Humanoid Robot Walking Optimization using Genetic Algorithms

Luiz Fernando Cirigliano Villela *Esther Luna Colombini*

Relatório Técnico - IC-PFG-17-07

Projeto Final de Graduação

2017 - Julho

UNIVERSIDADE ESTADUAL DE CAMPINAS

INSTITUTO DE COMPUTAÇÃO

Humanoid Robot Walking Optimization using Genetic Algorithms

Luiz Fernando Cirigliano Villela   Esther Luna Colombini

Abstract

The problem of creating fast and stable walking for humanoid robots is a very complex one due to the large degree of freedom and the external variables involved. This work tackles this problem by using a model free approach where each joint is represented by a sinusoidal function of time. The parameters of the functions for all actuated joints are optimized using a genetic algorithm. Experiments were performed with a NAO robot in a simulated environment under V-REP. The optimized robot was able to walk at a speed of $54cm/s$ in a straight line and for up to 200 meters without falling. Experiments were also carried out to evaluate the individuals capacity to adapt to different scenarios, such as walking up and down ramps. Results showed different movement patterns, a slower pace and more upright positions for the robot walking uphill.

The code and raw data of the experiments used in this project are hosted on [1].

1 Introduction

The field of robotics is a complex and fast evolving one. From industrial automation to military drones, they are increasingly more present in our lives. It also presents many challenges, as usually there are multiple variables in the environment that can affect their operation. One of the core challenges of robotics is movement. For some robots, this challenge can be simplified by the use of wheels, which provides a simpler and more stable movement path, but for others this is not applicable, as uneven terrains, steps, and other complications can get in the way. For these cases, that represent more general ones, humanoid robots are usually embodied in the environment and more complex locomotion patterns are required.

Making a humanoid robot walk is not a trivial task as there are many limbs and degrees of freedom (DOF) to control, and plenty of outside factors that affect how those limbs should move. To tackle this problem, evolutionary algorithms represent a good approach to find good solutions in this sea of possibilities. In this work, we present a system implemented for optimized bipedal locomotion based on genetic algorithms and sinusoidal functions [2].

2 Movement Model

A humanoid robot has many degrees of freedom, which makes the task of controlling its movement a very difficult one to achieve. To overcome this problem, two different approaches are defined in the literature: model-based and model-free.

A **model-based** approach will analyze the forces involved in the movement (figure 1), and create a model based on the resulting kinematic equations. One of these approaches is "Zero Moment Point" (ZMP), which calculates the point on the ground, P, where the resulting moment is $\vec{0}$. The algorithm then uses this point and the region of support in contact with the ground to decide whether the walk will be stable.

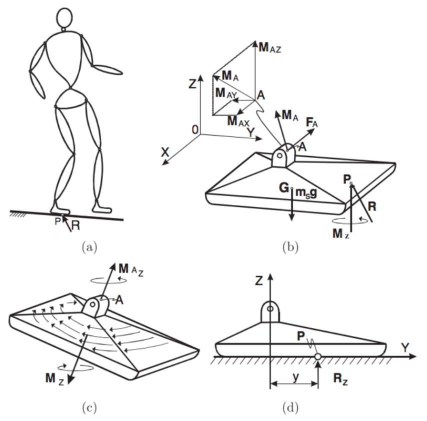


Figure 1: Forces acting on the foot sole of a walking biped. [3]

Model free approaches, as the name suggests, make little assumption regarding the forces acting on the movement, and the stability of the movement is evaluated through means of trial and error, usually requiring machine learning and/or optimization techniques.

Considering that walking is a periodic movement and that, usually, there is a symmetry between the right and left side of the body, where one side is always half period behind the other, the angular trajectories of the joints of a humanoid robot could be defined as a periodic function, generically written as a Fourier Series:

$$f(t) = \frac{a_0}{2} + \sum_{i=1}^{\infty} \left[a_i \cos(\frac{2n\pi t}{T}) + b_i \sin(\frac{2n\pi t}{T}) \right] \tag{1}$$

Fourier Series are an infinite sum of sines and cosines and in order to make them computationally feasible to solve, a Truncated Fourier Series (TFS) is usually applied by using a finite summation instead, such that:

$$f(t) = \frac{a_0}{2} + \sum_{i=1}^{N} \left[a_i \cos(\frac{2n\pi t}{T}) + b_i \sin(\frac{2n\pi t}{T}) \right] \tag{2}$$

In a bipedal locomotion task, each joint active in the movement has a TFS associated to it, where the period T is shared among all joints and the other parameters to the equation are independent. In fact, [4] demonstrated that TFS could be used to generate angular trajectories for locomotion control that are valid according to ZMP [2].

Model free formulas are usually paired with optimization methods such as genetic algorithms (GA) and particle swarm (PSO), that are used to progressively try out and evolve

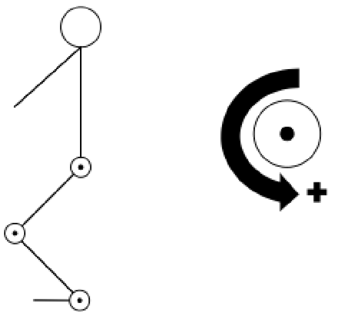


Figure 2: Leg joint schematic. [2]

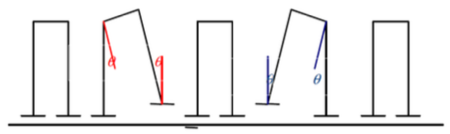


Figure 3: Schematic for the leg movement in the coronal plane (hip roll).[2]

these parameters based on some metric that is pre-defined to evaluate the robots performance.

In the experiments performed in this work, a model free approach that uses a single sine wave component as the TFS was adopted [5]. In addition, two different amplitudes are used, one for the first half of the cycle, and another one for the second. This allowed a joint to have a wider movement range in one direction, and a restricted one in the other.

A generic joint x could be described by the following equations:

$$\Theta_x(t) = \begin{cases} \Theta_{0x} + \alpha_x \sin(\frac{2\pi(t+\tau_x)}{T}) & \text{if } t - \tau \leq \frac{T}{2} \\ \\ \Theta_{0x} + \beta_x \sin(\frac{2\pi(t+\tau_x)}{T}) & \text{if } t - \tau > \frac{T}{2} \end{cases} \tag{3}$$

where Θ_x is angular position of the joint in time t, α_x, β_x and τ_x are constants.

The body joints used to control the walk of the robot were hip, knees, and ankle (pitch), as seen on Figure 2, plus arm movement through the shoulder joint, and also a degree of freedom in the coronal plane of the legs (Figure 3).

One issue with these equations is that the change from the robot's initial state to the moving state could be very abrupt, causing instability. To circumvent this problem, a scaling factor was added to Equation 3, and a multiplying factor in the amplitude was added in order to ease the transition into the moving stage. Let n be the number of the period (starting at $n = 0$), the equations for the initial transitions are given by:

$$\Theta_x(t) = \begin{cases} \Theta_{0x} + \frac{n}{4}\alpha_x \sin(\frac{2\pi(t+\tau_x)}{T}) & \text{if } n < 4 \\ \\ \Theta_{0x} + \alpha_x \sin(\frac{2\pi(t+\tau_x)}{T}) & \text{if } n >= 4 \end{cases} \tag{4}$$

Exceptions to this system are the ankle pitch and roll joints. Instead of being independent, their role is to compensate for the movement of hip and knees, making sure the foot is always parallel to the floor, preventing a lot of unstable scenarios.

2.1 NAO Robot

For the experiments, the chosen humanoid model was the NAO robot, by Aldebaran Robotics [6]. Figures 4, 5, 6 detail the dimensions and full degrees of freedom of its joints.

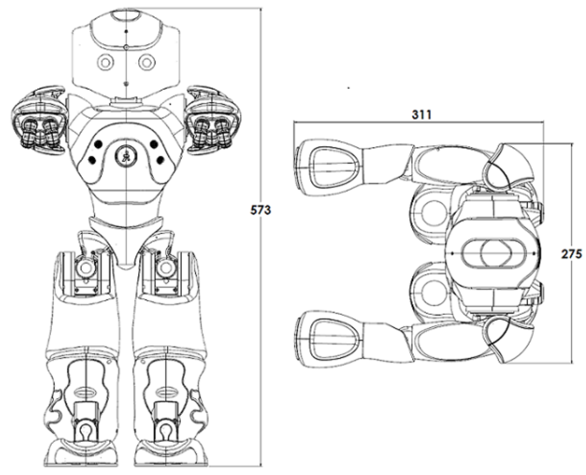


Figure 4: NAO H25 robot dimensions, in millimeters. [6]

To improve efficiency and convergence, each of the parameters for the controlled joints is either given a search interval or a fixed value (e.g: arm movement is always in opposing phase to the leg). This search space is given by table 1.

Table 1: Parameters and bounds for each joint, following conventions from equation 3. The interval for the movement period parameter T was [200.0ms, 600.0ms].

Θ_{0x} **(rad)**	α_x **(rad)**	β_x **(rad)**	τ_x **(t/t)**	**Joint**
[-0.5, 0.0]	[-1.0, 0.2]	[-0.4, 0.0]	0.0	Hip Pitch
[0.0, 2.0]	[0.2, 3.1]	0.0	[0.0, 0.5]	Knee Pitch
0.0	[0.0, 1.0]	[0.0, 1.0]	0.5	Shoulder Pitch
0.0	[0.0, 0.5]	0.0	0.0	Hip Roll

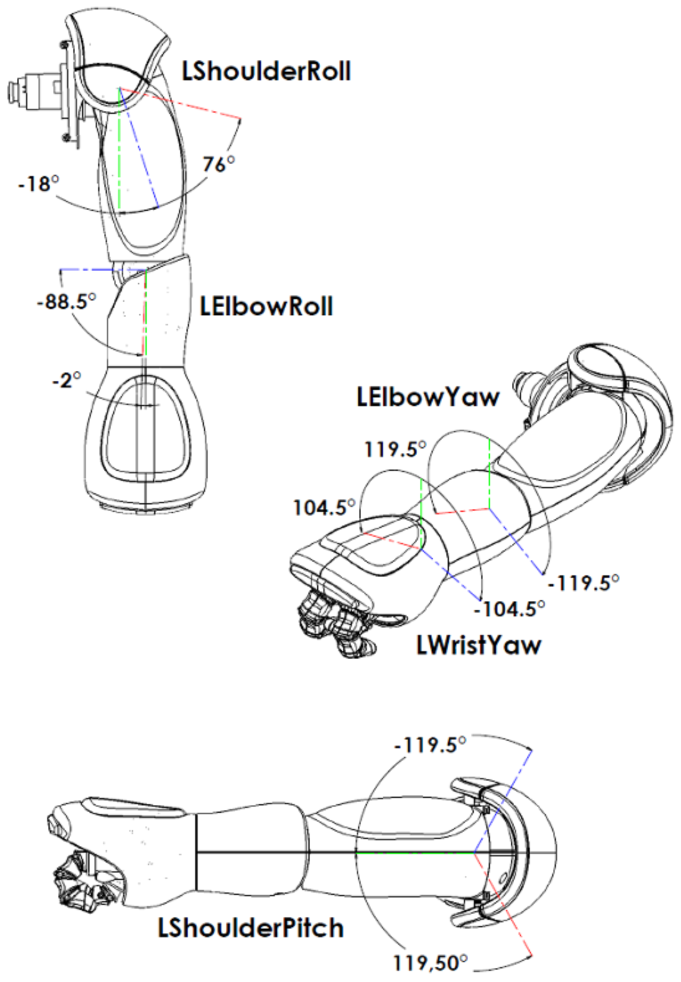


Figure 5: NAO arm detailed. [6]

3 Genetic Algorithm Application

Genetic Algorithms are inspired by the biological process of evolution and natural selection. They are a way of searching for good and robust solutions in a large search space, such as all the possible parameters for the joint equations in a robot. The algorithm is composed of a population of individuals that evolve through time. The phases of crossover, mutation and selection are repeated until a threshold of convergence or maximum number of iterations is reached. A fitness function measures the quality of an individual, that represents a particular solution for the problem.

There are many variations to genetic algorithms[7]. For this research, the optimization used a Steady State genetic algorithm, implemented in C++ utilizing GALib [8]. The Steady State GA creates an overlap at each generation by adding the newly generated offspring to the previous population, then removing the worst individuals out of both parents

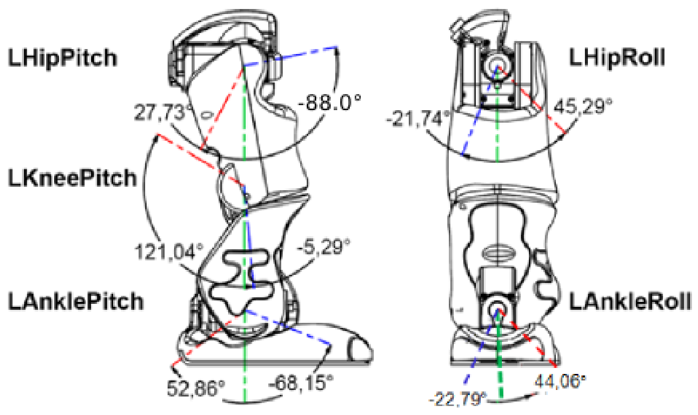


Figure 6: NAO leg detailed. [6]

and offspring to return the population to its original size. A replacement percentage can also be used to specify a quota of offspring and parents that should be selected at this stage. The best value for this is domain specific, and it is about finding a balance between keeping good existing solutions while maintaining variability.

In this work, GA was applied to find the best fit values for the constants of equation 3 for each controlled joint.

3.1 Selection

In this work, the end goal of the optimization is to find a robot that is able to walk the fastest in a straight line without falling. To that end, a combination of factors must be taken into account when defining the tests and the objective (scoring or fitness) function.

The objective function for this problem is particularly significant in initial stages, and thus should be more carefully designed to take into account the initial stages when most individuals simply fall straight away. A good combination of scoring components would overcome that initial barrier. For instance, even though the end objective was to have a robot that walked along the x axis, adding slight score bonuses for standing up, as well as walking in the y axis, helped the evolution at stages when most individuals would not be able to stand, let alone walk in any direction at all.

This function was defined as:

$$F(i) = 1 + 15 \times \frac{t_i}{600} + 10 \times D_i + 20 \times X_i \tag{5}$$

Where t_i is the number of simulation steps that the robot spent without falling, D_i is the total distance moved, and X_i is the distance moved in the X axis (negative values count as 0). The function also guarantees a positive value due to GALib utilizing log for processing the results.

Finally, after individuals have their score measured, the selection is made by a "Roulette Wheel Selection" [9], where the probability of an individual being selected to the next generation is proportional to its fitness compared to the rest of the population.

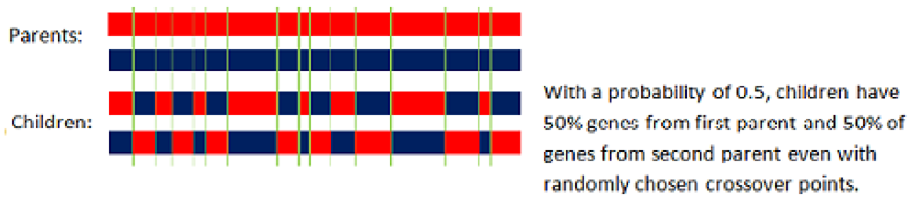

Parents:
Children:
With a probability of 0.5, children have 50% genes from first parent and 50% of genes from second parent even with randomly chosen crossover points.

Figure 7: Uniform crossover visualization.

3.2 Crossover

After selection, crossover is the method that generates the offspring for the next generation. The most common type is the uniform crossover (Figure 7), which simply copies half of the genes from one parent and half from the other. But, as the genes in this problem have a continuous distribution, we can use another method, known as BLX-alpha (also called Blend, algorithm detailed in section 3.2.1), to improve the genetic variability and convergence. Instead of just picking a gene from either parent, the parents now define an interval from which the new value is picked.

3.2.1 BLX-alpha algorithm

The BLX-alpha (blend) crossover algorithm, as presented on [10], is defined by:

```
1: select two parents $X(t)$ and $Y(t)$ from a parent pool
2: create two offspring $X(t+1)$ and $Y(t+1)$ as follows:
3: **for** i = 1 to n **do**
4:     $d_i = \|x_i(t) - y_i(t)\|$
5:     choose a uniform random real number $u$ from interval
       $[min(x_i(t), y_i(t)) - \alpha d_i, max(x_i(t), y_i(t)) + \alpha d_i]$
6:     $x_i(t+1) = u$
7:     repeat 5 and 6 for $y_i(t+1)$
8: **end for**
```

3.3 Mutation

Mutation in a genetic algorithm has, in real life, an important aspect in increasing the diversity of the population after a few generations.

The mutation process for the GA applied in this work utilized a Gaussian (or Normal) distribution method, where each gene has a probability p ($p = 5\%$) of suffering a mutation. If a mutation occurs, the new value of the gene follows a Gaussian distribution around the initial value, with a standard deviation of 1.0, as it can be seen in Figure 8.

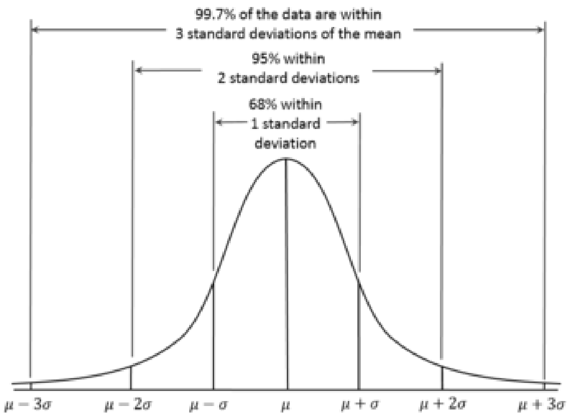


Figure 8: Probabilities in a Gaussian distribution.

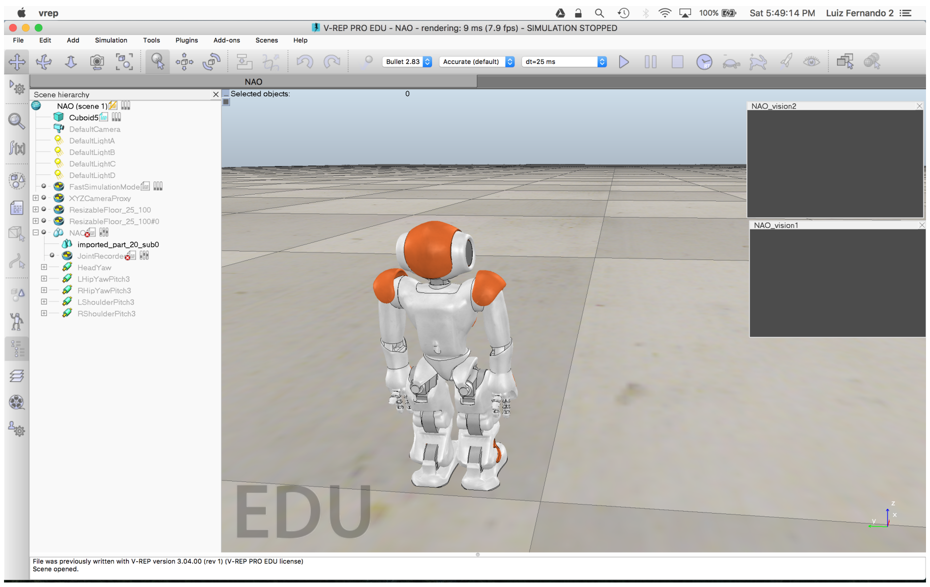
4 Simulations

All the simulations were done with a NAO [11] robot running on the V-REP Simulator [12], utilizing the Bullet v2.8.3 engine.

V-REP offers a remote API[13] to control the simulation (also referred to as the server) through a C++ interface (the client). The remote API allows the client to set, for instance, the absolute position of a "limb", and the angle for each joint. This is how the robot is controlled. At each simulation step the client calculates where the joint should be according to the GA individual being tested and then sends a command to the server to set the joint to that angle.

One of the issues that can arise from this server/client approach is that they can go out of sync, if, for instance the calculations on the client take too long. To prevent this issue, V-REP offers a synchronous mode, where the client is also responsible for the simulation step. So the simulation no longer happens in real-time, but instead at whatever pace the client is running.

As mentioned above, since we are dealing with a computational simulation, the engine is not actually running a continuous simulation. Instead, it takes "steps", small time intervals at which it calculates the forces and movement being exerted, and then interpolates to the next "step". The size of this step has to take into account a balance between computational power and accuracy. The default setting of the program utilizes a time step of 50ms for its simulations, but after initial tests, this time interval showed to be too large of a gap. With the robot movement period T being a vital component of the optimization, a large time step prevent T to decrease below a point of about 400ms, as then the step would become very large compared to the entire movement cycle. Reducing the step by half allowed for T to go down further, allowing faster walks and improving the results considerably.



V-REP PRO EDU - NAO - rendering: 9 ms (7.9 fps) - SIMULATION STOPPED

Bullet 2.83 | Accurate (default) | dt=25 ms

Selected objects: 0

```
File was previously written with V-REP version 3.04.00 (rev 1) (V-REP PRO EDU license)
Scene opened.
```

Figure 9: Screenshot of the simulation program, VREP, with the test scene loaded.

5 Experiments and Results

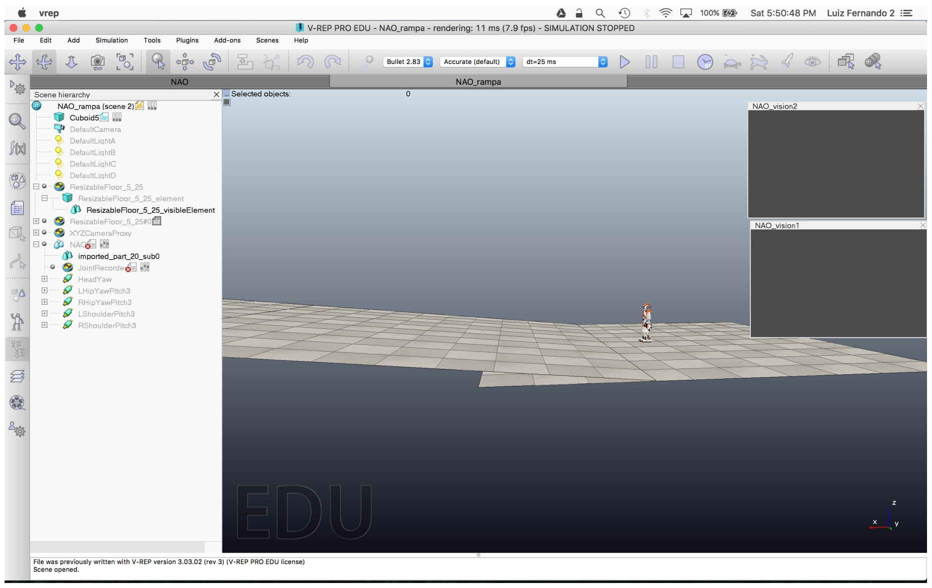
Two separate evolution experiments were conducted on different terrains. The **first** experiment considered a plane surface (Figure 9), and the **second** had terrain with a ramp with a 5^o inclination (Figure 10). The results and comparison between the two evolved individuals are presented next.

In both evolution processes, the robots were allowed to walk for up to 15 seconds or until they fell. The falls were detected by tracking the robot Z axis at each simulation step. In addition to this, on the ramp test, if the robot started drifting to the side, or going back down the ramp, the test would also be stopped. The process started with a random population of 150 individuals, and evolved through the process described on section 3 until there were no significant improvements between two subsequent generations. The replacement rate was set to 80% in both sets of experiments.

To benefit more stable walks, during the evolution process, each robot walked 3 times, and the average score of the objective function (Equation 5) was used as the final result.

5.1 Plane floor experiment

Figure 11 displays the evolution of individuals by generation. It can be seen that there is usually big improvements between generations up until generation 11, when the scores start to converge. After the evolution process, there were multiple individuals with similar scores, the one presented here was the one that achieved the most consistent result across tests, with a speed of **54 cm/s** over a period of 15 seconds, walking over 8 meters. This results present an improvement over previous algorithms (Table 5.1). Furthermore, according to [14], a maximum speed 44.47cm/s was achieved by the ZMP Controller implemented by



File was previously written with V-REP version 3.03.02 (rev 3) (V-REP PRO EDU license)
Scene opened.

Figure 10: Screenshot of VREP with the ramp scene loaded.

the NAO Devils team. In this case, our approach increased this speed by 21%. Figure 12 present the trajectory of the best robot at each new generation.

Table 2: Comparison of performance to the ones presented on a previous work [2].

Modelo	$\Delta X(m)$	**Desvio Padrão (m)**
magma-AF	3.93	0.4
Maximo	7.41	0.55
Cirigliano (Proposed approach)	8.1	0.1

Table 3: Best individual performance over 3 runs.

Run	**T (s)**	Δx **(m)**	Δy **(m)**	Vx **(m/s)**
1	15	8.1	-0.72	0.54
2	15	8.1	-0.98	0.54
3	15	8.1	-0.29	0.54
AVG	15	8.1	-0.663	0.54

More tests were carried out to evaluate the stability and performance of the most evolved individual, as detailed on Table 4. This particular individual was then allowed to walk without the time restriction, for a length of up to 200 meters. 10 runs were attempted, and the results are displayed on Figure 13. In most tests the robot walked at least 50 meters before falling, failing that mark in only one case, when it fell shortly after 30 meters. In one of the tests, the robot also walked the whole 200 meters, reaching the maximum distance limit.

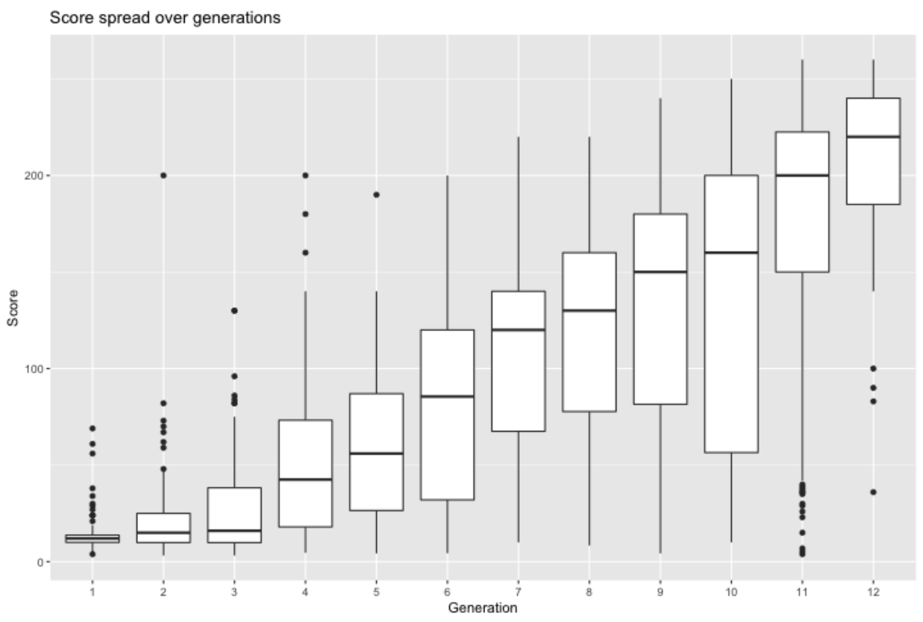


Figure 11: A boxplot graph representing the score distribution of individuals over generations.

Table 4: Joint parameters for the best individual, with a period of T=295ms.

Θ_{0x} **(rad)**	α_x **(rad)**	β_x **(rad)**	τ_x **(t/t)**	**Joint**
-0.211	-0.872	-0.117	0.0	Hip Pitch
0.622	3.29	0.0	0.171	Knee Pitch
0.0	0.357	0.307	0.5	Shoulder Pitch
0.0	-0.0128	0.0	0.0	Hip Roll

We also noticed that in the longer tests, the robot tends to deviate more from its straight path, and not always to the same side. This can be attributed to the multiple factors that are involved in a "real-world" simulation, such as motor friction, torque variations, uneven surfaces, and etc, what would require a feedback control in the system to correct the robot's path while walking.

5.2 Floor with ramp experiment

Initially there was an attempt to move the previously trained individual from the flat surface directly to the ramp, but this test failed, as the robot would fall as soon as it reached the inclination. The whole optimization process was then re-executed for the ramp scenario, in order to compare the differences between the evolved individuals. The results are presented in Tables 5 and 6.

As expected, the two scenarios generated different adaptions of individuals, as we can see in the comparison between the joint functions on Figure 14). For instance, the knee

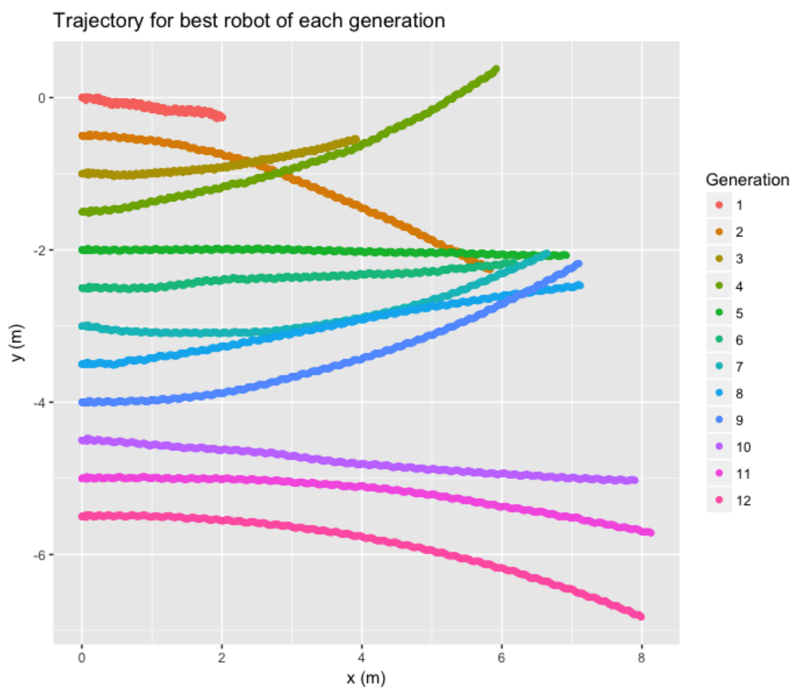


Figure 12: Trajectory of the best new robot at each generation (run #3).

Table 5: Best ramp individual performance over 3 runs.

Run	**T (s)**	**Δx (m)**	**Δy (m)**	**Vx (m/s)**
1	15	5	0.16	0.33
2	15	4.8	0.95	0.32
3	15	4.9	0.8	0.327
AVG	15	4.9	0.637	0.326

Table 6: Joint parameters for the best ramp individual, with a period of T=235ms.

Θ_{0x} (rad)	**α_x (rad)**	**β_x (rad)**	**τ_x (t/t)**	**Joint**
-0.176	-0.675	-0.222	0	Hip Pitch
0.572	1.29	0	0.127	Knee Pitch
0	0.463	0.709	0.5	Shoulder Pitch
0.0	-0.136	0.0	0.0	Hip Roll

amplitude for the ramp individual (Table 6) was 1.29 rad as opposed to 3.29 rad (Table 4) for the individual walking on a flat surface. Tilting the body forward on a plane was a way to gain speed, but with the addition of the slope, that movement ends up causing the robot to fall, leading to more conservative approaches.

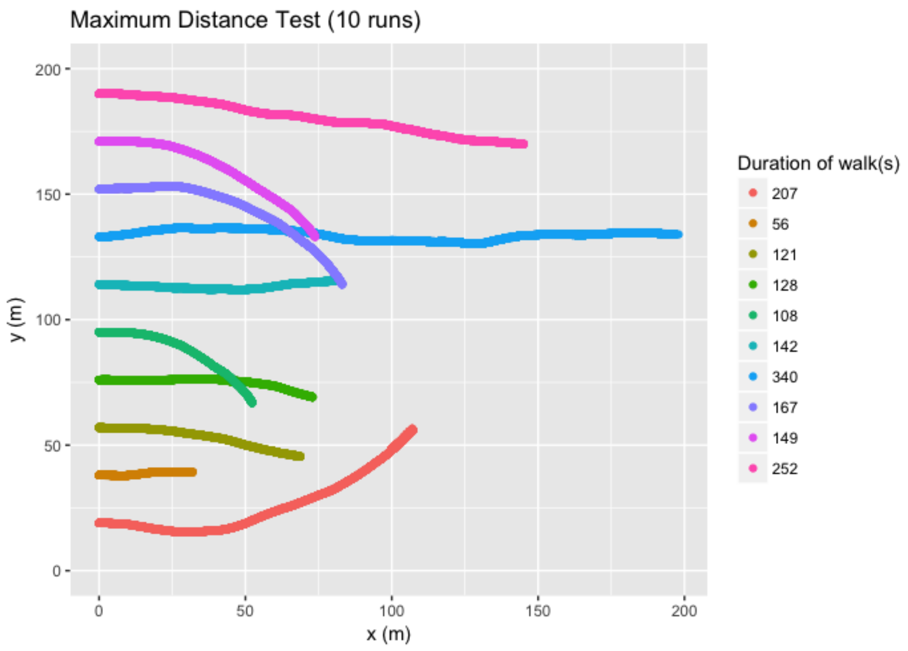


Figure 13: Trajectory of trained individual on a distance test. For this test, the robot was allowed to walk until it fell, up to a maximum of 200 meters.

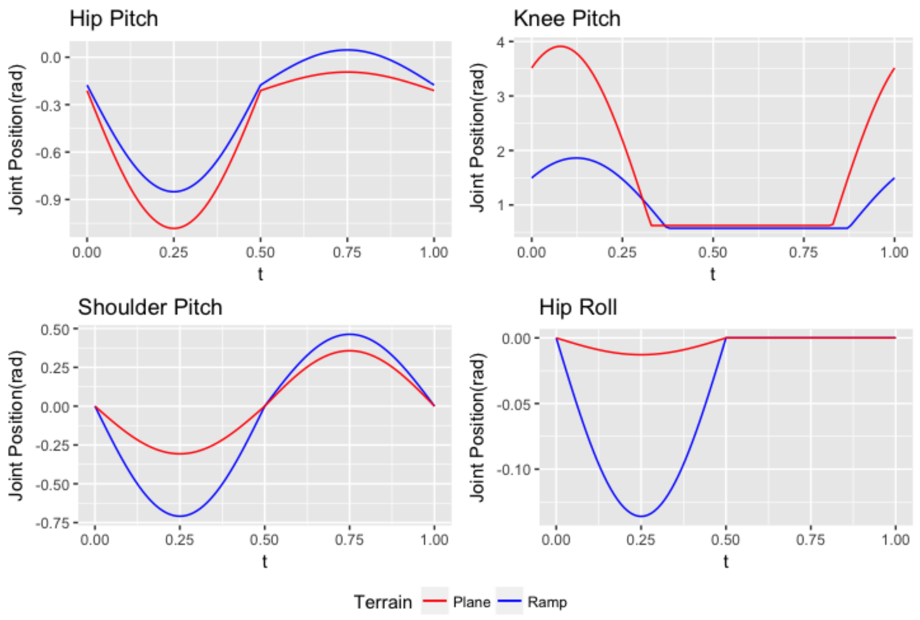


Figure 14: Comparison of each joint's movement function, for the best plane and best ramp individual.

6 Conclusion

In this work the problem of humanoid locomotion was tackled through a model free approach where each joint is represented by a sinusoidal function of time and its parameters are optimized by a genetic algorithm. Experiments were performed with a NAO robot in a simulated environment under V-REP and the optimized robot was able to walk at a speed of $54cm/s$ in a straight line and for up to 200 meters without falling.

The resulting work has shown that, despite the enormous search space for the problem treated and all the variables involved in humanoid movement, it can be treated by providing a solid and succinct foundation for the movement, and then allowing the genetic algorithms to find the best solutions.

Furthermore, another advantage of using this model-free evolutionary approach is that it takes care of environmental differences on its own. No changes in code were necessary in order to differentiate the learning processes for a plane versus a slope scenario, for instance. When we wanted to see how the robot would adapt to walking on a ramp with a 5^o inclination versus the plane, we simply had to re-execute the optimization process for the ramp scenario. If we were to test the robot on a ramp with a 2.5^o inclination, we would re-execute that same process. This would be useful in a real-world situation, where the robot has been trained with a variety of walk types for specific scenarios, and then detects which of those is the most similar to its current situation on the go, but this pattern recognition is a whole another problem in itself.

References

[1] Luiz Fernando Cirigliano. Project github repository. `https://github.com/LuizFernandoCirigliano/pfg`. (Accessed on 06/30/2017).

[2] Marcos Ricardo Omena de Albuquerque Maximo. Otimizacao de caminhada de robos humanoides, 2012.

[3] Miomir Vukobratović and Branislav Borovac. Zero-moment point—thirty five years of its life. *International Journal of Humanoid Robotics*, 1(01):157–173, 2004.

[4] Lin Yang, Chee-Meng Chew, Teresa Zielinska, and Aun Neow Poo. A uniform biped gait generator with offline optimization and online adjustable parameters. *Robotica*, 25(5):549–565, 2007.

[5] Nima Shafii, Siavash Aslani, Omid Mohamad Nezami, and Saeed Shiry. *Evolution of Biped Walking Using Truncated Fourier Series and Particle Swarm Optimization*, pages 344–354. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[6] Discover nao, the little humanoid robot from softbank robotics — softbank robotics. `https://www.ald.softbankrobotics.com/en/cool-robots/nao`. (Accessed on 06/24/2017).

[7] Gilbert Syswerda. A study of reproduction in generational and steady state genetic algorithms. *Foundations of genetic algorithms*, 2:94–101, 1991.

[8] Galib: Matthew's genetic algorithms library. `http://lancet.mit.edu/ga/`. (Accessed on 04/08/2017).

[9] Ga roulette wheel selection. `http://www.edc.ncl.ac.uk/highlight/rhjanuary2007g02.php`. (Accessed on 06/10/2017).

[10] Blend crossover (blx). `http://www.tomaszgwiazda.com/blendX.htm`. (Accessed on 05/18/2017).

[11] Nao - documentation — aldebaran 2.5.5.5 documentation. `http://doc.aldebaran.com/2-5/home_nao.html`. (Accessed on 05/23/2017).

[12] Coppelia robotics v-rep: Create. compose. simulate. any robot. `http://www.coppeliarobotics.com/`. (Accessed on 05/23/2017).

[13] Remote api functions (c/c++). `http://www.coppeliarobotics.com/helpFiles/en/remoteApiFunctions.htm`. (Accessed on 01/15/2017).

[14] Julian Cristiano, Miguel Angel Garcia, and Domenec Puig. On the maximum walking speed of nao humanoid robots. In *XII Workshop of Physical Agents*, pages 60–66, Albacete, Spain, September 2011.