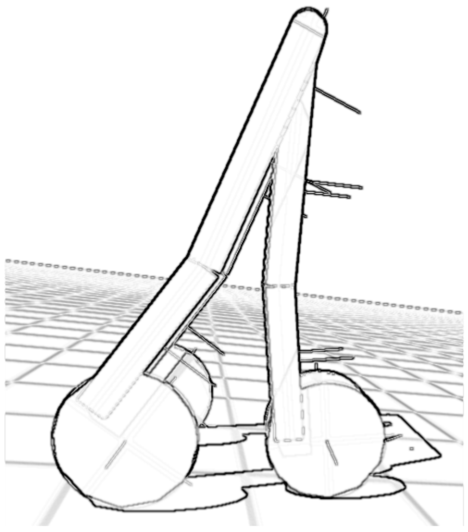
Computer Lab 5
Passive Walker

Mikica Kocic
miko0008@student.umu.se

2012-06-16



Mark what exercises you have done in this lab		*Points (by supervisor)*
Basic level (mandatory)	×	
Extra exercise #1: extended model	×	
Extra exercise #2: added sensors	×	
Extra exercise #3: an active walker	×	
Extra exercise #4: user controls	×	
Extra exercise #5: n/a		
The report is brief and to the point but still coherent and well structured.		
Late report		
Total points		

Tutors: **Martin Servin** and **John Nordberg**

1 Background

The purpose of this computer lab was to model and simulate a passive walker using an existing physics engine. Advanced level tasks were to extend the passive walker model, add sensors and motors, and implement a controlled active walker.

2 Implementation and Simulation

2.1 Theory

In 1987, McGeer started working on the concept of passive dynamic walking. Passive walking acts as a gait-like motion of a mechanism having no external actuation other than gravity. McGeer demonstrated by numerical and physical simulations models [1]–[4] that some mechanisms could achieve bipedal walking on small slopes without actuators or control systems. These passive systems can generate stable walking patterns just by gravitational effects, and, in addition, these motions are natural and energy-optimal.

2.2 Model

Since McGeer first studied the passive walk through simple models, different researchers have achieved additional insights in dynamics on these [5], [6], [9] and [10].

The analyzed passive dynamic walker in this lab is a planar walker model with knees based on [7] and [8]. The walker is shown on Figure 2.2–1, where its detailed dimensions are depicted in Appendix A.

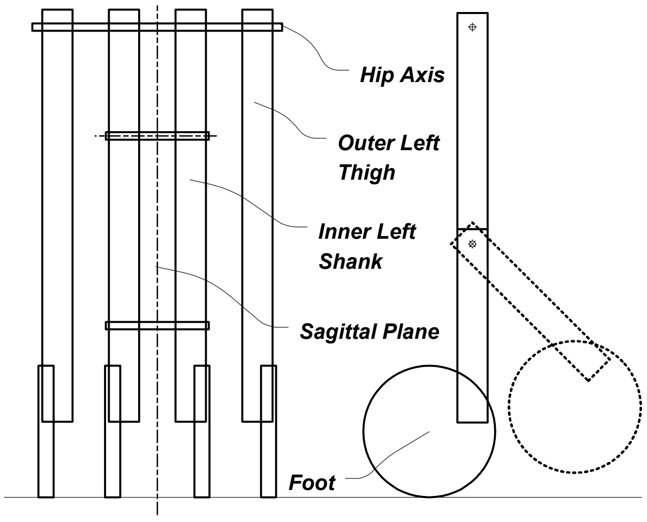


Figure 2.2–1: The schematic figure of the analyzed passive walker

The passive walker consists of four leg assemblies constrained to move in the sagittal plane to prevent lateral dynamics, namely two identical upper legs (thighs) and two identical lower legs (shanks). These leg assemblies are connected via hinge joints. The leg assemblies are named inner-left, inner-right, outer-left and outer-right, depending on their position to the sagittal plane. The regular contact between the walker and the plane is via circular feet. The mechanism, constrained only by the hip and knee joints, has six degrees of freedom. The thighs' orientations are given by the angle between inner and outer leg pairs on the hip axis. From the thighs to the shanks, two clockwise rotation angles are used. It is always assumed that the walker has at least one foot in contact with the ground. The basic model is extended using asymmetric mass distribution between upper and lower leg parts, and also by allowing negative knee angles, so the leg assemblies can take a concave shape.

2.3 Implementation

The passive walker model is implemented in C++ and it uses Bullet Physics library for physics simulations, and GLUT and OpenGL API for animations.

The core implementation consists of ~3k lines of code distributed in around 5 modules. The program documentation consists of ~1 lines of comment that follows javadoc convention. The resulting documentation can be generated from these comments using Doxygen tool.

The central part of the code is the `Machine` class, for which the include dependency graphs is shown on Figure 2.3–1. The good point to start browsing the code is `Main.cpp` file, which contains the main entry point.

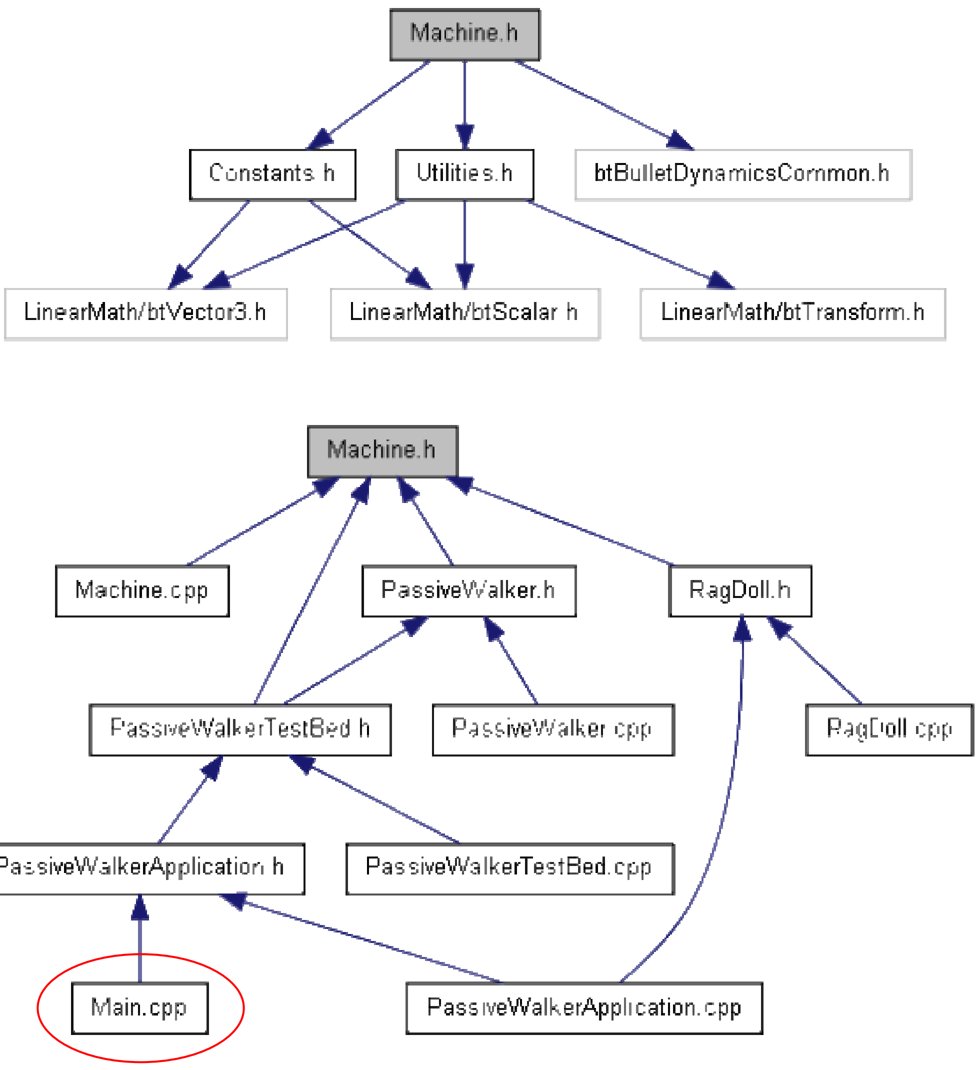


Figure 2.3–1: Include dependency graphs for `Mahine.h`

2.4 Compiling the Code

The code uses Bullet Physics (BT) library. The code has been tested against the BT library release v2.80 and linked with BT sub-libraries `BulletDynamics`, `BulletCollision` and `LinearMath`.

The animation part requires OpenGL libraries, which are usually distributed with an operating system, and the freeglut (GL Utility Toolkit) that can be downloaded from http://freeglut.sourceforge.net/

The default distribution contains binaries compiled with MS VS 2010 compiler. To rebuild binaries on Windows, use VS solution in folder `vs2010`.

To rebuild binaries on Linux, use `Makefile`; just type `make` on the command prompt. When compiling on Linux, you have to configure the location of the bullet physics library in `Makefile`. Olso, if you have non-standard location for freeglut (e.g. if it is not installed as a system package), you will also need to configure `GLUT_INC` and `GLUT_LIB` variables.

Here is a sample log showing the compilation process to get the stand-alone application on Linux:

```
mikica@triton:~/lab5$ make rebuild
 [C++  ]  obj/Constants.o
 [C++  ]  obj/Machine.o
 [C++  ]  obj/PassiveWalker.o
 [C++  ]  obj/PassiveWalkerTestBed.o
 [C++  ]  obj/PassiveWalkerApplication.o
 [C++  ]  obj/RagDoll.o
 [C++  ]  obj/Main.o
 [C++  ]  obj/OpenGL/DemoApplication.o
 [C++  ]  obj/OpenGL/GLShapeDrawer.o
 [C++  ]  obj/OpenGL/GLDebugDrawer.o
 [C++  ]  obj/OpenGL/GLDebugFont.o
 [LD   ]  bin/PassiveWalker
mikica@triton:~/lab5$
```

NOTE:

Please, use the windows executable that is distributed with this report to repeat the presented results.

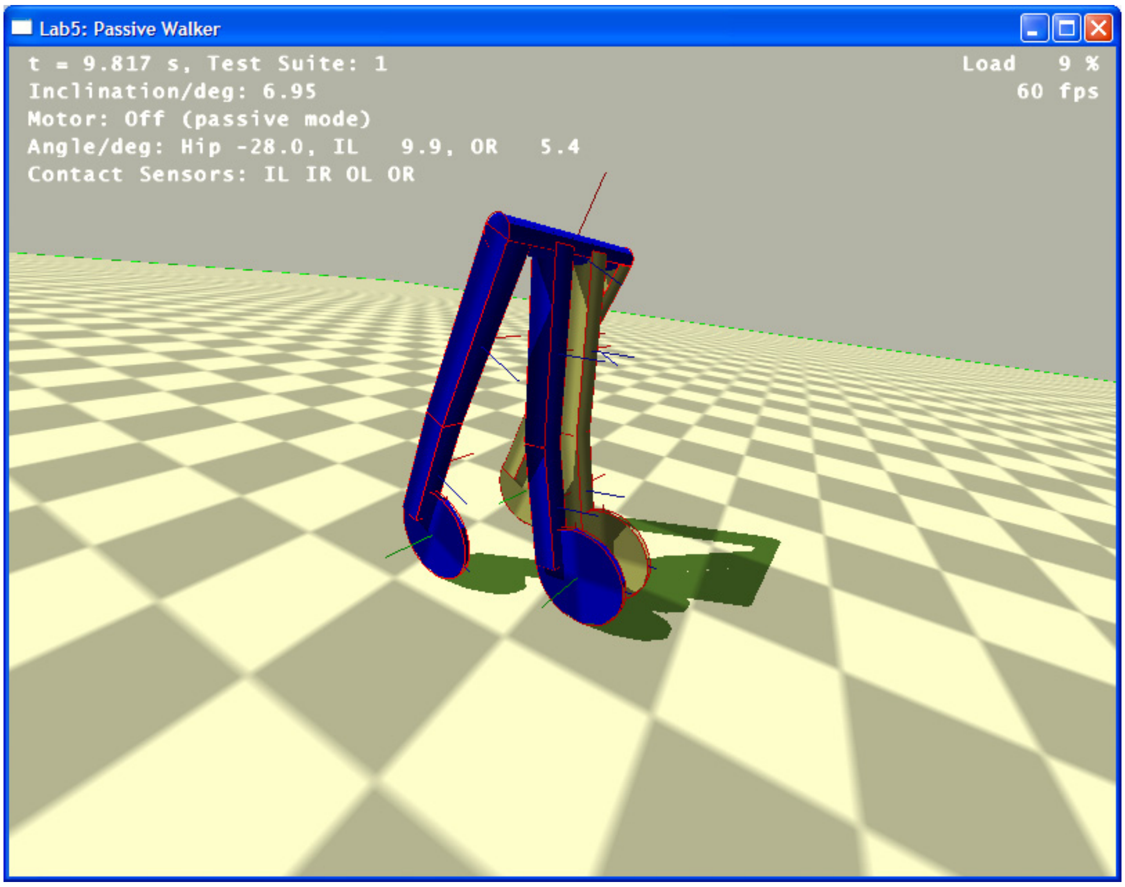
If you (re)compile the code, you will certainly need to provide new initial conditions!

To do this, select a desired test suite (by pressing `F1-F7`) and then press `F12` to start a Monte Carlo simulation. The application will gradually try to find longer and longer successful walks, starting with a minimum of 8-seconds duration goal. The results will be then written in '`config-`*`id`*`.txt`' file. After examining the resulting configuration file, you may pop-up the desired initial conditions to the front of the file.

More details on the initial conditions sensitivity can be found in the *Discussion* section.

2.5 Running Stand-Alone Simulations

To run a simulation, navigate to `bin` directory and start the `PassiveWalker` executable.
The gui should appear:



```
t = 9.817 s, Test Suite: 1                              Load   9 %
Inclination/deg: 6.95                                     60 fps
Motor: Off (passive mode)
Angle/deg: Hip -28.0, IL   9.9, OR   5.4
Contact Sensors: IL IR OL OR
```

Figure 2.5–1: Snapshot of the Lab5 PassiveWalker simulation window.

The application window is logically divided into two areas: central area that contains the simulated system with the passive walker and the ground plane, and top-left area that contains local time and various simulation parameters.

Beside the standard short-cut keys that are compatible with the Bullet GL Demo applications, the user can also use the following short-cut keys to control the passive walker simulation:

`F1, F2, … F8`	Start different predefined simulation test suite *id* = 1..8. ***Note:*** The initial conditions are loaded from the '`config-`*`id`*`.txt`' file.
`F12`	Starts the Monte Carlo simulation for the current test suite *id*.
`H`	Toggle displaying profiling statistics
`I`	Pause/continue simulation (toggle simulation execution)
`S or <enter>`	Simulation single-step; keep holding `<enter>` for slowed-down simulation
`W`	Toggle displaying objects as wire-frames vs. solid bodies

In active walker mode (after starting test-suite *id* 8 by pressing `F8`), one can control the active walker using the following short-cut keys:

`1`	Decrease hip angle limit parameter
`2`	Increase hip angle limit parameter
`3`	Decrease maximal leg angular velocity parameter
`4`	Increase maximal leg angular velocity parameter
`5`	Decrease angle difference between left and right knee motors parameter
`6`	Increase angle difference between left and right knee motors parameter

3 Results

3.1 Basic Level Tasks

The longest simulated walk was nine steps on an inclined ground plane with 6.95-degrees slope.
In this configuration, the passive walker's mass was 19 kg and its height was 0.688 m scaled 10 times i.e. the simulated walker's length was 6.88 m in the BT dynamics world. (This scaling was needed to meet the BT requirements not to have collision objects with sizes smaller than 0.1 units.)

The simulation snapshots of this walk are shown on Figure 3.1–1. The average step length was $d = 3.51$ m with the standard deviation of 0.25 m. This corresponds to the scaled length 0.35(3) m.

The average step time period was $T = 2.16$ s with the standard deviation of 0.04 s on the slope 6.95°, which means that the dimensionless step time period was $T / \sqrt{L / g} = 2.6$. This value is in agreement with the dimensionless time period 2.7 of the walker-2 as presented in [7] on the slope 6.7°**.**

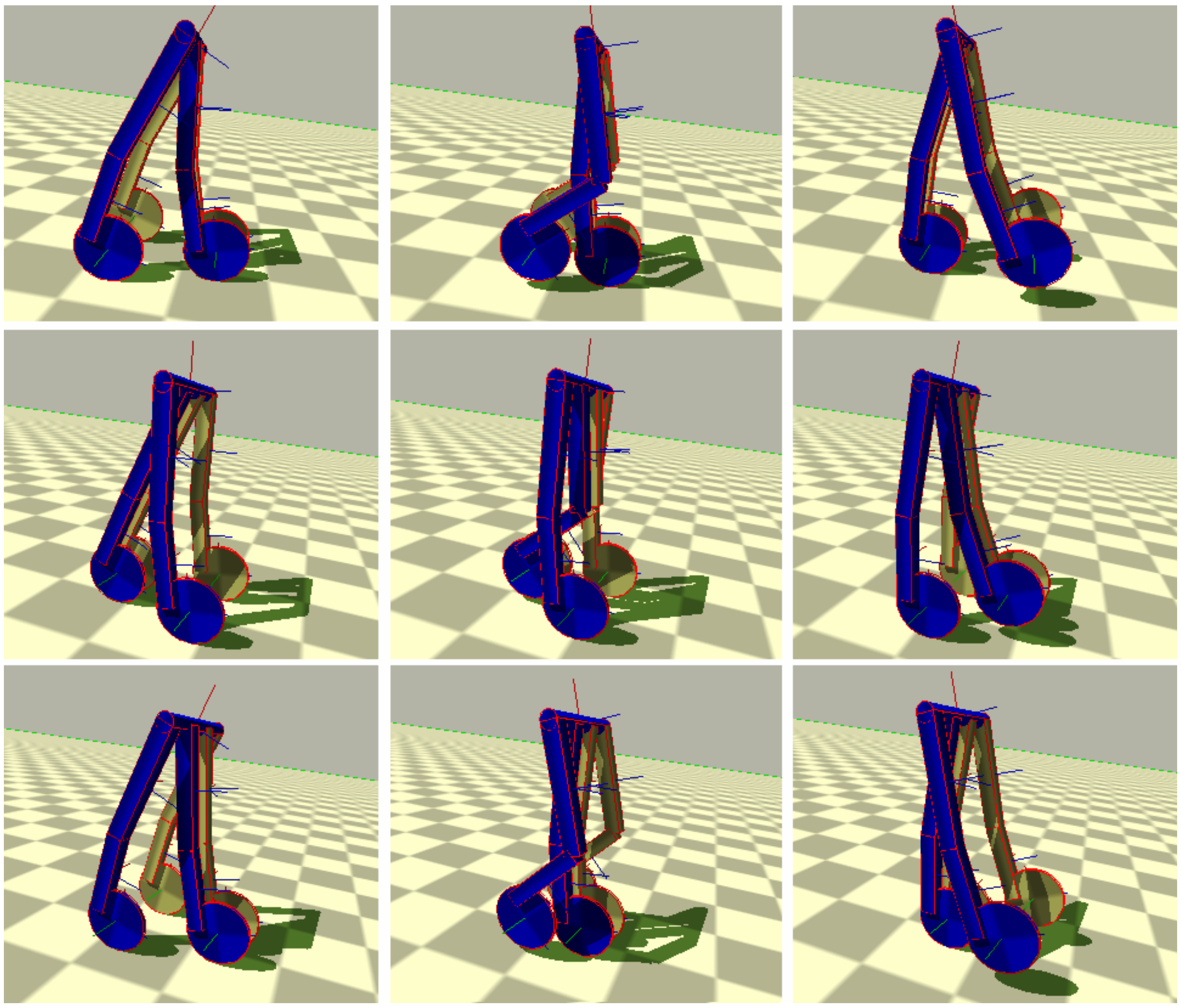


Figure 3.1–1: Snapshots of the passive walker simulation on an inclined plane with slope 6.95°

The diagram of the walking gait during the simulation is shown on Figure 3.1–2. Here, the time history of the hip and knee configuration characterizes the gait. This diagram, also referred to as an 'angle-angle' diagram, is essentially equivalent to the notion of a trajectory in the configuration space.

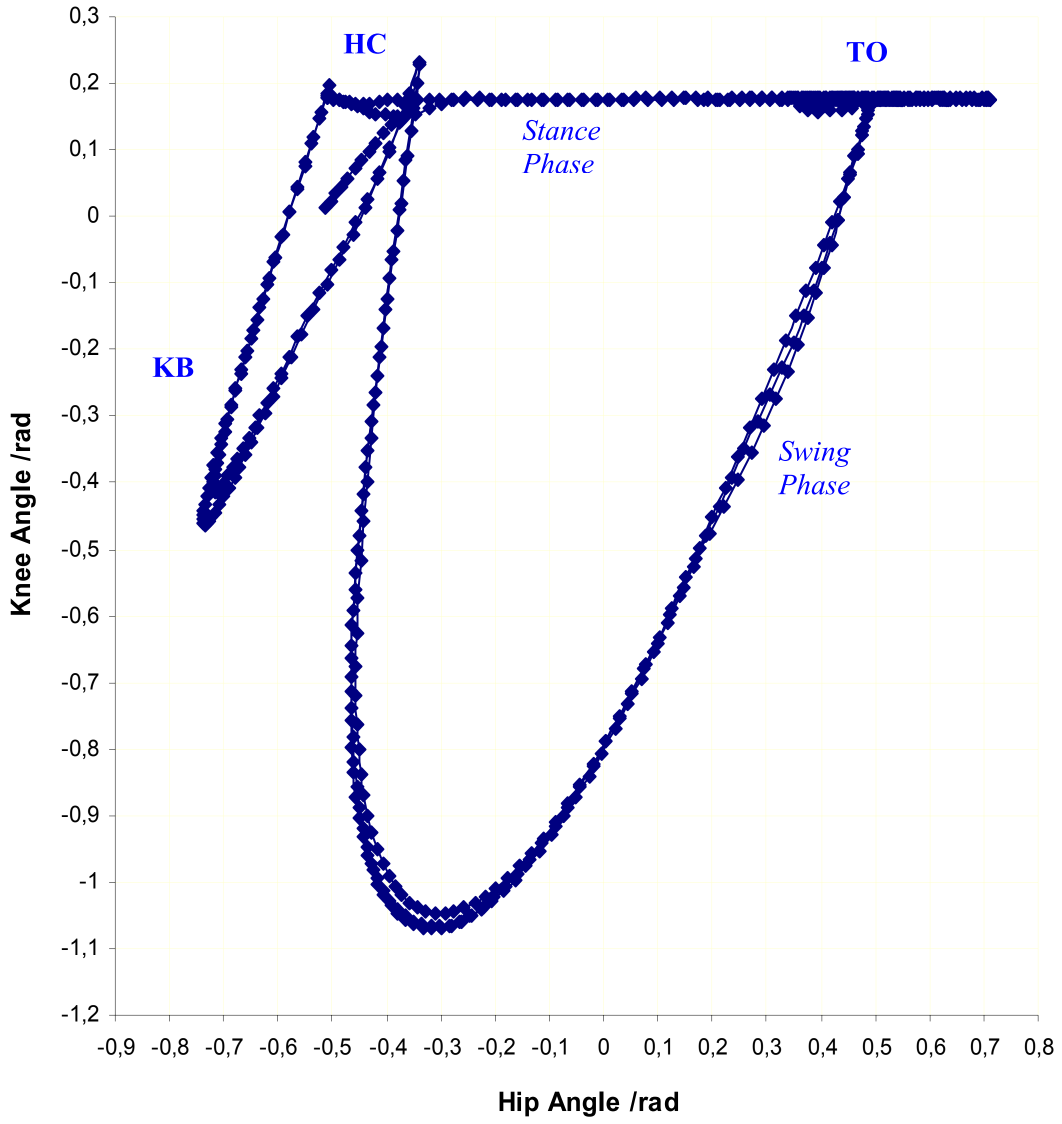


Figure 3.1–2: The angle-angle diagram of the walking gait on an inclined plane with slope 6.95°

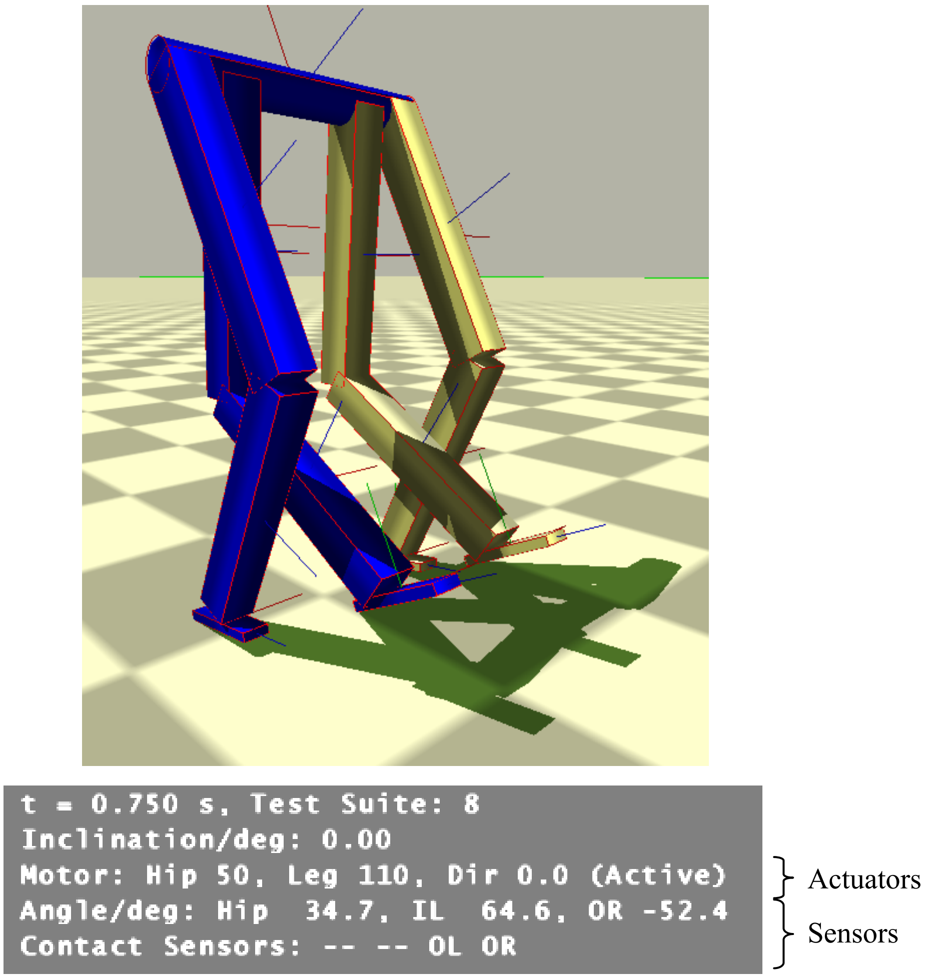
On this diagram, the toe-off (TO) is the point when a given foot leaves the ground at one of the lesser of the two peaks in knee extension. After TO, the leg is considered to be in the swing phase until it contacts the ground once again. The swing phase terminates when the heel makes contact with the ground where the knee reaches almost the full extension. At this point, called heel contact (HC), the leg enters the stance phase of the cycle where it remains until toe-off again. During the HC there is a knee bounce (KB) region where the knee joint bounces for a while until settled. The stance phase is the part of the obit between HC and TO. The complete stride is one orbit from TO to TO. The range of motion of both joints is indicated by the total area covered by the cycle. Knee flexion, where the angle gets smaller, is indicated by a downward motion in the *y*-direction, and knee extension is any upward motion on the plot. Just the same, hip rotation forward is to the right and backward rotation is to the left on the *x*-axis.

The effects of the friction can be seen on sliding of the standing foot during the stride. The average slip of the rear foot in the simulated case was 1.15 m (or 11.5 cm downscaled) with the standard deviation 0.07 m on the inclined plane with the slope 6.95°. The friction is important to hold the front leg in position during the swing phase and to hold the rear leg in position during the stance phase.

3.2 Advanced Level Tasks

The basic model is extended using asymmetric mass distribution between upper and lower leg parts, and also by allowing negative knee angles, so the leg assemblies can take a concave shape. This increases the stability of the passive walker since the center of mass of the walker is pushed forward to be above the center of the mass of the walker's feet. The negative knee angle can be observed as negative knee flexion on Figure 3.1–1 in the first frame of sample simulation of the passive walker. However, these extensions were also proven useful in the case of the active walker implementation where the cylinder feet are replaced with cuboid (flat) feet connected to shanks with ankle joints.

The sensors need to track the position of the active walker were angle sensors for the joints (angle between the legs at hip, and angles between thighs and shanks at knee joints) and contact sensors between the walker and the ground plane (hip axis and feet collision sensors). The image of the active walker with a sample sensor is shown on Figure 3.2–1. The hip-ground contact sensor is used to detect a condition when the walker is fallen on the ground plane, while the other sensors were used in the active walker's finite-state machine that controls the walker's actuators (angular motors at hip and knees' joints).



```
t = 0.750 s, Test Suite: 8
Inclination/deg: 0.00
Motor: Hip 50, Leg 110, Dir 0.0 (Active)
Angle/deg: Hip  34.7, IL  64.6, OR -52.4
Contact Sensors: -- -- OL OR
```

Figure 3.2–1: The sample image of the active walker with sensors showing the walker's hip and knee angles and its feet contacts with the ground plane.

The contact sensors were encapsulated in the `Machine::ContactSensor` class using the `btCollisionWorld::ContactResultCallback` class, while the angle sensors were encapsulated as a part of the `Machine::Joint` class that extends a general BT 6-DOF constraint. There exist five actuators: two at the inner/outer knees, two at the inner/outer foot ankles and one at the hip axis between the inner and outer pairs of thighs. These actuators are configured by the `Machine::Joint::SetAngularMotor` method and controlled by the finite-state machine in the `PassiveWalker::StateMachine` method, which executes transitions between inner/outer leg swing/stance phases depending on the actual sensed angles.

The sample animation of the active walker is shown on Figure 3.2–2, while the angle-angle diagram of the walking gait during the same simulation is shown on Figure 3.2–3.

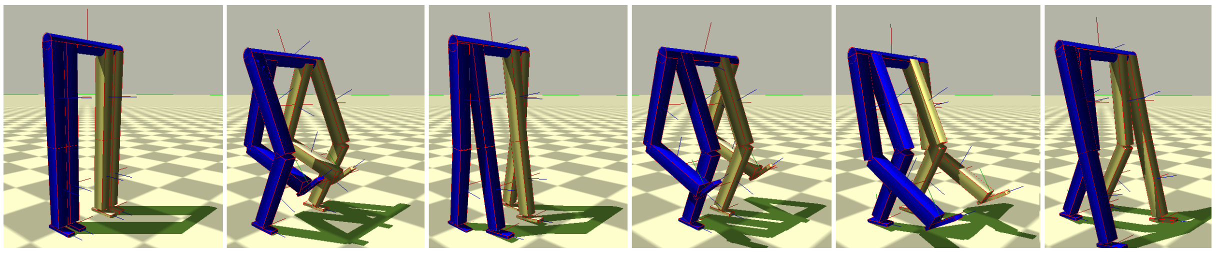


Figure 3.2–2: The sample animation of the active walker

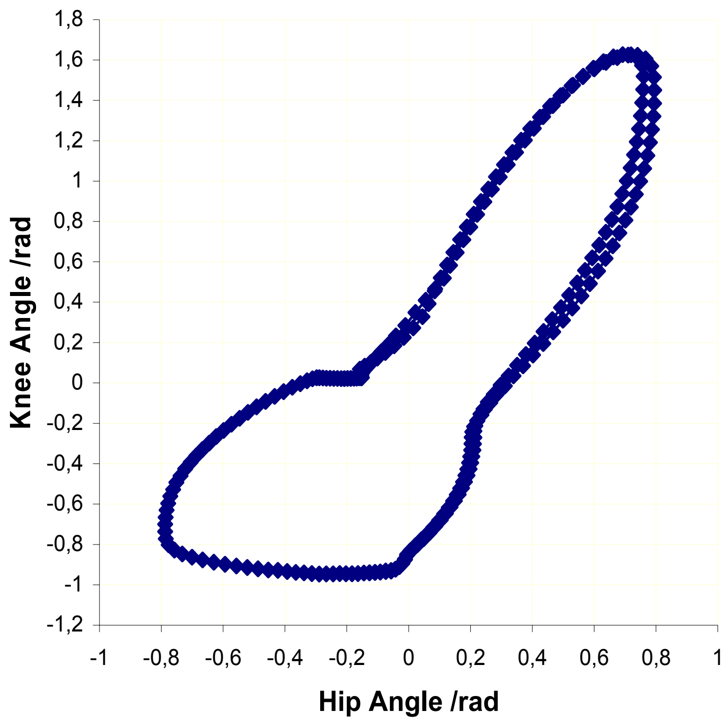


Figure 3.2–3: The angle-angle diagram of the active walker's walking gait.

The user can controll the active walker's speed can be controlled by setting two parameters: the maximal allowed hip angle (hip angle limit), and the maximal angular velocity of the upper legs (thighs). Short-cut keys '1' and '2' decreases and increases hip angle limit respectively, while short-cut keys '3' and '4' decreases and increases maximal allowed leg angular velocity respectively.

Since the implemented walker is a planar walker limited to a sagittal plane (the walker is effectively confined to a 2D movement), the change of direction was little harder to accomplish and it is done by inducing a slight angular offset between left and right leg motors. This angular 'disturbance' is controlled by short-cut keys '5' and '6' which decreases and increases the angular left/right motor offset respectively.

4 Discussion

There is an inherent difficulty in finding initial conditions that will produce a stable walk for a passive dynamic walker. In real-world models, the walker is swung a few times in the air and then put on the inclined plane at some point in time; this point is completely based on the experience of the user and his/her 'feeling' and feedback in hands.

However, to find initial conditions for a stable simulation is little bit harder and increases with the number of objects in the mechanical system. Typically, the simulated models are simplified to up to four objects that are analyzed in 2D. The implemented walker consists of 13 rigid bodies joined with 16 constraints, which gives the total of 169 state variables that should be set up at the beginning of the simulation. The presented passive walker in fact has six degrees of freedom, but still, it is difficult to pin-point exact location in the six-dimensional phase space. Further, the simulation is *very* sensitive to round-off errors and execution order of algorithms (e.g. induced by the order of creation of objects). It was discovered that by introducing as low as 10^{-8} relative error in initial parameters can cause a significant butterfly effect. Also, it was discovered that the kind of the *code compilation* has a significant effect on the stability; for instance, by turning debug mode or by changing the level of code optimization can render once stable initial set of parameters. All these problems are mainly caused by the user putting incoherent input initial conditions that cause an initial impulsive (jerky) behavior of the simulated passive walker in the first simulation time-step.

To solve this remedy, this simulation model uses a Monte Carlo method for finding initial conditions. The implemented algorithm starts with some given initial parameters loaded from 'config-*id*.txt' file, and randomly varies them while monitoring the duration of the resulting walks. The simulation stops when the 'walk duration' goal is exceeded, in which case the simulation parameters are appended to the 'config-*id*.txt' file. This 'walk duration' goal is increased each time the Monte Carlo simulation is run. The user can later examine these configuration files and bring the selected initial parameters to the beginning of the file.

References

[1] McGeer, Tad, *Passive dynamic walking*, International Journal of Robotics Research, pp 62-82, 1990

[2] McGeer, Tad , *Passive walking with knees*, Proceedings of the IEEE Conference on Robotics and Automation, pp 1640-1645, 1990

[3] McGeer, Tad, *Passive dynamic biped catalogue*, In Proceedings of the 2nd International Symposium of Experimental Robotics. Springer-Verlag, New York, 1991

[4] McGeer, Tad, *Dynamics and control of bipedal locomotion*, Journal of Theoretical Biology, pp 277–314, 1993

[5] Ruina, Andy, *Non-holonomic stability aspects of piecewise holonomic systems*, Reports on Mathematical Physics, 1997

[6] Garcia, M., Ruina, A., Coleman, M. *Some results in passive-dynamic walking*, Euromech Conference on Biology and Technology of Walking, Munich, Germany, March 1998.

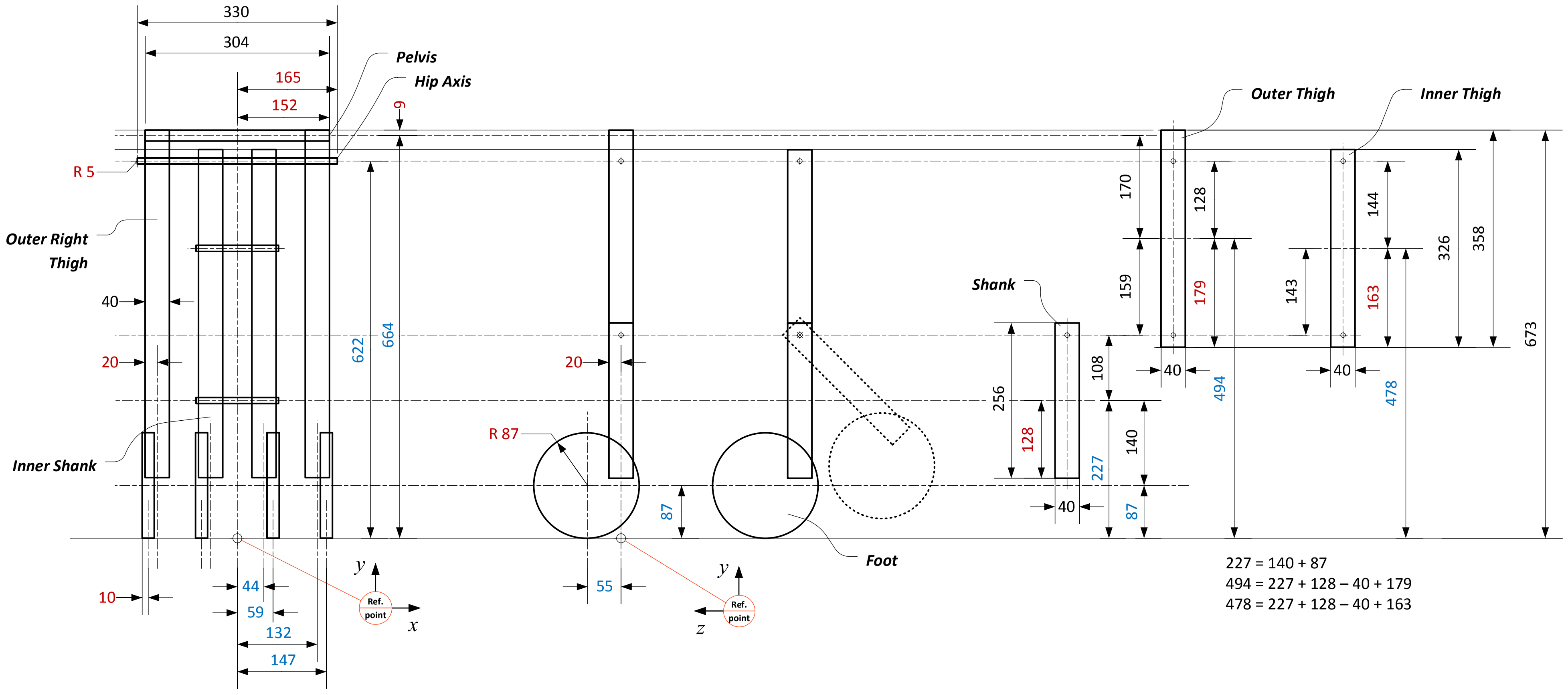
[7] Chen, J., Greaves, A., Lohrenz, I. & Wu, Q., *On Passive Dynamic Bipedal Walking*, Poster on Dynamic Walking Department of Mechanical and Manufacturing Engineering, University of Manitoba, 2008

[8] Koop, D., Ferley, D. & Wu, Q., *Passive Dynamic Bipedal Walking with Knees, The Effect of Parameter Variation*, Department of Mechanical and Manufacturing Engineering, University of Manitoba, North Central ASABE/CSBE Conference, 2008

[9] Connolly, Christopher I, et al., *A Computational Model for Legged Locomotion*, Technical Laboratory for Perceptual Robotics, University of Massachusetts, Report #96-14, 1996

[10] Honeycutt, C., Sushko, J. & Reed, K.B., *Asymmetric Passive Dynamic Walker*, IEEE International Conference on Rehabilitation Robotics, 2011

Appendix A – Passive Walker Dimensions



227 = 140 + 87
494 = 227 + 128 – 40 + 179
478 = 227 + 128 – 40 + 163

Part	Shape	Half Extent			Position			Orientation		
		x	y	z	x	y	z	x	y	z
Pelvis	Box	152	9	20	0	664	0	0	0	0
Hip Axis	Cylinder	5	165	5	0	622	0	0	0	$\pi/2$
Inner L/R Thigh	Box	20	163	20	±44	478	0	0	0	0
Outer L/R Thigh	Box	20	179	20	±132	494	0	0	0	0
Inner L/R Shank	Box	20	128	20	±44	227	0	0	0	0
Outer L/R Shank	Box	20	128	20	±132	227	0	0	0	0
Inner L/R Foot	Cylinder	87	10	87	±59	87	55	0	0	$\pi/2$
Outer L/R Foot	Cylinder	87	10	87	±147	87	55	0	0	$\pi/2$
Inner Leg Lock	Box	68	5	5	0	227	0	0	0	0

Notes: Cylinder height is specified along y-axis (requires rotation)
Position is given relative to the reference point

Joint			Body A, rel. pos.			Body B, rel. pos.			Body B, rotation		
Body A	*Body B*	*Type*	x	y	z	x	y	z	x	y	z
Pelvis	Outer Thigh	Fixed	±132	0	0	0	170	0			0
Inner Leg Lock	Inner Shank	Fixed	0	0	0	0	0	0			0
Inner Thigh	Hip Axis	Fixed	0	144	0	0	±44	0			$-\pi/2$
Outer Thigh	Hip Axis	Hinge ±90°	0	128	0	0	±132	0			$-\pi/2$
Outer Thigh	Shank	Hinge +0/-95°	0	−159	0	0	108	0			0
Inner Thigh	Shank	Hinge +0/-95°	0	−143	0	0	108	0			0
Shank	Foot	Fixed	0	−140	55	0	± 15	0			$-\pi/2$