Evolutive learning of walking gaits for Nao humanoid robot using the Gazebo simulator

Francisco Pérez[1] and Francisco Rivas[2] and José M. Cañas[3]

Abstract— **This contribution presents the work in progress for the automatic learning of walking gaits using an evolutionary algorithm. The humanoid movement is modelled as coupled waves of its limbs, which are coordinated by a Central Pattern Generator. The waves are defined with a set of parameters. The values of parameter set define the space of possible walking gaits. The evolutionary algorithm searches in that space a good walking gait such that maximizes the robot advance speed, provides stability and minimizes lateral deviation. The evolutionary algorithm proposes parameter combinations which are tested in a Nao humanoid inside the Gazebo simulator. The developed Nao support in Gazebo software and the implementation of this gait evaluation on the simulated humanoid are presented.**

I. INTRODUCTION

Walking gaits for humanoid robots is an important issue in robotics. Many research groups are focusing their studies on trying to find a good solution for this kind of locomotion. A clear example of this is the international scientific competition RocoCup. In this competition a group of robots have to play soccer all together and humanoids are used in some of their main leagues. Moreover, almost all major Japanese companies are working on their own prototypes. Perhaps, in order to introduce them into an emerging market for service robots in homes, thinking that this human appearance makes them more readily acceptable by people. One of the most important works is the Honda ASIMO (Advanced Step in Innovative MObility) robot, which is capable of speeds up to 6 km/h.

Despite of the major recent advances, locomotion of humanoid remains an open problem, yet being too far from the flexibility, robustness and plasticity of human natural movements of people.

Gaits generation in humanoid robots is part of a more general problem: the robots coordination of n joints actuators. Given a robot with n joints with its particular morphology (humanoid, quadruped, hexapod, legless...) the problem is to find the specific functions which establish the position of each joint so that the robot can move. In the literature there are three different approaches: classic, control tables and bio-inspired model. In the classical approach, trajectory functions for the outside of the limb are set, calculating every joint position using inverse kinematics. Path functions for the ends of the legs are set in the classical approach and the positions of the actuators are calculated using inverse kinematics. The implementation of these controllers requires an exhaustive mathematical model of the robot and overall computational cost will be high.

A different approach is the control tables proposed by Yim [1] for locomotion of its first modular robots. These tables store basically a vector of every joint position for each instant. The controller scans the table sending positions to the actuators. This controller is simple and can be performed using low-end microcontroller. However, it is very flexible, to change the movement involves simply on recalculating the table.

The third approach is the bio-inspired one, wherein the position functions are obtained from nature models. IJspeert [2] was the pioneer in applying the lamprey Central Pattern Generators (CPGs) for robots locomotion, getting movement in its robotic salamander named Amphibot [3]. The CPGs are specialized groups of neurons that produce rhythms to control muscle activity of living beings. Kurakawa et al. [4] used as principle the half center oscillator pattern of Matsuoka [5] to generate the modular robot rhythms of the M-tran III, which can take different morphologies.

CPGs have the property that in steady state behave as fixed-frequency oscillators. Using this concept, Gonzalez-Gomez [6] proposed a simplified model of sinusoidal oscillators with which he was able to manage the locomotion of a robotic snake in two dimensions, getting five different gaits and obtaining a smooth natural motion.

The most used technique for humanoid robots locomotion has been for many years the ZMP (Zero Moment Point). ZMP calculates the trajectory of the center of mass to get a smooth and stable gait. Such complex robots like ASIMO and HRP are using this algorithm. However, to make ZMP work properly you need to make a very precise modeling of the robot and actuators. Therefore, more and more authors are applying bio-inspired models for humanoids robots. The ZMP approach in combination with CPGs is suggested by Or et al [7] for real-time control of a humanoid robot with a flexible spine.

II. MOVEMENT MODEL

The gait of a humanoid robot is similar to the gait of a human. Humanoids have 2 feet with 3 joints in each leg: hip, knee and ankle. The gait of a biped robot, as human, is periodic. Our gait consists on chaining a number of steps. If we walk at a constant speed, each step are exactly alike. Therefore, we can simplify the gait of a humanoid robot in *steps*.

[1]Universidad Rey Juan Carlos, Spain
[2]Universidad Rey Juan Carlos, Spain `franciscomiguel.rivas@urjc.es`
[3]Universidad Rey Juan Carlos, Spain `josemaria.plaza@urjc.es`

Another important feature is the symmetry: to get a linear gait the movement of a leg must be symmetrical to the contrary only with a certain delay, exactly the half of a step length. To get a sufficiently stable gait making the robot not to fall, all the actuators involved must be in rhythmic and properly synchronized, in short, coupled.

Following the CPGs approach, for each actuator involved in the gait we will get a function that defines its movement in each step. Thus, the parameters do not depend on the number of the movement frame positions like movements with the control table. The gait will only depends on the parameters of the characteristic function, for example sinusoidals.

As shown in figure 1 the movements of the actuators do not correspond to sine waves in most cases. However, we try to model these trajectories with basis functions characterized with the same parameters as a sine wave: frequency (ω), amplitude (A_0), phase (β) and offset(γ), corresponding to the function a(t) = A_0 * sin (ω t + β) + γ. Each actuator will have only 4 parametres.

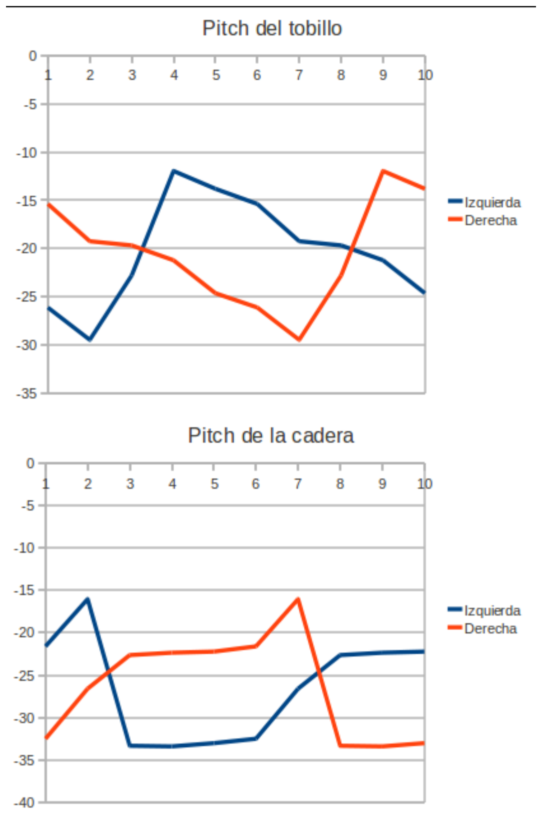

Fig. 1. Examples of gait actuator trajectories

From the 12 actuators involved in the gait we will avoid the yaw of the left and right hips. These yaw only control the hip rotation which affects to the trunk verticality. As we want the robot to walk with the greater stability both actuator are allways going to be set to 0. Each of the 10 remaining actuators will have a parameterized function with frequency, amplitude, phase and offset. So this parametrization will have 40 parameters to generate a gait fot the humanoid robot.

The frequency of all the actuators have to be the same as the step movement must start and end at the same time for all of them. Thus, we will have 3 independent parameters in each function (phase, amplitude, and offset) and a common frequency, so we really have 31 parameters.

In a straight gait the robot steps are symmetric and can obviate one side of the robot. We can set the movement on the left side from the right only by adding a fixed offset to the function that defines the movement. With this agreement we reduce the number of parameters to 16: 5 actuator with 3 parameters in each and the common frequency.

Fixing the hip offset as reference and defining the offset of the remaining joints on it, we can saved a parameter. That reference marks the initial position of the step, but irrelevant to the learning.

Another significant simplification is to fix the direct relationship on the actuators that perform the gait swing of the robot. The swing motion is performed by the robot to compensate the weight when lifting one leg during gait. The only actuators involved in this movement are the ankle and hip roll.

Changing the phases of the swing actuators means that the gait will not be rhythmic and will be uncoordinated. Based on this idea, we can set a direct dependence on the phase of the hip so that the phases of the swing actuator functions will only change varying the phase of the hip.

To control the amplitude of the swing we will introduce a parameter called *swing amplitude*. This amplitude is a constant by which all swing actuators will be multiplied. With this idea we can characterize the entire swing movement with a single parameter. So we eliminate the two amplitudes, which depend on the new parameter, the two phases, which are directly dependent on the phase of the hip pitch in order to have the rhythmic swing. Applying these concepts, all the gait of a humanoid robot will be reduced to control the 10 parameters listed on Table II.

Parameter	**min**	**max**	levels
1. Common frequency (γ)	0	-	-
2. Hip pitch amplitude	0	62	62
3. Hip pitch offset (γ)	-100	25	125
4. Knee pitch amplitude	0	65	65
5. Knee pitch phase (β)	-2π	2π	12
6. Knee pitch offset (γ)	0	130	130
7. Ankle pitch amplitude	0	60	60
8. Ankle pitch phase (β)	$-\pi$	π	6
9. Ankle pitch vertical offset (γ)	-75	45	120
10. Swing mplitude	0	100	100

TABLE I
PARAMETERS OF THE WALKING GAIT MODEL

III. NAO HUMANOID IN GAZEBO

We have chosen the Nao as humanoid robot and Gazebo as the simulator. There are two building blocks when programming the Nao support in Gazebo: create the robot model in Gazebo and the plugins that allow external applications to access to its sensors and actuators. Every plugin works as a dynamic library that is loaded at the start of Gazebo.

A. The humanoid model

The first step to build support to the Nao humanoid in Gazebo is to create a model inside the simulator with basic objects: links, joints and sensors. In addition to assembling all the pieces, the simulator offers Simulation Description Format (SDF), a way to write XML files defining the visual properties of our robot. To build the robot we joined some basic predefined blocks like hexahedrons, spheres, cylinders, etc. Several properties like geometrical size, mass, location in the world, inertia matrix and others can be associated to each block. These bodies are joined by hinges, which can give one or more directions of rotation. This provides our simulated model with different degrees of freedom. Every hinge in this model is built based on revolute joints, which commands the aperture of the hinge in the specified axis.

Following the next steps we designed and built every part of the simulated humanoid Nao: the arms, legs, body and head as can be seen in Figure 2. The arm and the camera will be detailed as representative samples. The properties of every part were set according to the Aldebaran Robotics documentation, including the center of masses, inertial matrices or the aperture range of the hinges, getting a realistic behavior at the simulation.

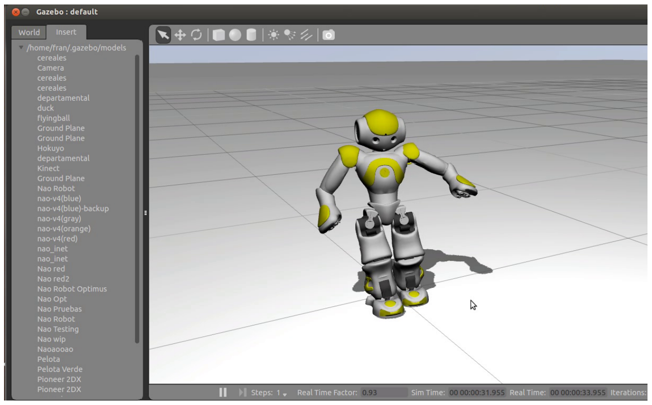

Fig. 2. Nao robot in Gazebo simulator

The humanoids arm in Gazebo is composed by humerus, ulna, radius and hand which were modelled as two rectangular hexahedrons, one for the humerus and another one for the rest. These two bodies are joined by an elbow hinge. This hinge gives the regular elbows degree of freedom, which allows to bend and stretch the arm. Besides, it allows another additional degree of freedom available in the real Nao, one that allows you to rotate the forearm with respect to the humerus. To achieve these two degrees of freedom at the elbow we used two hinge joints in the model with a dummy block of negligible mass in between. To add a skin is as simple as specify the path to a Collada12 file (.dae) in the visual tag. The Nao robot skins were built with Blender and added to this model.

The humanoid head is modelled as an sphere, with its skin, and incorporates two camera sensors, as in the real robot. To do that, it is necessary to add two sensors in the model. This sensor is created based on the SonyVID30 camera, already supported by Gazebo. This camera provides images with a size of 320x240 pixels, with an horizontal field of view (HFOV) of 60 degrees and a refresh of 10 frames per second. Once incorporated to the head, we developed the basic programming interface which allows to get data from the camera to see what the Nao robot in Gazebo is watching every moment. In order to move the head, we developed a neck hinge with two degrees of freedom: one for the pitch (tilt), and one for the yaw (pan).

B. Gazebo plugins

Once the Gazebo humanoid model was built, developers can program applications that use its sensors and command movements to its actuators. The simulator provides a low level API with different modules in order to have a realistic simulation, such as methods to get the position of a link, simulate a depth camera sensor or give an easy way to have a PID controller in the simulation, among others.

We write our robot applications using the JdeRobot open source robotics middleware [], so we developed several Gazebo plugins that wrap such low level API and provide a high level API, in terms of the ICE object interfaces used in that middleware. They provide external access to simulated Nao sensors and actuators from other JdeRobot components.

A Gazebo plugin must have at least three elements: the line in which you register the plugin, the `Load` method in which you load every element of the SDF file you want to take over control and the `OnUpdate` method that will run iteratively in a loop to do the work.

Several standard ICE interfaces in JdeRobot were used, like `Image` interface for Nao cameras or `encoders` interface for humanoid hinge positions. Other were specifically developed for this humanoid, like the body joints. Under the position commands a control loop is executed inside the plugin, it reads the hinge position using the Gazebo low level API to its encoders and commands orders to the hinge motors to keep the reference position sent to the high level API. The simulated robots neck is special: it is the only hinge that allows movement both in position and in speed. With this feature the neck can be commanded to an specific position or commanded to move to an specific speed in certain axis.

IV. EVOLUTIVE LEARNING

With the proposed motion model a huge number of walking gaits may be represented (not all). Discretizing the scope of the ten parameters of Table II and fixing the common frequency, there would be 2^{51} possible walking gaits. Many of them are nonsense, make the robot fall, others make the robot steps very long and slow o short and fast. Such huge space may be searched using a brute force approach but we propose the use of an evolutive algorithm that use a fitness function to explore such space of tentative walking gaits.

The fitness function is not analytical, as that would be cumbersome. Instead, we propose the use of a simulator, commanding the simulated humanoid robot to move according to each tentative walking gait during a given testing time interval. The results of that motion are objectively measured in terms of stability, speed and lineality of the generated motion. The combination of these criteria gives a quality function.

We have developed the software shown at Figure 3. The `Pose3D` plugin continuously registers the current motion position in 3D. The `NaoWalk` plugin implements the tentative walking gait, translating the parameters of a given walking gait into particular position commands to each robot joint along the testing time interval. It also accepts the parameters of each walking gait to be tested and places the simulated humanoid in a neutral starting position and posture before receiving the next `start-testing` command.

In addition, the `NaoWalk` plugin measures three quality indicators of the tested walking gait. The *speed* is measured as the distance already travelled along the testing interval from the starting point. The *lineality* is measured as the final side deviation from the ideal trajectory. The *stability* is measured as the accumulated deviation around the straight trajecdtory. In case of the robot falling the quality indicators of such parameter set are considered zero.

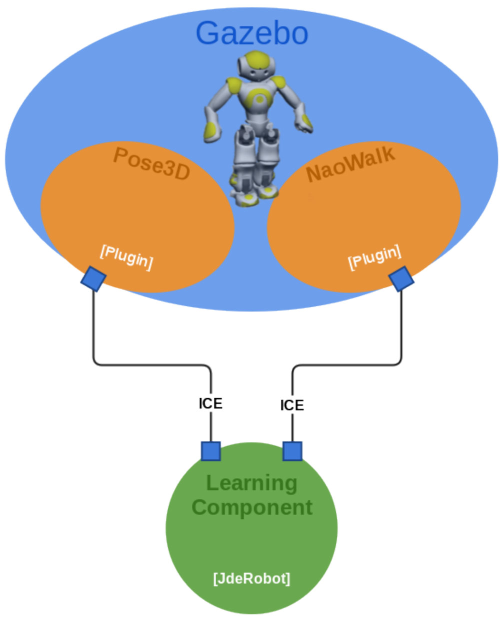

Fig. 3. Evolutionary algorithm component connecting to the Gazebo simulator, where the candidate walking gaits are tested.

The `Learning Component` implements the evolutive algorithm, including the evolutive operators like mutation, crossover, elitism, etc. and the fitness computation from the basic quality indicators. It keeps a population of tentative walking gaits that evolves depending on the fitness of its members and the application of operators. The population trends to move towards higher fitness values. This software component generates tentative gait parameters which are sent to the simulator to be tested. It receives the quality indicators from the `NaoWalk` plugin. This way it explores the parameter space, searching for an optimal walking gait.

V. CONCLUSIONS

In this contribution the developed support for Nao humanoid in Gazebo has been presented. A physical and appearance model of the robot parts was created. In addtion several plugins were programmed to provide access to robot sensors and actuators (joint motors) from external applications in the JdeRobot framework.

Legged robot motion is by far more complex than wheeled robot locomotion. A movement model for a humanoid robot has been proposed which uses coupled waves to coordinate humanoid limbs motion. The waves are defined with a set of ten parameters.

The span of possible walking gaits generated by different parameter values is huge. The support for Nao in Gazebo has been proposed to help in searching efficient walking gaits for the humanoid in such space. The work in progress in this line has been presented with two Gazebo plugins: one that provides the robot position in the simulated world in real time and a second one that implements a particular walking gait for a time interval, gathers gait quality indicators (like robot average speed, side deviation, whether it causes the robot to fall or not) and may restart the simulation with another different walking gait again.

Future lines include the full programming of the evolutionary algorithm that searches in the parameter space instead of using a brute force approach. A second envisioned work is to test the best walking gait found in the simulator into the real Nao robot.

ACKNOWLEDGMENT

This research has been partially sponsored by Project RoboCity2030-III-CM (S2013/MIT-2748), which is funded by Programas de Actividades I+D from Community of Madrid and cofunded by Structural Funds of the European Union.

REFERENCES

[1] M. Yim, Locomotion with unit-modular reconfigurable robot. PhD thesis, Stanford University, 1995.
[2] A. J. Ijspeert, Design of artificial neural oscillatory circuits for the control of lamprey and salamander-like locomotion using evolutionary algorithms. PhD thesis, Department of Artificial Intelligence, University of Edinburgh, 1998.
[3] A. J. Ijspeert and A. Crespi, Online trajectory generation in an amphibious snake robot using a lamprey like central pattern generator model, in Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA 2007), pp. 262268, 2007.
[4] H. Kurokawa, K. Tomita, A. Kamimura, S. Kokaji, T. Hasuo, and S. Murata, Distributed self-reconfiguration of m-tran iii modular robotic system, I. J. Robotic Res., vol. 27, no. 3-4, pp. 373386, 2008.
[5] M. Kiyotoshi, Mechanisms of frequency and pattern control in the neural rhythm generators, Biological Cybernetics, vol. 56, pp. 345-353, July 1987.
[6] J. Gonzalez-Gomez, Robotica Modular y Locomocion: Aplicacion a Robots Apodos. PhD thesis, Escuela Politecnica Superior, Universidad Autonoma de Madrid, 2008.
[7] O. Jimmy, A hybrid cpg-zmp controller for the real-time balance of a simulated flexible spine humanoid robot, Trans. Sys. Man Cyber Part C, vol. 39, pp. 547561, September 2009.