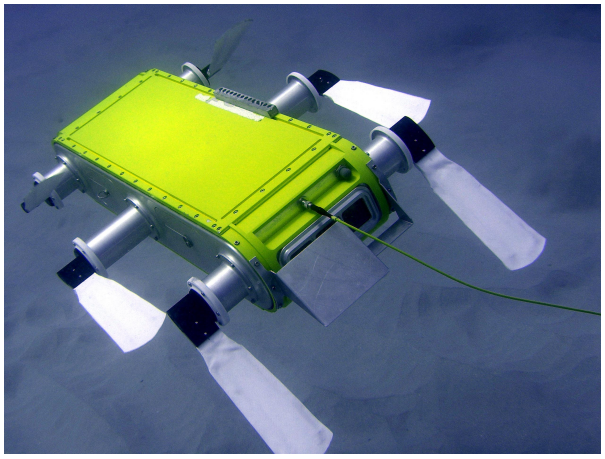


(a) 3D One-Leg Hopper

(b) QRIO



(c) Quadruped



(d) AQUA

Figure 1.2: A sample of legged robots: (a) The 3D One-Leg Hopper from the MIT Leg Lab. Reprinted from [34]. (b) The Sony QRIO biped. Reprinted from [42]. (c) Quadruped four-legged robot from the MIT Leg Lab. (d) The AQUA amphibious hexapod.

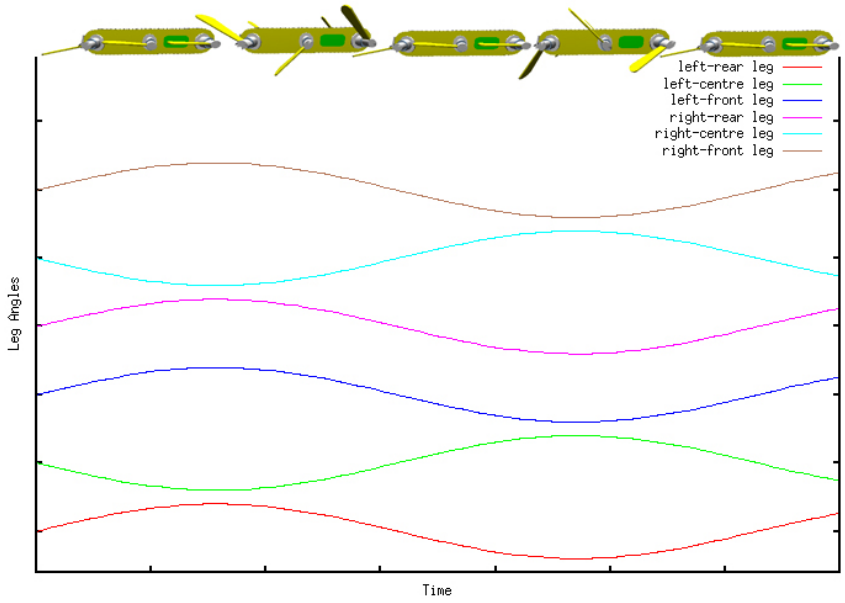


Figure 1.3: A graph of the leg angles involved in a surge gait accompanied by samples from the gait itself. Each leg rotates between $-22.5°$ and $+22.5°$. The two centre legs are phase-offset because through experimentation it was discovered that this increased vehicle stability.

Given the limitations of hand-crafted gaits, especially in high degree-of-freedom environments, it is important to develop strategies to automate this process through automatic gait synthesis. It is this problem that is addressed in this work.

1.1 Problem statement

This thesis examines the problem of generating gaits to move an underwater legged robot between two configuration states. The problem is summarized graphically in Figure 1.4. This work develops an algorithm that automacially constructs the pattern of leg motions that enables a vehicle to move from its start to goal positions and orientations. This

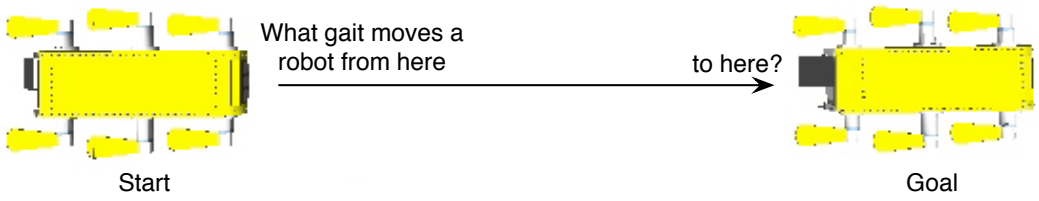


Figure 1.4: The problem: which pattern of leg motions moves the robot to a specific location?

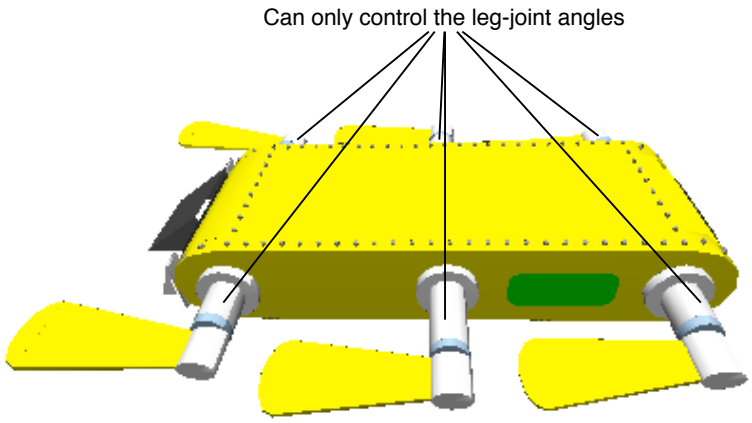


Figure 1.5: A major constraint on the gait synthesis system is that the dynamics of the vehicle cannot be directly controlled. Instead the leg joint-angles must be oscillated to generate the desired motion.

problem is complicated by the fact that there is no mechanism for direct control over the vehicle's position and orientation. Rather control exists only for the joint angles for each of the robot's legs (Figure 1.5). So the problem can be rephrased as: "what pattern of joint angles applies the appropriate forces to the robot so that it moves to a specific location and orientation (system state) given an initial state of the robot?"

In this work the problem of gait synthesis is separated into two different tasks: (i) the

automatic synthesis of a small alphabet of gaits that meet specific hydrodynamic properties as well as specific constraints on leg orientations at the beginning and end of the gaits, and (ii) the development of a strategy to enable these gaits to be sequenced so as to provide complex longer term motion strategies. Individual gait synthesis is formulated in terms of a non-linear optimization process that is tackled using simulated annealing [23]. Rather than attempting to construct a closed-form version of the dynamic model of the vehicle, a black-box hydrodynamics simulation approach is used instead. The simulated annealing system optimizes the control inputs (leg parameters) subject to pre-specified goal constraints. This results in a general approach to gait synthesis that makes very few assumptions about the specifics of vehicle dynamics. The process of sequencing gaits to form complex motions is treated as a classical AI problem and is addressed via a graph search algorithm [31]. This search process takes place in a 6DOF state space which treats the problem independently of the dynamics and kinematics of the vehicle. Instead the search process is constructed so that the kinematics and dynamics constraints required at the start of each gait are properly matched with those that hold at the end of the previous gait. Breaking the process of determining long-term gait strategies to move the vehicle into these two components provides significant computational savings.

Although the solutions presented in this thesis focus on the problem in general and are directed towards underwater legged robots generally, it has specific application to the AQUA amphibious hexapod (Figure 1.2(d)) [9, 15]. The AQUA project is a joint

research project between McGill University, York University and Dalhousie University to develop an amphibious mobile sensing platform. The AQUA robot is capable of walking on land, wading through the surf zone and swimming in open water. It has several on-board cameras and an inertial measurement unit. AQUA was designed to examine robotic contributions to the *site acquisition and scene reinspection* (SASR) task [9, 15]. In a SASR task the robot walks out into the water under operator control and is teleoperated to a particular location from which it will gather sensor measurements. Once near the site, the robot achieves an appropriate pose from which to undertake extensive sensor readings. After making the measurements, the robot returns to the starting position autonomously. Based on the measurements made while teleoperated the robot should return autonomously to the site to collect additional data at a later date. The complex 6DOF movement needed for SASR tasks makes custom hand-made gaits largely impractical.

1.2 Structure of this work

The remainder of this thesis is organized as follows. Chapter 2 provides an overview of robotic locomotion, especially legged locomotion and surveys existing approaches to gait synthesis in both land and sea-based legged robotic vehicles. It also reviews work related to graph search techniques. Chapter 3 describes the approach taken in this work to synthesize an alphabet of underwater swimming gaits for legged robots. Chapter 4 details gaits that have been synthesized using this approach. These gaits are

then used to form an alphabet of available gaits. Chapter 5 discusses the use of the gaits depicted in Chapter 4 in a 6DOF path planner which can generate complex long-term motions for the vehicle while avoiding pre-mapped obstacles in the environment. Chapter 6 provides some experimental validation for the combination of the gait alphabet and the path planner. Finally, Chapter 7 provides a summary of the work done, as well as indicating possible directions for future work. Details of the hydrodynamic model and specific details of the hydrodynamic simulations used in this work are provided in the Appendix.

Chapter 2 Previous work

Perhaps it should go without saying, but for a mobile robot to be effective, it must be mobile. Thus technologies to enable a vehicle to move in a safe and intelligent manner are critical components of any mobile robot and their study is a central aspect of robotics research. This chapter briefly reviews the state of the art in robotics related to mobility, with particular emphasis on techniques that are appropriate for long-term motion plan generation for legged vehicles. This chapter also provides a short introduction to classical graph searching techniques with particular emphasis on their ability to solve planning problems.

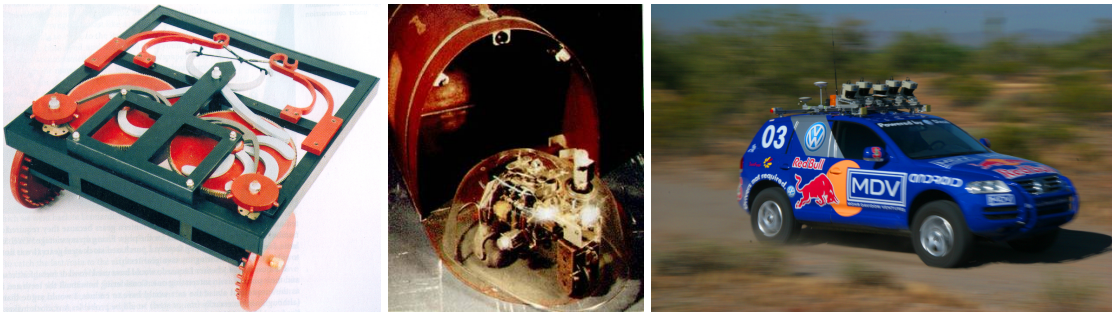
2.1 Locomotion

The Oxford American Dictionary defines locomotion as the "movement or the ability to move from one place to another." Mobile robots employ a number of different mechanisms to achieve motion. We begin by evaluating critical classes of mechanisms and provide examples of mobile robots that use these techniques.

2.1.1 Wheeled and tracked locomotion

Undoubtedly the most common and simplest way for a robot to exhibit useful motion is through the use of wheels or tracks. Wheeled locomotion has been in use for several thousand years due to its efficiency, stability, and operational simplicity. Wheels have been the mechanism of choice for many mobile robots throughout the ages. Leonardo Da Vinci designed, and reportedly built, a three-wheeled programmable automaton circa 1478 [38] (see Figure 2.1(a)). His autonomous cart was programmed using various cams attached to the top of the large barrel gears. These cams varied the direction and velocity of the vehicle allowing it to execute fairly complex paths. In 1948 W. Grey Walter developed the Bristol Tortoise [47] (Figure 2.1(b)), a fully autonomous electrical wheeled robot that could perform tasks such as wandering without mishap and navigating back to its recharging hutch. More recently, wheel-based autonomous vehicles have been able to successfully navigate 142 miles of the Mojave desert during the Defense Advanced Research Projects Agency (DARPA) Grand Challenge. Stanford University's winning entry, Stanley (Figure 2.1(c)), completed the course in 6h, 53 min and 58 sec.

Wheeled systems are simple to construct, they work well on terrains ranging from relatively even surfaces to gentle slopes, and can perform at a wide range of speeds. Tracked vehicles essentially extend wheels to provide a larger region of ground contact and permit a greater range of terrain to be traversed (Figure 2.2(b)). Unfortunately,

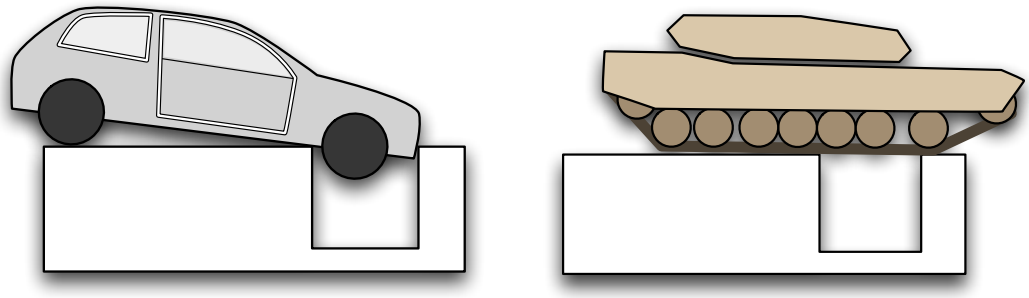


(a) Da Vinci's programmable cart (b) Bristol tortoise (c) Stanley

Figure 2.1: Autonomous wheeled robots. (a) A re-creation of Da Vinci's three-wheeled programable automaton. Reprinted from [38]. (b) Walter's 1948 wheeled mobile robot, the Bristol tortoise. Reprinted from [10]. (c) Stanford University's winning entry into the 2005 DARPA Grand Challenge, Stanley. Reprinted from [44].

there are limitations to wheeled and tracked vehicles: Perhaps most crucial is their need for ground support along the entire path of motion [10] (Figure 2.2(a)). This criterion is not always guaranteed to be met in rough terrain such as forests, disaster areas, and during planetary exploration. For environments lacking the ground contact needed for wheeled or tracked vehicles, alternative locomotion strategies are required.

Of all the types of robot locomotion, wheeled vehicles that use a differential drive system (those in which the two driving wheels are mounted on a shared axis with separate motors) are perhaps the easiest to model. Their position and orientation can be solved relatively easily by using a forward kinematics approach that maps control inputs (wheel-ground contact speeds) to vehicle pose as a function of time [10]. In other words, it is possible to compute where the vehicle will be at any time t based on the control parameters v_l, the ground velocity of the left wheel, v_r, the ground velocity of the right



(a) (b)

Figure 2.2: Wheeled and tracked vehicles. (a) Wheeled vehicles require ground support along the entire path of motion. (b) Tracked vehicles extend the ground contact zone and can thus traverse a greater range of terrain.

wheel, and the distance between the wheels l, (see Figure 2.3). At any instance of time a differential drive vehicle will rotate around its instantaneous centre of curvature (ICC) which is defined as follows. Let

$$R = \frac{l}{2}\frac{(v_l + v_r)}{(v_r - v_l)}$$

then the ICC is given by the vector

$$\overrightarrow{ICC} = [x - Rsin(\theta), y + Rcos(\theta)],$$

where the robot's current position is (x, y). The vehicle will rotate around the ICC at a rate ω given by:

$$\omega = \frac{v_r - v_l}{l}.$$

By integrating this rotation about the ICC as a function of time it is possible to compute the pose of the vehicle as a function of time. Although this type of forward kinematic

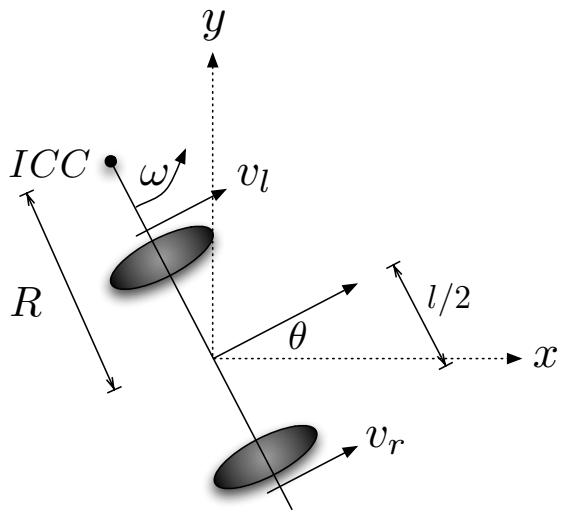


Figure 2.3: Differential drive kinematics. By varying the velocity of the left and right wheel a differential drive vehicle is able to control its pose.

pose estimation is possible for simple wheeled robots it is not sufficient for more complex locomotive strategies including legged vehicles.

Although the kinematics of wheeled robots are relatively straightforward, their inability to deal with a range of ground surfaces and workplaces lacking a ground altogether has lead to the development of alternative locomotive strategies including snake-like locomotion and robots using thrusters and legs. These locomotive strategies are considered in the following sections.

2.1.2 Body undulation locomotion

Robots that are biologically inspired are becoming ever more popular. Snake-like robots move by using the same serpentine locomotion as their vertebrate namesakes. Figure 2.4

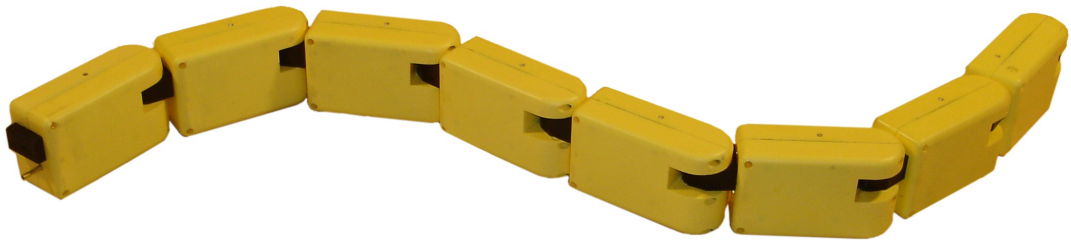


Figure 2.4: The AmphiBot II, an amphibious snake-like robot. Reprinted from [7].

depicts AmphiBot II [7], an amphibious snake-like robot. AmphiBot II achieves motion by using linked non-linear oscillators to produce travelling waves both on land and in open water.

There are several advantages to these snake-like vehicles which all stem from their flexibility: First, because of their shape, and with the addition of lights, cameras and other sensors, they can be used for search and rescue operations in highly unstructured environments such as that found in the rubble associated with disaster areas. If the robot is designed for amphibious operation the set of searchable disaster areas includes environments such as flooded mines and ship wrecks. Second, their ability to scale structures the same way that snakes scale trees (see NASA's snakebot, Figure 2.5) provides a level of mobility that is not possible with other technologies. Third, since each link in the robot can be designed as a modular unit, the robot can employ self-reconfiguraion techniques [21]. This allows a snake-like robot to separate into smaller units to explore a large area, then re-join to overcome larger obstacles. The disadvantages to snake-like

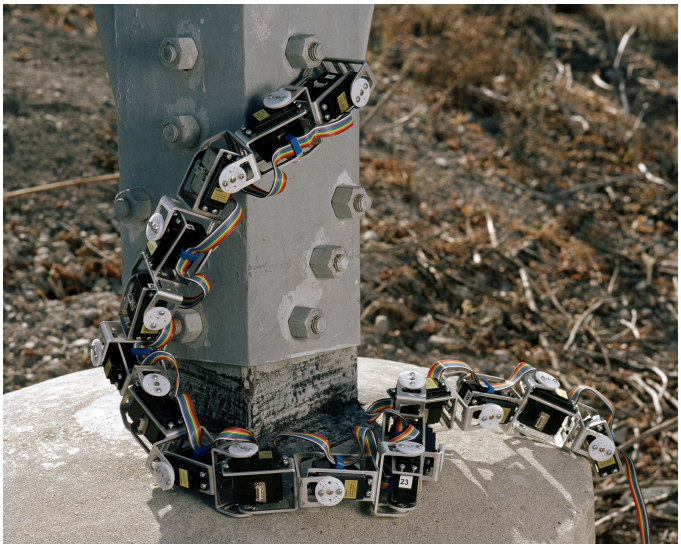


Figure 2.5: The NASA snakebot. (Photograph courtesy of NASA.)

robots include [45]: limited payload capacity, poor power efficiency, and a very large number of degrees-of-freedom which must be controlled.

A common mechanism for controlling snake-like robots is the central pattern generator (CPG) [6, 7, 40]. A CPG is a neural network capable of producing co-ordinated oscillatory signals without oscillatory inputs. For example the AmphiBot II (Figure 2.4) is controlled via a sine-wave based CPG which takes oscillation amplitude and phase difference as input and generates the oscillatory output signals which are sent to the motors [7]. The CPG and motors are designed to produce a travelling wave from the head to the tail of the robot which produces serpentine locomotion on land and in the water.

Modelling the motion of a snake-like robot is a complicated exercise in kinematics and dynamics [45]. For snake robots without wheels the dynamics of the interaction

between the surface of the robot and the surface of the ground must be modelled. Typically the dynamics presented with non-wheeled snake robots are based on a Coulomb or viscous-like friction model [40, 45].

2.1.3 Propeller-driven locomotion

Propellers or thrusters are used in both airborne and underwater vehicles. Figure 2.6 depicts two such vehicles: Figure 2.6(a) shows the Multipurpose Security and Surveillance Mission Platform (MSSMP) [29]. The MSSMP is a distributed network of remote sensors mounted on vertical-takeoff-and-landing vehicles used for a wide range of surveillance, security, and tactical missions. Figure 2.6(b) shows the ASV 6000 SASS Q from Autonomous Surface Vehicles Ltd [48]. The Survey Autonomous Semi-Submersible (SASS) technology permits small unmanned semi-submersible vehicles to deploy sensors to gather data from above or below the sea at significant distances away from a ship or from the shore.

Propellers have a long history of use in marine and aviation vehicles. Compared to wheeled and tracked vehicles, propeller driven vehicles exhibit two obvious advantages: they do not require ground contact, and they allow the vehicle to operate in 6DOF environments. There are, however, several disadvantages to propeller-based locomotion: First is the inability to station-keep or come to rest without potentially influencing (disturbing) the environment. The rotation of the propellers can disturb dust or silt which

(a) MSSMP (b) SASS

Figure 2.6: Propeller driven mobile vehicles. (a) Murphy and Bott's Multipurpose Security and Surveillance Mission Platform, reprinted from [29]. (b) The ASV 6000 SASS Q from Autonomous Surface Vehicles Ltd.

reduces visibility, not only for the robot itself, but also for other robots, humans and animals in the area. Second, since propellers generally rotate quickly and are typically not made from compliant materials, they are not safe for human operators to touch or interact with.

Controlling vehicles with propellors or thrusters involves modelling the vehicle's dynamics (i.e., the forces acting on the vehicle). This stands in contrast to wheeled vehicles which can often be successfully modelled solely via kinematics. Figure 2.7 illustrates a simple dynamics model for a fixed-wing aircraft. The motion of the vehicle is modelled by integrating the forces acting on the vehicle over time. Changing the vehicle's motion involves manipulating the controllable forces acting on the vehicle; for example, changing the vehicle's thruster configuration to generate a net force in the desired direction. A

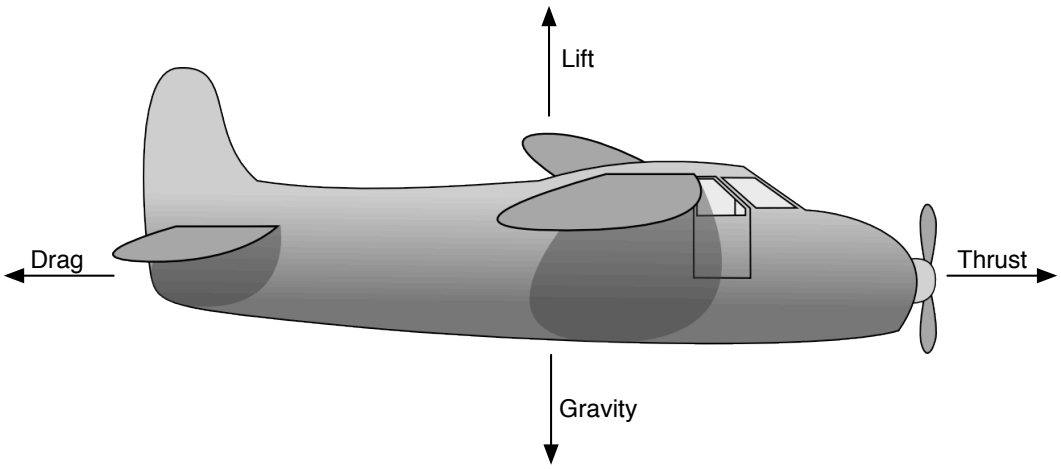


Figure 2.7: A simplified model of the forces acting on a fixed-wing aircraft. The propeller spins generating a thrust force. The reaction force to thrust is drag, the force of the air pushing on the aircraft. The movement of the wing through the air causes a lift force which must overcome the gravity force for the vehicle to become airborne.

common strategy for unmanned aerial vehicles (UAV) and unmanned underwater vehicles (UUV) is to combine thrust generating systems with controllable reaction surfaces that manipulate the slipstream passing over the vehicle. This approach is illustrated in Figure 2.8 which shows the UTA SDDR 94 tail-sitter aerial robot. The single downward facing propellor provides thrust and the black fins act as controllable reaction surfaces as the prop-wash moves past them providing the robot with both stability and manoeuvrability.

Solving the dynamics problem for an arbitrary vehicle is relatively straightforward in theory, but can be very complex in pratice. The basic approach is to develop a set of equations that describe the forces acting on the vehicle and then integrate these forces

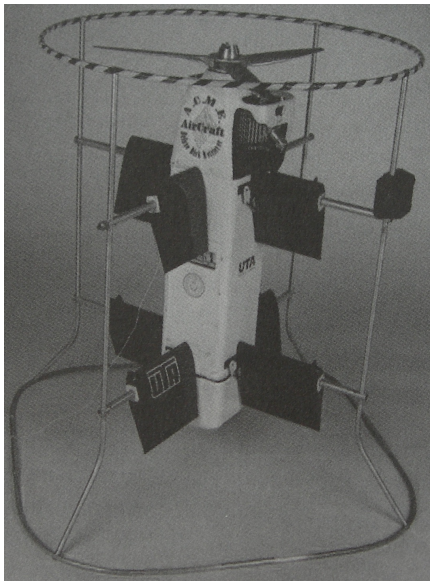


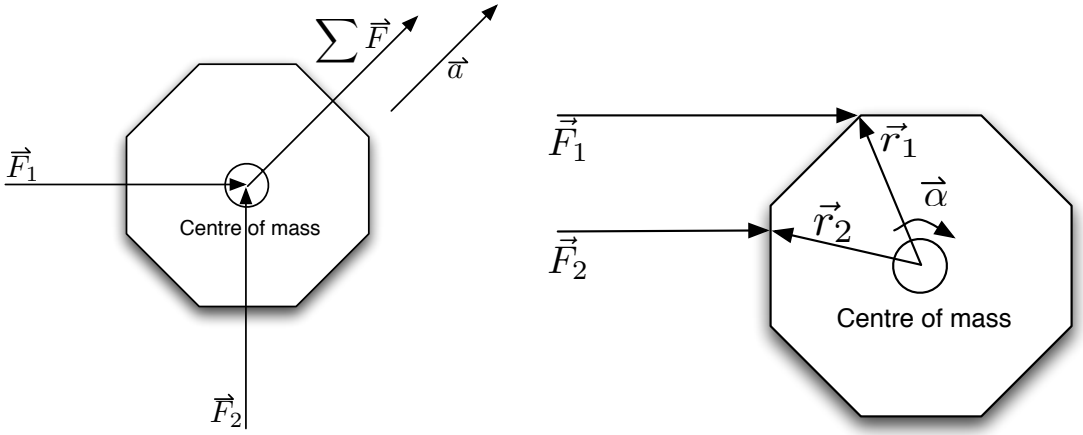
Figure 2.8: The UTA SDDR 94 tail-sitter aerial robot. Reprinted from [10].

to estimate the vehicle's state as a function of time. The dynamics model used in this thesis is presented below (see [3, 11, 12] for full details): The linear motion of a rigid body is derived from Newton's Second Law which states that acceleration of a body is proportional to the sum of the forces acting on the body (Figure 2.9(a)):

$$\sum \overrightarrow{F_i} = m\overrightarrow{a}$$

where $\overrightarrow{F_i}$ are vector forces applied to the body and expressed in the global (inertial) coordinate frame, m is the mass of the body and $\overrightarrow{a}$ is the resultant linear acceleration. Solving for $\overrightarrow{a}$ gives:

$$\overrightarrow{a} = \frac{\sum \overrightarrow{F}}{m}.$$



(a) Linear acceleration (b) Angular acceleration

Figure 2.9: Linear and angular dynamics. Linear acceleration comes from a summation of all forces acing on the body. Angular acceleration comes from the sum of the torques. A torque is derived from the cross product of the force vector and the vector representing the distance from the centre of mass to the force application point ($\overrightarrow{M}_{cg} = \overrightarrow{F} \times \overrightarrow{r}$).

Integration of linear acceleration, $\overrightarrow{a}$, over time gives linear velocity, $\overrightarrow{v}$:

$$\overrightarrow{v} = \int_0^t \overrightarrow{a}\ dt.$$

Further integration of the linear velocity of the body over time gives the position of the body, $\overrightarrow{p}$:

$$\overrightarrow{p} = \int_0^t \overrightarrow{v}\ dt.$$

Angular motion takes a similar form (Figure 2.9(b)):

$$\overrightarrow{M}_{cg} = I\overrightarrow{\alpha}$$

where $\overrightarrow{M}_{cg}$ is the angular moment acting on the centre of gravity of the body, I is a tensor (known as an inertial tensor) that represents the distribution of mass over the object and

$\overrightarrow{\alpha}$ is the resultant angular acceleration. The inertial tensor is evaluated by imagining the object to be divided into N small volume elements, each of which has a mass m_k at a position (x_k, y_k, z_k) in a local co-ordinate frame about the centre of mass. Then I is a 3×3 matrix given by

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}$$

where

$$\begin{aligned} I_{xx} &= \sum_{k=1}^{N} m_k(y_k^2 + z_k^2) \\ I_{yy} &= \sum_{k=1}^{N} m_k(x_k^2 + z_k^2) \\ I_{zz} &= \sum_{k=1}^{N} m_k(x_k^2 + y_k^2) \\ I_{xy} = I_{yx} &= -\sum_{k=1}^{N} m_k x_k y_k \\ I_{xz} = I_{zx} &= -\sum_{k=1}^{N} m_k x_k z_k \\ I_{yz} = I_{zy} &= -\sum_{k=1}^{N} m_k y_k z_k. \end{aligned}$$

$\overrightarrow{M}_{cg}$ can be expressed as:

$$\overrightarrow{M}_{cg} = \sum_i \overrightarrow{r}_i \times \overrightarrow{F}_i$$

where $\overrightarrow{F}_i$ is a force acting on the body, and $\overrightarrow{r}_i$ is the vector from the centre of mass to the application point of $\overrightarrow{F}_i$ on the body, expressed in a coordinate system aligned with the centre of mass of the vehicle. These two expressions for $\overrightarrow{M}_{cg}$ can be used to solve

for the angular acceleration, $\overrightarrow{\alpha}$. From $\overrightarrow{\alpha}$, the angular velocity, $\overrightarrow{\omega}$ can be extracted:

$$\overrightarrow{\omega} = \int_0^t \overrightarrow{\alpha}\ dt$$

Integration of the angular velocity with respect to time gives the orientation, $\overrightarrow{\Omega}$:

$$\overrightarrow{\Omega} = \int_0^t \overrightarrow{\omega}\ dt$$

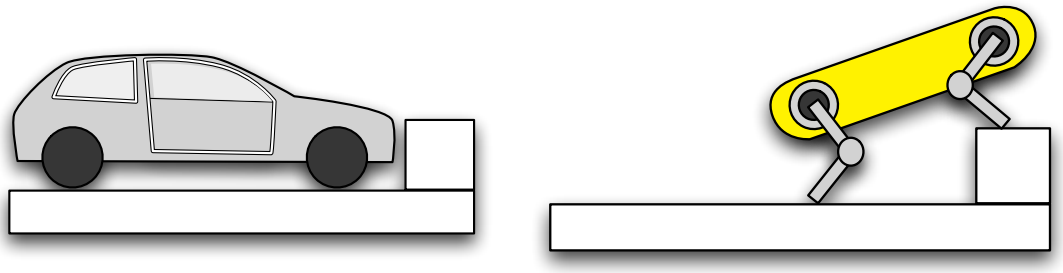
The critical observation here is that given some initial starting pose (position and orientation), knowledge of the mass distribution I, and the forces acting on the vehicle, it is possible to integrate the forces over time to determine the position and orientation of the vehicle as a function of time. In practice this integration process is quite complex and the computation can be sensitive numerically.

Modelling the dynamics of a vehicle involves obtaining accurate estimates of the forces acting on the vehicle and modelling their interaction properly. Doing this task accurately is extremely complex and a large modelling literature exists for for both aircraft (aerodynamic modelling – see [2, 41]) and watercraft (hydrodynamic modelling – see [4, 33]). One general observation about the process of accurate dynamics modelling is that ever more complex models are possible (see the references above for details). Accurate models require extremely complex representations of how actions are translated to specific forces on the vehicle and it is instructive to note that in ship and aircraft design such modelling often requires the development of scale physical models of the device(s) to build such models empirically.

In this thesis a relatively simple hydrodynamic model of a legged amphibious vehicle is used. Full details of this hydrodynamic model can be found in the Appendix. In essence this hydrodynamic model represents the vehicle as a solid block representing the body (with mass comparable to that of the real vehicle) and represents the thrust generated by the legs as thrust vectors acting on the hips of the vehicle, in the same direction as the legs and with a force proportional to the instantaneous angular velocity of the legs. Appropriate buoyancy and drag forces complete the hydrodynamic simulation.

2.1.4 Legged locomotion

A fourth class of robot locomotion is legged locomotion. Like robots modelled after snakes, legged robots are also often biologically inspired. Animals have relied on legged locomotion for hundreds of millions of years, and legged robots capitalize on this obviously successful form of motion. The advantages associated with terrestrial legged vehicles are outlined in [36]: approximately half the earth's surface is inaccessible to wheeled or tracked vehicles, but much of that area is still exploitable by legged animals. Legged locomotion is superior to wheeled and tracked vehicles for crossing obstacles and is mechanically superior over a variety of soil conditions. Furthermore, legged vehicles have an advantage over wheeled and tracked vehicles in that they can take purchase on discrete footholds, which wheels and tracks cannot [10] (see Figure 2.10). Another advantage to legged locomotion is its ability to operate in underwater 6DOF environments.



(a) (b)

Figure 2.10: An advantage of legged locomotion. (a) Wheeled vehicles cannot take purchase on discrete footholds and are therefore defeated by relatively small obstacles. (b) Legged vehicles are able to take purchase on discrete footholds and can therefore overcome obstacles that well exceed their body clearance.

In the water legs can be used as paddles or fins to swim in the open water and legs can also be used in their more "traditional" way to walk along solid surfaces such as the sea floor. In the underwater domain, the ability to move freely in open water is obviously the main advantage to this form of locomotion, but another advantage to a legged vehicle is in station-keeping, especially on the sea-floor. A legged robot can gently rest on the sea-floor without creating the same disturbance caused by a propeller based robot attempting the same task (see Figure 2.11) [15].

One biologically inspired legged robot is RHex [39] (Figure 2.12(a)). RHex is a hexapod robot designed by researchers from the University of Michigan and McGill University. RHex's locomotion has been modelled after the compliant legs of a Blaberus Cockroach [39]. This design allows the robot to travel at speeds up to one body length per second and traverse terrain with high variations well exceeding its body clearance.

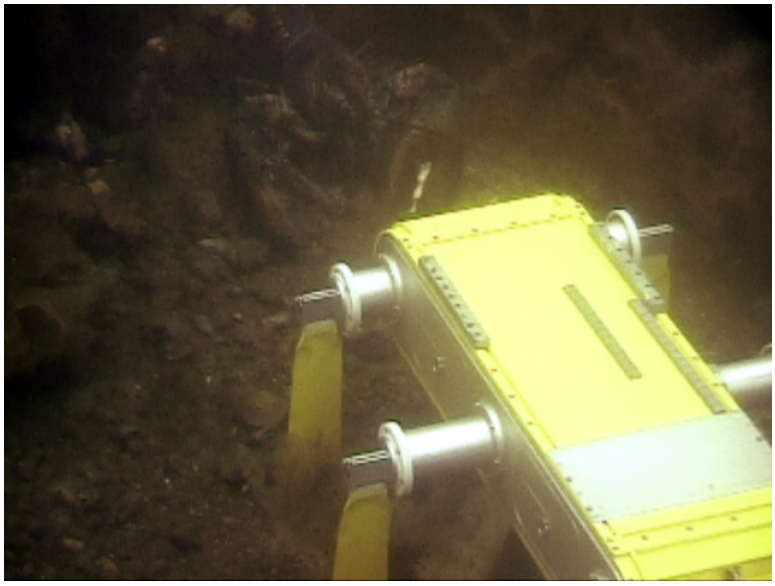
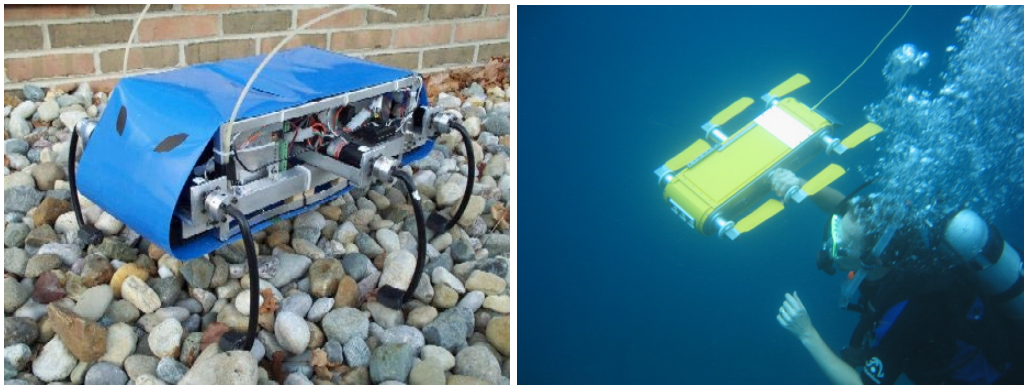


Figure 2.11: The AQUA amphibious hexapod rests on the ocean floor to observe a lobster off the coast of Halifax, Nova Scotia.

Figure 2.12(b) shows the AQUA amphibious hexapod. AQUA, a successor to RHex, was developed by research teams at McGill University, York University, and Dalhousie University. AQUA is capable of walking on land, wading through the surf zone, and swimming in open water to depths of 30m. With two front-facing cameras, one rear-facing camera and an internal inertial measurement unit, AQUA is capable of swimming over a coral reef or ship wreck and reproducing a 3D model of the structure. This is invaluable given how time-intensive, error-prone, and potentially dangerous it is to manually survey underwater structures.

Robotic legged locomotion researchers face difficult challenges in both engineering and software controller development. From an engineering standpoint legged robotics



(a) RHex (b) AQUA

Figure 2.12: Biologically inspired hexapod robots. (a) The RHex robot. Reprinted from [39]. (b) The AQUA amphibious hexapod accompanied by project Engineer Chris Prahacs.

present challenges due to the limited space, weight and power budgets and the required robustness of the device. By evidence of the number of legged vehicles in existence, it is also clear that these problems are tractable. From a software standpoint the difficulties encountered in generating locomotive strategies for legged robots are very similar to those required for snake-like robots, in that the high dimensionality of the locomotive mechanisms make autonomous control very challenging. Legged robots move by changing their leg joint angles as a function of time. The task of generating useful joint angle patterns (gaits) is explored in the next section.

2.2 Gaits

The term gait refers to the pattern of leg placements throughout the locomotion cycle of a legged system. For a given legged system many different gaits are possible. Gaits

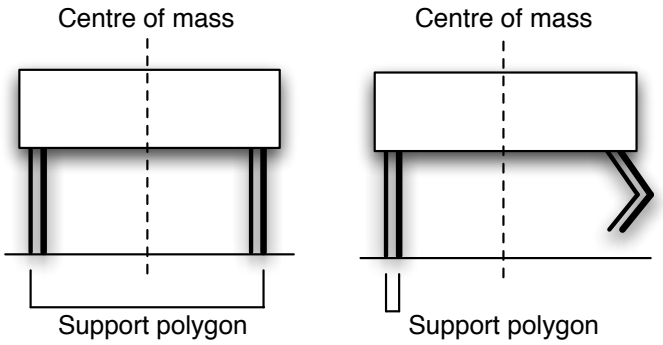
(a) Periodic

(b) Aperiodic

Figure 2.13: Foot placements in periodic and aperiodic gaits. (a) A periodic gait is a repeating pattern of foot placements. (b) An aperiodic gait is one in which the foot placements do not form a repeating pattern, such as when jumping over a gap in rough terrain.

can be classified as being periodic or aperiodic. A periodic gait is one in which a single pattern of leg placements repeats multiple times, such as the pattern used by humans when walking or running over even ground (Figure 2.13(a)). An aperiodic gait is one in which the sequence of leg placements varies throughout the course of the gait, such as walking over broken terrain. In this type of environment a human would choose their next step carefully and the gait varies depending on the perceived optimal foot placement (Figure 2.13(b)).

Terrestrial gaits can also be classified as being statically stable or dynamically stable. A statically stable gait is one in which at any point in the gait cycle, the centre of mass of



(a) Static (b) Dynamic

Figure 2.14: Statically and dynamically stable gaits of a quadruped. (a) When the centre of mass lies inside the base of support, the gait is said to be statically stable. (b) When the centre of mass lies outside the base of support, the gait is said to be dynamically stable. If this gait were to be stopped as above, the robot would fall.

the body lies inside the polygon created by the projection of the legs in contact with the support surface (Figure 2.14(a)). This means that a statically stable gait can be stopped at any time without the subject falling over. A dynamically stable gait is one in which there is some moment in the gait where, if stopped, the vehicle would fall because the centre of mass lies outside the support polygon (Figure 2.14(b)). A hop is an example of a dynamically stable gait. If a hop is "stopped" mid-flight, the vehicle will fall.

A land-based gait can be broken up into support and flight phases. The support phase occurs when a leg is in contact with the ground. The flight phase occurs when the leg is lifted off the ground, usually to move the foot to a new location.

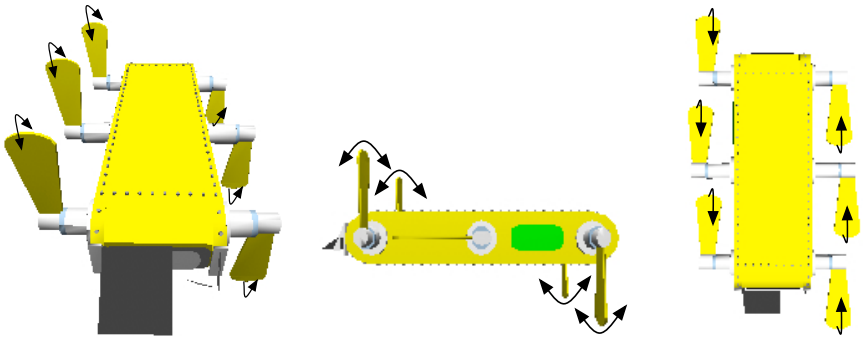
2.2.1 Underwater gaits

Underwater legged vehicle control has many features in common with propeller and thruster vehicle control. Underwater, legs act like the reaction surfaces found on propeller-driven robots. On a propeller-driven vehicle, propellers force fluid over reaction surfaces. The force applied to the reaction surface by the fluid changes the vehicle's orientation. Underwater legged robots use a slightly different technique, they move their legs through the fluid, but use the same resultant reactionary force for propulsion. There are several ways in which underwater legged locomotion differs from its land-based counterpart:

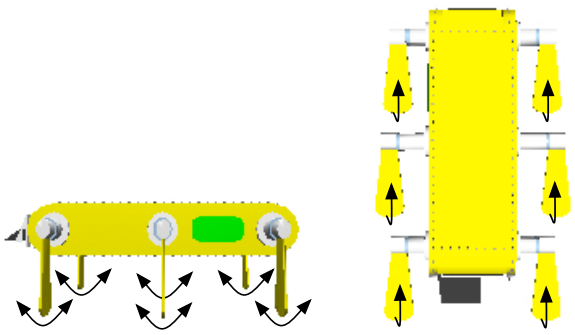
- The underwater environment is 6DOF and the gaits can potentially provide movement and reorientation over the six degrees of freedom. This means that potentially gaits can exist for a wide range of motions including roll, pitch, yaw, heave, sway, and surge (Figure 2.15).

- When the vehicle is fully suspended in open water the concept of dynamically stable gaits is meaningless because there is no surface contact. This removes the risk of falling, however it also removes the ground as a reference plane, requiring underwater robots to rely more heavily on sensors such as inertial measurement units (IMU) to gauge their orientation.

- Since the drag due to water is 800-fold greater than drag due to air [28], there is



(a) Roll (b) Pitch (c) Yaw



(d) Heave (e) Surge

Figure 2.15: Examples of underwater gaits using the AQUA amphibious hexapod. The marked legs oscillate sinusoidally $\pm 22.5^{\circ}$ around the leg positions shown. (a) A roll gait will cause the vehicle to rotate around the longitudinal axis. (b) A pitch gait will cause the vehicle to rotate around the transverse axis. (c) A yaw gait will cause the vehicle to rotate around the medial axis. (d) A heave gait will cause the vehicle to move along the medial axis. (e) A surge gait will cause the vehicle to move along the longitudinal axis. The design of the vehicle prohibits a sway gait.

no flight phase in the gait during which a leg can be moved without affecting the motion of the body. For example, on land a legged system walks by picking up one leg, moving it forward and planting it on the ground. Due to the relatively low drag due to air, the motion of the body is generally unaffected by the motion of this leg during this phase of the gait. Underwater, a leg incurs drag throughout the entire gait cycle, which applies forces on the body. Underwater gait design must take this extra drag into account to ensure that the net force throughout the entire gait cycle results in motion in the desired direction.

Whether a legged vehicle operates on land or in the water, it moves by executing a planned set of leg motions to move the vehicle forward. The process of developing a pattern of leg motions to perform this is known as gait synthesis. This problem is considered in the next section.

2.3 Gait synthesis

Gait synthesis is the process of (automatically) generating leg motions that move a vehicle between two configuration states (see Figure 1.4). A configuration state represents the robot's position, orientation, linear and angular velocity, and leg positions[1]. Gait synthesis is difficult in part due to the large search space associated with the problem. In the case

[1]Note that higher order derivatives with respect to time may also be included in the state - eg. leg velocities - but these higher order effects will not be taken into account in this work.

of a legged hexapod, such as the AQUA vehicle, where each leg has a 360° workspace, brute-force generation of a 10 second gait at 100 leg angles per second with a resolution of 1° becomes a search space of $(360 \times 6)^{1000} = 2160^{1000}$ possible configurations. That means that generating a 10 second gait for the vehicle that meets some specific motion constraint results in a search problem with more than 10^{3000} configurations to search through. Optimizations can be made to this search space, such as eliminating leg angle changes that lie outside the capabilities of the physical device. For example, say each leg has acceleration and velocity constraints such that it can only transit $[-2^\circ, +2^\circ]$ in a given time-step, leaving 5 position choices for each leg in that instant, this still leaves a search space of 10^{1000} configurations. While a complete search through a space this large is possible, at a rate of one leg angle configuration per millisecond it would take over 10^{990} years to complete the search. Clearly a brute force search is impractical and other techniques are required.

Given the size of the search space the application of heuristics has become a popular approach to gait synthesis. By using different techniques to shrink the search space and applying an appropriate search technique, researchers have been able to synthesize robot gaits for a variety of vehicles. The following sections review existing heuristic techniques for gait generation.

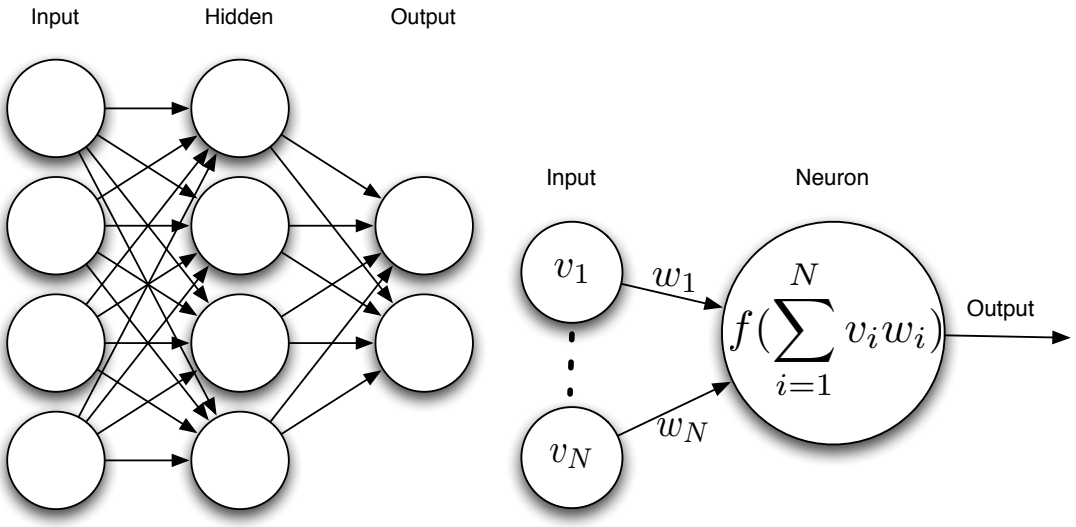


Figure 2.16: Artificial Neural Networks. (a) depicts a high level view of the structure of a neural network. (b) illustrates the mechanics of a single neuron embedded in a neural network, a weighted sum of the inputs passed through a function f, typically a sigmoid function.

2.3.1 Neural nets

In an effort to model robot gait synthesis after biological gait synthesis, many researchers (e.g., [5, 22, 24, 27, 43, 49]) have turned to neural networks (see [1]) to model the complex, non-linear relationship between the robot's sensory input and desired trajectory to the joint angle outputs required to propel the vehicle. In its most basic form, a neural network reads in a set of inputs and, after filtering the data through a series of hidden layers, produces a set of outputs (see Figure 2.16(a)). The basic mechanism embedded in each neuron is depicted in Figure 2.16(b). Each neuron performs a weighted sum of the inputs it receives and applies a non-linear function to the result (typically a sigmoid

function) before passing this response to the next layer. This type of neural network is called a feed-forward neural network. Initially the weights within the neural network are chosen at random and a training set of data is then applied to the network. During training the weights are adjusted until the network produces the correct results based on the training set. This type of supervised neural network requires an appropriate weight set and a strategy for training the neural network. Various training strategies exist although back-propagation [37] is a popular choice.

Training a neural network involves feeding a set of exemplars through the neural net and adjusting the weights via a learning algorithm. The type of training set used depends on how the inputs and outputs are encoded (see below). Examples of training sets used in neural network gait synthesis research include the output of linear control methods the neural network must attempt to out-perform [24], reference actuator signals designed to output known leg-motion responses [5], and gaits generated using another optimization technique (e.g., genetic algorithms) [43].

A challenging aspect to the application of neural networks to gait synthesis is determining how to encode the input and output data. Clearly requiring an output node for each leg-joint at each timestep is inefficient. Researchers have developed gait synthesis systems based on neural networks that have taken a reference signal (e.g., a sine wave) as input and output commands in the leg-actuator space [5]. Others use the current dynamics configuration of the robot (e.g., hip angle and hip joint angular velocity) as input

to the neural network which outputs the required hip actuator torque values necessary to keep the robot balanced [24].

There are several problems associated with applying neural networks to gait synthesis. Encoding gait parameters such that they can be embedded in a neural network implementation is a non-trivial exercise. Furthermore, choosing an appropriate training algorithm with which to adjust the weights of the network as well as how many hidden neuron layers are necessary is also difficult.

2.3.2 Genetic algorithms

Another biologically inspired technique that has been used for gait synthesis is genetic algorithms (GA) [16]. Genetic algorithms are a search strategy structured on the mechanics of natural selection and natural genetics. The algorithms combine survival of the fittest with randomized information exchange to seek out an optimal solution. Genetic algorithms model the solution to a search problem as a set of artificial creatures. In each iteration (generation) a new set of creatures is created by combining pieces of the fittest creatures from the previous generation along with some randomization process to avoid local minima. As future generations are created the creatures evolve towards an optimal solution set.

One requirement to implementing a genetic algorithm is that the solution guesses (creatures) must be encoded in such a way that the combination of segments from mul-

tiple valid creatures results in a valid creature. Another requirement is the existence of a fitness function that provides a metric for the error associated with a creature in terms of suitability as an optimal search solution.

Each generation of a genetic algorithm has three steps. The first step, reproduction, involves creating a new generation as carbon copies of the previous generation based on the previous generation's fitness values. Creatures seen as 'better' will contribute more offspring to the next generation. The second step is the crossover. Here the newly copied creatures are paired (mated) at random and – for each pair – an integer k is chosen at random between 1 and the length of the string l. Then for each pair, two new creatures are created by swapping the characters between $k+1$ and l inclusive. For example if a creature pair is made up of A_1 and A_2

$$A_1 = 0\ 1 \mid 0\ 1$$

$$A_2 = 1\ 1 \mid 0\ 0$$

and k was chosen to be 2 (indicated by the symbol '|'), the new creatures would be

$$A_1^{'} = 0\ 1\ 0\ 0$$

$$A_2^{'} = 1\ 1\ 0\ 1.$$

The third step is mutation. With a very small probability the creature strings undergo random alterations to a value at a string position. This random alteration helps avoid the creatures becoming stuck in local minima on their way to an optimal solution.

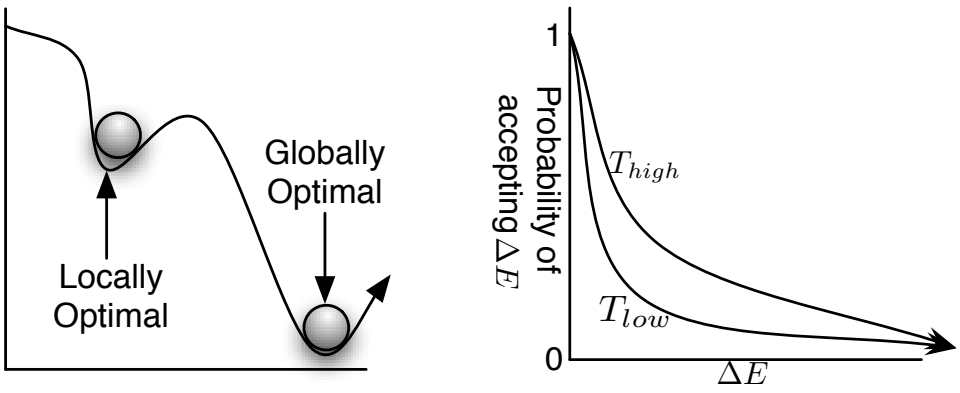
Genetic algorithms have been applied to the task of gait synthesis. In [17] genetic algorithms were used to synthesize leg motions for a hexapod walking robot, using the characteristics of the legs' motions as the genotype. Each member of the population was tested for fitness by looking for specific circumstances (moving the leg from front-to-back under reasonable load, moving the leg from back-to-front under zero load). In [18] GAs were used to synthesize gaits for simulated four and six-legged robots. The genotype used was a vector of floating-point joint angles and the fitness of each member of the population is measured by both the distance travelled by the vehicle and several onboard sensors (gyros and bumpers) to maintain stability.

Another way genetic algorithms are used in gait synthesis is training gait generating neural networks. In [43] GAs were used to synthesize near-optimal walking gaits for a biped robot. These gaits are used as a training set for a gait synthesizing neural network. In [26] GAs are used to tune a neural network which synthesized gaits for a six-legged robot.

There are several problems with the application of genetic algorithms to gait synthesis. Encoding the gait parameters into the string must be done with care. Also, tuning the mutation parameters as well as the size of each generation must be chosen carefully.

2.3.3 Simulated annealing

Search strategies that operate by gradient descent have difficulty avoiding local minima because the algorithms can only take "downhill" steps. In order to find the globally optimal solution an algorithm should be able to escape local minima (see Figure 2.17(a)). Simulated annealing is one such search strategy. Simulated annealing is the application of the Metropolis algorithm from statistical mechanics to the field of combinatorial optimization [23]. The Metropolis algorithm was originally used in condensed matter physics to provide an efficient simulation of a collection of atoms in equilibrium at a given temperature. In each step of the algorithm an atom is given a small random displacement and the resulting change in the energy of the system, ΔE, is computed. If $\Delta E \leq 0$ then the displacement is accepted and the new configuration with the displaced atom is used as the starting point for the next iteration. If $\Delta E > 0$ then the displacement is treated probabilistically with the probability that this new configuration is accepted given by $P(\Delta E) = e^{(-\Delta E/T)}$. If a random number chosen from within the range $(0, 1)$ is less than $P(\Delta E)$ then the new configuration is retained, otherwise the original configuration is used in the next iteration. The temperature T is initially set to a high value but as time goes on the search is 'cooled' and the value of T is reduced thus reducing the number of 'uphill steps' taken by the algorithm. The algorithm simulates the thermal motion of atoms in thermal contact with a heat bath at temperature T (see Figure 2.17).



(a) (b)

Figure 2.17: Properties of the simulated annealing search strategy. (a) Simulated annealing allows for uphill steps which can prevent the system from becoming trapped in a local minima. (b) The probability that an uphill step is taken depends on the size of the step, ΔE, as well as the current temperature, T. As the temperature is lowered, the probability that large values of ΔE will be accepted diminishes.

In the physical sense, annealing involves heating a material to its melting point and lowering the temperature slowly, spending a great deal of time near the material's freezing point. If the material is allowed to cool sufficiently slowly the material will form a (near) perfect crystalline structure. If the material is allowed to cool too quickly it will fall out of equilibrium resulting in a crystalline structure with many defects. In a simulated annealing approach to optimization this same approach is used to find optimality in large data sets. Simulated annealing applies the Metropolis algorithm to a complex search problem and lowers the temperature slowly to find an optimal solution.

The advantage to simulated annealing over an iterative improvement search strategy,

where only displacements that lower the energy of the system are accepted, is the possibility of avoiding locally optimal solutions when a more globally optimal solution exists. As an example Figure 2.17(a) shows an energy function for which the minima is being sought. An iterative improvement strategy will only make downhill steps, so the search strategy can become trapped in a locally optimal minima and miss the more globally optimal solution altogether. On the other hand, simulated annealing uses the probability function $P(\Delta E)$ to allow for stochastic uphill steps. The probability that an uphill step will be taken depends on the size of ΔE as well as the shape of the exponential curve which changes as the temperature, T, is lowered (Figure 2.17(b)).

To apply simulated annealing to a search problem, four ingredients are needed: a concise description of a configuration of the system; a random generator of "moves" or rearrangements of the elements in a configuration; a quantitative objective function containing the trade-offs that have to be made; and an annealing schedule of the temperatures and length of times for which the system is to be evolved.

Simulated annealing has been applied to the application of gait patterns for a quadruped robot [46]. In this work simulated annealing was used to control the phase difference between the oscillators in the double-jointed legs of a quadruped robot. The gait synthesis engine was, under certain conditions, able to obtain trot and canter patterned gaits for the vehicle.

Although there is little literature regarding the use of simulated annealing for gait

synthesis, it has found application in a wide variety of other optimization tasks: Gao and Tian used a modified simulated annealing algorithm for mobile robot path planning [13]. Pai and Sreeram used simulated annealing in the identification of military targets from images captures from unmanned reconnaissance planes [32]. Applications of simulated annealing to computer circuit design as well as the travelling salesman problem, along with further details of simulated annealing can be found in [23].

2.3.4 Gait transition

Gait transition is the process of switching between two gaits usually for the purpose of altering speed or direction. The process involves generating an aperiodic leg motion that blends two distinct periodic leg motions and it is a complex task for any legged robot. On land the problem is made simpler by the fact that the behaviour of a leg can be changed during the flight phase of the gait with minimal or no effect on the rest of the body, due to the relatively low drag coefficient of air compared to the friction due to ground contact. An example of gait transitions in the land-based legged robot domain is presented in [19]. In [19] Hodgins describes techniques for walk-to-run and run-to-walk transitions for a two-legged robot (see Figure 2.18). The techniques involve analyzing the control algorithms for the two gaits to find points where the oscillations are similar. The controller then adds or removes energy to transform each oscillation into the corresponding oscillation for the new gait. For example, the controller might extend the

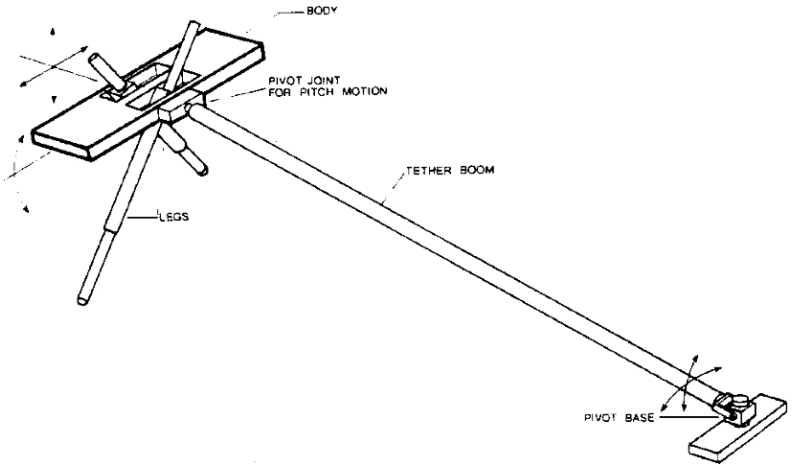


Figure 2.18: A sketch of the biped running robot used for gait transition experiments in [19].

leg during the stance phase of the walk gait to add the energy required to initiate the flight phase of the running gait.

In the underwater domain the problem complexity of gait transition is compounded by the fact that a body suspended in a fluid cannot freely articulate its limbs without incurring unwanted drag forces. This means that an underwater robot does not have the same freedom to modify the behaviour of its legs while performing a gait transition. There is very little material available in the literature that deals with how underwater systems, either biological or robotic achieve gait transitions. From the robotic side this stems primarily from the fact that there are very few legged underwater robots. On the biological systems side, some work has been conducted that addresses the question of why fish switch between sets of fins during swimming (e.g., [8]), however that research does not address the nature of how this transition occurs.

Given a sufficiently large set of sample gaits, one possible mechanism for transitioning between gaits is to only match up gaits when the necessary kinematic and dynamic constraints are met at the point of transition. For example, suppose that the set of gaits all have leg angles of zero at the beginning and end of every gait. Then gaits can be transitted as long as the dynamic constraints that must be true at the start of one gait are met by the completion of another. The problem then becomes one of searching through the set of gaits in order to match the proper gaits together. Research in this type of task is presented in the next section.

2.4 AI search techniques

Path planning for mobile robots is a broad field of research. The basic path planning problem is determining a path in the vehicle's configurations space between an initial and final configuration [10]. The classic approach to mobile robot path planning is to discretize the workspace into nodes between which the robot is capable of transitioning. These nodes represent the robot's configuration spaces. The nodes are embedded into a connected graph representing all possible transitions between nodes. The objective then is to find the "shortest" or "best" path between the start and goal configurations in the graph (see Figure 2.19).

A number of algorithms for determining the least-cost path through connected graphs, such as the configurations-space graphs found in mobile robot path planning,

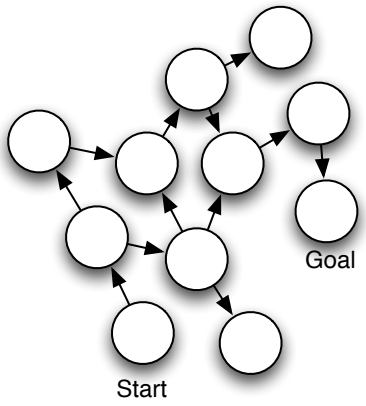
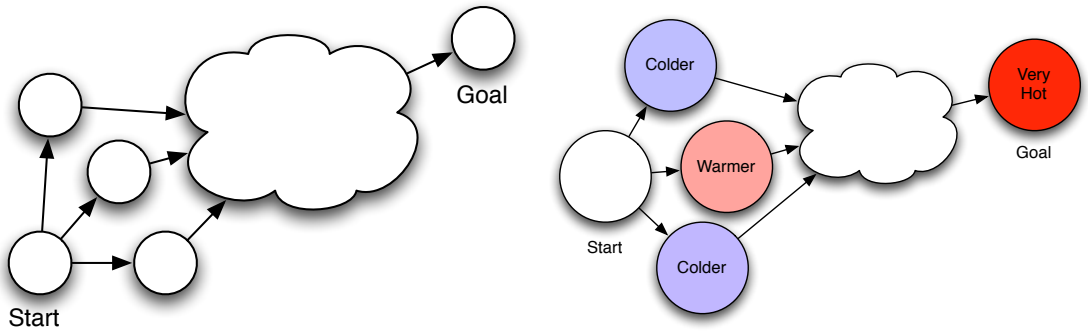


Figure 2.19: Graph search techniques are designed to find the "least-cost" path between two nodes in a graph.

have been developed. Graph search strategies can be defined as being uninformed or informed. An uninformed search is one in which from any given node no heuristic exists to provide an educated guess on which adjacent node has the highest probability of leading toward the goal node (see Figure 2.20(a)). In this case the choice of the next node to search is basically random. An informed search moves through a search space in which a heuristic exists to guide node choices (see Figure 2.20(b)).

An example of an informed graph search algorithm is A* (read A-STAR) [31]. A* is a heuristic directed optimal path graph search algorithm which finds the least-cost path between two nodes. The algorithm requires several pieces of information to be included in each node. Each node must have a *parent marker* which will point backwards to the node through which the lowest-cost path back to the start node can be reached. Once the algorithm has finished it is these parent markers which direct the least cost path backward



(a) Uninformed search (b) Informed search

Figure 2.20: Graph search strategies are designed to get from a start node to a goal node. Optimized search strategies try to answer the question "which node should be searched next in order to find the goal more quickly?" These search strategies lie between uninformed and informed searches. (a) Uninformed searches provide no information to help optimize the graph search. There is no way to intelligently choose which next node to search. (b) Informed searches are used when information exists in order to intelligently choose the next node to search which has the highest priority of leading to the goal node.

from the goal node to the start node, thereby defining the path. Nodes must also have three cost values defined: G the cost of the path from the start node to the current node, H a lower-bound "heuristic estimate" of the distance from the current node to the goal node, and F the distance-plus-cost value defined as $F = G + H$. Provided that G is an upper bound of the cost of the best path from the start to the current node and H is a lower bound of the cost of the best path from the current node to the goal, then A* is guaranteed to find the optimal path from the start to the goal, assuming that such a path exists [31]. A* is provided as Algorithm 2.1 at the end of this chapter.

There are limits to the configuration space graph search approach. Chief among these

is that the search space can be very large and it is possible that a search could traverse the entire graph. As the size of the graph grows, so too do the computational resources required to perform the search.

2.5 Summary

Mobile robots employ a range of mechanics to achieve locomotion. Wheeled and tracked vehicles are simple but limited in terms of their exploitable terrain. Snake-like mobile robots can exploit a wide range of terrain but are limited in payload and operational simplicity. Propellor based vehicles can operate underwater or in the air but sacrifice the safety involved in handling an operational device. Legged vehicles, many biologically inspired, are mechanically complicated but can exploit a wide range of terrain including the underwater domain where legged systems excel at station-keeping without disturbing the environment.

Legged vehicles move by executing gaits, patterns of joint angle motions as a function of time. These gaits can be cyclic (periodic gaits) or acyclic (aperiodic gaits) and for terrestrial vehicles gaits may be statically or dynamically stable. Gait synthesis is the process of generating gaits in an automatic manner. Gait synthesis involves a search through a large configuration space where heuristics become essential compared to brute-force gait generation. In the past neural nets, simulated annealing and genetic algorithms have been used to generate gaits for terrestrial legged vehicles, but generating gaits for

underwater legged vehicles is a relatively unexplored research area.

Although a range of heuristic techniques exist for gait synthesis including neural nets and genetic algorithms, simulated annealing – an optimization technique which has proved very successfully in other fields – has received little attention with respect to terrestrial and aquatic vehicle gait generation. Given an appropriate set of gaits a range of established techniques exist to link these gaits together provided that the end state of the gaits meet the dynamic preconditions for subsequent gaits. The key difficulty (and to date an open problem) is the automatic synthesis of an alphabet of gaits for a legged vehicle. Given the potential for simulated annealing to address the task of gait synthesis it seems likely that a popular mobile robot path planning solution could be applied to the alphabet of gaits to generate complex motions in obstacle strewn environments.

The next chapter describes an underwater gait generation system involving the generation of an alphabet of small gaits using simulated annealing. Later a description of the path planner used to intelligently combine the alphabet of gaits into complex maneuvers is provided.

Algorithm: A*

add the start node to the open list;
repeat
    search the open list for the node with the lowest F cost;
    call this node the current node;
    move the current node to the closed list;
    foreach *node adjacent to the* current node **do**
        if *it is blocked by an obstacle or on the* closed list **then**
            ignore it;
        end
        if *it is not on the* open list **then**
            add it to the open list;
            record the current node as this node's parent;
            update this node's G, H and F costs;
        end
        if *it is on the* open list *and the F cost of reaching this node via the* current node *is less than this node's previously recorded F cost* **then**
            record the current node as this node's parent;
            update this node's G, H and F costs;
        end
    end
until *the* goal node *has been added to the* closed list *or the* open list *becomes empty* ;
If a path has been found, follow the parent markers from the goal node back to the start node to extract the optimal path

Algorithm 2.1: The A* algorithm for finding the least-cost path between two nodes in a connected graph. After [31].

Chapter 3 Motion synthesis

Calculating the patterns of leg joint angles required for a legged vehicle to achieve a certain pose from an initial state (determining its inverse dynamics) is a complex problem. In this thesis we divide the problem into two sub-problems: (i) generating an alphabet of simple gaits, and (ii) linking the gaits together while meeting the dynamic and configuration constraints imposed by the linked gaits to form complex maneuvers. This chapter details a solution to the first sub-problem, that of generating a language of simple gaits that can be later linked together (detailed in Chapter 5) to form more complex vehicle motions. This chapter provides a high-level overview of the gait alphabet generation system, details of the individual components, and how they fit together to generate an alphabet of gaits for underwater legged vehicles. Finally, sample results of the process are presented and a summary of the chapter is provided.

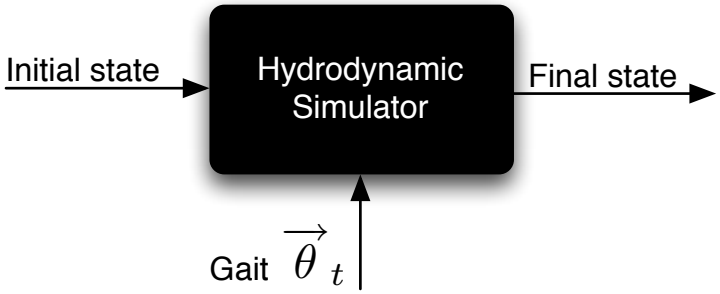


Figure 3.1: The hydrodynamic simulator is treated as a black box. Given an initial state and a gait – a series of leg angles as a function of time – (Figure 3.2), it outputs the final position reached by the vehicle after executing the gait for some period $t \in [0, T]$.

3.1 Gait synthesis

At its highest level the gait synthesis system answers the question posed in Figure 1.4. More formally, for a generic legged vehicle we seek the sequence of leg angles as a function of time which will have a robot achieve a destination state given some initial state of the vehicle. The gait synthesis system uses a simulated annealing engine to make small iterative changes to an initial guess gait until the synthesized gait is capable of transiting a legged vehicle between an initial and final position (see Algorithm 3.1). The linchpin of the system presented here is the modelling of the vehicle's hydrodynamics via a black-box simulator (Figure 3.1).

```
Algorithm: Gait Synthesis System
best gait = initial guess gait;
while temperature > ε do
    for N times for this value of temperature do
        new gait = tweak(best gait);
        ΔError = error( simulate( new gait ) ) - error( simulate( best gait ) );
        if ΔError ≤ 0.0 then
            best gait = new gait;
        else
            if rand() < e^(−ΔError/temperature) then
                best gait = new gait;
            end
        end
    end
    temperature = temperature × Δtemperature;
end
```

Algorithm 3.1: The gait synthesis algorithm is outlined here and described in this chapter. The system makes small changes to an initial guess gait until the gait is capable of transiting a vehicle between an initial and final configuration state. The temperature cools by an amount proportional to the current temperature.

The simulator takes in an initial vehicle state and a gait to be executed, and outputs the final state reached after the execution of a potential gait for a predefined period of time. The initial and final vehicle states include the position $\overrightarrow{p}$, linear velocity $\overrightarrow{v}$, orientation $\overrightarrow{\Omega}$, angular velocity $\overrightarrow{\omega}$, and leg joint angles of the vehicle $\overrightarrow{\theta}_t$, along with any higher derivatives that may be appropriate. The gait is a series of joint angles, one for each leg of the vehicle given as a function of time (see Figure 3.2).

The gait synthesis system can be conceptualized as a simulated annealing loop which uses the hydrodynamic simulator to compute the energy of the system represented as the difference (error) between the achieved final state of the vehicle and the goal final

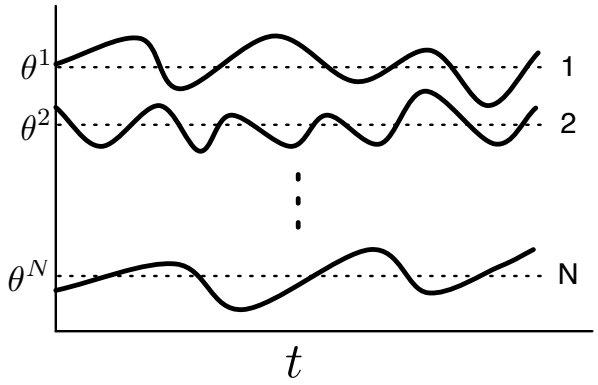


Figure 3.2: The gait, $\overrightarrow{\theta}_t = (\theta_t^1, \theta_t^2, \cdots, \theta_t^N)$: A series of leg joint angles, one for each N legs. These joint angles represent a gait for an underwater legged vehicle. As legs change orientation with respect to time, the forces generated by their motions propel the vehicle.

state (see Figure 3.3). Utilizing a simulated annealing approach the system makes small iterative changes to a seed gait until the simulator returns a final vehicle state that is (approximately) the same as the desired final vehicle state or fails. The result is a gait capable of moving the simulated robot between the initial and final states. The following sections detail each component of the gait synthesis system outlined in Figure 3.3.

3.1.1 Seed gait

The gait synthesis system requires a seed gait as an initial guess to begin the search process. The seed gait is represented as leg joint angles as a function of time (Figure 3.2). The choice of the seed gait is meant to assist the system in reaching its goal. For example, if the gait to be synthesized is a surge gait, the initial guess might be a sine-based gait such as the gait shown in Figure 1.3. Alternatively, the system might be supplied with a seed

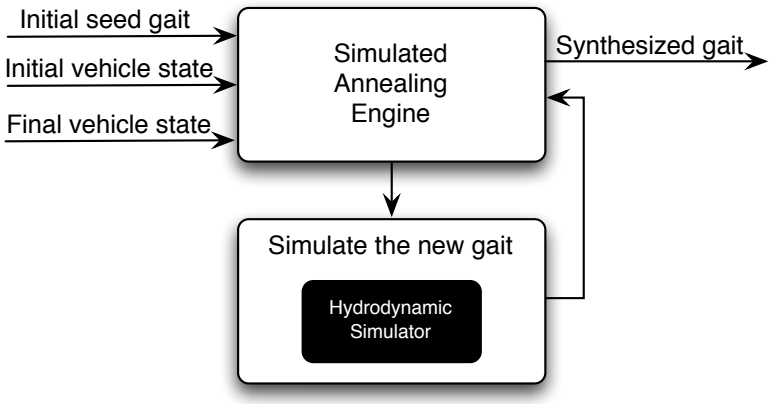


Figure 3.3: A high-level representation of the simulated annealing loop used by the gait synthesizer.

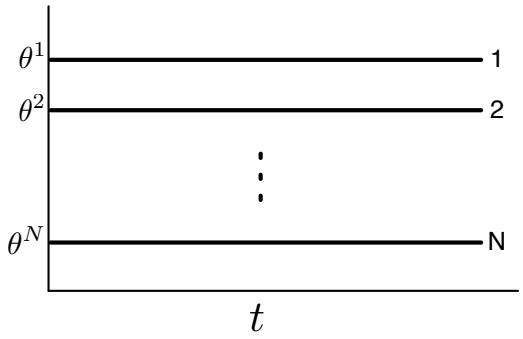
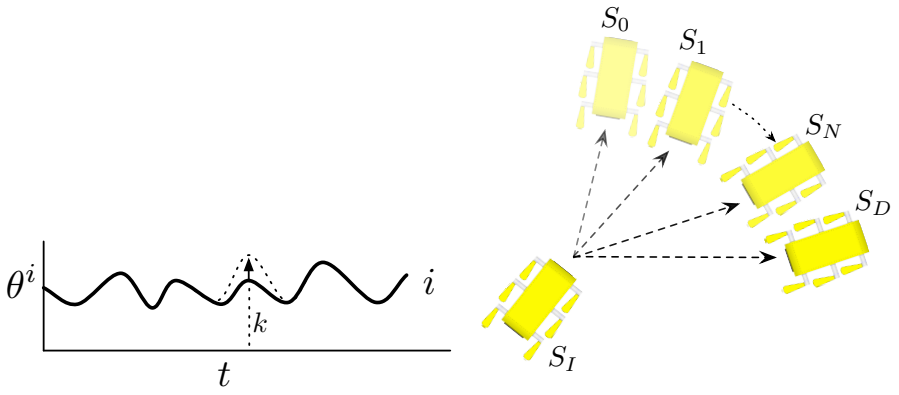


Figure 3.4: A gait where all legs remain at rest can be used as the seed gait of the system.

gait where all the legs remain static (Figure 3.4). The final constraint on the seed is that the first and last set of leg angles must match specified start and goal leg configurations. Typically this configuration is defined as all the leg joints lying at 0° on the longitudinal axis of the vehicle although other start and end configurations are possible.



(a) The iteration (b) The goal

Figure 3.5: Gait modification and the goal of the simulated annealing engine. (a) Gait modification involves iteratively tweaking the gait by choosing a random leg, i, at a random time throughout the gait, k, and displacing the angle slightly. (b) The goal of this process is to search for the leg angle tweaks that cause the robot to move between a given initial state, S_I to a given destination state, S_D.

3.1.2 Simulated annealing engine

The simulated annealing engine makes small iterative changes to the gait until the execution of the gait causes the simulated robot to transit between its designated initial and final states with a minimum error (Figure 3.5). At each step the engine chooses a random leg i, at a random time-step t throughout the gait k, and changes the leg angle of leg i at time t by a random amount $\Delta\theta$ in the range $\pm\Delta\theta_{max}$. For the AQUA vehicle $\Delta\theta_{max} = 2^\circ$ (Figure 3.5(a)):

$$\theta_k^i \leftarrow \theta_k^i + \Delta\theta$$

The aim of this process is to make small changes to the gait until the gait is capable of transiting the simulated robot between the designated initial and final states (Figure 3.5(b)). It is important to note that the black-box nature of the hydrodynamic simulator has implications to this process. The gait modifying engine acts without knowledge of the dynamics of the vehicle: It sends the simulator a gait and the simulator returns the state reached by the virtual vehicle at the end of the gait execution. If the final state returned from the simulator is the closest yet seen to the desired end state, the new gait is saved as the 'best-yet', if not the gait is only accepted if it is accepted based on the simulated annealing process. This tweak-simulate-analyze loop occurs a number of times per temperature value to give the simulated annealing engine ample time at a given temperature to find a minimal configuration state. Following [23] the temperature is then reduced by an amount proportional to the current temperature and the tweak-simulated-analyze loop is then repeated. Once the simulated annealing process has finished the 'best-yet' gait is returned to the user.

Simulated annealing requires several tuning parameters: the starting temperature T, the rate at which the system cools ΔT (expressed as cool rate as a function of search iteration), and the number of attempts made at each temperature. Most of the gaits generated for the simulated AQUA vehicle in this thesis were performed with $T = 0.01$, $\Delta T = .90$ and 600 iterations per temperature (well below the number of attempts per temperature recommended in [23] but necessary for completion within a reasonable amount of time,

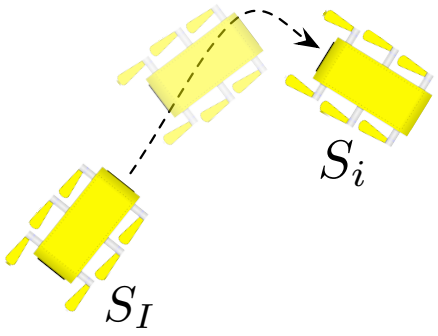


Figure 3.6: A virtualized robot executes the gait.

but which was sufficient to affect a solution). Several gaits in the alphabet required alternative T values; specific parameter values are described in Chapter 4 along with the gaits for which they were used.

The blindness of the search process to the vehicle's dynamics makes the search process very generic in terms of the vehicle configurations it can support. It also allows for more (or less) complex hydrodynamic models to be used without change to the basic system structure. Furthermore, the search process can be easily tasked with the generation of more constrained and 'broken leg gaits' in which additional constraints such as one or more legs are not functioning are introduced.

The hydrodynamic simulator is used to test each gait as it filters through the gait synthesis system (Figure 3.6). For each iteration, the simulator starts the virtual vehicle in the initial state S_I. The simulated robot then executes the current gait and the simulator returns the robot's state as a function of time (executing the gait) which is used to evaluate

the effectiveness of the gait being evaluated.

3.1.3 Evaluation of the final state

The gait synthesis system terminates the search if a suitable gait has been found or if a predefined number of iterations has been exceeded. Successful termination requires the development of a criterion that evaluates if the current gait has developed a final system state that is considered "close enough" to the desired final state (Figure 3.5(b)).

In the current implementation the measurement of the error between the achieved end state and the desired end state is computed as the Euclidean distance between the state vectors of the achieved and goal states. The error includes components representing the position error E_p, the linear velocity error E_v, the orientation error E_Ω, and the angular velocity error E_ω. Each component is computed as follows for the achieved end state a and the goal end state I:

Position error:

$$E_p = ||\overrightarrow{p}_a - \overrightarrow{p}_I||.$$

Linear velocity error:

$$E_v = ||\overrightarrow{v}_a - \overrightarrow{v}_I||.$$

Orientation error is the magnitude of the quaternion required to rotate the achieved ori-

entation quaternion q_a into the ideal orientation quaternion q_i:

$$q_e = q_a^{-1} \times q_i$$

$$E_\Omega = 2\ cos^{-1}(\frac{q_e[w]}{|q_e|})$$

where $q_e[w]$ represents the scalar component of the quaternion. Angular velocity error:

$$E_\omega = ||\overrightarrow{\omega}_a - \overrightarrow{\omega}_I||.$$

Once the individual error components have been computed, they are linearly combined:

$$E = \alpha E_p + \beta E_v + \gamma E_\Omega + \phi E_\omega$$

The weighting values $(\alpha, \beta, \gamma, \phi)$ are used to normalize the error values. Different values of the weighting parameters can be used to reflect the relative importance (and scale) of the errors. In the work performed here $(\alpha, \beta, \gamma, \phi) = (1, 1, 1, 1)$.

3.1.4 Sample synthesis

The following is an example of the gait synthesis system presented in this chapter. This section demonstrates the generation of a yaw-left gait for the virtual AQUA amphibious hexapod (see Figure 3.7). The system is tasked with generating a gait to transit the virtual vehicle between the two configurations listed in Table 3.1. As described above the gait synthesis system requires an initial guess gait. For this example the system is provided with a 15 second gait where initially all the legs begin in the rest-position $(\theta_i = 0)$

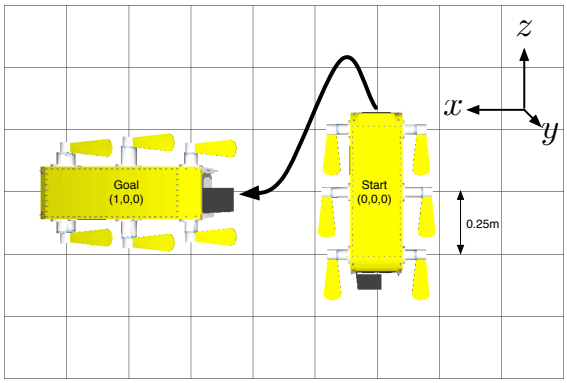


Figure 3.7: The sample gait will be a yaw left gait. The robot should turn left $90°$ and translate from $(0, 0, 0)$ to $(1, 0, 0)$.

	position	**roll**	**pitch**	**yaw**	**linear velocity**	**angular velocity**
start	$(0, 0, 0)$	$0°$	$0°$	$0°$	$(0, 0, 0.1)$	$(0, 0, 0)$
goal	$(1, 0, 0)$	$0°$	$0°$	$90°$	$(0.1, 0, 0)$	$(0, 0, 0)$

Table 3.1: The virtual vehicle's start and goal configurations for the sample yaw-left gait. Note that the goal state is relative to the initial state.

and the left-side legs rotate $180°$ (in-unison) while the right-side legs remain at the rest-position (see Figure 3.8). The final elements required by the gait synthesis system are the parameters for the simulated annealing engine. The initial temperature T is set to 0.01 and the cooling rate ΔT is set to 0.9, so at every iteration T cools by 10%.

Once all of the initial requirements have been met, the gait synthesis system is tasked to generate the requested yaw-left gait. Set to the above values for T and ΔT a 15 second gait generation takes roughly 94 minutes to run on a 3 GHz Intel Xeon Mac Pro (though this could be reduced by stopping the synthesis after several iterations of T have

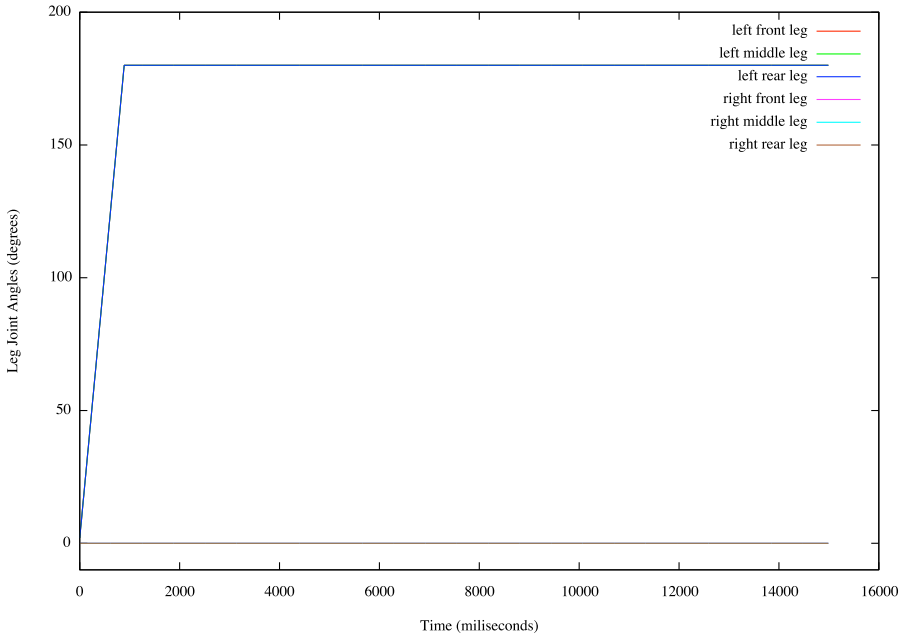
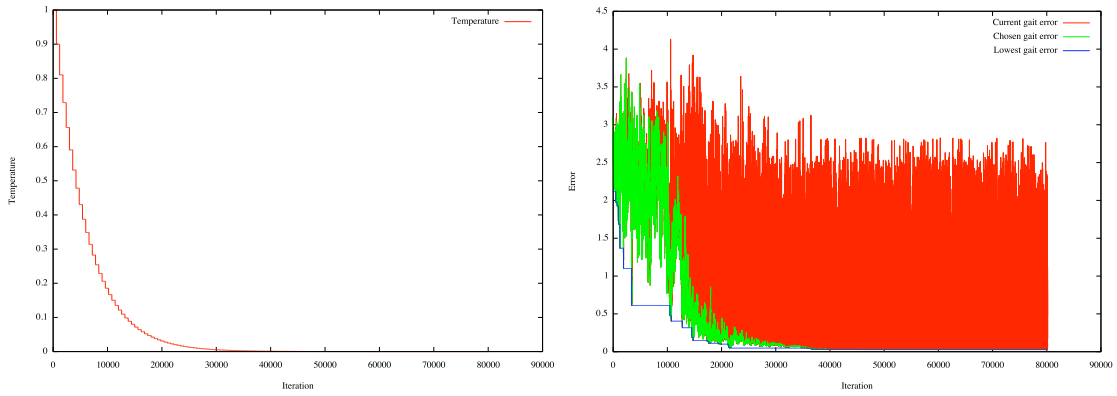


Figure 3.8: The initial guess gait provided to the gait synthesis system. At the beginning of the 15 second gait the left-side legs rotate 180° while the right-side legs remain still. This gait is used to seed the gait synthesis system and will be refined until it is capable of transiting the vehicle between the start and goal configurations.

passed without a drop in the configuration error). As the gait synthesis system runs, the error associated with the vehicle's final configuration compared to the goal configuration is graphed against the temperature T used by the simulated annealing engine (see Figure 3.9). What should be clear from the graph is that the error associated with the gait's final configuration tends towards zero as T cools to zero. In the above run, the gait synthesis system turned the initial guess gait (Figure 3.8) into the gait shown in Figure 3.10. When this gait is applied to the simulated AQUA vehicle the result is indeed a left turn with a 1m forward transit (Figures 3.11 and 3.12). The final state reached by the vehicle in this sample gait is presented in Table 3.2.



(a) Temperature (b) Configuration Error

Figure 3.9: As the temperature T cools (a), the system accepts fewer and fewer uphill steps for the chosen gait and the error tends toward zero (b). For every iteration of T the gait is tweaked and tested 600 times. When the algorithm completes the search, the gait with the lowest error is returned.

This example gait shows that given appropriate initial parameters, that is a good initial guess gait and appropriate values for T and ΔT, the gait synthesis system can generate a gait which comes very close to matching the desired final configuration. Note that the values (T and ΔT) control the simulated annealing engine and must be chosen carefully. If the temperature T is too high then too many 'uphill' steps will be taken and engine may not be able to recover. If T is too low then only 'downhill' steps will be taken and the system may become trapped in a local-minimum. If the cooling rate ΔT is too quick the engine may not have sufficient time to make the 'uphill' steps necessary to avoid the local-minima. A slow cooling rate for ΔT is preferable, but it comes at the cost of execution speed as more iterations are required. The benefits associated with finding appropriate initial parameter values come from minimizing the configuration error at the

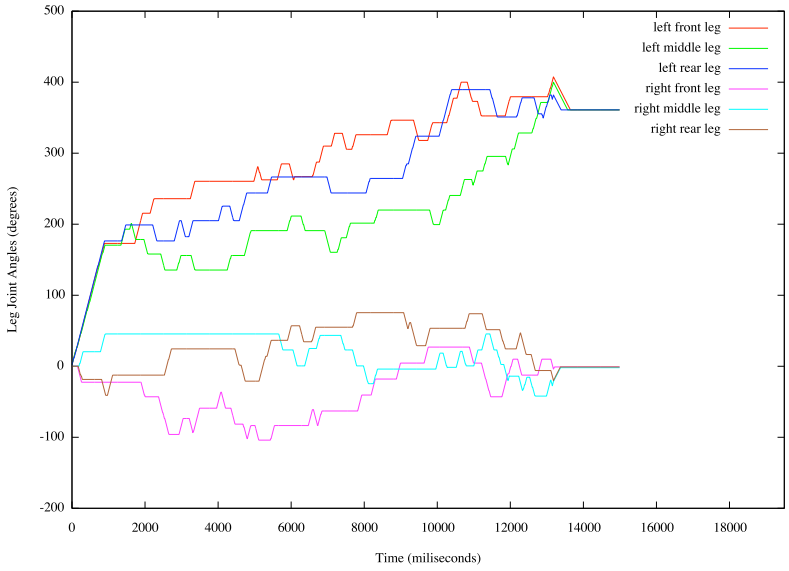
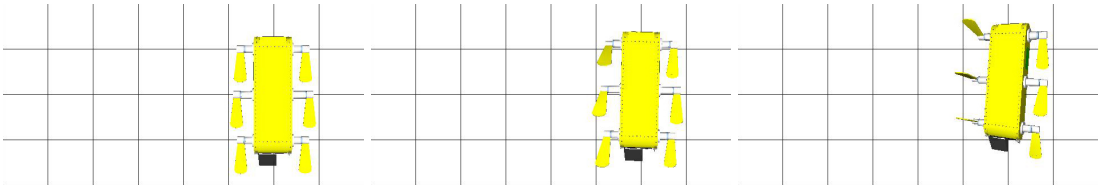


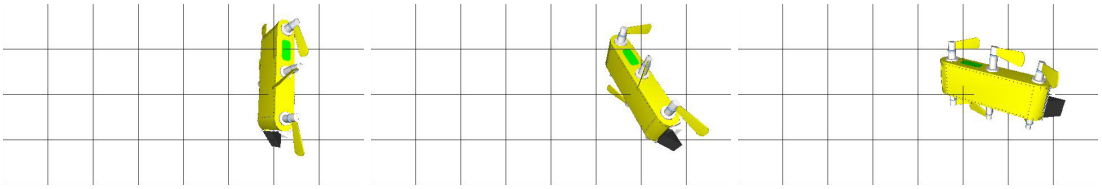
Figure 3.10: Leg-joint angles from the yaw-left gait generated by the gait synthesis system.

end of the gait which in-turn reduces the cumulative configuration error as the gaits are linked together to form complex maneuvers (see Chapter 5).

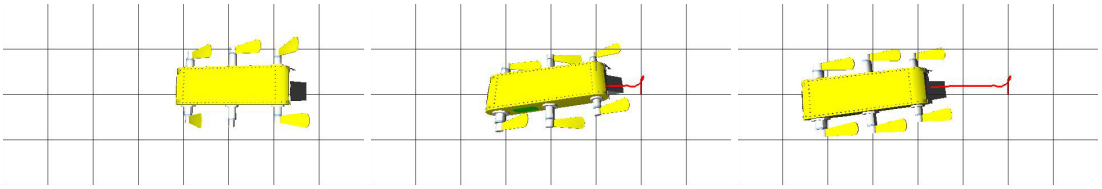
To further decrease the error between the achieved final configuration and the goal final configuration the gait synthesis system can be run a second (or third) time using the synthesized gait as the initial guess gait. The above synthesized yaw-left gait was used as the initial guess gait (T and ΔT remained the same) and once the new gait had been generated the final configuration error was reduced by 5% (see Figure 3.13 and Table 3.3).



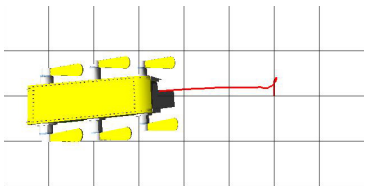
(a) $t = 0$s (b) $t = 0.2$s (c) $t = 0.5$s



(d) $t = 1.7$s (e) $t = 4.7$s (f) $t = 7.7$s

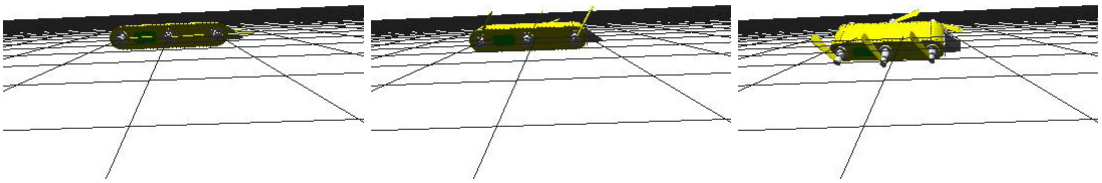


(g) $t = 10.7$s (h) $t = 12.5$s (i) $t = 13.6$s

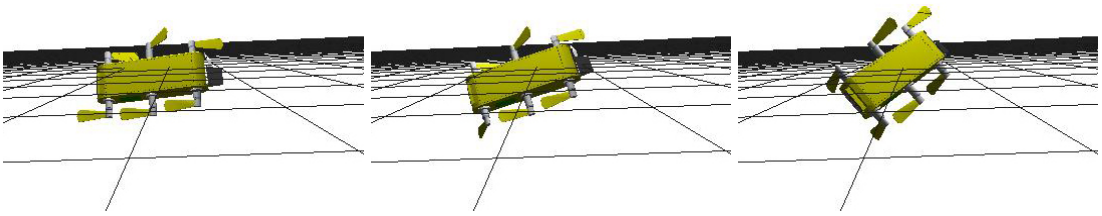


(j) $t = 15$s

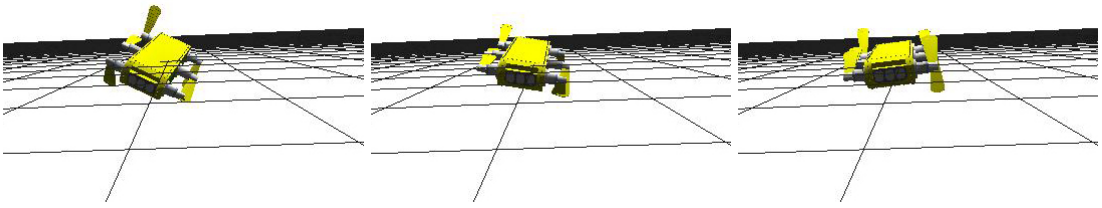
Figure 3.11: A top-down view of the simulated AQUA robot executing the simulated yaw-left gait. The frames are labelled with the time (in seconds) at which they were captured throughout the gait. The path of the robot is drawn in red.



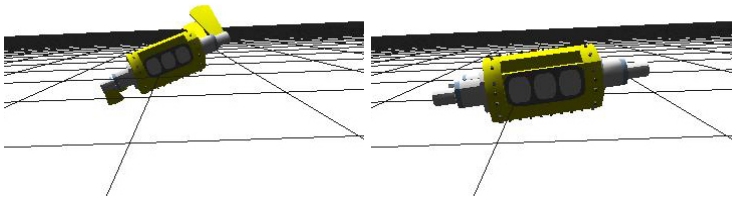
(a) $t = 0$s (b) $t = 0.3$s (c) $t = 0.6$s



(d) $t = 1.2$s (e) $t = 2.3$s (f) $t = 5.4$s



(g) $t = 7.8$s (h) $t = 9.5$s (i) $t = 10.7$s



(j) $t = 13.2$s (k) $t = 15$s

Figure 3.12: A side-on view of the simulated AQUA robot executing the simulated yaw-left gait. The frames are labelled with the time (in seconds) at which they were captured throughout the gait.

	position	**orientation**
goal	$(1, 0, 0)$	$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix}$
achieved	$(0.99, 0.0, -0.02)$	$\begin{pmatrix} -0.10 & -0.06 & 0.99 \\ 0.09 & 0.99 & 0.07 \\ -0.99 & 0.09 & -0.10 \end{pmatrix}$
	linear velocity	**angular velocity**
goal	$(0.1, 0, 0)$	$(0, 0, 0)$
achieved	$(0.1, 0.0, -0.01)$	$(0.0, -0.01, -0.01)$
	final configuration error	
goal	0.0	
achieved	0.024044	

Table 3.2: A comparison of the goal configuration to the configuration achieved by the virtual legged vehicle after executing the synthesized gait. The gait was generated through one run of the synthesis system.

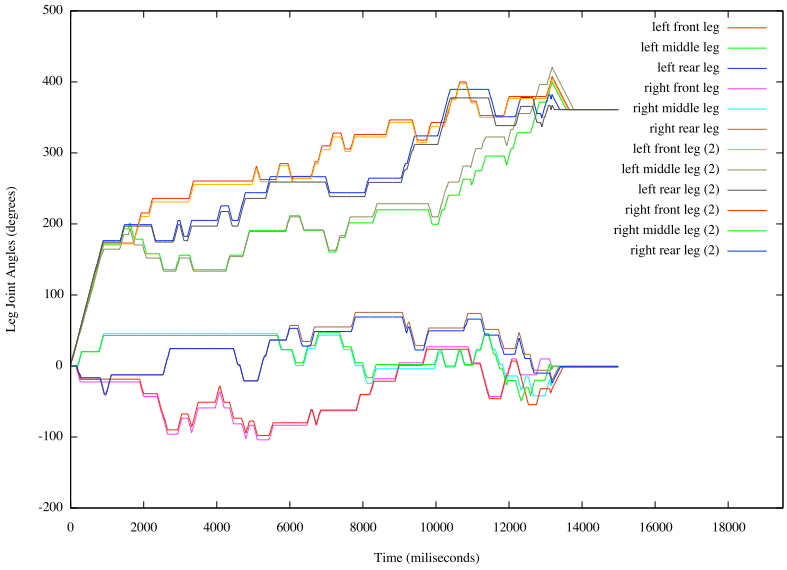


Figure 3.13: Two separate leg-joint angle sets. The first is from the yaw-left generated above. This gait was used as the initial guess and the system was run again. The second set of leg-joing angles (labelled '2') is the result of the second synthesis run. The second synthesis resulted in a gait with a final configuration-state error 5% less than the first synthesized yaw-left gait.

	position	**orientation**
goal	$(1, 0, 0)$	$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix}$
achieved	$(1.00, 0.0, 0.01)$	$\begin{pmatrix} -0.08 & -0.09 & 0.99 \\ 0.02 & 1.00 & 0.08 \\ -1.00 & -0.02 & -0.08 \end{pmatrix}$
	linear velocity	**angular velocity**
goal	$(0.1, 0, 0)$	$(0, 0, 0)$
achieved	$(0.11, 0.0, -0.01)$	$(0.0, 0.0, 0.0)$
	final configuration error	
goal	0.0	
achieved	0.023025	

Table 3.3: A comparison of the goal configuration to the configuration achieved by the virtual legged vehicle after executing the second synthesized gait (seeded by the first synthesized gait). The gait was generated with two passes through the synthesis system.

3.2 Summary

This chapter outlined the use of simulated annealing to generate gaits for a generic legged underwater vehicle. The process begins with a start and goal configuration for the vehicle, an initial guess for the gait and starting temperature and cooling rate for the simulated annealing engine. The gait generation system then makes small iterative changes to the initial guess gait and will accept the changes based out the probabilities dictated by the simulated annealing engine. As the temperature of the system cools the engine converges towards a gait that will transit the vehicle between the start and goal configurations.

Generation of a gait requires the specification of the initial and final states of the vehicle and the duration of the gait. The initial and final configuration-states include parameters for position, orientation, linear velocity, angular velocity, and the leg-joint angles. The state information is sufficient to fully specify the internal state of the vehicle (here the joint angles) as well as the the instantaneous dynamic state of the vehicle. Here the dynamic state is given in terms of the position and velocity values in both position and orientation. Clearly higher temporal derivatives in both the vehicle state (e.g., instantaneous leg velocities) and dynamic state (e.g., acceleration) could be included in the state specification.

The gait synthesis system presented above can be used applied iteratively. That is, the output gait of the engine can be re-used as the initial guess gait and the simulated

annealing process can be run again (often improving on the previously generated gait). However this process provides diminishing returns.

The simulated annealing engine requires an initial guess gait to be provided. In some cases (such as the surge-forward gait) the initial guess gait can be a still-gait where the legs remain in the rest position. In other cases (such as the yaw gaits and a surge-backwards gait) the system benefits from initial guess gaits in which the legs are repositioned at the beginning of the gait. The better the choice for an initial guess gait, the fewer uphill steps the simulated annealing engine must take and the better the likelihood of synthesizing a near-optimal gait. For example, a surge-backwards gait requires the legs to be rotated from their rest position 180° before oscillating to move the vehicle backwards. However any leg motion (positive or negative) from the rest position will generate forward thrust which will result in a *larger* final configuration-state error. Thus, uphill steps (accepting a configuration with a larger error) are required before the legs achieve an orientation where they can apply backwards thrust. Because simulated annealing is designed to take relatively few uphill steps (compared to downhill steps) it will be difficult for the engine to make all the uphill steps required to rotate all the legs 180°. Instead we can provide the engine with an initial guess gait where all the legs are rotated 180° at the beginning of the gait.

Throughout the gait synthesis process gaits are tweaked with small changes and then applied to a virtual vehicle. The final configuration-state reached by the vehicle is com-

pared to an ideal goal configuration-state to determine if the new tweaked gait is superior to previous gaits, if so the new gait is tweaked again and the process repeats. The comparison of final configuration-states involves computing the configuration-state error of the gaits compared to the goal state. This error is computed as a weighted sum of the position, orientation, linear velocity, and angular velocity differences between the achieved final configuration and the goal configuration. The weights in the error summation allow the values to be scaled in terms of importance (e.g., perhaps position is more important than angular velocity). This error function could be augmented with further parameters to achieve a more precise configuration-state. For example, higher temporal derivatives (linear and angular acceleration) could be added, as could the angular velocity of the legs.

A sample gait, a yaw-left gait was shown as an example of the gait synthesis system in operation. The next chapter shows the application of the technique developed here to synthesize a large alphabet of gaits for the AQUA vehicle.

Chapter 4 An alphabet of gaits

Chapter 3 outlined the gait synthesis system and an example gait was provided. This chapter describes the synthesis of a number of simple gaits for the AQUA vehicle. These synthesized gaits form an alphabet of gaits that will be used by the gait linking process described in the following chapter.

4.1 The gaits

The following gaits were chosen to form the alphabet of gaits for the simulated vehicle. Each gait moves the robot 1m in some direction and many change the vehicle's orientation by 90°. The motions associated with each gait were selected based on several criteria: First, that they could be performed by the real AQUA vehicle (the sway gait is absent due to this criterion). Second, that they move the vehicle a uniform distance in a 6DOF environment. Third, each gait should begin and end with the legs at the rest position (aligned with the body). These criteria simplify the path planning process outlined in the next chapter.

Details of the gaits generated are provided below. For each sample gait tables are provided listing the start and goal configuration-states, and the simulated annealing parameters used. All orientations are in degrees, while all distance measurements are in meters. Graphs of the initial seed gait used, and the synthesized gait are also shown. In terms of the resulting gait a table of the goal and achieved configuration-states, and still frames from the gait execution by the simulated AQUA vehicle are shown. Also shown is the configuration error associated with each synthesized gait. This value is generated using the the equation from Section 3.1.3 and is used to evaluate how close the synthesized gait brings the vehicle to the goal configuration-state.

4.1.1 Yaw Left

The yaw left gait moves the robot 1m to the left with a 90° left turn. The seed gait provided to the gait synthesis system has the left-side legs rotating 180°.

	position	**roll**	**pitch**	**yaw**	**linear velocity**	**angular velocity**
start	$(0, 0, 0)$	0°	0°	0°	$(0, 0, 0.1)$	$(0, 0, 0)$
goal	$(1, 0, 0)$	0°	0°	90°	$(0.1, 0, 0)$	$(0, 0, 0)$

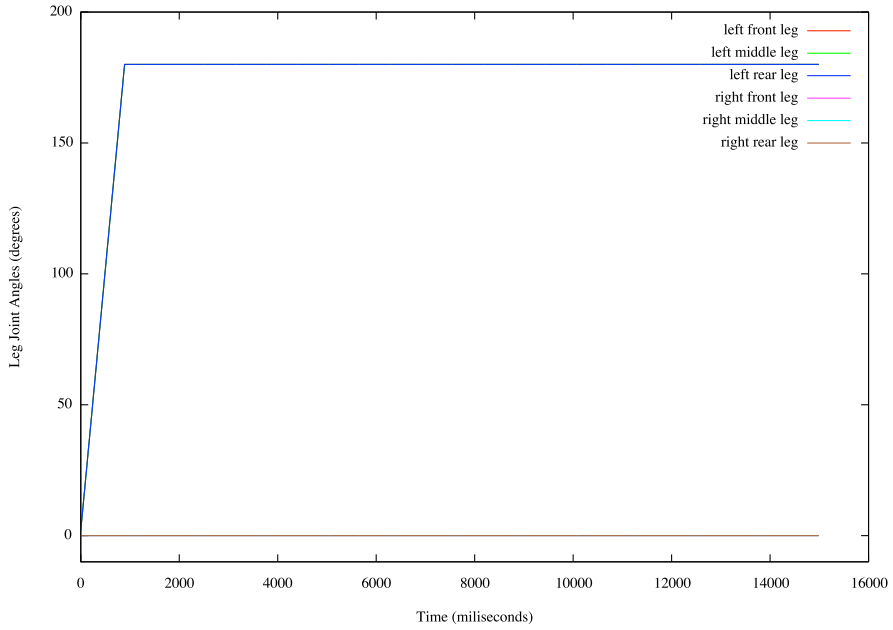
Table 4.1: The start and goal configuration-states for the yaw-left gait.

Starting temp. (T)	**Cooling rate (ΔT)**	**Attempts per temp.**
0.01	0.9	600

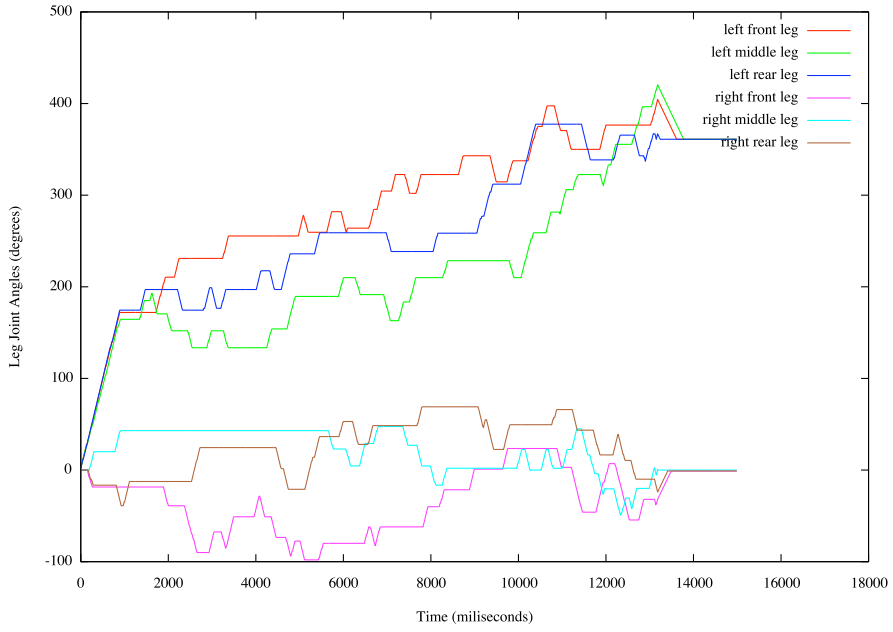
Table 4.2: The simulated annealing parameters used to generate the yaw-left gait.

	position	**orientation**
goal	$(1, 0, 0)$	$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix}$
achieved	$(1.00, 0.0, 0.01)$	$\begin{pmatrix} -0.08 & -0.09 & 0.99 \\ 0.02 & 1.00 & 0.08 \\ -1.00 & -0.02 & -0.08 \end{pmatrix}$
	linear velocity	**angular velocity**
goal	$(0.1, 0, 0)$	$(0, 0, 0)$
achieved	$(0.11, 0.0, -0.01)$	$(0.0, 0.0, 0.0)$
	final configuration error	
goal	0.0	
achieved	0.023025	

Table 4.3: The goal and achieved configuration-states, and final error value for the yaw-left gait.

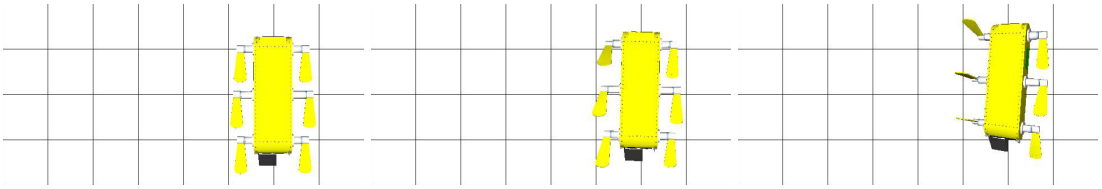


(a)

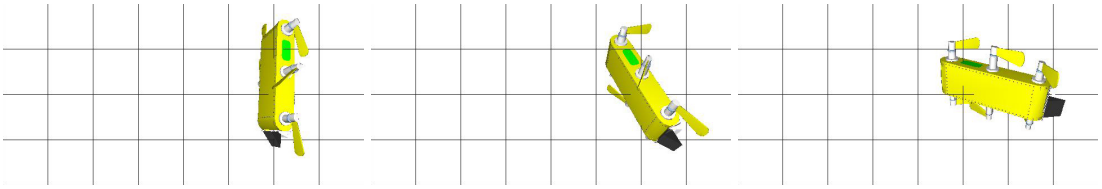


(b)

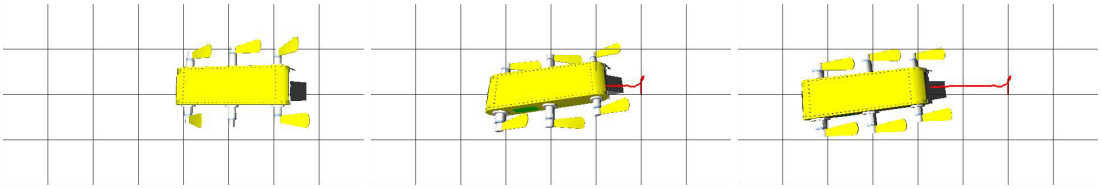
Figure 4.1: The initial seed gait (a) and the synthesized yaw-left gait (b).



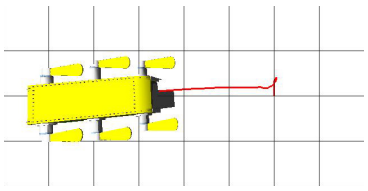
(a) $t = 0$s (b) $t = 0.2$s (c) $t = 0.5$s



(d) $t = 1.7$s (e) $t = 4.7$s (f) $t = 7.7$s



(g) $t = 10.7$s (h) $t = 12.5$s (i) $t = 13.6$s



(j) $t = 15$s

Figure 4.2: A top-down view of the simulated AQUA robot executing the synthesized yaw-left gait. The frames are labelled with the time (in seconds) they were captured throughout the gait. The path of the robot is drawn in red.

4.1.2 Yaw Right

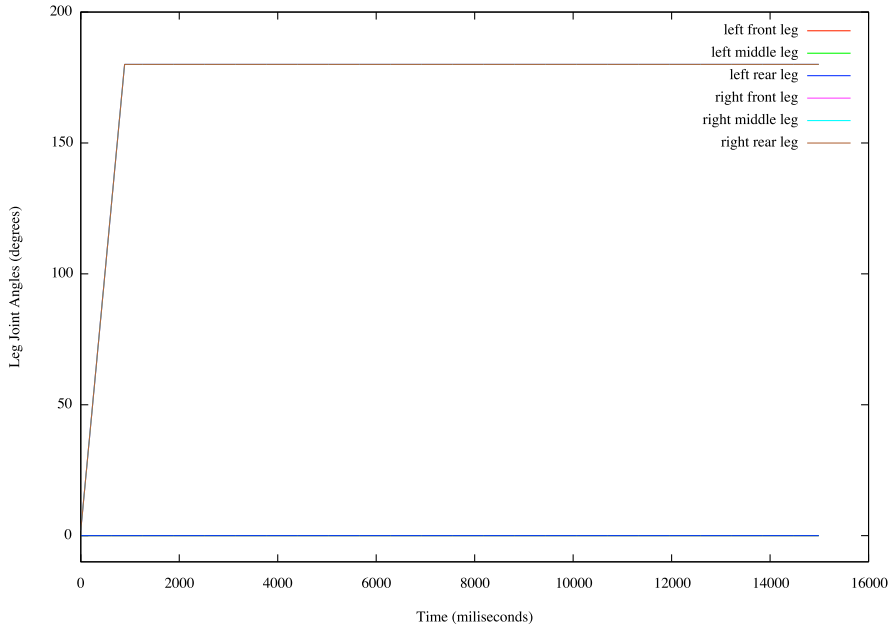
The yaw right gait moves the robot 1m to the right with a 90° right turn. The seed gait provided to the gait synthesis system has the right-side legs rotating 180°.

	position	**roll**	**pitch**	**yaw**	**linear velocity**	**angular velocity**
start	$(0, 0, 0)$	0°	0°	0°	$(0, 0, 0.1)$	$(0, 0, 0)$
goal	$(1, 0, 0)$	0°	0°	-90°	$(-0.1, 0, 0)$	$(0, 0, 0)$

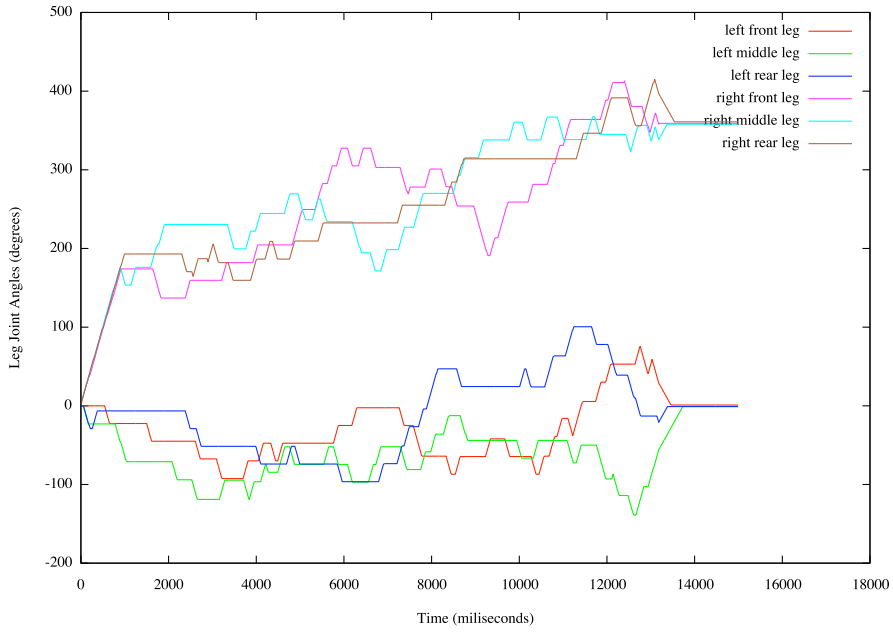
Table 4.4: The start and goal configuration-states for the yaw-right gait.

Starting temp. (T)	**Cooling rate (ΔT)**	**Attempts per temp.**
0.01	0.9	600

Table 4.5: The simulated annealing parameters used to generate the yaw-right gait.

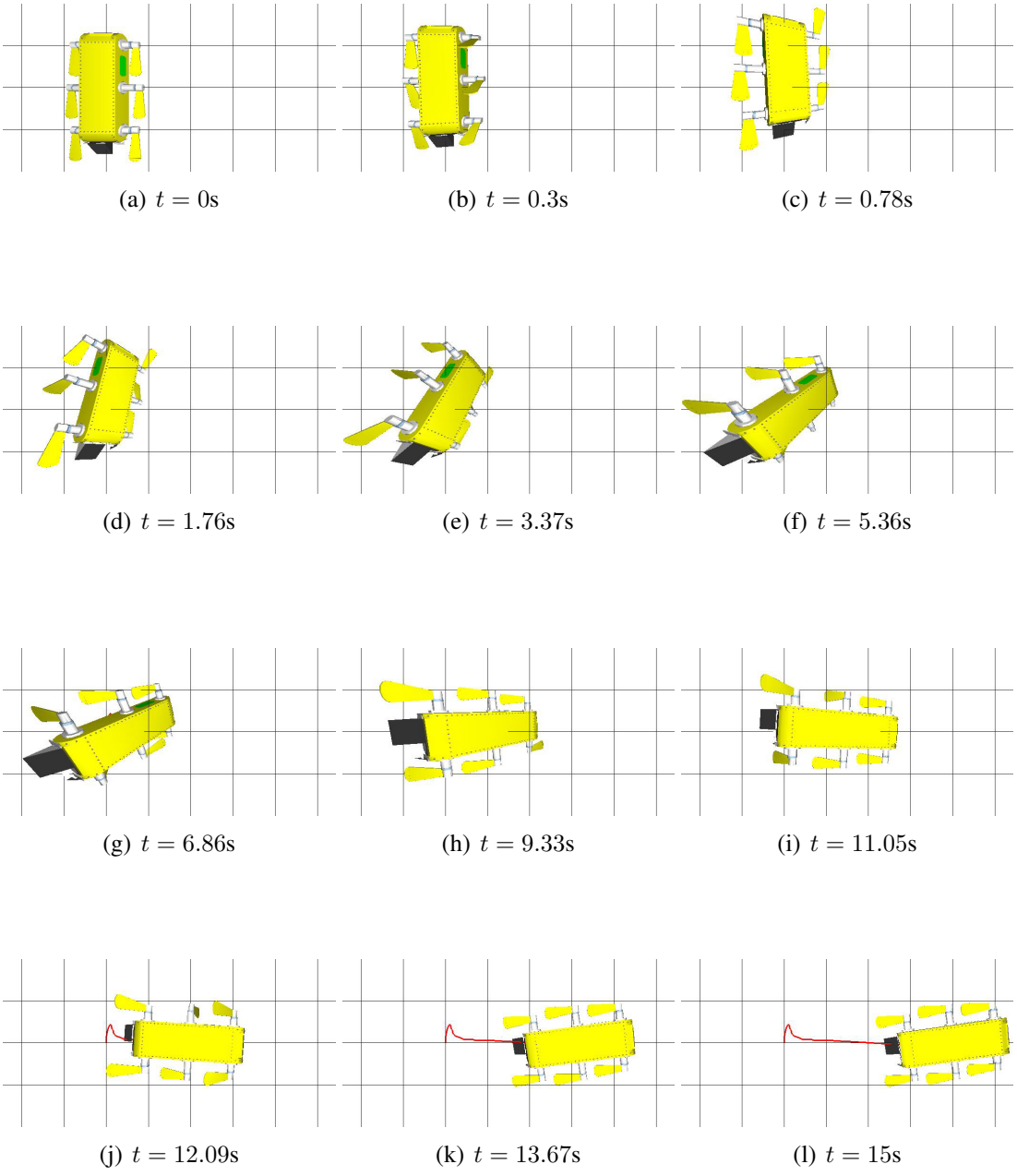


(a)



(b)

Figure 4.3: The initial seed gait (a) and the synthesized yaw-right gait (b).



(a) $t = 0$s (b) $t = 0.3$s (c) $t = 0.78$s

(d) $t = 1.76$s (e) $t = 3.37$s (f) $t = 5.36$s

(g) $t = 6.86$s (h) $t = 9.33$s (i) $t = 11.05$s

(j) $t = 12.09$s (k) $t = 13.67$s (l) $t = 15$s

Figure 4.4: A top-down view of the simulated AQUA robot executing the synthesized yaw-right gait. The frames are labelled with the time (in seconds) they were captured throughout the gait. The path of the robot is drawn in red.

	position	**orientation**
goal	$(-1, 0, 0)$	$\begin{pmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$
achieved	$(-.99, 0.0, 0.0)$	$\begin{pmatrix} 0.14 & 0.00 & -0.99 \\ 0.09 & 0.99 & 0.02 \\ 0.99 & -0.09 & 0.14 \end{pmatrix}$
	linear velocity	**angular velocity**
goal	$(-0.1, 0, 0)$	$(0, 0, 0)$
achieved	$(-0.1, 0.0, 0.0)$	$(0.0, 0.0, 0.0)$
	final configuration error	
goal	0.0	
achieved	0.007818	

Table 4.6: The goal and achieved configuration-states, and final error value for the yaw-right gait.

4.1.3 Surge

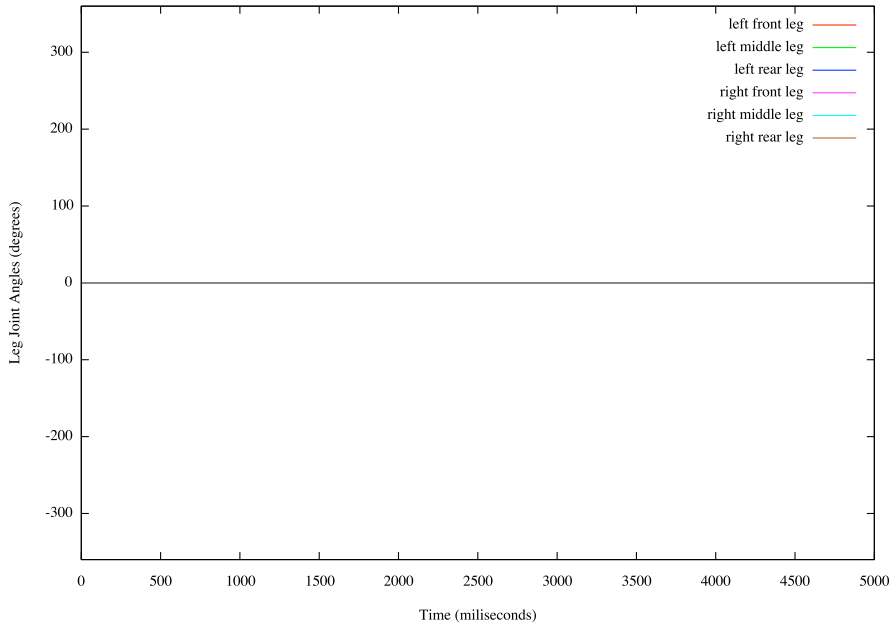
The surge gait moves the robot 1m forward with no change in orientation. The seed gait provided to the gait synthesis system has all the legs in the rest position.

	position	**roll**	**pitch**	**yaw**	**linear velocity**	**angular velocity**
start	$(0, 0, 0)$	$0°$	$0°$	$0°$	$(0, 0, 0.1)$	$(0, 0, 0)$
goal	$(0, 0, 1)$	$0°$	$0°$	$0°$	$(0, 0, 0.1)$	$(0, 0, 0)$

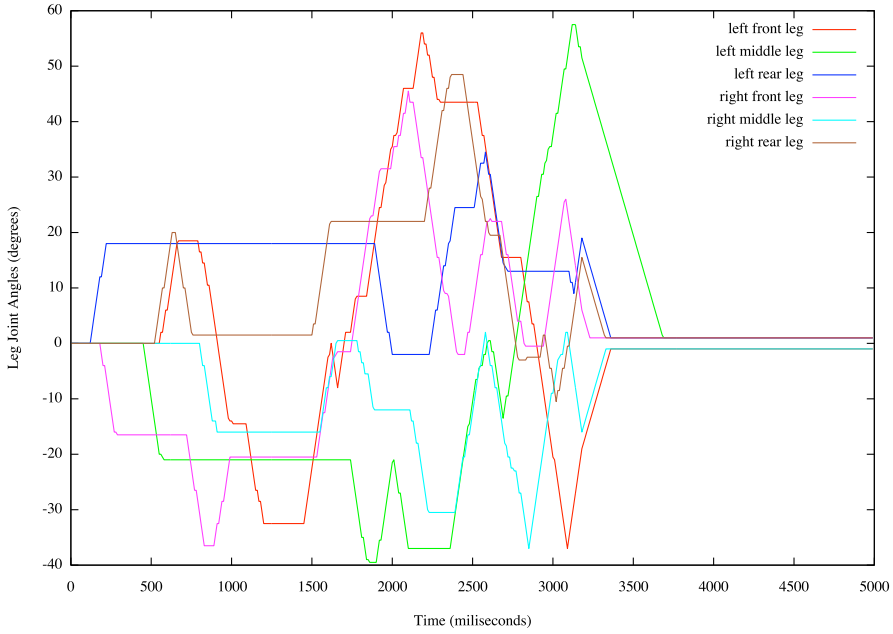
Table 4.7: The start and goal configuration-states for the surge gait.

Starting temp. (T)	**Cooling rate (ΔT)**	**Attempts per temp.**
0.01	0.9	600

Table 4.8: The simulated annealing parameters used to generate the surge gait.

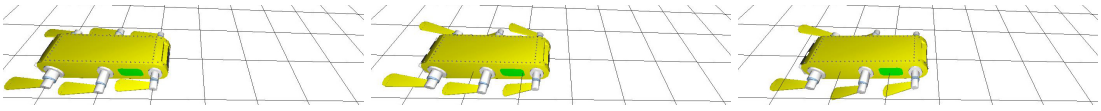


(a)

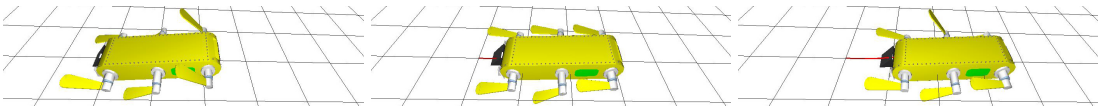


(b)

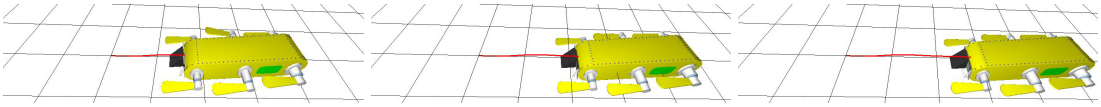
Figure 4.5: The initial seed gait (a) and the synthesized surge gait (b). The seed gait has all legs at the rest position ($0°$).



(a) $t = 0$s (b) $t = 0.71$s (c) $t = 1.15$s



(d) $t = 2.06$s (e) $t = 2.81$s (f) $t = 3.19$s



(g) $t = 3.57$s (h) $t = 4.1$s (i) $t = 5$s

Figure 4.6: A side-on view of the simulated AQUA robot executing the synthesized surge gait. The frames are labelled with the time (in seconds) they were captured throughout the gait. The path of the robot is drawn in red.

	position	**orientation**
goal	$(0, 0, 1)$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
achieved	$(0.0, 0.0, 0.99)$	$\begin{pmatrix} 0.99 & 0.04 & 0.02 \\ -0.05 & 0.98 & 0.09 \\ -0.02 & -0.09 & 0.99 \end{pmatrix}$
	linear velocity	**angular velocity**
goal	$(0, 0, 0.1)$	$(0, 0, 0)$
achieved	$(0.0, 0.0, 0.11)$	$(0.0, 0.0, 0.0)$
	final configuration error	
goal	0.0	
achieved	0.013673	

Table 4.9: The goal and achieved configuration-states, and final error value for the surge gait.

4.1.4 Pitch Up

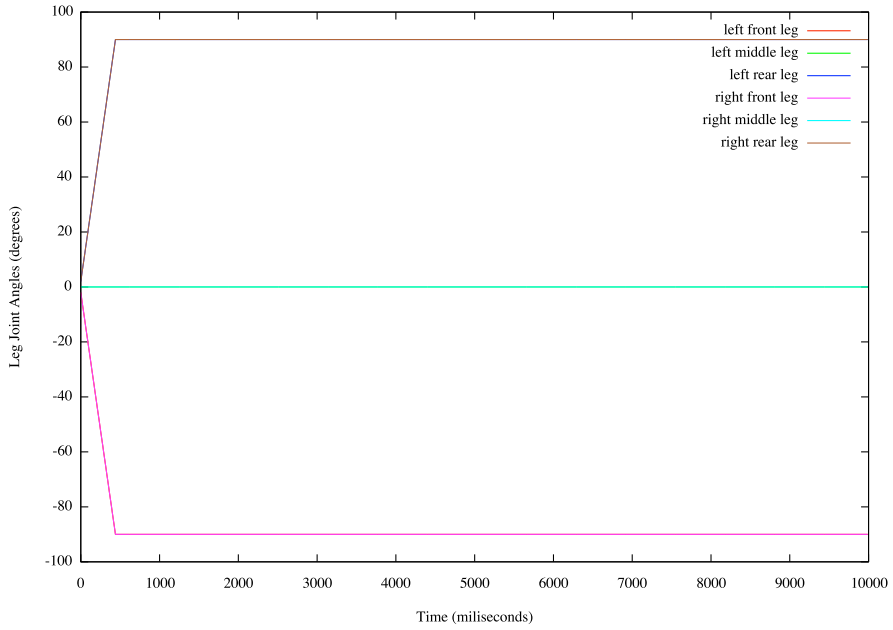
The pitch-up gait moves the robot 1m up with a 90° upward pitch around the $x-$axis. The seed gait provided to the gait synthesis system has the front two legs rotating downwards by 90° and the back two legs rotating upwards by 90°.

	position	**roll**	**pitch**	**yaw**	**linear velocity**	**angular velocity**
start	$(0, 0, 0)$	0°	0°	0°	$(0, 0, 0.1)$	$(0, 0, 0)$
goal	$(0, 1, 0)$	0°	-90°	0°	$(0, 0.1, 0)$	$(0, 0, 0)$

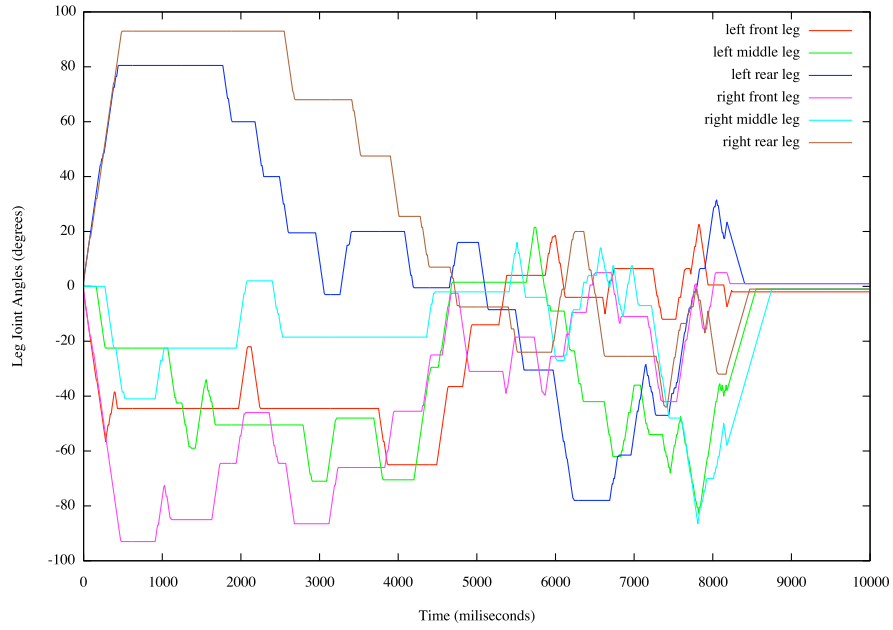
Table 4.10: The start and goal configuration-states for the pitch-up gait.

Starting temp. (T)	**Cooling rate (ΔT)**	**Attempts per temp.**
0.01	0.9	600

Table 4.11: The simulated annealing parameters used to generate the pitch-up gait.

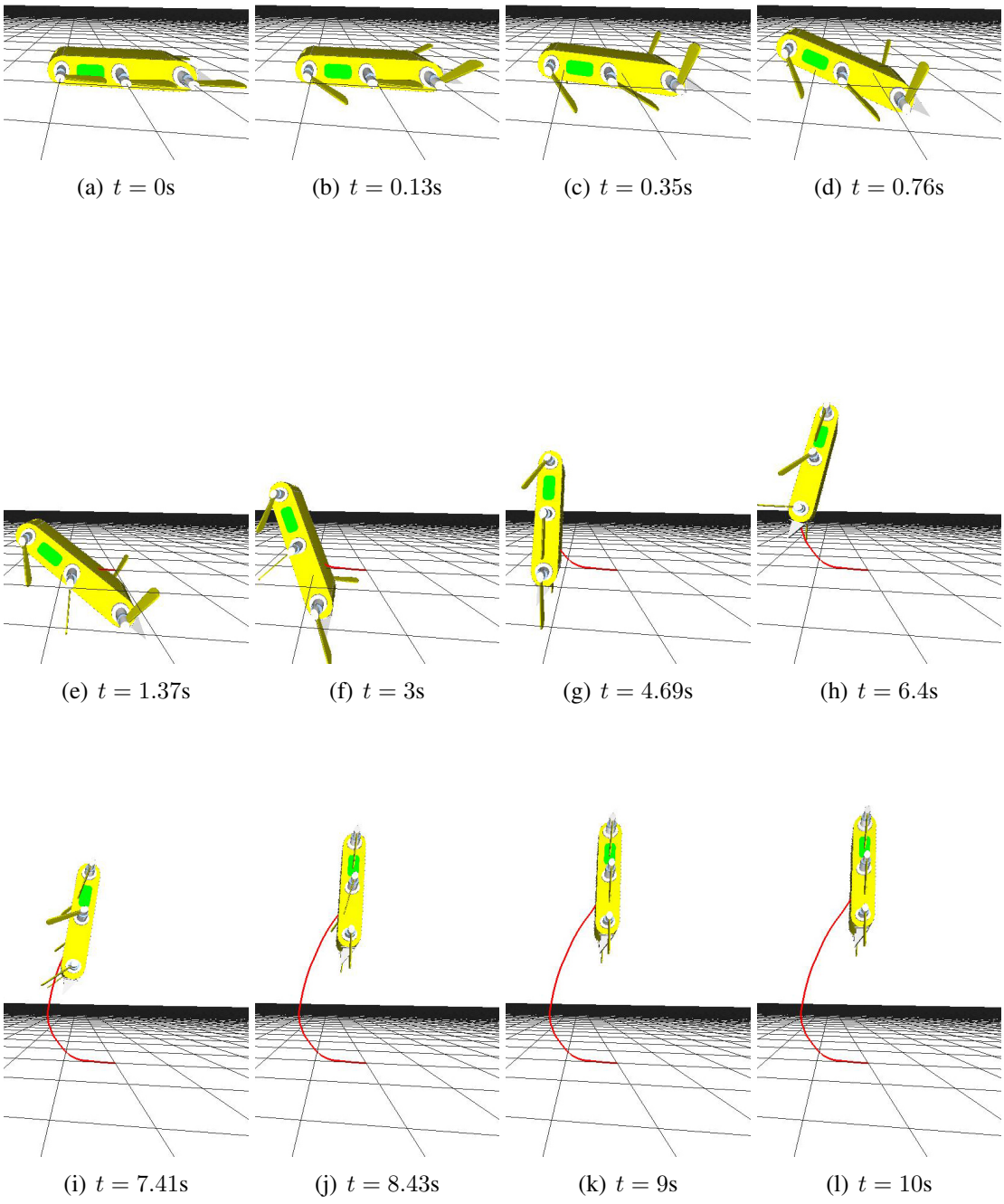


(a)



(b)

Figure 4.7: The initial seed gait (a) and the synthesized pitch-up gait (b).



(a) $t = 0$s (b) $t = 0.13$s (c) $t = 0.35$s (d) $t = 0.76$s

(e) $t = 1.37$s (f) $t = 3$s (g) $t = 4.69$s (h) $t = 6.4$s

(i) $t = 7.41$s (j) $t = 8.43$s (k) $t = 9$s (l) $t = 10$s

Figure 4.8: A side-on view of the simulated AQUA robot executing the synthesized pitch-up gait. The frames are labelled with the time (in seconds) they were captured throughout the gait. The path of the robot is drawn in red.

	position	**orientation**
goal	$(0, 1, 0)$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix}$
achieved	$(0.0, 1.0, 0.0)$	$\begin{pmatrix} 0.99 & 0.03 & 0.1 \\ -0.09 & -0.04 & 0.99 \\ 0.04 & -0.99 & -0.04 \end{pmatrix}$
	linear velocity	**angular velocity**
goal	$(0, 0.1, 0)$	$(0, 0, 0)$
achieved	$(0.0, 0.02, -0.01)$	$(0.0, 0.0, 0.0)$
	final configuration error	
goal	0.0	
achieved	0.007818	

Table 4.12: The goal and achieved configuration-states, and final error value for the pitch-up gait.

4.1.5 Pitch Down

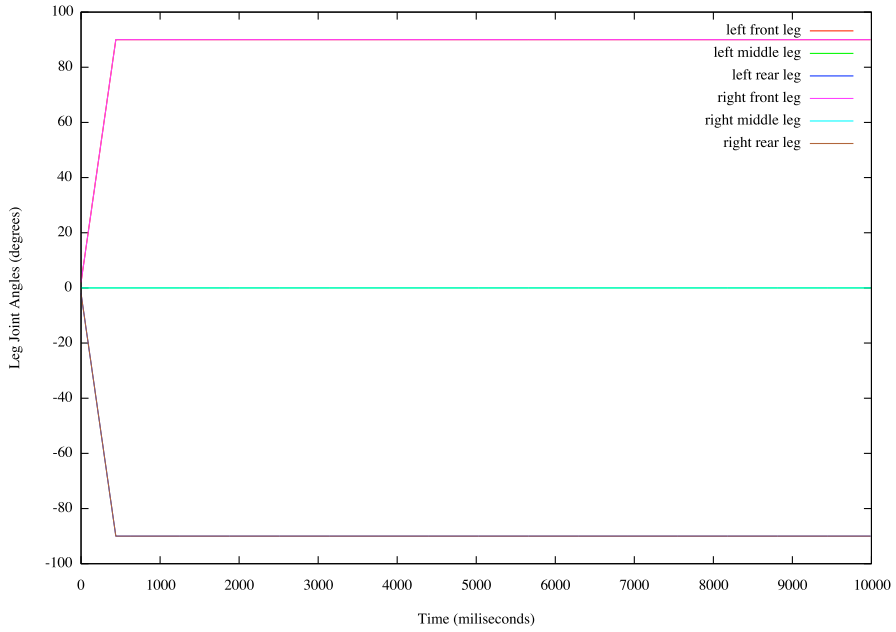
The pitch-up gait moves the robot down 1m with a 90° downward pitch around the $x-$axis. The seed gait provided to the gait synthesis system has the front two legs rotating upwards by 90° and the back two legs rotating downwards by 90°.

	position	**roll**	**pitch**	**yaw**	**linear velocity**	**angular velocity**
start	$(0, 0, 0)$	0°	0°	0°	$(0, 0, 0.1)$	$(0, 0, 0)$
goal	$(0, -1, 0)$	0°	90°	0°	$(0, -0.1, 0)$	$(0, 0, 0)$

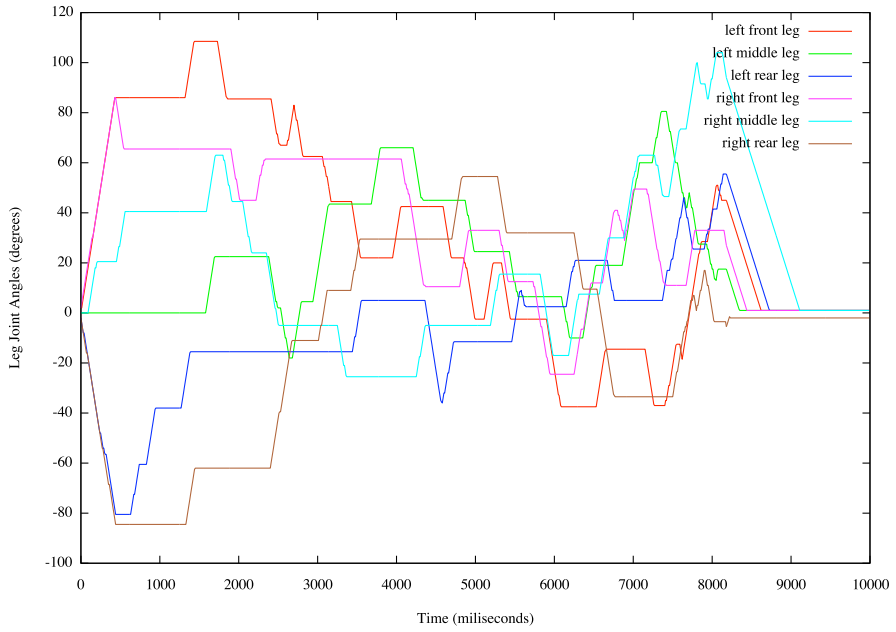
Table 4.13: The start and goal configuration-states for the pitch-down gait.

Starting temp. (T)	**Cooling rate (ΔT)**	**Attempts per temp.**
0.01	0.9	600

Table 4.14: The simulated annealing parameters used to generate the pitch-down gait.

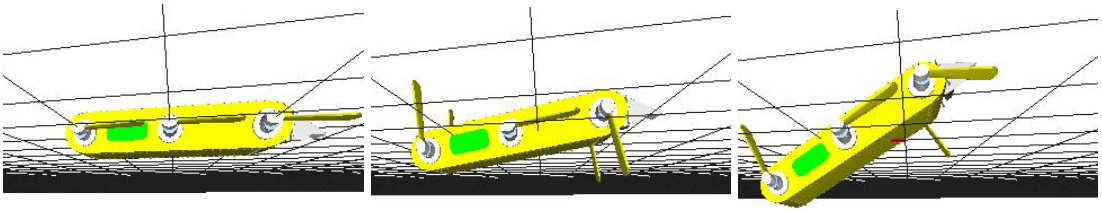


(a)

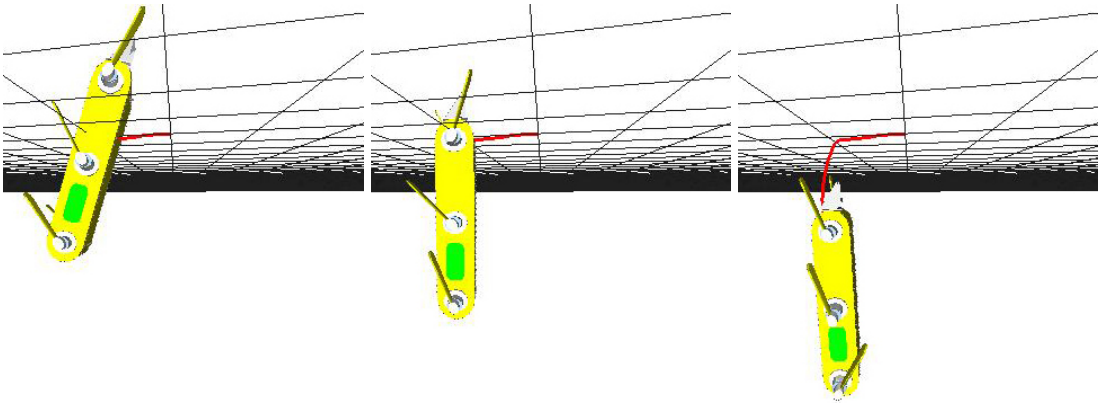


(b)

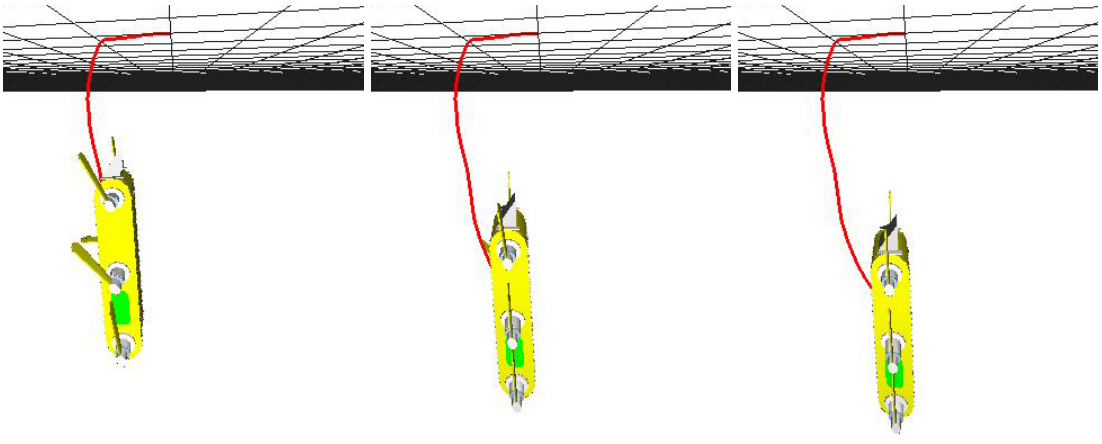
Figure 4.9: The initial seed gait (a) and the synthesized pitch-down gait (b).



(a) $t = 0$s (b) $t = 0.47$s (c) $t = 1.09$s



(d) $t = 3.14$s (e) $t = 4.69$s (f) $t = 6.56$s



(g) $t = 7.75$s (h) $t = 8.68$s (i) $t = 10$s

Figure 4.10: A side-on view of the simulated AQUA robot executing the synthesized pitch-down gait. The frames are labelled with the time (in seconds) they were captured throughout the gait. The path of the robot is drawn in red.

	position	**orientation**
goal	$(0, -1, 0)$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}$
achieved	$(0.0, -1.0, 0.0)$	$\begin{pmatrix} 1.0 & 0.04 & -0.06 \\ -0.06 & -0.08 & -1.0 \\ -0.04 & 1.0 & -0.08 \end{pmatrix}$
	linear velocity	**angular velocity**
goal	$(0, -0.1, 0)$	$(0, 0, 0)$
achieved	$(0.0, -0.02, -0.01)$	$(0.0, 0.0, 0.0)$
	final configuration error	
goal	0.0	
achieved	0.104858	

Table 4.15: The goal and achieved configuration-states, and final error value for the pitch-down gait.

4.1.6 Roll Left

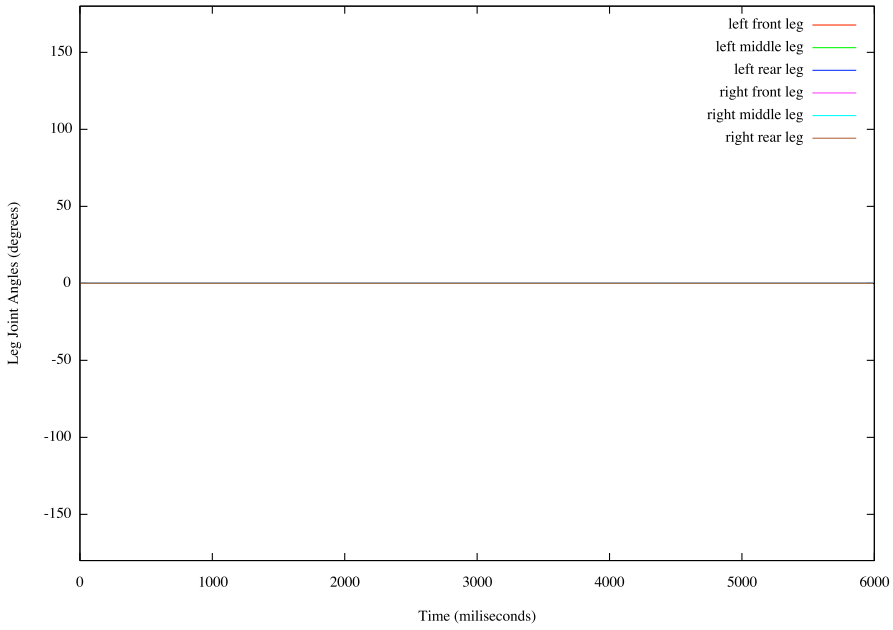
The roll-left gait moves the robot forward 1m with a 90° counter-clockwise rotation around the $z-$axis. The seed gait provided to the gait synthesis system has all the legs in the rest position.

	position	**roll**	**pitch**	**yaw**	**linear velocity**	**angular velocity**
start	$(0, 0, 0)$	0°	0°	0°	$(0, 0, 0.1)$	$(0, 0, 0)$
goal	$(0, 0, 1)$	-90°	0°	0°	$(0, 0, 0.1)$	$(0, 0, 0)$

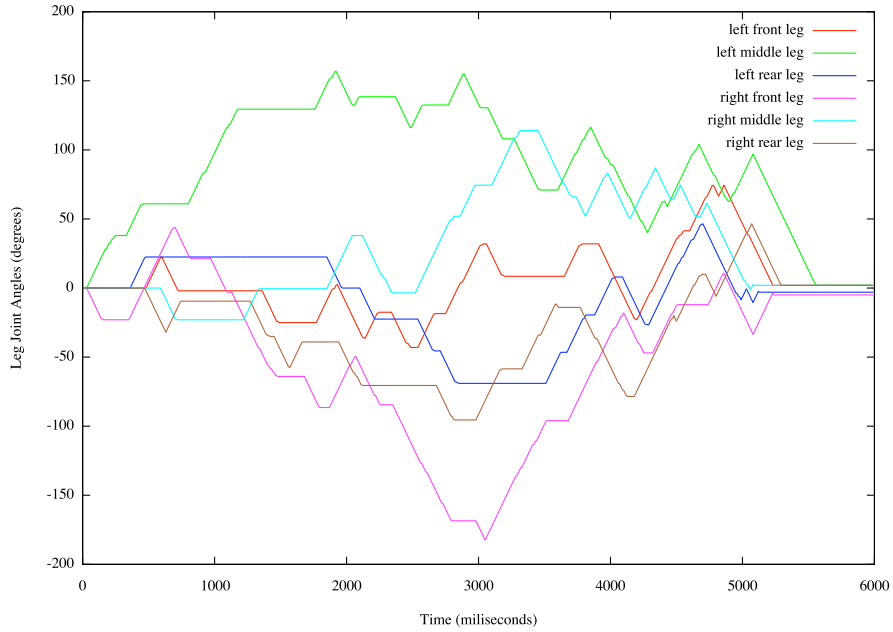
Table 4.16: The start and goal configuration-states for the roll-left gait.

Starting temp. (T)	**Cooling rate (ΔT)**	**Attempts per temp.**
0.1	0.9	600

Table 4.17: The simulated annealing parameters used to generate the roll-left gait.

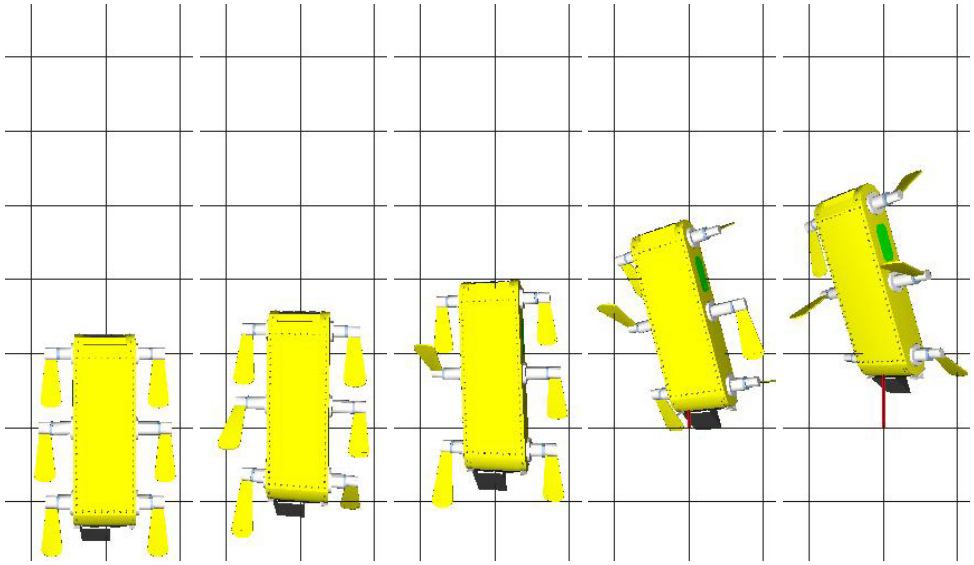


(a)

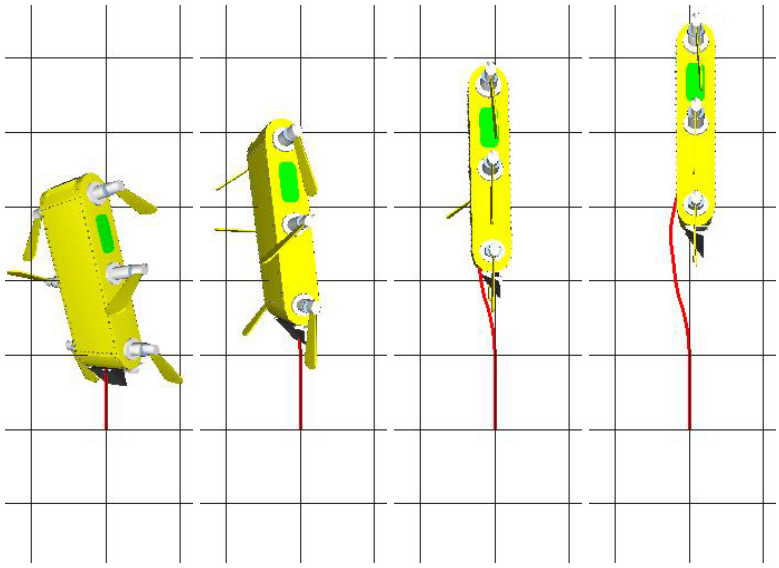


(b)

Figure 4.11: The initial seed gait (a) and the synthesized roll-left gait (b).



(a) $t = 0$s (b) $t = 0.63$s (c) $t = 1.23$s (d) $t = 2.43$s (e) $t = 3.41$s



(f) $t = 3.86$s (g) $t = 4.7$s (h) $t = 5.29$s (i) $t = 6$s

Figure 4.12: A top-down view of the simulated AQUA robot executing the synthesized roll-left gait. The frames are labelled with the time (in seconds) they were captured throughout the gait. The path of the robot is drawn in red.

	position	**orientation**
goal	$(0, 0, 1)$	$\begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
achieved	$(0.02, 0.0, 0.99)$	$\begin{pmatrix} 0.0 & 1.0 & 0.0 \\ -1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{pmatrix}$
	linear velocity	**angular velocity**
goal	$(0, 0, 0.1)$	$(0, 0, 0)$
achieved	$(-0.04, -0.01, 0.13)$	$(0.01, -0.01, 0.0)$
	final configuration error	
goal	0.0	
achieved	0.094334	

Table 4.18: The goal and achieved configuration-states, and final error value for the roll-left gait.

4.1.7 Roll Right

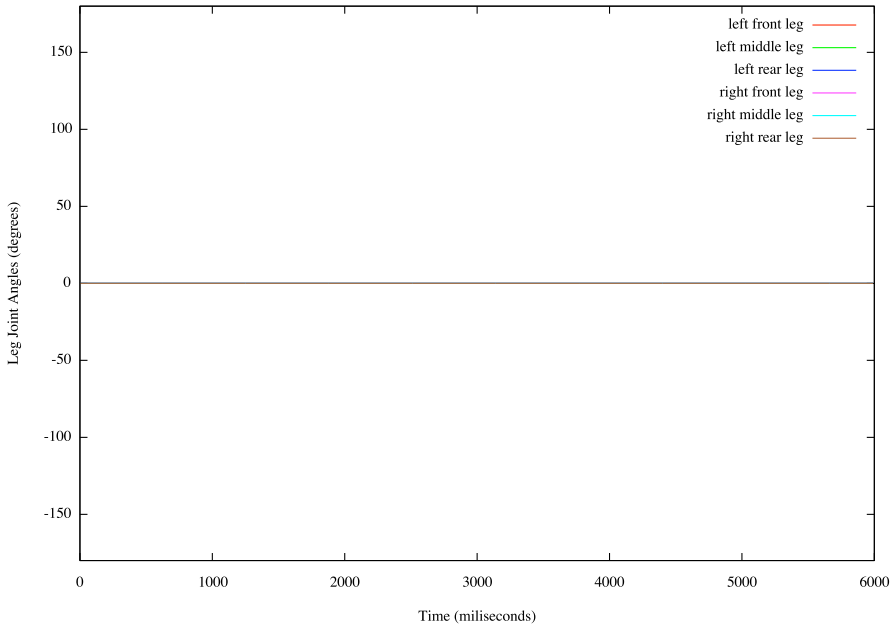
The roll-right gait moves the robot forward 1m with a 90° clockwise rotation around the $z-$axis. The seed gait provided to the gait synthesis system has all the legs in the rest position.

	position	**roll**	**pitch**	**yaw**	**linear velocity**	**angular velocity**
start	$(0, 0, 0)$	0°	0°	0°	$(0, 0, 0.1)$	$(0, 0, 0)$
goal	$(0, 0, 1)$	90°	0°	0°	$(0, 0, 0.1)$	$(0, 0, 0)$

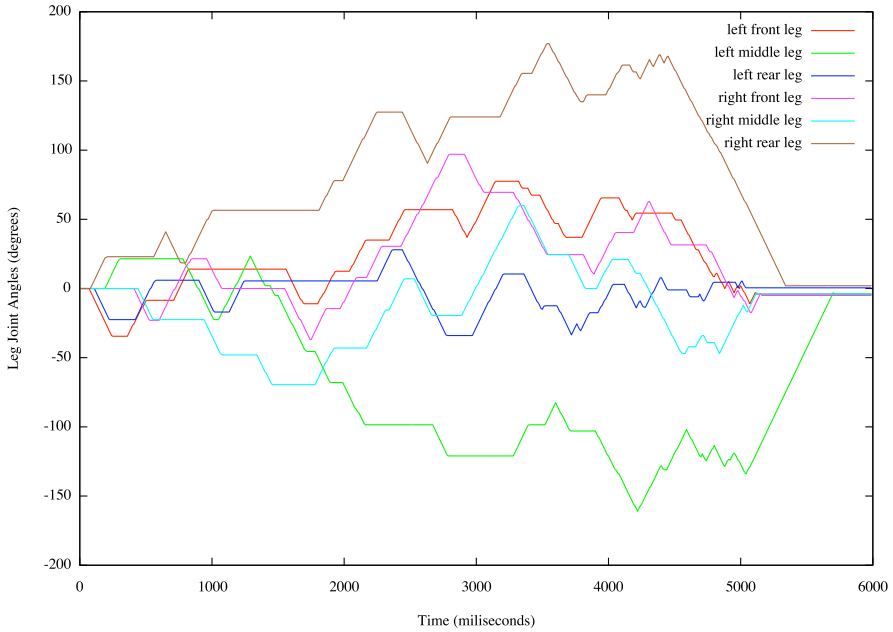
Table 4.19: The start and goal configuration-states for the roll-right gait.

Starting temp. (T)	**Cooling rate (ΔT)**	**Attempts per temp.**
0.01	0.9	600

Table 4.20: The simulated annealing parameters used to generate the roll-right gait.

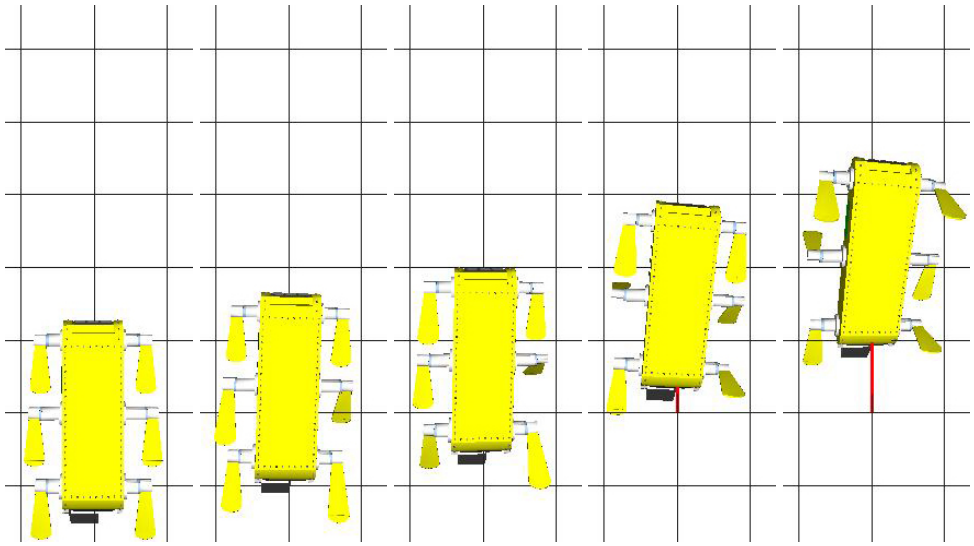


(a)

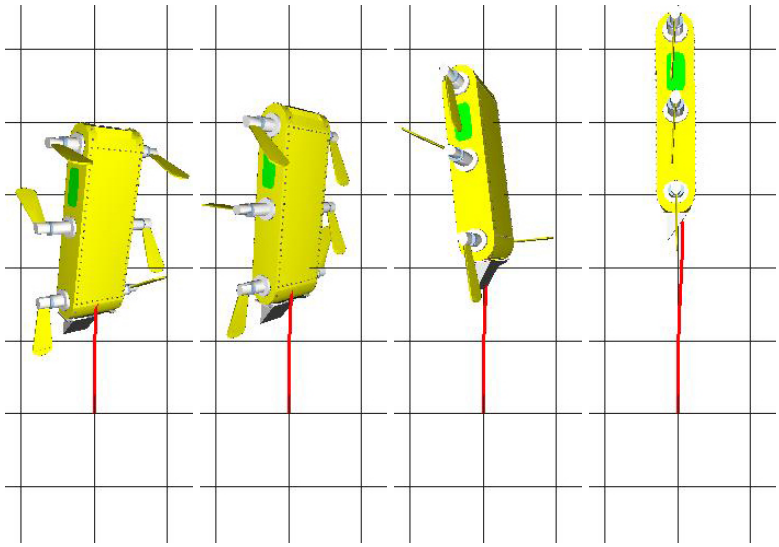


(b)

Figure 4.13: The initial seed gait (a) and the synthesized roll-right gait (b).



(a) $t = 0$s (b) $t = 0.72$s (c) $t = 1.14$s (d) $t = 2.07$s (e) $t = 2.64$s



(f) $t = 3.18$s (g) $t = 3.57$s (h) $t = 4.9$s (i) $t = 6$s

Figure 4.14: A top-down view of the simulated AQUA robot executing the synthesized roll-right gait. The frames are labelled with the time (in seconds) they were captured throughout the gait. The path of the robot is drawn in red.

	position	**orientation**
goal	$(0, 0, 1)$	$\begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
achieved	$(0.0, 0.0, 1.0)$	$\begin{pmatrix} 0.0 & -1.0 & 0.0 \\ 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{pmatrix}$
	linear velocity	**angular velocity**
goal	$(0, 0, 0.1)$	$(0, 0, 0)$
achieved	$(-0.01, -0.01, 0.12)$	$(-0.01, -0.06, 0.0)$
	final configuration error	
goal	0.0	
achieved	0.089474	

Table 4.21: The goal and achieved configuration-states, and final error value for the roll-right gait.

4.1.8 Heave Up

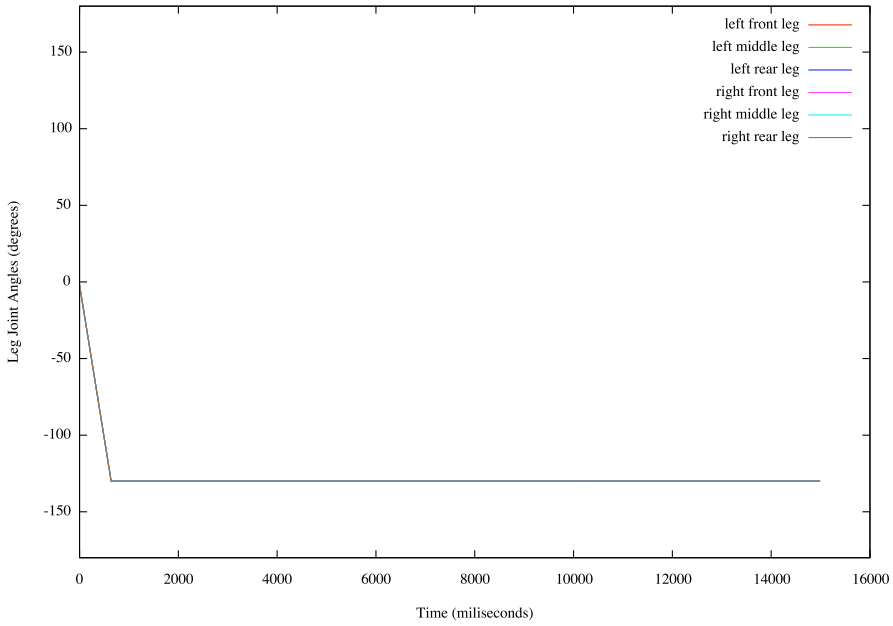
The heave-up gait moves the robot up 1m without a rotation. The seed gait provided to the gait synthesis system has all the legs rotate 130° downwards. This helps the vehicle negate the initial forward velocity and directs it upwards.

	position	**roll**	**pitch**	**yaw**	**linear velocity**	**angular velocity**
start	$(0, 0, 0)$	0°	0°	0°	$(0, 0, 0.1)$	$(0, 0, 0)$
goal	$(0, 1, 0)$	0°	0°	0°	$(0, 0, 0.1)$	$(0, 0, 0)$

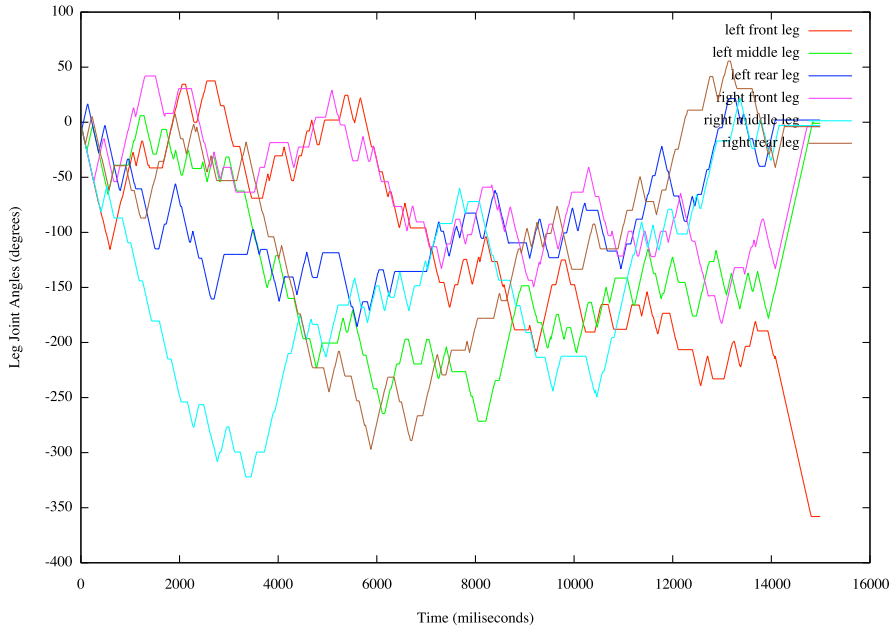
Table 4.22: The start and goal configuration-states for the heave-up gait.

Starting temp. (T)	**Cooling rate (ΔT)**	**Attempts per temp.**
0.01	0.9	600

Table 4.23: The simulated annealing parameters used to generate the heave-up gait.

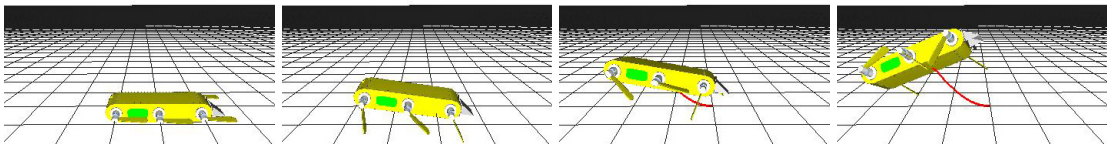


(a)

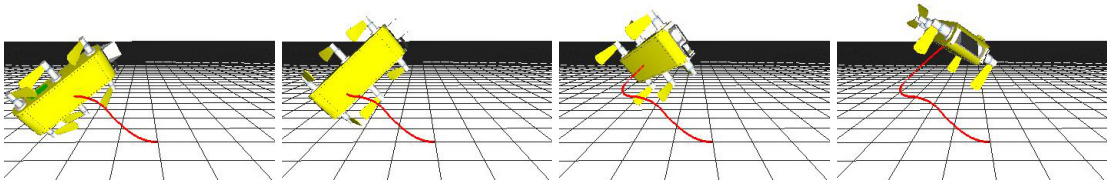


(b)

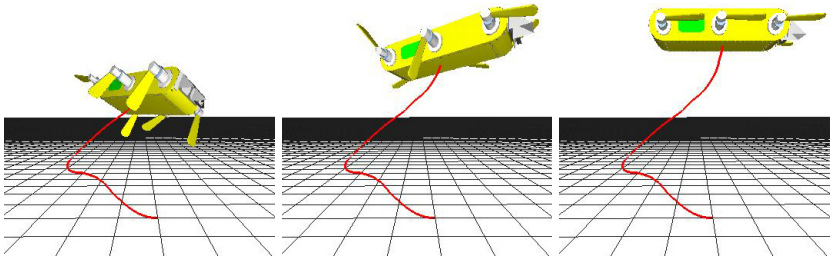
Figure 4.15: The initial seed gait (a) and the synthesized heave-up gait (b).



(a) $t = 0$s (b) $t = 0.76$s (c) $t = 1.65$s (d) $t = 2.81$s



(e) $t = 4.24$s (f) $t = 6.32$s (g) $t = 7.75$s (h) $t = 9.94$s



(i) $t = 11.07$s (j) $t = 12.89$s (k) $t = 15$s

Figure 4.16: A side-on view of the simulated AQUA robot executing the synthesized heave-up gait. The frames are labelled with the time (in seconds) they were captured throughout the gait. The path of the robot is drawn in red.

	position	**orientation**
goal	$(0, 1, 0)$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
achieved	$(0.0, 0.99, 0.0)$	$\begin{pmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{pmatrix}$
	linear velocity	**angular velocity**
goal	$(0, 0, 0.1)$	$(0, 0, 0)$
achieved	$(0.0, 0.02, 0.05)$	$(0.01, 0.02, 0.0)$
	final configuration error	
goal	0.0	
achieved	0.091149	

Table 4.24: The goal and achieved configuration-states, and final error value for the heave-up gait.

4.1.9 Heave Down

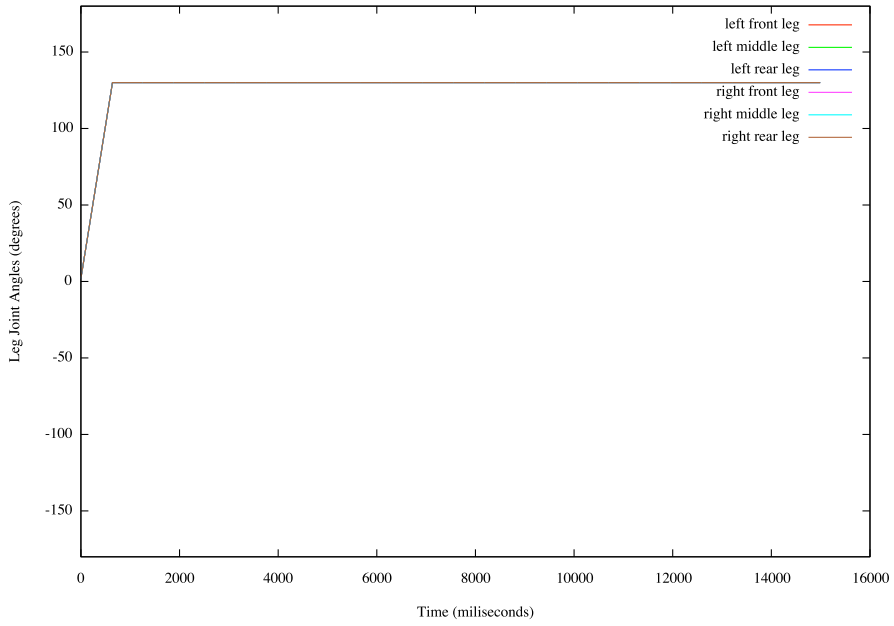
The heave-down gait moves the robot down 1m without a rotation. The seed gait provided to the gait synthesis system has all the legs rotate 130° upwards. This helps the vehicle negate the initial forward velocity and directs it downwards.

	position	**roll**	**pitch**	**yaw**	**linear velocity**	**angular velocity**
start	$(0, 0, 0)$	0°	0°	0°	$(0, 0, 0.1)$	$(0, 0, 0)$
goal	$(0, -1, 0)$	0°	0°	0°	$(0, 0, 0.1)$	$(0, 0, 0)$

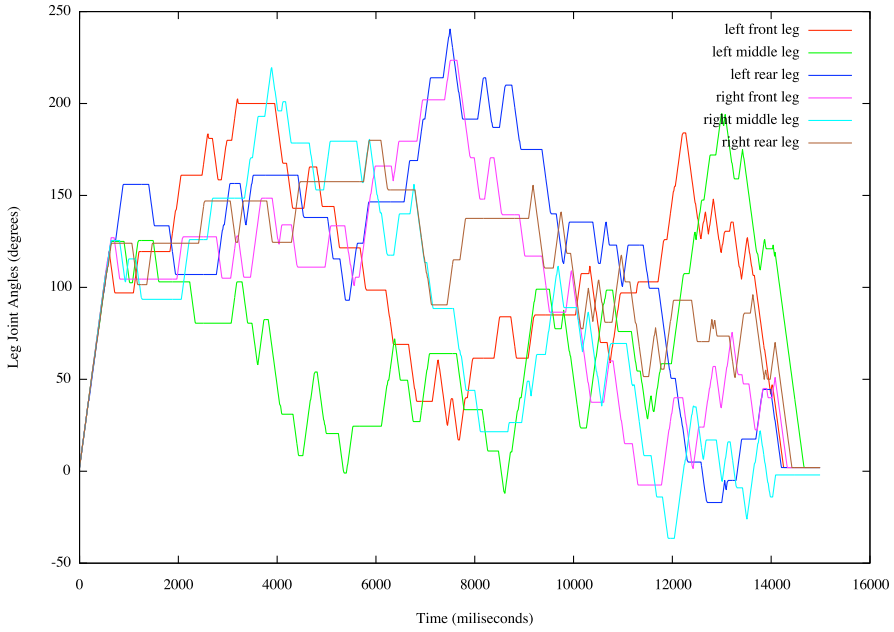
Table 4.25: The start and goal configuration-states for the heave-down gait.

Starting temp. (T)	**Cooling rate (ΔT)**	**Attempts per temp.**
0.01	0.9	600

Table 4.26: The simulated annealing parameters used to generate the heave-down gait.

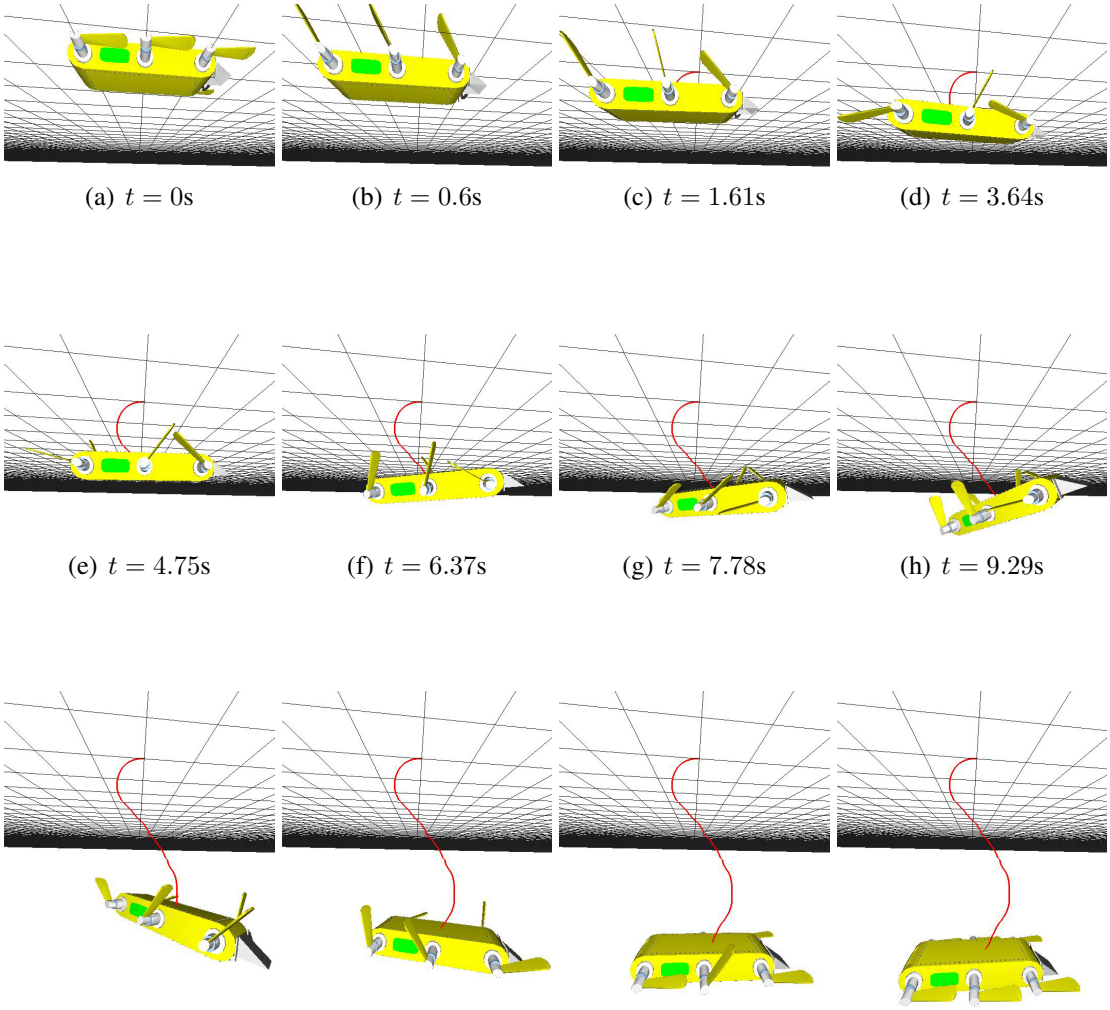


(a)



(b)

Figure 4.17: The initial seed gait (a) and the synthesized heave-down gait (b).



(a) $t = 0$s (b) $t = 0.6$s (c) $t = 1.61$s (d) $t = 3.64$s

(e) $t = 4.75$s (f) $t = 6.37$s (g) $t = 7.78$s (h) $t = 9.29$s

(i) $t = 11.97$s (j) $t = 13.67$s (k) $t = 14.41$s (l) $t = 15$s

Figure 4.18: A side-on view of the simulated AQUA robot executing the synthesized heave-down gait. The frames are labelled with the time (in seconds) they were captured throughout the gait. The path of the robot is drawn in red.

	position	**orientation**
goal	$(0, -1, 0)$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
achieved	$(0.01, -1.0, 0.0)$	$\begin{pmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{pmatrix}$
	linear velocity	**angular velocity**
goal	$(0, 0, 0.1)$	$(0, 0, 0)$
achieved	$(0.0, -0.03, 0.04)$	$(0.0, 0.02, 0.0)$
	final configuration error	
goal	0.0	
achieved	0.105550	

Table 4.27: The goal and achieved configuration-states, and final error value for the heave-down gait.

4.1.10 Accelerate

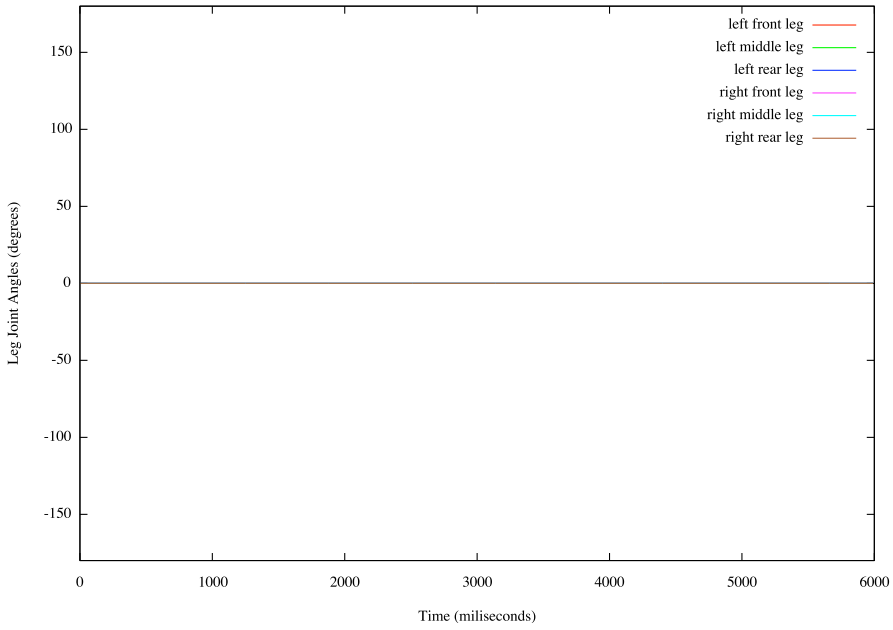
The accelerate gait moves the robot down from a static position forward 1m and accelerate it up to $0.1m/s$. The seed gait provided to the gait synthesis system has all the legs in the rest position.

	position	**roll**	**pitch**	**yaw**	**linear velocity**	**angular velocity**
start	$(0, 0, 0)$	$0°$	$0°$	$0°$	$(0, 0, 0)$	$(0, 0, 0)$
goal	$(0, 0, 1)$	$0°$	$0°$	$0°$	$(0, 0, 0.1)$	$(0, 0, 0)$

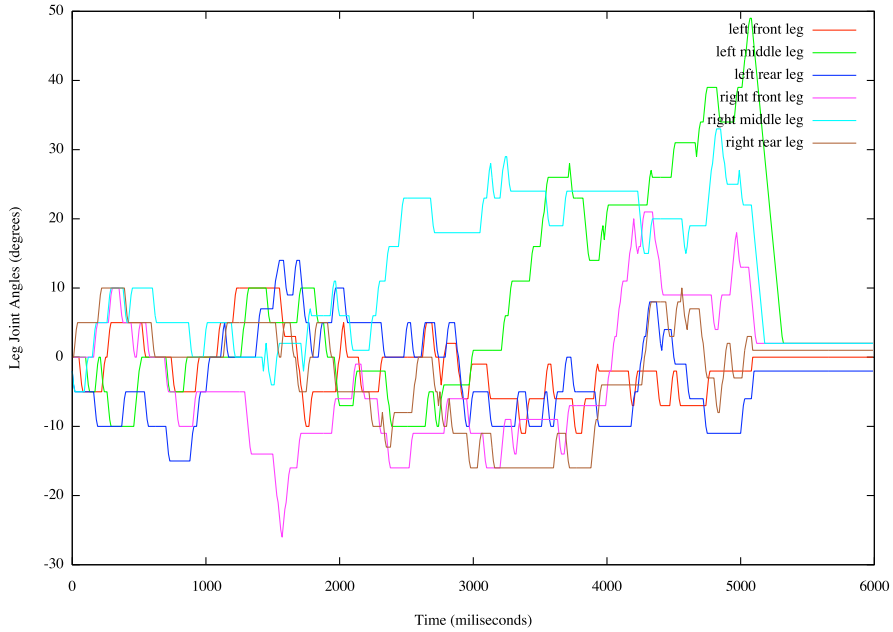
Table 4.28: The start and goal configuration-states for the accelerate gait.

Starting temp. (T)	**Cooling rate (ΔT)**	**Attempts per temp.**
0.01	0.9	600

Table 4.29: The simulated annealing parameters used to generate the accelerate gait.

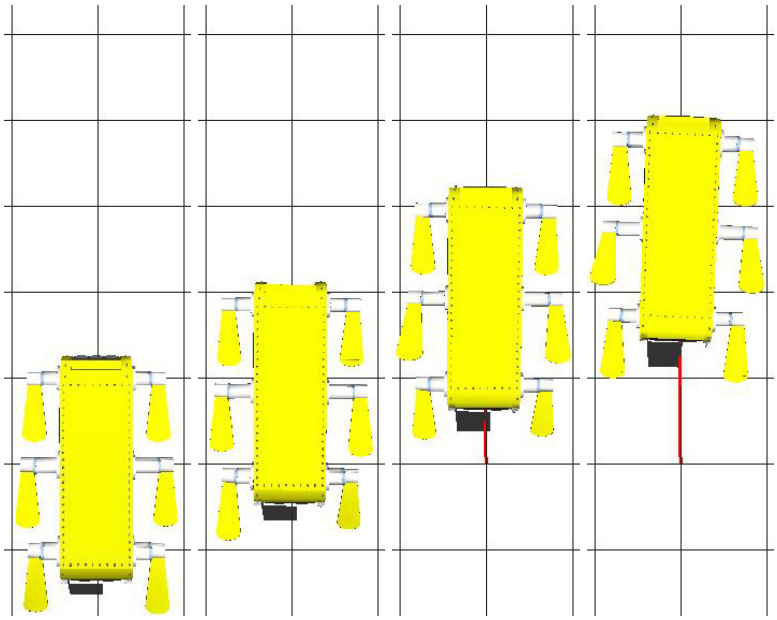


(a)

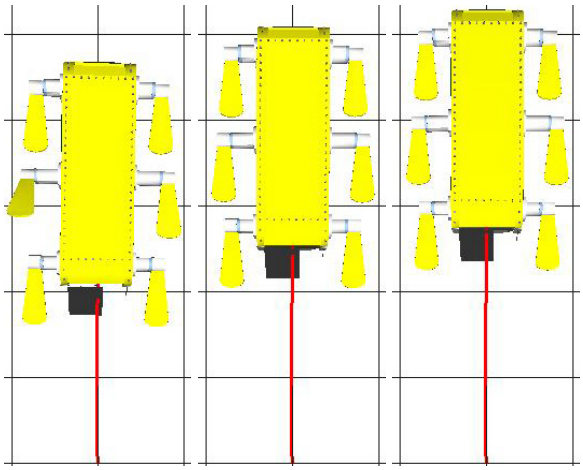


(b)

Figure 4.19: The initial seed gait (a) and the synthesized accelerate gait (b).



(a) $t = 0$s (b) $t = 2.26$s (c) $t = 3.6$s (d) $t = 4.47$s



(e) $t = 5.13$s (f) $t = 5.64$s (g) $t = 6$s

Figure 4.20: A top-down view of the simulated AQUA robot executing the synthesized accelerate gait. The frames are labelled with the time (in seconds) they were captured throughout the gait. The path of the robot is drawn in red.

	position	**orientation**
goal	$(0, 0, 1)$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
achieved	$(-0.01, -0.01, 1.0)$	$\begin{pmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.01 \\ 0.0 & -0.01 & 1.0 \end{pmatrix}$
	linear velocity	**angular velocity**
goal	$(0, 0, 0.1)$	$(0, 0, 0)$
achieved	$(0.0, -0.02, 0.14)$	$(0.01, 0.0, 0.0)$
	final configuration error	
goal	0.0	
achieved	0.069134	

Table 4.30: The goal and achieved configuration-states, and final error value for the accelerate gait.

4.1.11 Decelerate

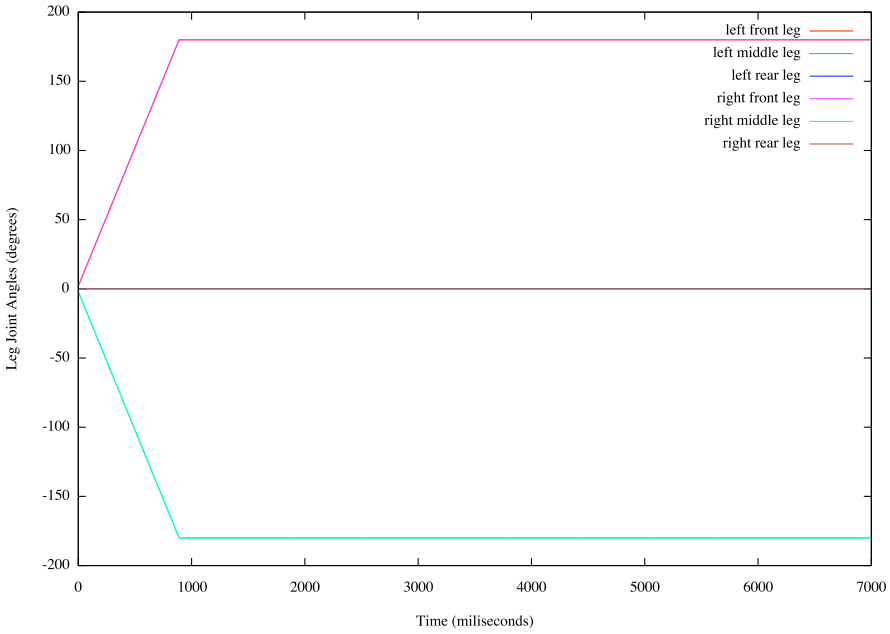
The decelerate gait halts the forward momentum of a robot travelling at $0.1m/s$ bringing it to rest. The seed gait provided to the gait synthesis system has the front two legs rotate 180° upwards and the middle-two legs rotate 180° downwards. This leg configuration helps negate the forward momentum and allows the gait synthesis system to generate a gait which brings the vehicle to rest.

	position	**roll**	**pitch**	**yaw**	**linear velocity**	**angular velocity**
start	$(0, 0, 0)$	0°	0°	0°	$(0, 0, 0.1)$	$(0, 0, 0)$
goal	$(0, 0, 0)$	0°	0°	0°	$(0, 0, 0)$	$(0, 0, 0)$

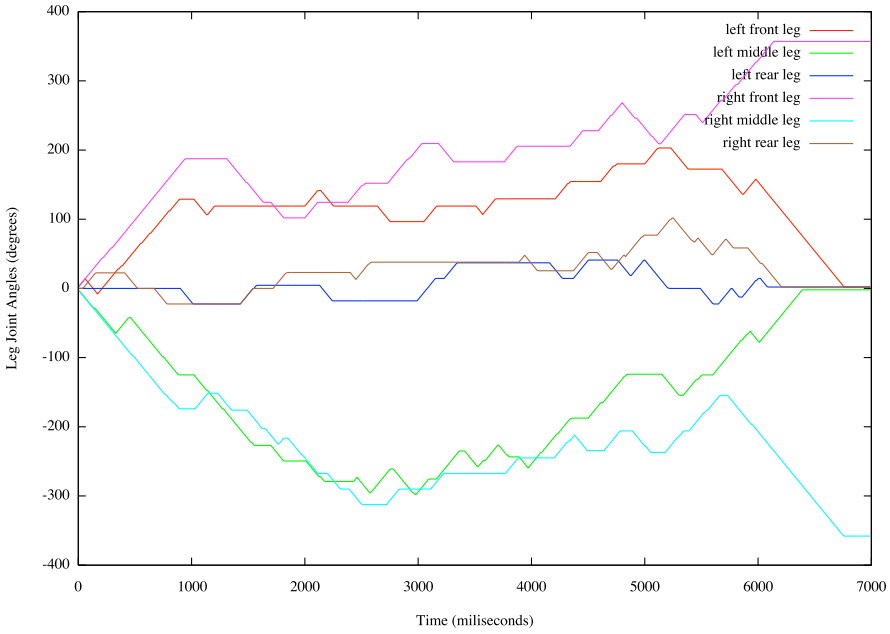
Table 4.31: The start and goal configuration-states for the decelerate gait.

Starting temp. (T)	**Cooling rate (ΔT)**	**Attempts per temp.**
0.01	0.9	600

Table 4.32: The simulated annealing parameters used to generate the decelerate gait.

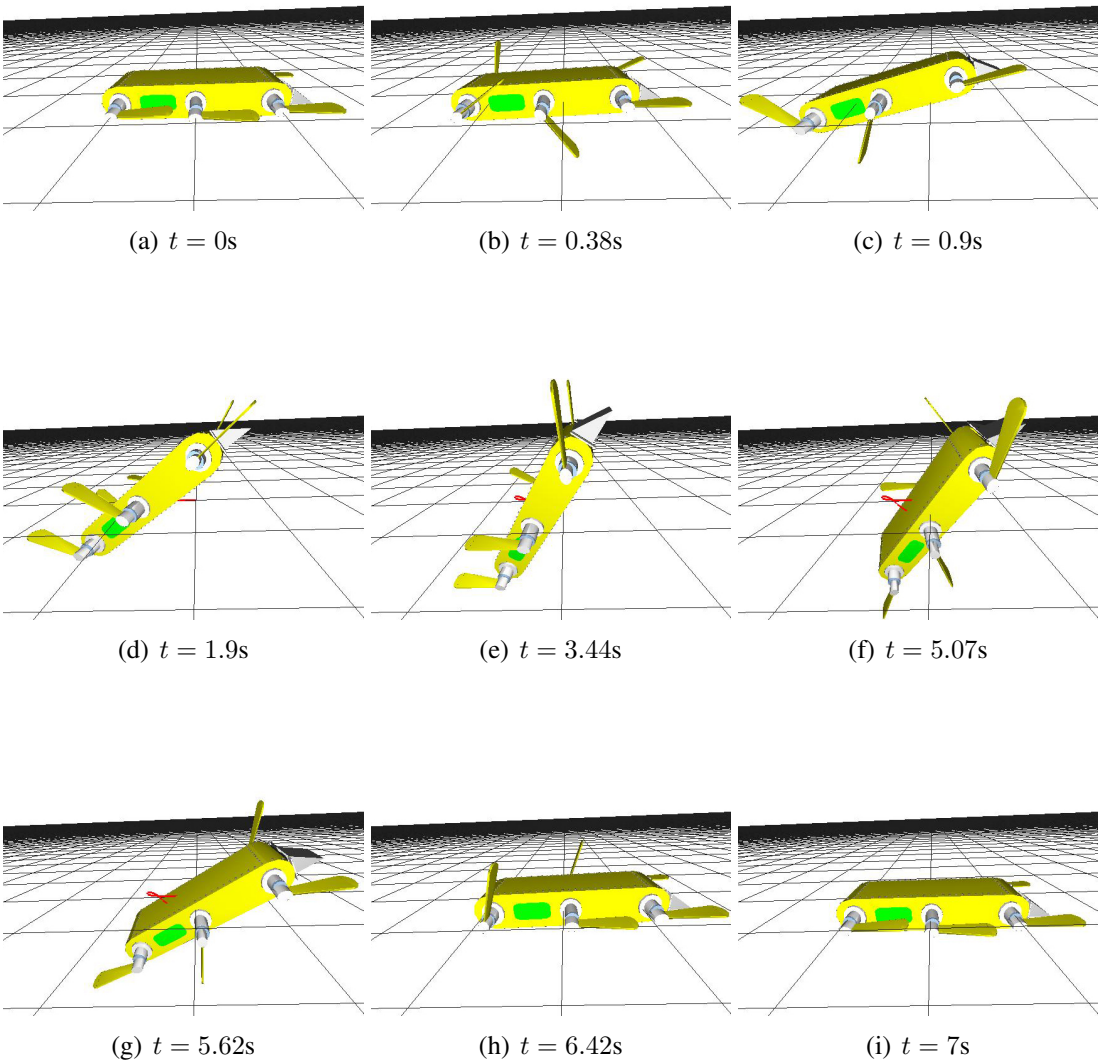


(a)



(b)

Figure 4.21: The initial seed gait (a) and the synthesized decelerate gait (b).



(a) $t = 0$s (b) $t = 0.38$s (c) $t = 0.9$s

(d) $t = 1.9$s (e) $t = 3.44$s (f) $t = 5.07$s

(g) $t = 5.62$s (h) $t = 6.42$s (i) $t = 7$s

Figure 4.22: A side-on view of the simulated AQUA robot executing the synthesized decelerate gait. The frames are labelled with the time (in seconds) they were captured throughout the gait. The path of the robot is drawn in red.

	position	**orientation**
goal	$(0, 0, 0)$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
achieved	$(-0.01, -0.01, 1.0)$	$\begin{pmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{pmatrix}$
	linear velocity	**angular velocity**
goal	$(0, 0, 0)$	$(0, 0, 0)$
achieved	$(0.0, -0.02, 0.03)$	$(0.0, 0.0, 0.0)$
	final configuration error	
goal	0.0	
achieved	0.076355	

Table 4.33: The goal and achieved configuration-states, and final error value for the decelerate gait.

4.2 Summary

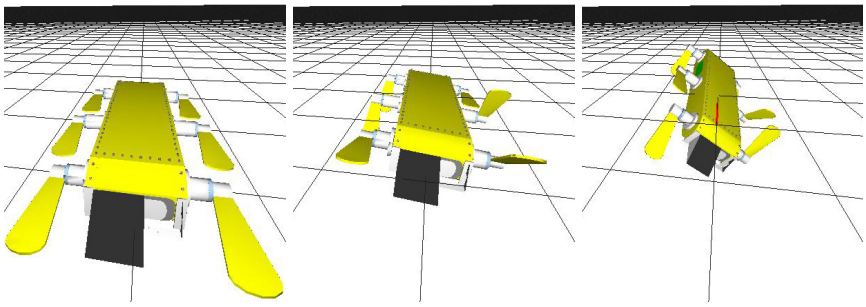
This chapter presented an alphabet of synthesized gaits for the AQUA underwater vehicle. Each gait in the alphabet provides a unique motion that moves the vehicle by 1m and many change the orientation of the vehicle by 90°, several also change the vehicle's velocity. The choice of the simulated annealing parameters as well as the choice of the seed gait are important to the successful synthesis of gaits. If the starting temperature is not high enough the simulated annealing engine may not be able to escape local minima during the search process. Likewise, if the initial seed gait provided to the engine while synthesizing a surge-forward gait has all the legs flipped 180° from the correct position, the temperature will need to be set much higher in order to accept the motion caused by correcting the leg orientations. There are several things to note however: First, the gaits presented are by no means the only possible gaits which can be synthesized. Second, the gaits presented above (and indeed any gait synthesized by the technique presented) are not necessarily the best possible gaits. There may exist "more optimal" and indeed an "optimum" gait given a specific error metric, but simulated annealing is not guaranteed to find it in practice.

The choices of which motions to synthesize for inclusion in the gait alphabet is driven by their use in the path-planner explained in the next chapter. In this case discretizing the space into 1m grids and synthesizing gaits that move the vehicle to adjacent grids makes

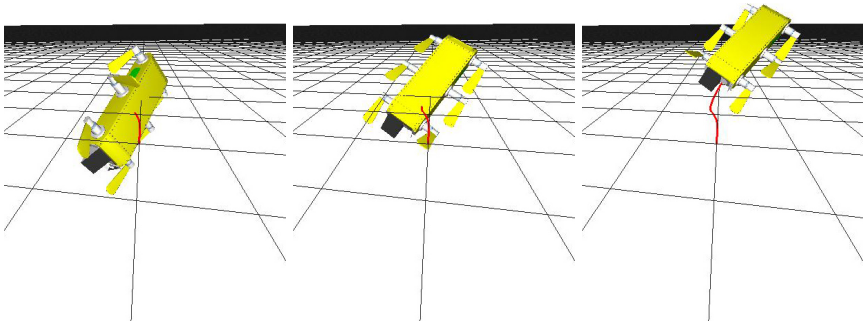
generating a path for the robot fairly simple. However, there are an endless number of other motions which could be synthesized by this system. An example of one such gait is presented in Figure 4.23 where the gait synthesis system generated a gait to move the robot to $(-1, 1, 1)$ with a 45° yaw and a 45° pitch. Furthermore, the gaits generated by the synthesis system do not need to share initial and final linear and angular velocities, this was another simplification made for the sake of linking the gaits together as described in the next chapter.

The gaits synthesized for inclusion in the gait alphabet may not represent the truly *optimal* leg joint-angle patterns. That is, since the simulated annealing algorithm makes no guarantee regarding the production of the optimal solution to the search problem, there may be gaits which can outperform those presented above. This is especially true for special cases such as synthesizing the "fastest" gait. Figure 4.24 illustrates a race between a robot executing the hand-tuned surge gait illustrated in Figure 1.3 (sinusoidal leg joint-angle motions) and a robot executing a surge gait synthesized by the system presented here. Even when the surge gait is synthesized using the goal position as the only error metric (adding other constraints would sacrifice speed to achieve a specified orientation, etc.) the synthesized gait cannot outperform a hand-tuned gait in this simulated drag-race. However hand-tuning a gait to turn 90° and come to a stop is nearly impossible whereas the system presented here can synthesize it with relative ease.

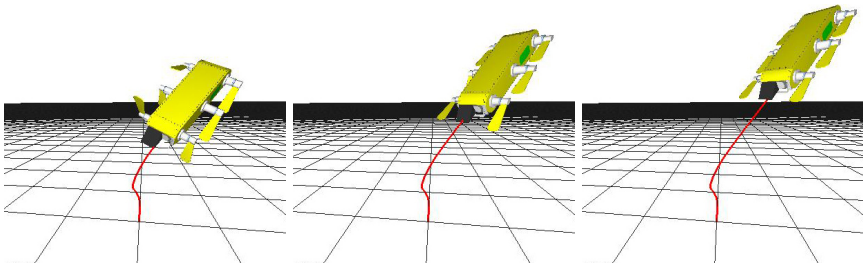
In the next chapter a mechanism by which these gaits are combined to form complex



(a) $t = 0$s (b) $t = 0.7$s (c) $t = 1.63$s

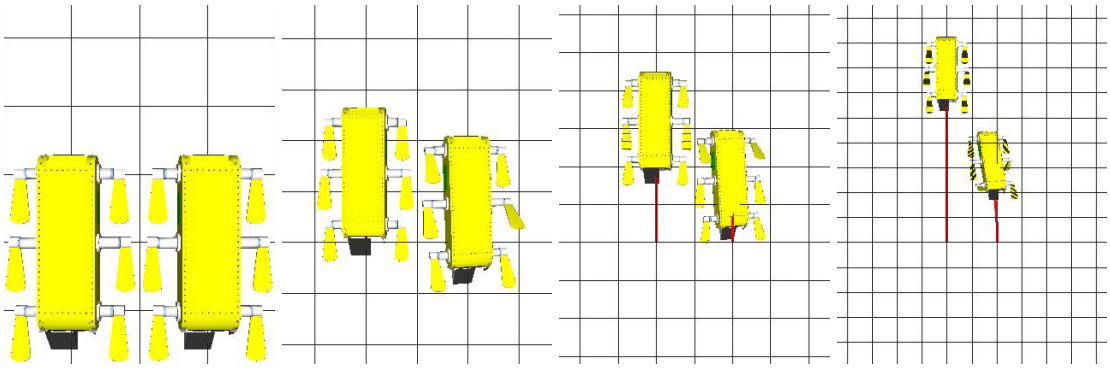


(d) $t = 2.47$s (e) $t = 3.59$s (f) $t = 4.51$s

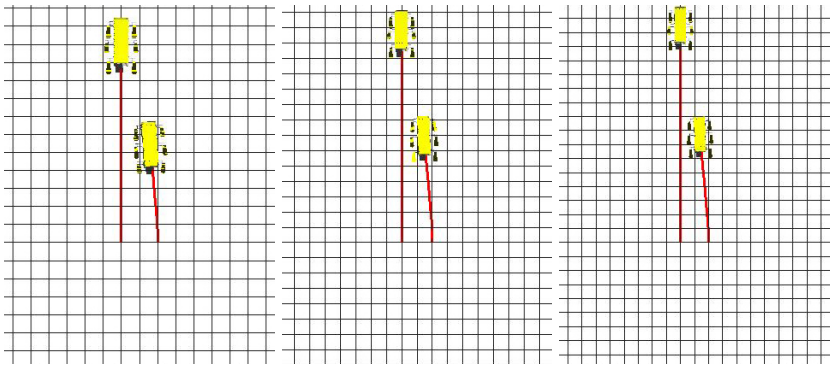


(g) $t = 4.94$s (h) $t = 5.47$s (i) $t = 6$s

Figure 4.23: A view of the simulated AQUA robot executing a synthesized gait. The frames are labelled with the time (in seconds) they were captured throughout the gait. The path of the robot is drawn in red.



(a) $t = 0$s (b) $t = 1.02$s (c) $t = 1.75$s (d) $t = 2.88$s



(e) $t = 3.95$s (f) $t = 4.58$s (g) $t = 5$s

Figure 4.24: A race between a robot executing a hand-tuned surge gait (left) and a robot executing a synthesized surge gait (right). To ensure the synthesized gait moves the robot as quickly as possible the only error constraint used is the goal position (4m forward). The frames are labelled with the time (in seconds) they were captured throughout the gait. The path of the robot is drawn in red.

motions is detailed.

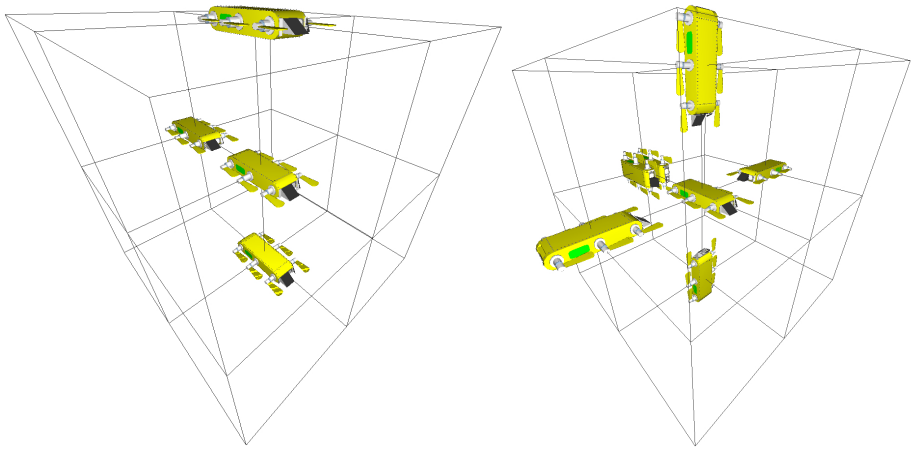
Chapter 5 Path planning with the gait alphabet

Chapter 3 outlined the system for generating an alphabet of gaits. This system was used to synthesize the gaits presented in Chapter 4. Each of these member gaits is useful on its own but the gaits must be linked together to form more complex maneuvers. Which gaits are linked together and the ordering of the sequence of gaits depends on the required maneuvers. This chapter presents an automatic gait sequencer used to choose which gaits should be linked in order to move the virtual vehicle from point A to B while avoiding obstacles in its path.

The problem of navigating the vehicle between two configurations is formulated as a path-planning problem and approached using classical artificial intelligence graph search techniques. The environment is discretized into $1m^3$ 3D grids with six possible orientations per location. The choice of discretizing the environment into a grid is a simplification in order to make the path planning and gait synthesis easier to perform since the individual gait alphabet is based on axis-aligned motions with 1m motions per time step. This is by no means the only approach. More complex individual gaits would require

a more sophisticated representation in terms of state for the gait linking, but the basic approach presented in this chapter would still apply. Discrete locations in space and orientation are represented as nodes in a graph, valid configurations for the robot in the 6DOF environment. The structured alphabet of gaits developed in the previous chapter (see Figure 5.1) are designed to transit the robot within this discretized space. A single node in the graph has edges corresponding to all possible directly reachable configurations via the gaits in the alphabet (see Figure 5.2). A graph search technique is used to decide which edges should be followed (which gaits should be linked) to move between different configurations subject to the defined start and goal locations and the presence of obstacles within the robot's environment. Within the graph representation above the problem becomes that of finding a path (preferably a least-cost path) through the graph between a start node and a goal node while avoiding obstacles. The nodes in the path represent gaits which – when linked together – will transition the robot between the start and goal configurations (see Figure 5.3). To perform this graph search the A* algorithm is used [31] (see Algorithm 2.1).

One issue that must be addressed when linking gaits together is the small errors associated with each gait. No individual gait is perfect in terms of its ability to move the robot within this grid. Thus while retaining this grid-like representation of space and orientation the actual orientation and position of the vehicle (as computed by the gait synethesis process) is used.



(a) Translation gaits (b) Rotation gaits

Figure 5.1: The alphabet of gaits synthesized in Chapter 3 were chosen to transit the robot 1m from its current position and with a possible change in orientation. These figures represent the origin and goal positions for each member gait of the alphabet. 5.1(a) Translation gaits (surge forward, heave up and heave down). 5.1(b) Rotation gaits (yaw left, yaw right, pitch up, pitch down, roll left and roll right – separated for clarity).

5.1 Application of A* to motion synthesis

A* is a best-first graph search algorithm which finds the least-cost path between two nodes in a graph whose edges have associated weights (costs). The A* algorithm requires the definition of two cost functions: an upper bound on the cost from the current node n back to the start node ($g(n)$), and a lower bound of the cost from the current node n to the goal node ($h(n)$). Here we define this cost function in terms of the number of node-to-node transitions (required individual gaits) needed to move the robot from the start node to the goal node (i.e., the fewest linked gaits required to transit the vehicle between the start configuration-state and the goal configuration-state). According to the

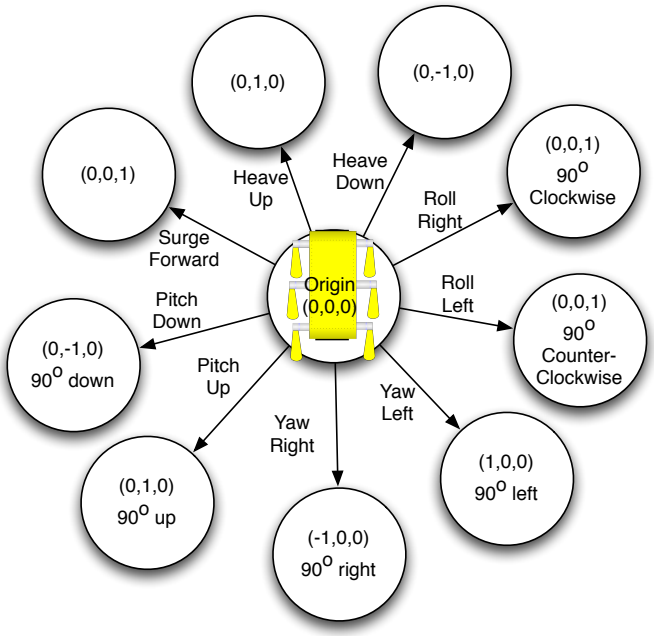
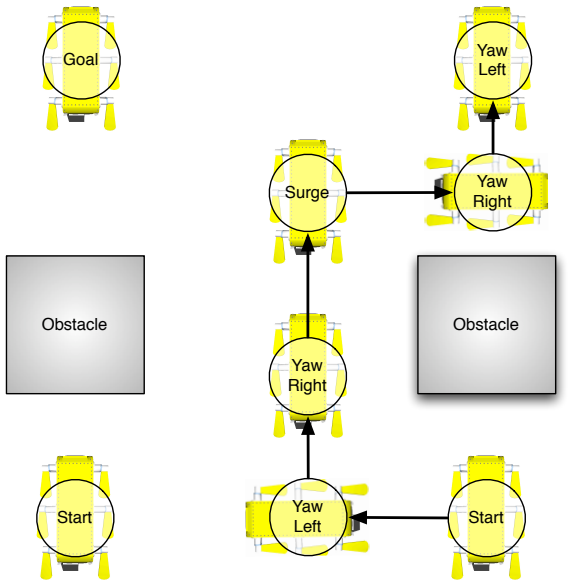


Figure 5.2: If possible vehicle configurations are nodes in a graph, then the adjacent nodes can be reached via member-gaits from the alphabet. This figure shows the origin configuration-state and its adjacent nodes. These are also depicted in Figure 5.1.

A* algorithm, $h(n)$ is a heuristic measure of the lowest possible 'cost' from the current node to the goal node. We define $h(n)$ as the minimum number of gaits that might be needed to reach the goal node from the current node. Given the nature of the individual gaits that make up the gait alphabet, this estimate is found using the Manhattan distance between the current position and the goal position. More specifically:

$$h(n) = |x_g - x_n| + |y_g - y_n| + |z_g - z_n|.$$

This is a good lowest-cost measure because each gait from the alphabet moves a unit distance along a single axis, matching the grid-based distance of the Manhattan function.



(a) The problem (b) A possible path

Figure 5.3: The problem (a): Is there a linear combination of gaits which will take the vehicle from the start configuration to the goal configuration? One possible path is illustrated in (b).

Using this metric the algorithm is able to search outwards from the current nodes starting with the adjacent nodes which have configuration-states 'closest' to the configuration-state of the goal node (i.e., gaits which will move the robot closer to the goal are given priority in the graph search). Finally, $f(h) = g(n) + h(n)$.

Each node in the graph contains two representations of the position and orientation of the robot: The first is the discretized configuration state, the position is discretized into 25cm blocks and the orientation is discretized into one of 24 axis-aligned orientations. Maintaining the discretized representation simplifies the task of marking a node as visited (i.e., a node will not be searched if it is fewer than 25cm and 45° from a

previously searched node). The second representation is the actual position and orientation reached by executing one of the gaits. This representation is required because each gait in the alphabet includes some position and orientation error. Using the 'ideal' goal configuration-state for each gait in the path planner results in the robot drifting off course. By using the actual final configuration-state for each gait the path planner can more accurately generate a sequence of gaits which transits the vehicle through the environment. Under this implementation moving from one node to the next involves combining the actual position and orientation changes, then discretizing the achieved configuration-state to compare against the list of previously searched nodes.

5.2 Linking gaits into complex maneuvers

The key element of the application of the path planner to the generation of complex maneuvers is the ability to link the smaller gaits together to form longer gaits. For the system presented in this thesis this is accomplished by using a common leg joint-angle configuration which each of the smaller gaits begins and ends. This way any gait can lead into any other gait in the alphabet. This is by no means the only way to link the small gaits together, see the Chapter 5 Summary for a discussion of alternative approaches.

In the system presented here member gaits of the alphabet presented in Chapter 4 are linked via direct concatenation of their leg joint-angles. For example linking the heave-down gait (see Section 4.1.9) and the surge gait (see Section 4.1.3) generates the

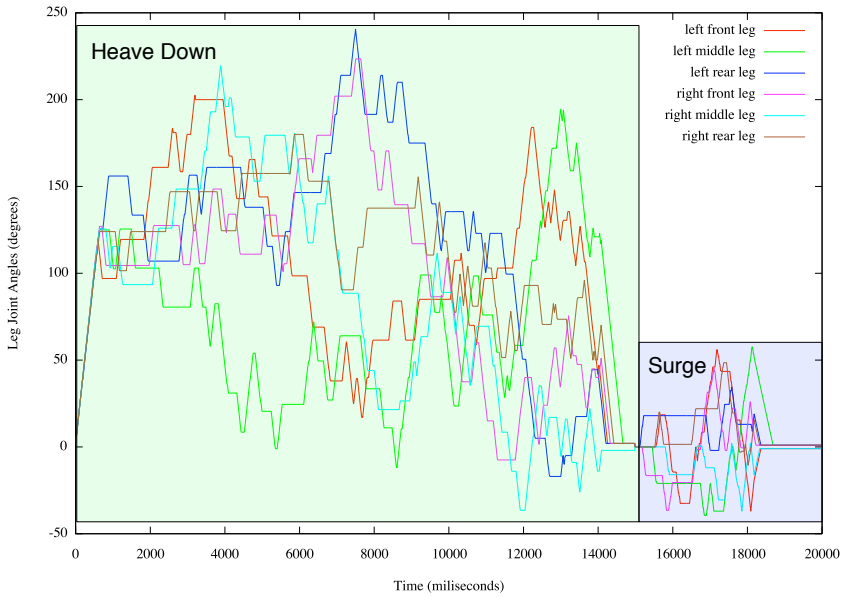
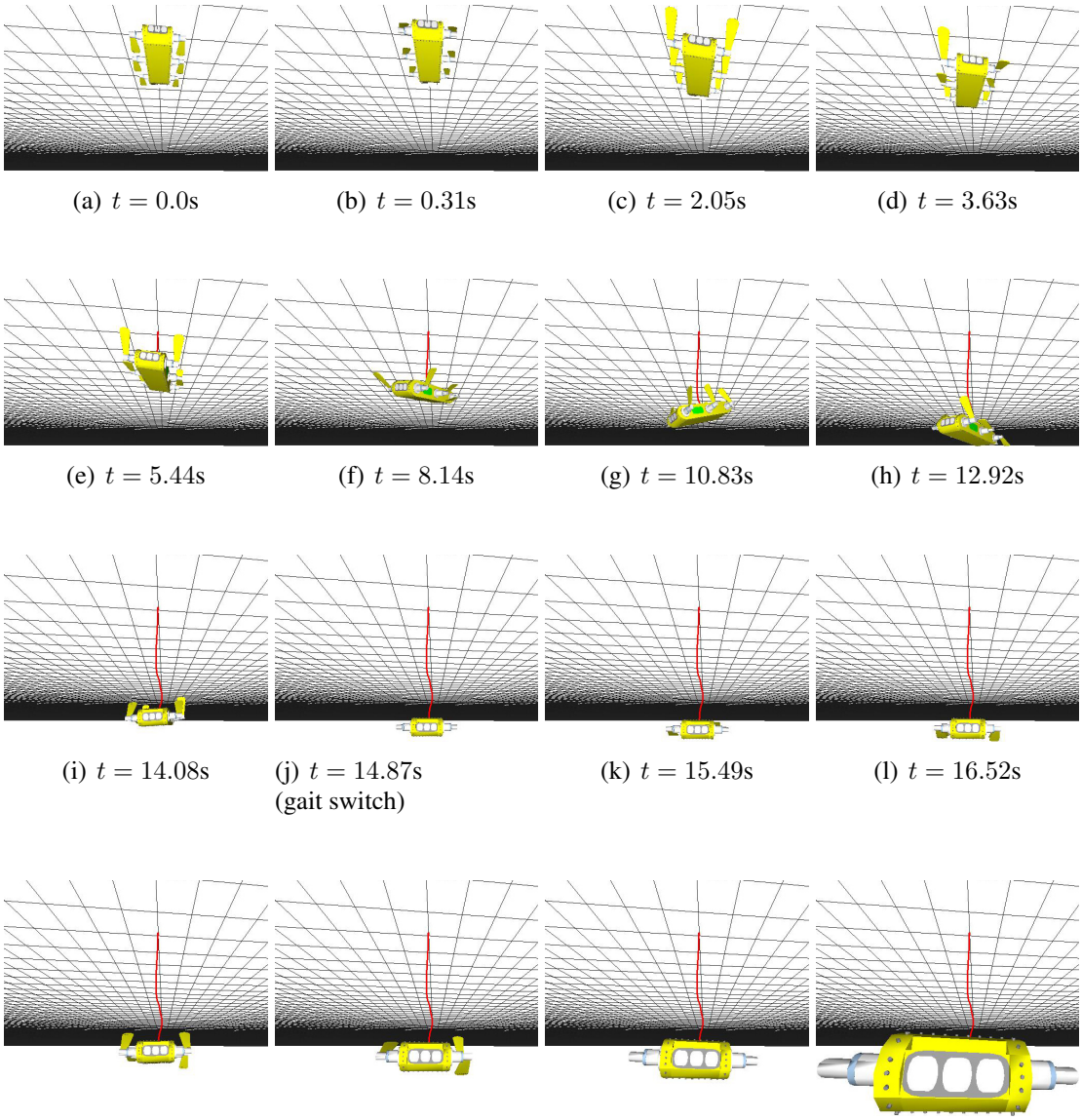


Figure 5.4: To link two gaits the leg joint-angle patterns are concatenated. In this case the surge follows the heave-down gait.

joint-angle profile in Figure 5.4. Once the gaits have been linked together the new gait can be run through the hydrodynamic simulator and applied to the robot (see Figure 5.5).

5.3 Path-planning example

Here a worked example is presented to explain how the A* algorithm is used to choose which gaits are required to move the vehicle between two configuration-states. To limit the search space for this example we will confine the available gaits to those that move the vehicle in the xz (horizontal) plane: surge, yaw-left and yaw-right. For a starting configuration state the vehicle is placed at the origin $(0, 0, 0)$ facing down the z-axis. The



(a) $t = 0.0$s (b) $t = 0.31$s (c) $t = 2.05$s (d) $t = 3.63$s

(e) $t = 5.44$s (f) $t = 8.14$s (g) $t = 10.83$s (h) $t = 12.92$s

(i) $t = 14.08$s (j) $t = 14.87$s (gait switch) (k) $t = 15.49$s (l) $t = 16.52$s

(m) $t = 17.13$s (n) $t = 18.1$s (o) $t = 18.61$s (p) $t = 20$s

Figure 5.5: A head-on view of the simulated AQUA robot executing a complex maneuver: a heave-down followed by a surge-forward. The frames are labelled with the time (in seconds) they were captured throughout the gait. The path of the robot is drawn in red.

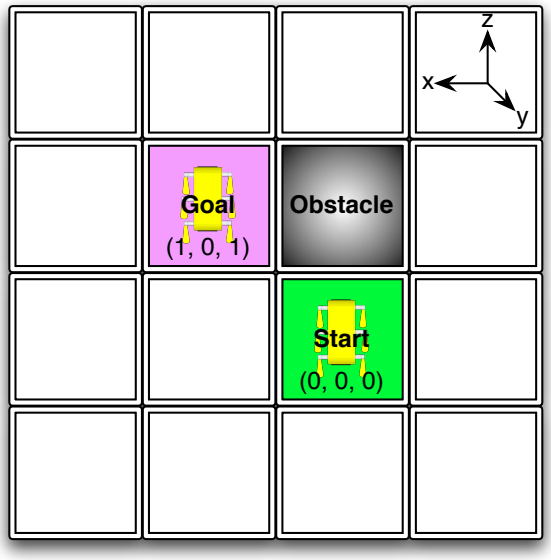


Figure 5.6: The start and goal configurations for the path planning example. The robot needs to move one meter to the left and one meter forward. The A* algorithm is used to decide which gaits are required to reach this goal.

goal configuration-state is 1m to the left and one meter forward $(1, 0, 1)$ with no rotation. Also, an obstacle is placed at $(0, 0, 1)$. Figure 5.6 illustrates the example scenario.

The first step is to add the start node to the *closed* list which marks this node as 'visited'. From the start node we examine all adjacent nodes for which gaits exist to transit the robot between the start node and the adjacent node (see Figure 5.7). In other words, we simulate the application of the three gaits (surge, yaw-left, and yaw-right) to the vehicle at the current position (the start position) and examine the nodes reached by the vehicle at the end of each gait. We examine these adjacent nodes and compute their g, h and f weights. First, we examine the node reached by the surge gait. Since it is blocked by an obstacle we add the node to the *blocked* list so it will not be visited again. Second,

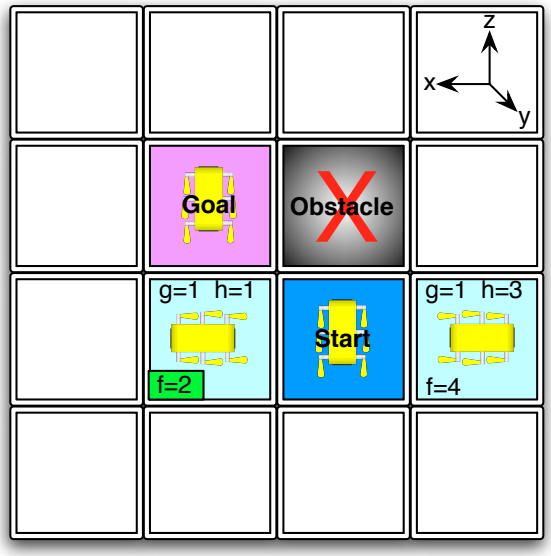


Figure 5.7: From the start node we examine the adjacent nodes and compute their g, h and f weights.

we calculate the weights of the node reached by the yaw-left gait and add the node to the search algorithm's *open* list. Third, we calculate the weights of the node reached by the yaw-right gait and add the node to the *open* list. Once all the adjacent nodes have been added to either the *blocked* or *closed* list we choose the next node to examine by selecting the node with the lowest f weight (highlighted in green in Figure 5.7), in this case the node representing the yaw-left gait. This node becomes the current node, we add it to the *closed* list and begin examining its adjacent nodes.

From the current node we examine the three adjacent nodes which can be reached by the yaw-left, yaw-right and surge gaits respectively and compute their weights (see Figure 5.8). Since none of the adjacent nodes are blocked they are all added to the *open*

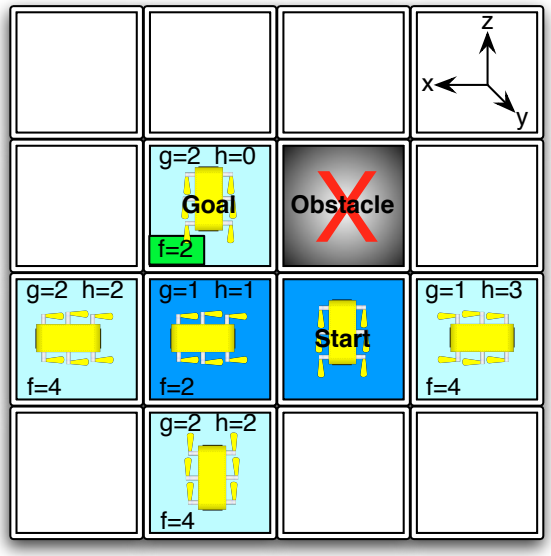


Figure 5.8: From the current node we examine the adjacent nodes and compute their g, h and f weights.

list. Again we choose the adjacent node with the lowest f weight to be our new current node and add it to the *closed* list (highlighted in green in Figure 5.8) (see Figure 5.9). In this case the node we just added was the goal configuration-state node so we can trace back to the start node and extract the path. In other words, we now know the shortest path from the start configuration-state to the goal configuration-state requires the execution of a yaw-left gait and then a yaw-right gait. These gaits can be linked together and then applied to the robot in the hydrodynamic simulator (see Figure 5.10).

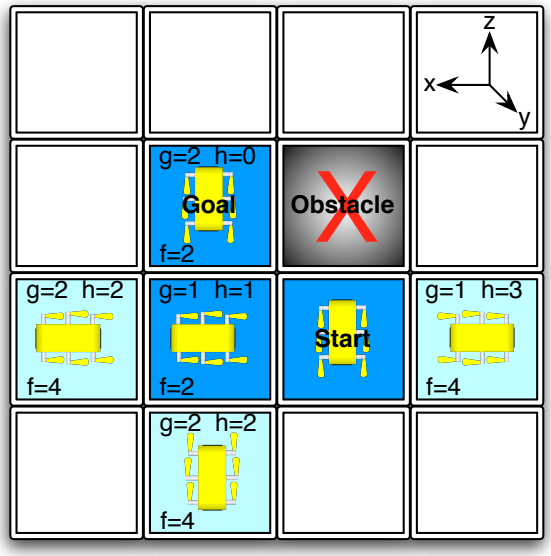
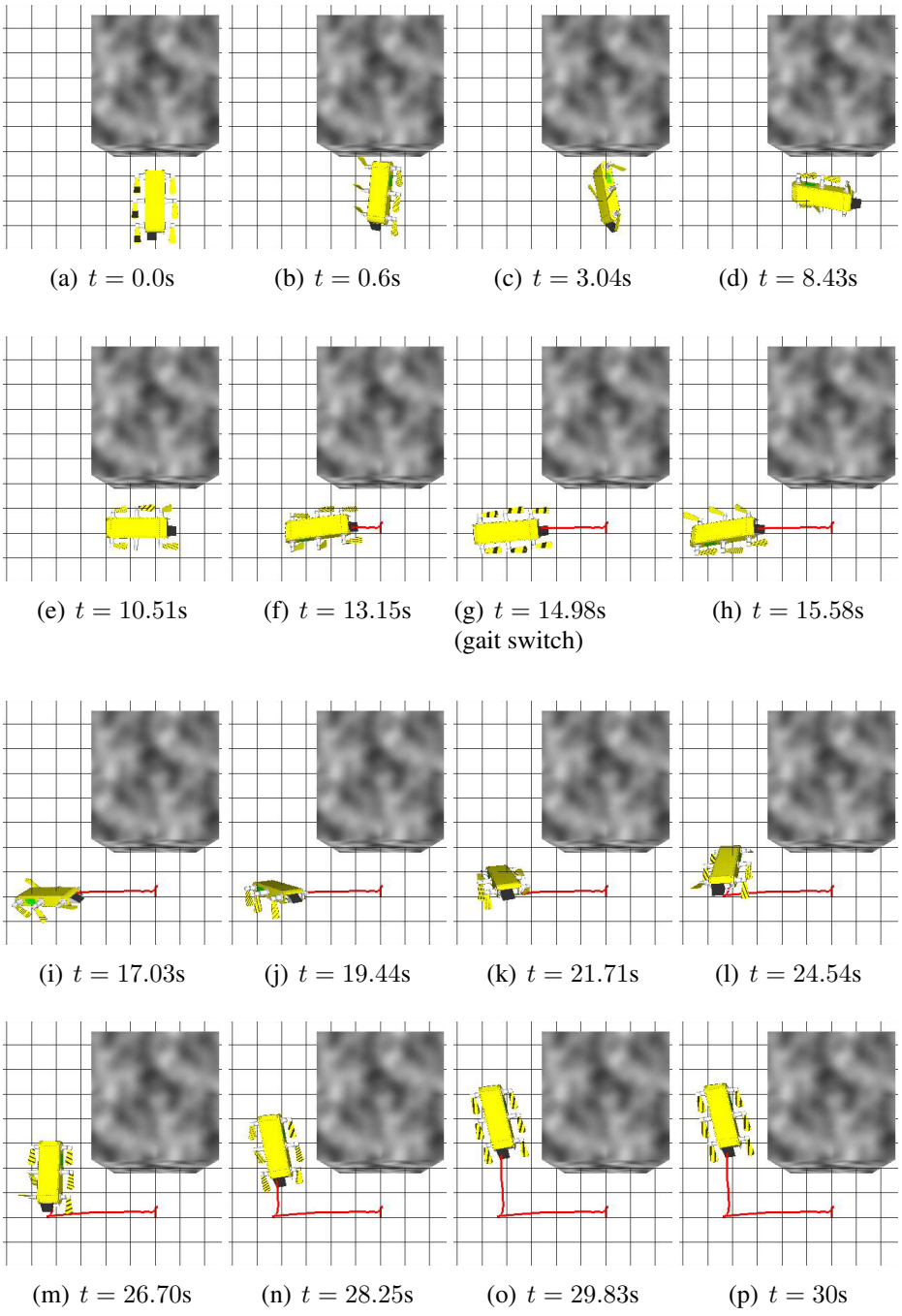


Figure 5.9: Once the goal node is added to the *closed* list we have found the fewest number of gaits required to transit the robot between the start and goal configuration-states.

5.4 Dynamics constraints and considerations

Though the above example was simplified, the real path-planner is slightly more complicated by the need to match the vehicle dynamics required by each gait. Many of the gaits in the alphabet have initial and final linear velocity requirements of 0.1m/s forward. Because of these constraints almost any gait can be linked to any other gait. There is however, one special circumstance which must be accounted for: since initially the vehicle is at rest the first step in the path planner must be to apply the accelerate gait to bring the vehicle's forward velocity up to 0.1m/s. At the moment, the only accelerate gait in the alphabet also moves the vehicle forward 1m. The drawback to this technique is that



(a) $t = 0.0$s (b) $t = 0.6$s (c) $t = 3.04$s (d) $t = 8.43$s

(e) $t = 10.51$s (f) $t = 13.15$s (g) $t = 14.98$s (gait switch) (h) $t = 15.58$s

(i) $t = 17.03$s (j) $t = 19.44$s (k) $t = 21.71$s (l) $t = 24.54$s

(m) $t = 26.70$s (n) $t = 28.25$s (o) $t = 29.83$s (p) $t = 30$s

Figure 5.10: A top-down view of the simulated AQUA robot executing the gaits suggested by the path-planner: a yaw-left gait followed by a yaw-right gait. The frames are labelled with the time (in seconds) they were captured throughout the gait. The path of the robot is drawn in red.

if there is an obstacle 1m in front of the vehicle's initial position (as in the above example) the path planner will be unable to find a path. This shortcoming could be avoided by synthesizing several accelerate gaits which move the vehicle in different directions. Also, since it is usually desirable to have the vehicle come to rest at the goal position, a decelerate gait also appears as a member-gait of the alphabet. The path-planner automatically adds the decelerate gait to the end of the sequence to make the vehicle come to rest at the end of the gait execution.

5.5 Summary

Given initial and goal configuration-states the path planner presented here will return a list of the gaits (from the gait alphabet presented in Chapter 4) which – when applied to the vehicle – will move the robot along an obstacle free path (if one exists) between the start and goal configuration-states that minimizes the given cost function. The path planner attempts to find an optimal path between the start and goal positions by embedding the configuration-states reachable via the alphabet of gaits into a connected graph where adjacent nodes represent adjacent configuration-states for which a gait exists to transit the robot between the two states. Extracting the required sequence of gaits is formulated as a graph-search problem for which the A* algorithm is well suited. Once a list of required gaits has been extracted from the path the gaits can be concatenated since their initial and final leg angles must match.

It is important to note that having a single common leg configuration through which all gaits begin and end is by no means the only way to affect gait transitions. One possible solution would be for more gaits to be generated and added to the alphabet which begin and end in other leg configurations. For example, to link a surge gait to a heave gait, a new surge gait could be generated which ends with the six legs pointing down so that when the heave gait begins any leg motion would thrust the robot up towards the goal configuration-state.

Another possible solution would be to use the system presented in this thesis to generate 'transition gaits'. The simulated annealing engine could be used to synthesize small gaits which would re-orient the legs from the final joint-angle configuration of one gait to the initial joint-angle configuration of the next gait, without introducing any unwanted motion. Under this system the gait alphabet would include gaits such as 'link-surge-to-heave-up' and 'link-yaw-left-to-stop'.

The main drawbacks of this path-planning and gait sequencing technique are related to long-distance path-planning, vehicle dynamics and numerical error in the simulation. Since this technique discretizes the environment into 25cm grids it would be computationally prohibitive to synthesize a gait that moves the robot across the Atlantic Ocean from New York to London. For the motion synthesis system presented here the environment must be computationally tractable, though as computing power increases the size of the representable environment grows as well.

The second drawback of the path-planner presented here is its incomplete model of the vehicle's dynamics. Most of the gaits in the alphabet are generated with an initial velocity of 0.1m/s and most of the gaits are generated with a *goal* velocity of 0.1m/s. However, just as the generated gaits contain some position and orientation error, so do they contain linear and angular velocity error. This results in gaits with *final* linear velocities not quite equal to 0.1m/s. As these gaits are linked by the path-planner the final linear velocity of one gait becomes the initial linear velocity of the next, resulting in gaits which begin with initial velocities other than 0.1m/s instead of exactly 0.1m/s (the way the gaits were generated). This inconsistency can cause the linked gaits to drive the robot off course since the initial conditions of the gait are not exactly met.

A third drawback to this approach is numerical error introduced in several places throughout the motion synthesis pipeline described in this thesis. The hydrodynamic simulator will introduce error to the system through integration of the forces acting on the vehicle. As the vehicle states and gaits are passed through the motion synthesis pipeline the values are rounded to six-decimal place floating point numbers. This round-off error will also contribute to the overall error of the system. As gaits are linked together these sources of error accrue and can cause the robot to deviate from its intended course.

There are several techniques which could be used to mitigate the configuration-state error introduced by numeric and dynamic error: A reactive algorithm could be developed to correct for small configuration-state errors before the vehicle drifts too far off

course, similar to the way a monopod robot attempts to remain balanced over its support column [35]. Another possible solution would be to use the hydrodynamic simulator to test the gait chains inside the path planner at every step. This would eliminate the error due to inconsistent dynamics as the configuration-state reached by the current gait chain would exactly match the output of the simulator (since they would be one and the same). This solution would come at the cost of execution speed.

The next chapter presents several examples of the combination of the gait alphabet and the path-planner moving the simulated AQUA vehicle through virtual environments.

Chapter 6 Experimental validation

This chapter presents several experiments run using the automatic motion synthesis system to move the virtual AQUA vehicle through different obstacle-strewn environments.

6.1 Experiment 1

The goal of this experiment is to have the motion synthesis system plan a path and combine the gaits required to move the robot around a U-shaped obstacle with a 180° yaw (see Figure 6.1). The path planner generates a list of the the minimum number of gaits required to move the robot from the start configuration to the goal configuration:

- Accelerate
- Heave up
- Surge
- Roll right
- Roll right
- Pitch up
- Heave down

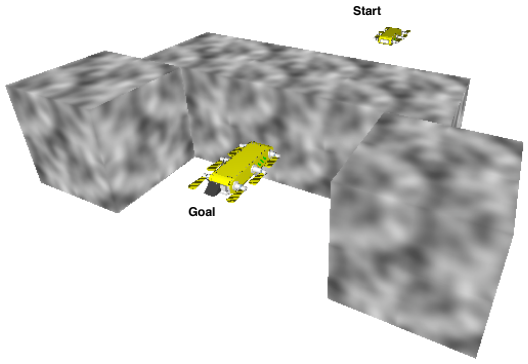


Figure 6.1: The goal of this experiment is to generate the motion required to have the robot maneuver around the obstacle and come to rest facing back towards the start position.

- Pitch up
- Decelerate

Once these nine gaits are combined, they are applied to the vehicle through the hydrodynamic simulator. This results in the robot swimming around the obstacles to the goal configuration (see Figure 6.2).

In this scenario the vehicle came to rest 31cm and 10° from the goal configuration.

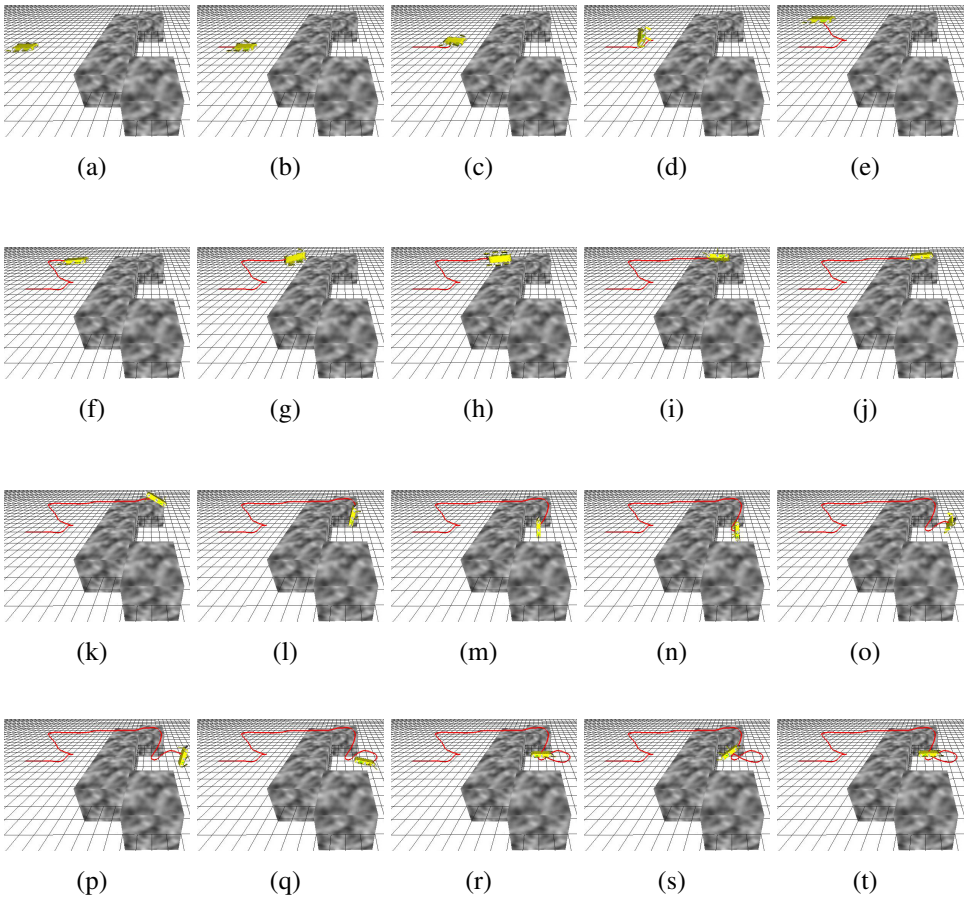
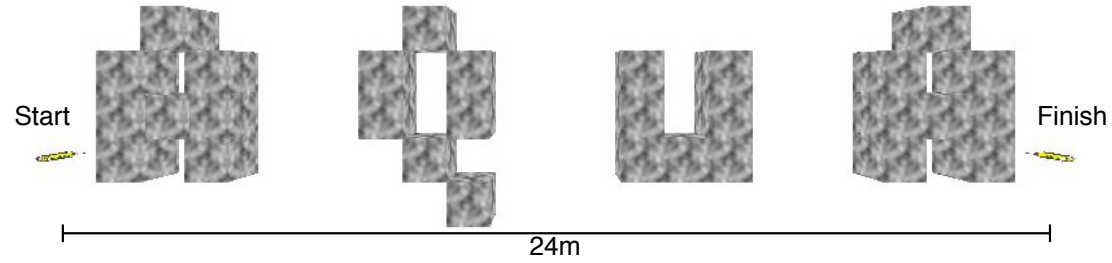
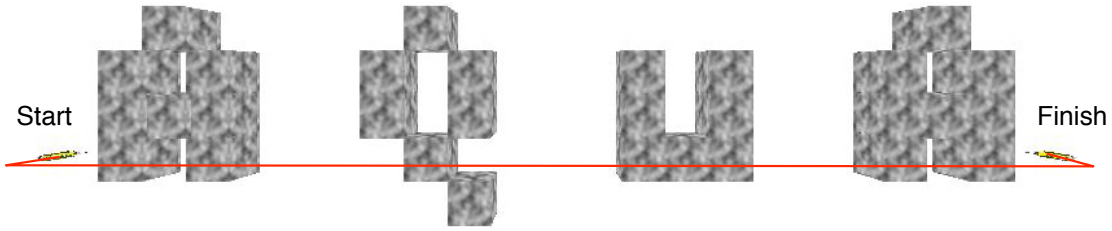


Figure 6.2: A top-down view of the simulated AQUA robot executing the synthesized gaits necessary to complete Experiment 1. The path of the robot is drawn in red.



(a)



(b)

Figure 6.3: The setup for Experiment 2. (a) The start and end configuration-states for this experiment. (b) The expected path for the robot.

6.2 Experiment 2

The goal of this experiment is to demonstrate how the system handles motion synthesis for large motion (i.e., long gait chains). The task is to generate the gait required to move the robot 24m around several obstacles, perform a 180° yaw and come to rest. The initial and final positions are shown in Figure 6.3(a) and the ideal path is shown in Figure 6.3(b).

The path planner generates a list of the the minimum number of gaits required to move the robot from the start configuration to the goal configuration:

- Accelerate
- Roll right
- 19 $\times$ Heave down

- Pitch down
- Roll right
- Roll right
- Surge
- Roll right
- Heave down
- Yaw Left
- Decelerate

Once these 29 gaits are combined, they are applied to the vehicle through the hydrodynamic simulator. This results in the robot swimming around the obstacles to the goal configuration (see Figure 6.4, and top-down view Figure 6.5). In this scenario the vehicle came to rest 1.85m and 14° from the goal configuration.

6.3 Summary

In this chapter several experiments were run to determine the strengths and weaknesses of the motion synthesis system developed in this thesis. Each experiment had the path planner select (from the gait alphabet) the the minimum cost function that would carry the robot between the start and goal configuration-states. These gaits were concatenated together and applied to the vehicle in the hydrodynamic simulator to assess how the gait-chains perform.

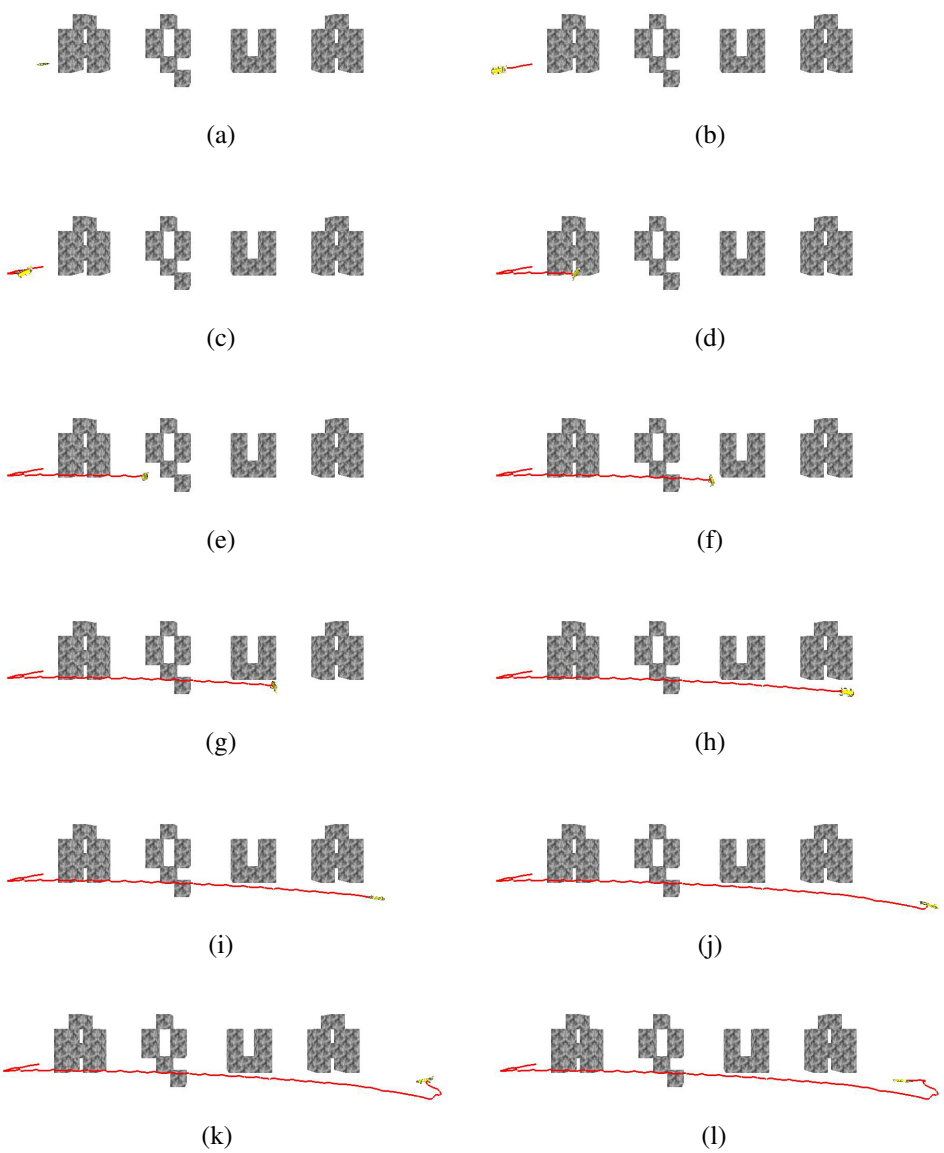


Figure 6.4: A side-on view of the simulated AQUA robot executing the synthesized gait generated for Experiment 2. The path of the robot is drawn in red.

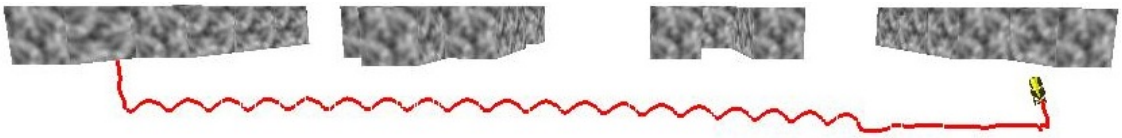


Figure 6.5: A top-down view of the simulated AQUA vehicle executing the gait synthesized in Experiment 2. The robot's path is drawn in red.

Clearly the dynamic and numerical error discussed in the previous chapter have a drift effect on longer gait chains. As the gait chains lengthen more error is accumulated, pushing the vehicle off-course. This error accumulation will always be a problem when 'perfect' gaits are difficult (if not impossible) to synthesize and you have a discretized state-space.

The best way to mitigate this error is to augment the system with online simultaneous localization and mapping (SLAM) capabilities [20] which would construct a map of the underwater environment and help provide odometry for use in trajectory correction. SLAM accompanied by an on-line path-planner would be capable of executing course correction and eliminate many of the shortcomings assocaited with the technique presented here.

The next chapter provides a summary of the work done along with future work which would refine and validate the techniques used in this motion synthesis system.

Chapter 7 Summary and future work

This thesis describes the development of a system for automatically generating complex underwater maneuvers for legged underwater robots and applies the system to the problem of motion generation for a simulated version of the AQUA amphibious hexapod. The maneuver generation system consists of two sub-components: First, a simulated annealing engine capable of automatically generating a short duration gait given constraints on the state of the vehicle (position, orientation, linear and angular velocity) both before and after the gait execution is used to generate an alphabet of maneuvers. The simulated annealing engine generates the required joint-angle patterns which transits the vehicle between the start and goal configurations. If two gaits are generated such that the higher order derivatives of their start and end configurations match then the gaits can be joined together to generate more complex motions. Second, given a start and end state in a (possibly obstacle ridden) environment, determining which members of the gait alphabet should be combined to transit the robot from a start configuration-state to a goal configuration-state while avoiding the obstacles is addressed using classical

path planning techniques. By combining these two systems leg motions can be automatically generated to move the AQUA amphibious hexapod through an obstacle scattered environment. Simulation of the AQUA vehicle through a simple hydrodynamic model demonstrates the effectiveness and generality of the approach.

There are several areas in this work which would benefit from further research. The most obvious of which is validation of the gait synthesis system on the real AQUA vehicle. Without access to the real robot and due to the lack of empirical information regarding the dynamics of the vehicle, this work relies on a simplified hydrodynamics model for the robot. This means that the gaits generated by the system presented here will likely need to be retuned in order to apply them to the real vehicle. However, there are many ways in which this system could be adapted to the real vehicle, all of which involve ground-truth tracking of the robot's position and orientation (as well as higher order derivatives with respect to time). For instance, in conjunction with several cameras to track the vehicle's position, and the vehicle's on-board inertial measurement unit (IMU) to track orientation, the approach could be used generate gaits for the real vehicle operating in open water.

Another area that would benefit from further research is correcting the configuration-state error over time. In the current implementation, synthesizing motions requiring many gaits chained together result in the small configuration-state errors associated with each small gait accruing as the chain becomes longer. This results in the vehicle drifting

off its prescribed path and potentially into obstacles. This drift is caused by two factors. First, the dynamics of the vehicle are not fully accounted for in the path-planner. Second, numerical error in the simulator and round-off error in other parts of the motion-synthesis pipeline can also cause position and orientation error. This error requires an on-line mechanism to help keep the vehicle on the correct path.

Simulated annealing was chosen as the heuristic engine used to drive search process of the motion synthesis system presented in this thesis. There are other heuristic search techniques which could be used instead. For example, it would be fairly simple to replace the current simulated annealing engine with a genetic algorithm (GA). GAs have been successfully used in gait synthesis and and other optimization search tasks and they would be good fit for this gait synthesis system as well. Neural networks would be a more complicated alternative due to the challenge of mapping the gaits to the levels of neurons. However like GAs, neural networks have found success in gait synthesis and more general AI applications, and the system presented here could certainly benefit from investigation into the application of neural networks or other search-based heuristics.

The system described in this thesis relies on a fairly simple hydrodynamic model for the virtual AQUA underwater vehicle to evaluate the effect of each gait as it applies to the robot's configuration-state. Though the simple hydrodynamic model used meets the needs of this work, there is certainly room for a more accurate (and complex) hydrodynamic model. Some research has gone into developing a more complete hydrodynamic

model for AQUA (see [14]). That work was primarily concerned with the thrust forces generated by the vehicles legs and did not include complicated fluid dynamic properties such as vorticies or an accurate model of how drag effects the vehicle's body. In practice, the development of a high fidelity hydrodynamic model would likely involve significant experimental work with the vehicle mounted on a force-torque stand in the presence of different velocity water columns as well as requiring an accurate model of how the legs and body shed vorticies and produce drag. But this is certainly an interesting direction for future work.

Another research endeavour from which this system would benefit would be the inclusion of the same sensory inputs which are available to the real AQUA vehicle. For example, the AQUA robot has several cameras and an inertial measurement unit (IMU). Information (real or simulated) from these sensors could help the system generate smoother gaits by seeking a smooth inertial profile, and – combined with the egomotion tracking presented in [20] – could help establish the underwater egomotion of the vehicle thereby providing a mechanism to identify and adjust for any course deviation during the gait execution.

Finally, the system presented here would benefit from a path planner capable of multiple waypoint navigation. This would allow the vehicle to leave from a given point travel around the environment (e.g., reef or fish stock surveillance) and return to the start position. This is a key feature of the site acquisition and scene reinspection task for which

AQUA was built, so closing this loop is of great interest.

Appendix A Simulator details

The motion synthesis system presented in this thesis uses a hydrodynamic simulator to determine how each synthesized gait affects the motion of the simulated legged vehicle. The process of simulating the vehicle's motion includes developing a model for both the body of the vehicle and how forces are applied to the vehicle. The body of the virtual AQUA vehicle is modelled as a 19kg solid rectangular prism (see Figure A.1) which is neutrally buoyant with both centre of mass and centre of buoyancy equal to the geometric centre. The body of the vehicle moves as a result of forces applied on it. In this case the legs apply thrust forces on the vehicle and the reactionary force of the water pushing back against the vehicle applies drag forces. The legs apply instantaneous thrust forces proportional to their rotational velocity. For each leg i there is a force f_i:

$$f_i \propto \dot{\theta}_i.$$

The direction of the thrust force is equivalent to the vector created by projecting the leg forward through the hip attachment joint (see Figure A.2). Each of these six forces (one for each leg) are applied to the vehicle at their respective hip joints at each time-step.

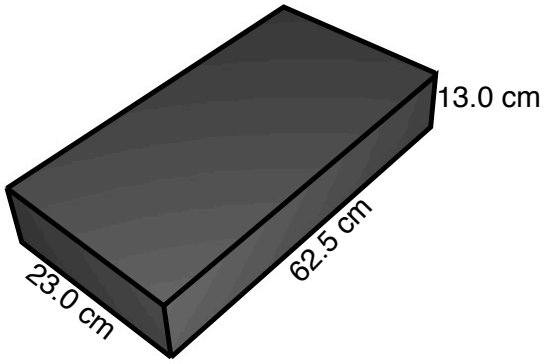


Figure A.1: The body of the AQUA vehicle is modelled as a solid rectangular prism in the hydrodynamic simulator.

The drag forces are a result of the water pushing back on the vehicle as it moves. For each face of the body F_i both linear and angular drag are calculated. Linear drag is a force vector acting in the opposite direction of the linear velocity vector. The magnitude of this force depends on how many faces the water is pushing against as the vehicle moves, that is, faces for which the normal forms a positive dot-product with the linear velocity vector. So the linear drag force is:

$$f_{ld} \propto \sum_{i=0}^{N} \begin{cases} F_i^n \cdot v \leq 0 & 0, \\ F_i^n \cdot v > 0 & F_i^A F_i^n \cdot v \end{cases}$$

where v is the linear velocity, F_i^A is the area of face i and F_i^n is the normal of face i.

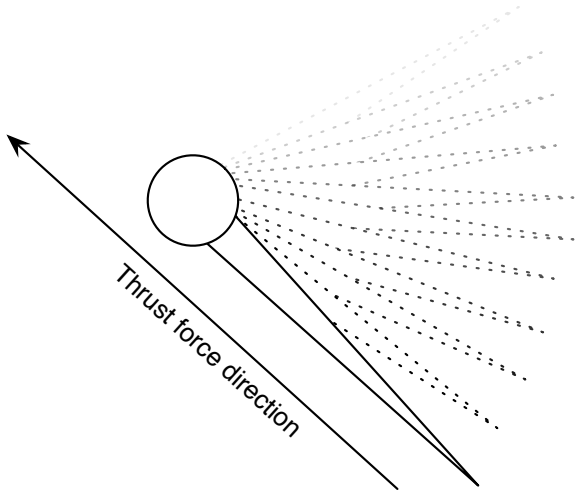


Figure A.2: Each leg applies a thrust force on the vehicle proportional to its rotational velocity.

The linear drag force is applied to the centre of mass of the vehicle. The angular drag is the result of the fluid pushing back on the vehicle as it rotates. Each face for which the normal has a non-zero dot-product with the angular velocity vector contributes to the scale of the drag torque force. This torque force is applied in the opposite direction of the angular velocity vector. The magnitude of the angular drag force is defined as:

$$f_{ad} \propto \sum_{i=1}^{N} (\frac{1}{2} F_i^a F_i^n \cdot \dot{\theta}).$$

At each time-step, these forces are calculated and applied to the model of the vehicle's body. The Open Dynamics Engine (ODE)[2] is used to integrate the forces applied on the body. Given the forces as a function of time, ODE returns the state of the vehicle

[2]http://www.ode.org/

(position, orientation, linear velocity, angular velocity). So at each time-step leg motions are converted into thrust forces, the current linear and and angular velocity are converted into drag forces, and the physics engine integrates these forces and returns the updated state of the vehicle.

Bibliography

[1] H. Abdi. A neural network primer. *Journal of Biological Systems*, 2(3):247–283, 1994.

[2] J. D. Anderson. *Fundamentnals of Aerodynamics*. McGraw-Hill series in aeronautical and aerospace engineering. McGraw-Hill, Boston, 3rd edition, 2001.

[3] D. M. Bourg. *Physics for Game Developers*. O'Reilly & Associates, Inc., January 2002.

[4] A. Brian. *Ship Hydrostatics and Stability*. Butterworth-Heinemann, Oxford, 2003.

[5] J. Cappelletto, P. Estevez, W. Medina, L. Fermin, J. M. Bogado, J. C. Grieco, and G. Fernandez-Lopez. Gait synthesis and modulation for quadruped robot locomotion using a simple feed-forward network. In *Artificial Intelligence and Soft Computing ICAISC 2006*, pages 731–739. Springer Berlin / Heidelberg, 2006.

[6] J. Conradt and P. Varshavskaya. Distributed central pattern generator control for a serpentine robot. In *Joint Int. Conf. on Artif. Neural Networks and Neural Information*, pages 338–341, June 2003.

[7] A. Crespi and A. J. Ijspeert. AmphiBot II: an amphibious snake robot that crawls and swims using a central pattern generator. In *Proc. 9th Int. Conf. on Climbing and Walking Robots (CLAWAR 2006)*, pages 19–27, September 2006.

[8] E. G. Drucker. The use of gait transition speed in comparative studies of fish locomotion. *Amer. Zool.*, 36(6):555–566, 1996.

[9] G. Dudek, P. Giguere, C. Prahacs, S. Saunderson, J. Sattar, L. Torres-Mendez, M. Jenkin, A. German, A. Hogue, A. Ripsman, J. Zacher, E. Milios, H. Liu, P. Zhang, M. Buehler, and C. Georgiades. AQUA: An amphibious autonomous robot. *Computer*, 40(1):46–53, Jan 2007.

[10] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*. Cambridge University Press, Cambridge, 1st edition, 2000.

[11] R. P. Feynman, R. B. Leighton, and M. Sands. *The Feynman Lectures on Physics, Mainly Mechanics, Radiation, and Heat*. Addison-Wesly Publishing Company, 1963.

[12] P. M. Fishbane, S. G. Gasiorowicz, and S. T. Thornton. *Physics for Scientists and Engineers*. Pearson Prentice Hall. Upper Saddle River, NJ, third edition edition, 2005.

[13] M. Gao and J. Tian. Path planning for mobile robot based on improved simulated annealing artificial neural network. In *Third IEEE Int. Conf. Natural Computation*, volume 3, pages 8–12, August 2007.

[14] C. Georgiades. Simulation and Control of an Underwater Hexapod Robot. Master's thesis, McGill University, Febuary 2005.

[15] C. Georgiades, A. Hogue, H. Liu, A. Ripsman, R. Sim, L. A. Torres, P. Zhang, C. Prahacs, M. Buehler, G. Dudek, M. Jenkin, and E. Milios. AQUA: an aquatic walking robot. In *The 7th Unmanned Underwater Vehicle Showcase, Proceedings UUVS'04*, 2004.

[16] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[17] T. Gomi and K. Ide. Evolution of gaits of a legged robot. In *IEEE Int. Conf. on Fuzzy Systems*, volume 1, pages 159–164, May 1998.

[18] M. R. Heinen and F. S. Osorio. Applying genetic algorithms to control gait of physically based simulated robots. *IEEE Congress on Evolutionary Computation (CEC)*, pages 1823–1830, July 2006.

[19] J. K. Hodgins. Biped gait transitions. In *IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 2092–2097, Yorktown Heights, NY, April 1991. IEEE.

[20] A. Hogue. *SensorSLAM: an investigation into the use of sensor parameters within the SLAM framework*. PhD thesis, York University, Toronto, Ontario, Canada, 2008. In Preparation.

[21] P. Jantapremjit and D. Austin. Design of a modular self-reconfigurable robot. In *Australian Conf. on Robotics and Automation*, pages 38–43, 14-15 November 2001.

[22] J. JUANG. Fuzzy neural network approaches for robotic gait synthesis. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 30:594–601, August 2000.

[23] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[24] D. M. A. Lee and W. H. ElMaraghy. A neural network solution for bipedal gait synthesis. *Int. Joint Conf. on Neural Networks (IJCNN)*, 2:763–768, June 1992.

[25] M. Leslie. The man who stopped time. *Stanford Magazine*, May 2001.

[26] M. A. Lewis, A. H. Fagg, and G. A. Bekey. *Recent Trends in Mobile Robots*, chapter Genetic algorithms for gait synthesis in a hexapod robot, pages 317–331. World Scientific, New Jersey, 1993.

[27] J. Lin and S. Song. Modeling gait transitions of quadrupeds and their generalization with CMAC neural networks. In *IEEE Transactions on Systems, Man and Cybernetics*, volume 32, pages 177–189. IEEE, August 2002.

[28] M. M. Martinez, R. J. Full, and M. A. R. Koehl. Underwater punting by an intertidal crab: A novel gait revealed by the kinematics of pedestrian locomotion in air versus water. *The J. of Exp. Bio.*, 201:2609–2623, 1998.

[29] D. Murphy, J. Bott, W. Bryan, J. Coleman, D. Gage, and H. Nguyen. MSSMP: No Place to Hide. Proceedings AUVSI'97, Baltimore, MD, 3-6 June 1997.

[30] E. Muybridge. The Horse in motion. "Abe Edgington," owned by Leland Stanford; driven by C. Marvin, trotting at a 2:24 gait over the Palo Alto track, 15th June 1878, 1878.

[31] N. J. Nilsson. *Principles of Artificial Intelligence*. Tioga Publishing Company, Palo Alto, CA, 1980.

[32] G. A. V. Pai and S. Sreeram. Military target identification using simulated annealing. In *9th Int. Conf. on Neural Information Processing*, volume 4, pages 2064 – 2068. IEEE, November 2002.

[33] W. G. Price and R. E. D. Bishop. *The dynamics of ships : proceedings of a Royal Society discussion meeting held on 28 and 29 June 1990*. The Royal Society, London, June 1991.

[34] M. H. Raibert. Legged robots. *Commun. ACM*, 29(6):499–514, 1986.

[35] M. H. Raibert. *Legged Robots That Balance*. MIT Press, 1986.

[36] C. Ridderström. *Legged locomotion: Balance, Control and Tools — From Equation to Action*. PhD thesis, The Royal Inst. of Technology, 100 44 Stockholm, Sweden, MAY 2003.

[37] R. Rojas. *Neural Networks - A Systematic Introduction*. Springer-Verlag, Berlin, New-York, 1996.

[38] M. E. Rosheim. *Leonardo's Lost Robots*. Springer Verlag, 2006.

[39] U. Saranli, M. Buehler, and D. E. Koditschek. RHex: A simple and highly mobile hexapod robot. 20(7):616–631, July 2001.

[40] M. Sfakiotakis and D. P. Tsakiris. Neuromuscular control of reactive behaviors for undulatory robots. *Neurocomputing*, 70(10-12):1907–1913, 2007.

[41] R. F. Stengel. *Flight Dynamics*. Princeton University Press, Princeton, N.J., 2004.

[42] F. Tanaka and H. Suzuki. Dance interaction with qrio: a case study for non-boring interaction by using an entrainment ensemble model. In *IEEE Int. Workshop on Robot and Human Interactive Communication*, pages 419–424, September 2004.

[43] Z. Tang, C. Zhou, and Z. Sun. *Advances in Natural Computation*, volume 3611, chapter Humanoid Walking Gait Optimization Using GA-Based Neural Network, pages 252–261. Springer Berlin / Heidelberg, 2005.

[44] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, and L.-E. Jendro. Stanley, the robot that won the DARPA Grand Challenge. *Journal of Field Robotics*, 23(9):661–292, 2006.

[45] A. A. Transeth and K. Y. Pettersen. Developments in Snake Robot Modeling and Locomotion. In *Proceedings of the 9th International Conference on Contro, Automation, Robotics and Vision*, pages 1–8, 5-8 December 2006.

[46] K. Tsujita, M. Kawakami, and K. Tsuchiya. A study on optimal gait pattern of a quadruped locomotion robot. *IEEE Int. Conf. on Systems, Man and Cybernetics*, 1:745–749, October 2004.

[47] W. G. Walter. *The Living Brain*. Duckworth, London, 1953.

[48] H. W. Young and S. J. Phillips. A Novel Platform for Data Gathering. In *Proc. of the IEEE/OES Seventh Working Conference on Current Measurement Technology*. IEEE, 2003.

[49] Y. F. Zheng. A neural gait synthesizer for autonomous biped robots. In *Intelligent Robots and Systems '90. 'Towards a New Frontier of Applications', Proceedings. IROS '90. IEEE International Workshop on*, volume 2, pages 601–608. IEEE, July 1990.