Chapter 8

Fuzzy PD Controller in NAO System's Platform

Edgar Omar López-Caudana and
César Daniel González Gutiérrez

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/63979

Abstract

Humanoid robotic platforms rarely achieve the desire trajectory because of the deviation generated during the robot walking. This problem is due to different circumstances such as robot manufacturing, wear and tear of mechanic parts, or variations of floor flatness. Currently, one of the humanoid robots on the market is the robotic platform developed by Aldebaran Robotics called NAO robot, and it is used for different purposes where the robot needs to navigate into controlled spaces. NAO presents the issue of deviation during walking; therefore, a Fuzzy PD Controller is developed and implemented for this platform to reduce the orientation error and to ensure reliability during navigation. Inertial sensors are used to get the orientation reference and for feedback of the closed-loop control. Consequently, a robust control was implemented and tested in different conditions of floor and velocity during the robot's navigation such as robot races and maze resolution. Experimental results show that fuzzy controller achieves significant improvements in the trajectories of NAO.

Keywords: NAO, Fuzzy Controller, Biped walking, trajectory, robot

1. Introduction

Currently, robotics is an area with many challenges such as stabilization, communications, manipulation, path planning, vision, and so on. The challenge becomes even more problematic when robots are designed for being autonomous with the task of interacting with the world independent of the environment. Humanoid robots are one of the different structures that exist, and it is a bio-inspired development. The idea of designing artificial intelligence entities is almost as old as humanity [1], and its origins focus on the creations of robots with human capabilities. Al-Jazari designed a humanoid structure as he wrote in the text "The Science of Ingenious

Mechanisms" in the thirteenth century [2]. In 1495, Leonardo da Vinci designed and built a mechanical structure that looks like an armored knight, and his work was inspired in the researches and achievements reached by the Greek culture [3].

Due to the advances in digital computation in the second half of the twentieth century, researches had the opportunity to introduce significant computation into robots for sensing, control, and actuation. The achievements were developed around 1970 by Kato and Vukobratovic researches [1] after the revolutionary advances in technologies in the field of legged robots. Then, the first robot that integrates capabilities of sensing, locomotion, and manipulation was WABOT-1, developed by Ichiro Kato et al. at Waseda University of Japan [4]. After that, the research group of Ichiro Kato developed WABOT-2, a robot which could read notes and play piano. In 1986, Honda began a research group about humanoid robotics, and in 1996, Honda introduced robot P2 which was followed by P3 in 1997 and by Asimo in 2002 [3]. In the world, many universities and research centers keep developing different humanoid robots. Aldebaran Robotics are working with NAO, Romeo, and Pepper robots, ASIMO is developed by Honda, MIT is working on Atlas and NASA's Valkyrie, QRIO by Sony, and HRP by Kawada [1].

Unlike industrial robots that operate in fixed environments, humanoid robots must operate under various and changeable environmental conditions and complete a wide range of tasks, with the characteristic of being autonomous [5]. The previous conditions stand different issues to solve such as locomotion, manipulation, artificial vision, cognition, and communication. Bipedal walking is one of the research issues in humanoid robots because it allows robots to interact and to move through the environment in which they are involved. Currently, one of the most used control methods to solve this problem is the zero-moment point (ZMP) and it sits within the support polygon of the robot's foot to ensure that the foot remains planted on the ground, assuming that friction is high enough to avoid slipping [1]. This algorithm was proposed by Vukobratovic and Stepanenko in 1972 [1]. Reaction mass pendulum (RMP) is a multibody inverted pendulum model, and it is inspired in human walking model [6].

The two previous algorithms are used to achieve bipedal walking; however, they need more algorithm layers to complete the full control of the robot for following trajectories. The most common algorithms are modulated playback, mathematical synthesis, passive dynamic walking, and physics-based heuristics [7]. Kastsian et al. [8] maximize the speed of a compass-like biped robot for a passive dynamic walking through applying the normal vector method. Yilmaz et al. in [9] used a natural ZMP reference trajectory and Fourier series approximation-based method which was proposed for a straight walk for the robot SURALP. Kurazume et al. [10] use a method for generating a straight-legged walking pattern for a biped robot using up-and-down motion of an upper body.

NAO is a robot developed by Aldebaran Robotics, and its locomotion is based on a dynamic model of the linear inverse pendulum inspired by work of Kajita and Tani [11] and is solved using quadratic programming [12]. The developed locomotion allows users for controlling the displacement of the robot in x, y, and z according to its body or global frame. One can generate motions through assigning values to lineal velocities or defining position targets. However, the robot is incapable of achieving with precision those targets, and it always has deviations on its trajectory. Aldebaran proposed a method called *ALVisualCompass* to correct this problem

http://dx.doi.org/10.5772/63979

based on artificial vision. Nevertheless, it consumes a lot of computational resources and the algorithm is dependent of the fact that at least part of the reference image is visible while the compass is running. This robot has different sensors such an accelerometer and a gyroscope which can be used to solve this issue. In this chapter, a PD fuzzy controller is proposed to eliminate this problem using inertial sensors and the implemented locomotion infrastructure in the platform.

2. Fuzzy PD controller

2.1. Mamdani Fuzzy Inference System

In 1965, Lotfi Asker Zadeh proposed the bases of a new logic system in his paper *Fuzzy Sets* [13]. Fuzzy logic was created to emulate the human logic and take decisions despite of the inaccurate received information from the environment. It is a system based on linguistic variables depicted by experts; for example, when humans describe the temperature of a day, they used words such as "hot," "cold," "too hot," and so on. In 1975, Ebrahim Mamdani et al. used the fundaments of fuzzy logic to create fuzzy logic controller for a steam engine and boiler combination in his paper "An experiment in linguistic synthesis with a fuzzy logic controller" [14]. Mamdani Fuzzy Inference System has three stages: fuzzification, inference system, and defuzzification.

Fuzzification process consists in mapping a crisp value into the membership functions along the discursive universe to get a fuzzy value [15]. **Figure 1** depicts the whole process, where the line is the crisp value (input to the system), and it is evaluated in the membership function. The value of that function is called a fuzzy value.

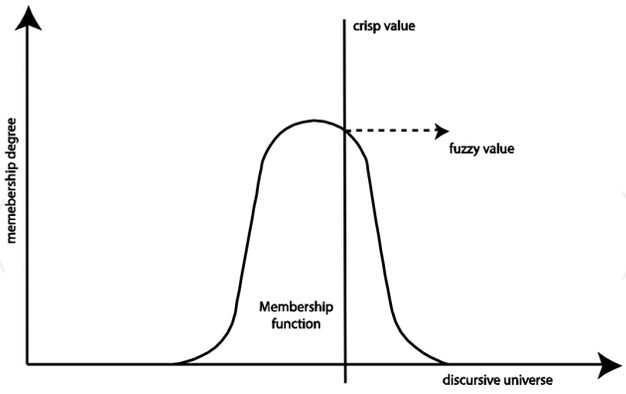

Figure 1. Fuzzification process.

The inference system performs the reasoning mechanism based on the rule if-then. This step has three components: a rule base, which contains the fuzzy rules of the system; a database, which contains all the membership functions of each input and output [15], and the last component is the reasoning component that executes the inference system. The next expression depicts the reasoning mechanism:

If x is A, then y is B

Some examples are “If pressure is high, then volume is small” and “If a tomato is red, then it is ripe.”

Finally, defuzzification consists in extract a crisp value from the output fuzzy sets. In general, there are five methods for defuzzification: centroid of area, bisector of area, mean of maximum, smallest of maximum, and largest of maximum [15]. **Figure 2** depicts the complete Mamdani Fuzzy Inference System.

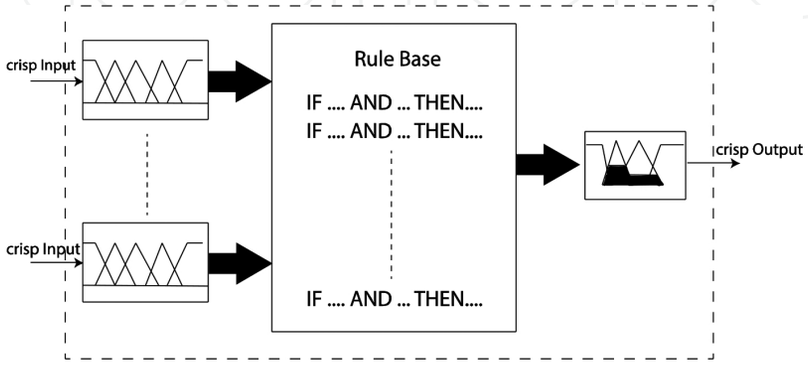

Figure 2. Mamdani Fuzzy Inference System.

2.2. Fuzzy PD controller

A classical closed-loop control system is depicted in **Figure 3**. The reference is compared with the actual value of the system to get a difference. The error is the input to the implemented control system, which response is the input to the model of the plant. One of the most used controllers is the PD controller, and its equation is described as follows:

$$u(t) = K_p e(t) + K_d \dot{e}(t)$$

where $e(t) = r(t) - y(y)$

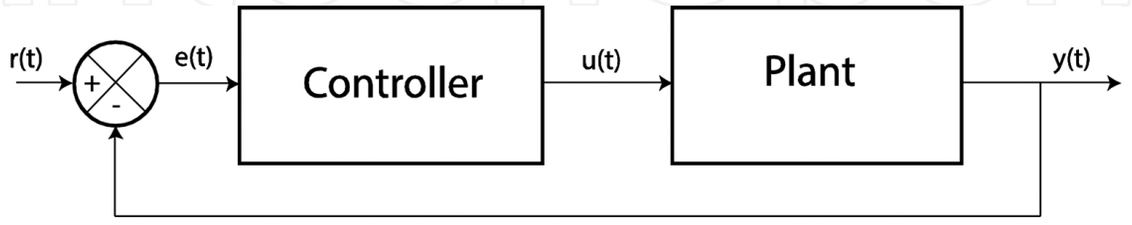

Figure 3. Basic control system.

In **Figure 4**, a step response with the desired value is shown. The different pointed marks are conditions that can be described with linguistic variables. For example, the pointed dots in

http://dx.doi.org/10.5772/63979

Figure 4 can be described as "A value with positive error and negative derivative of error for dot 1," "A value with negative error and zero derivative of error for dot 3," "A value with zero error and zero derivative of error for dot 7," and so on. For each condition, the controller varies its output and can be described as "positive," "negative," "zero," and so on.

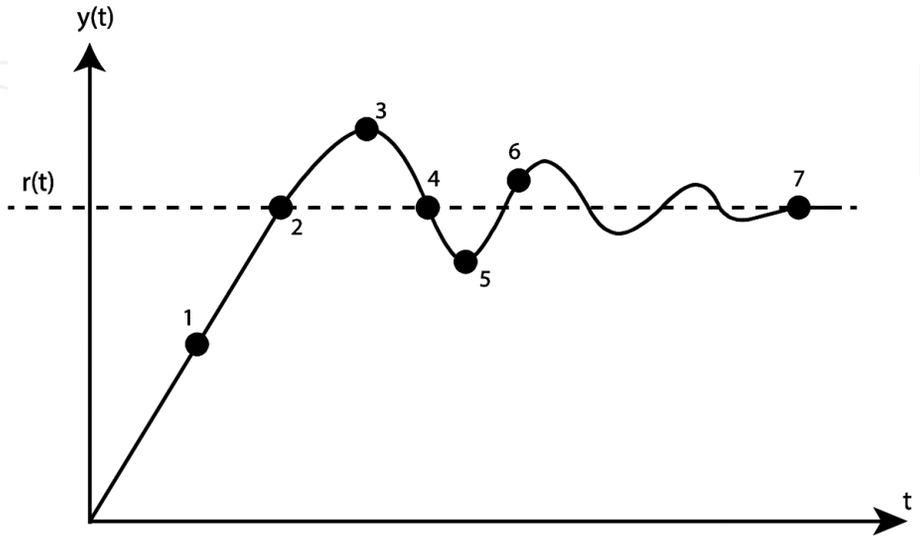

Figure 4. Step response.

For the above-mentioned examples, a fuzzy PD controller can be implemented with two inputs (error and derivative of the error), where each input must have n-membership functions that represent the linguistic variables. **Figure 5** depicts the final design of a fuzzy PD control system.

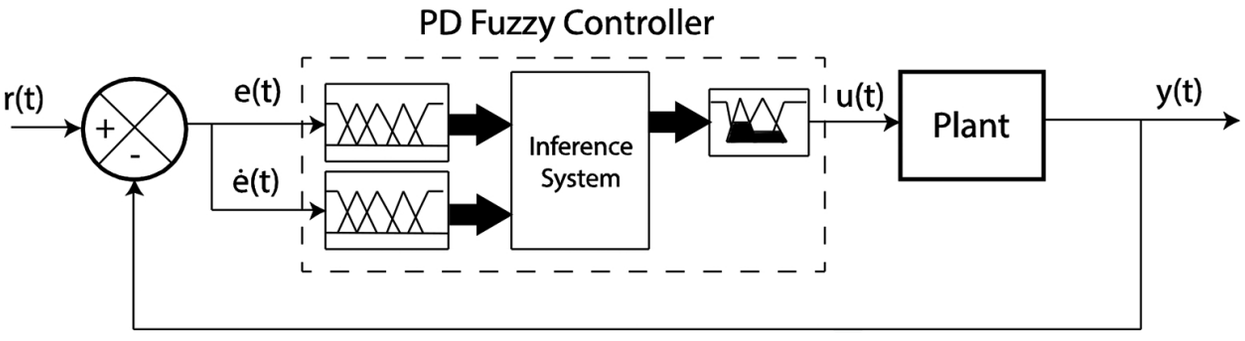

Figure 5. Fuzzy PD control system.

2.3. Robot environment

Bipedal walking robots have different challenges depending on the environment in which they are enrolled. The kind of soil in which robots walk affects directly in the dynamics of the robot due to the friction forces that are generated during motions. If the soil coefficient friction is

enough high, the robot could not walk and also will demand high consumption in motors; on the other hand, if the biped robot walk over soil with low coefficient friction, it could slide and fall down. Also, when the soil presents impurities such as small rocks, fissures, mud and so on, the robot stabilization can be affected and also its trajectory. Another issue is the mechanical deterioration, the engines of the motors that allow to move robot articulations will present a wear depending on the manufacturing material, and then, the mathematical model of the robot will be affected. In addition, there could be external forces that may change or eliminated bipedal walking.

NAO is a humanoid robot created in 2006 by Aldebaran Robotics in France. This robot has 25 degrees of freedom (DOF) and has a height of 58 cm. It also contains inertial sensors such as accelerometer and gyroscope, ultrasonic sensors, capacitive touch sensor, bumpers, infrared sensors, and cameras. This robot is used for competitions, presentations, teaching, social assistant, and exhibitions. **Figure 6** shows the NAO robot structure.

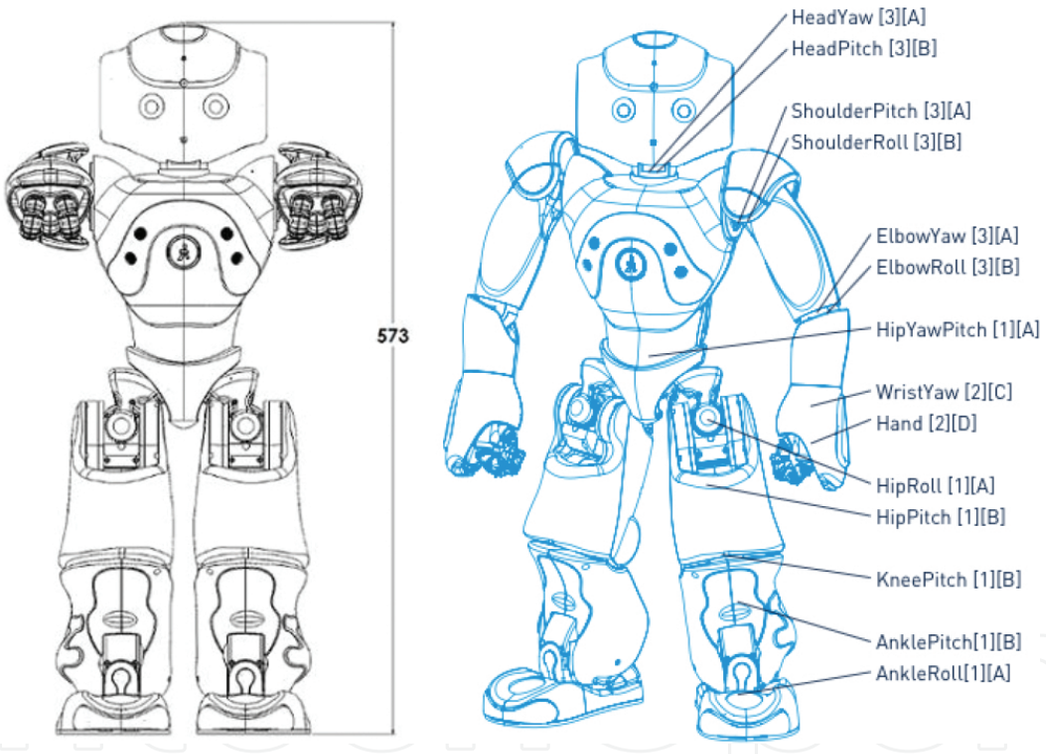

Figure 6. NAO robot and locations engine (pictures taken from *Aldebaran Robotic: NAO Documentation*).

According Aldebaran, NAO System presents nest characteristics:

- 25 Degrees of Freedom (Degrees of Freedom)
- Stepped omnidirectional
- Two grasping hands
- ATOM Z530 CPU 1.6 GHz

http://dx.doi.org/10.5772/63979

- Flash Memory SDRAM 256 MB/2 GB
- Inertia sensor with two-axis gyroscope and 3-axis accelerometer.
- 1x RJ45 Ethernet port—10/100/1000 Base T and Wi-Fi IEEE 802.11b/g
- 2x video cameras (960p @ 30fps), better sensitivity in VGA. View—239° horizontal, vertical view of 68°. HD resolution.
- Object recognition
- Detection and Face Recognition

Text to Speech:

Two speakers and multilanguage voice synthesis (Spanish and English Preloaded)

Four microphones and voice recognition multilanguage (Spanish and English preloaded)

Supports multiple programming languages.

It has special programming and simulation software.

According to degrees of freedom of the Robot

Head: 2 DOF

Arms: 5 DOF

Pelvis: 1 DOF

Legs: 5 DOF

Hands: 1 DOF

Image Engine 2.27 Locations

Battery:

Type: Lithium-Ion

Voltage: 21.6/2.15 Ah

Maximum load voltage: 2 A

Charging time: 5 h

Battery life: 60 min (using principal)

90 min (normal use)

Mother board:

- Processor: Intel ATOM Z530
- Cache memory: 512 KB

- Clock speed: 1.6 GHz
- Ram memory: 1 GB
- Flash memory: 2 GB
- Wireless: IEEE 802.11b/g/n

Camera:

- Model: MT9M114
- Resolution: 1.22 MP
- Pixels: 1288 × 968
- Number of images: 30 images per second

Engines:

- Motor Type 1:
- Model: 22NT82212P
- No load speed: 8300 rpm ± 10%
- Load speed: mNm 68 ± 8%
- Continuous torque: 16.1 mNm max

Motor Type 2:

- Model: 17N88208E
- No load speed: 8400 rpm ± 12%
- Load speed: 9.4 mNm ± 8%
- Continuous Torque: 4.9 mNm max
- Motor Type 3:
- Model: 16GT83210E
- No-load speed: 10,700 rpm ± 10%
- Load speed: 14.3 mNm ± 8%
- Continuous torque: 6.2 mNm max

As mentioned before, robots present a wide range of issues depending on the environment. NAO robot also presents these problems. The robot includes different locomotion methods to control the robot based on a mathematical model of the inverse pendulum. One can set a target velocity on x-, y-, and z-axis or even in positions to follow a trajectory. However, the robot exposes often deviations of the generated trajectories.

Some work on NAO robots and motion analysis can be reviewed in [16], where a report on omnidirectional walk engine focused on use of robot soccer is done; or in [17], where some

http://dx.doi.org/10.5772/63979

kind of control strategy which contains a controller using quantitative feedback theory is used to establish a control method. Both cases show different environments in order to get better movement control methods, but our proposal is made of a PD fuzzy controller. In [18], we can observe how a neural network or fuzzy logic techniques can be utilized to achieve basic behavioral functions necessary to mobile robotics system, as we show in next section in our proposal.

2.4. Proposed solution

NAO has two three-axis inertial sensors. Accelerometer provides measurements in range of ±2g, while gyroscope can measure angular velocities in range of ±500°/s. It is important to have reliable values to perform control techniques implementation and can obtain by using potentiality of accelerometer and gyroscope data. According to Higgins [19], a complementary filter can be used to acquire an accurate angle position. **Figure 7** shows the body framework, and it illustrates the way the robot can follow different trajectories applying lineal velocity on x-axis and adjusting the path by applying turns (z angular velocities) in z-axis [20].

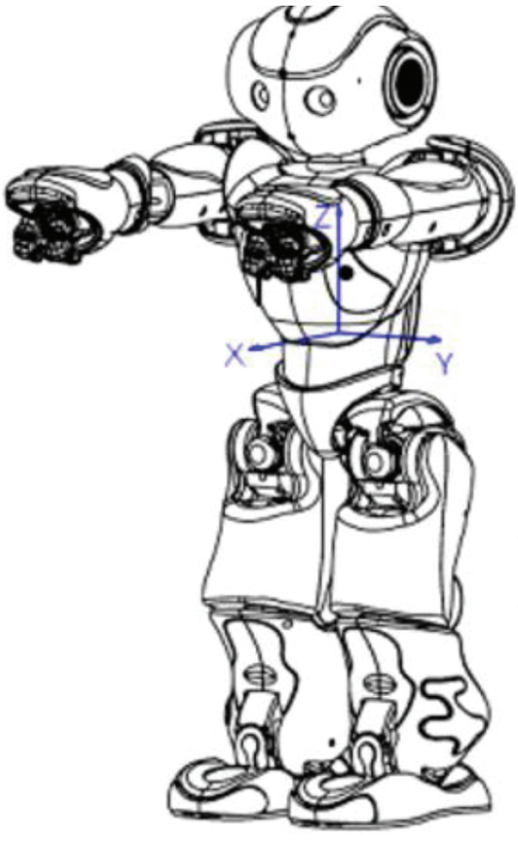

Figure 7. NAO robot body frame (picture taken from *Aldebaran Robotics: NAO Documentation*).

In this section, two controllers are proposed. The first proposed PD fuzzy controller was designed with the aim of following a given angle reference respect to z-axis. **Figure 8** shows

the whole process to build a fuzzy PD controller. According to that, the first step requires to define the system type. Multiple input–single output is presented for this controller due to the error, derivative of the error, and the z angular velocity output. The next step consists in selecting the discursive universe for each input and output. Discursive universe is selected according to the characteristics of the variable and the expert. To achieve that, the expert must know the robot behavior. Therefore, NAO was configured to display the error and its derivative with respect an initial reference in console. After getting a set of values, error and derivate ranges were defined. The error range goes from –40° to 40° and the normalized derivate error

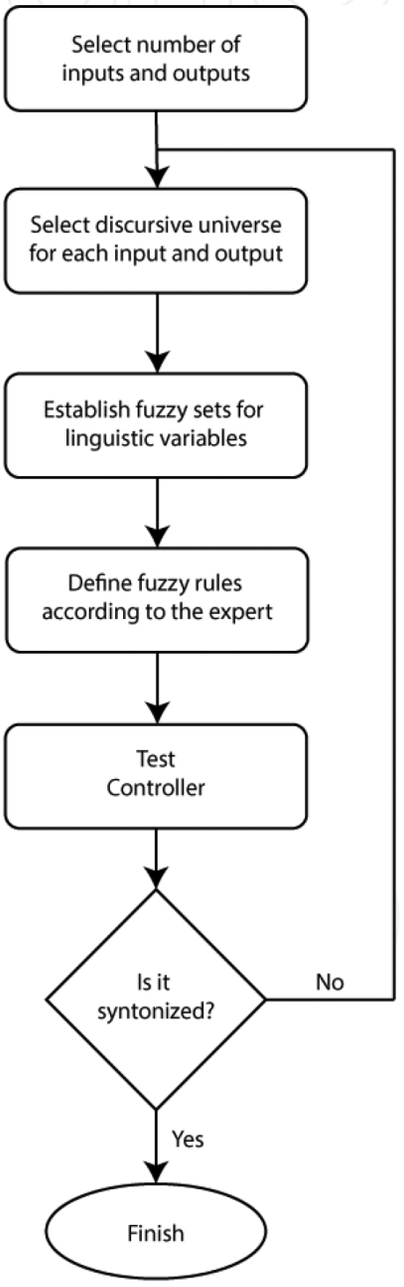

Figure 8. Flow chart for tuning Fuzzy Controller.

http://dx.doi.org/10.5772/63979

goes from -1 to 1. Similarly, the robot was programmed to make turns around its axis in a range of a normalized velocity. A first proposed discursive universe for the output was settled from -0.4 to 0.4.

Once the discursive universe is established, a number of fuzzy sets must be defined to describe linguistic variables. The error input (**Figure 9**) has five membership functions labeled as VERY NEGATIVE (VN), NEGATIVE (N), ZERO (Z), POSITIVE (P), and VERY POSITIVE (VP), and its discursive universe goes from -0.6981 to 0.6981 (in radians). Second input (derivative of the error (**Figure 10**)) is designed with three membership functions labeled as NEGATIVE (N), ZERO (Z), and POSITIVE (P), and the discursive universe was defined from -1 to 1 with the intention to be normalized. Finally, the output (**Figure 11**) has five membership functions labeled as VERY RIGHT (VR), RIGHT (R), ZERO (Z), LEFT (L), and VERY LEFT (VL) which describe the direction and magnitude of z velocity.

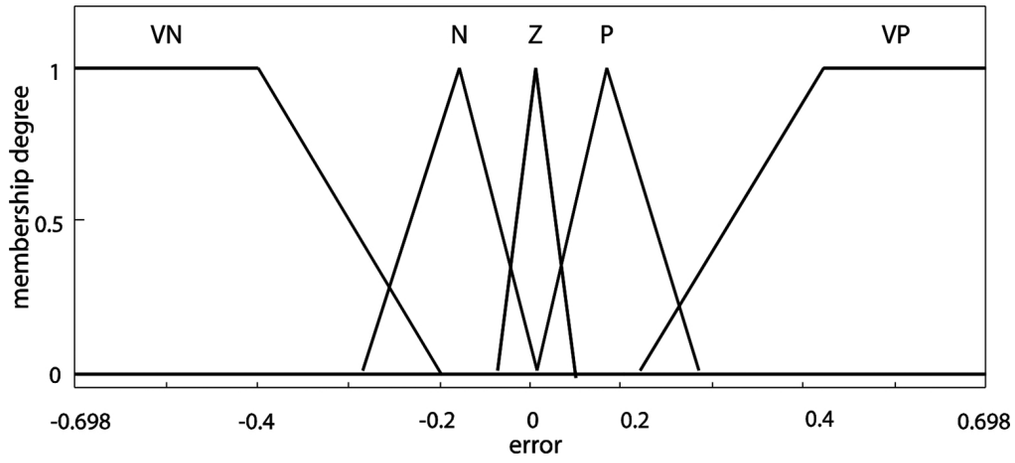

Figure 9. Membership functions of error input from controller 1.

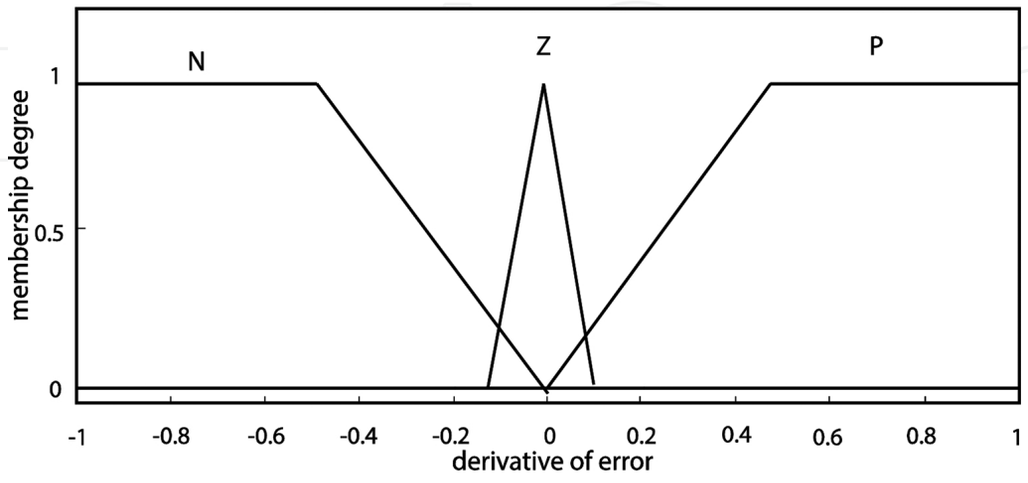

Figure 10. Membership functions of the derivative of error input from controller 1.

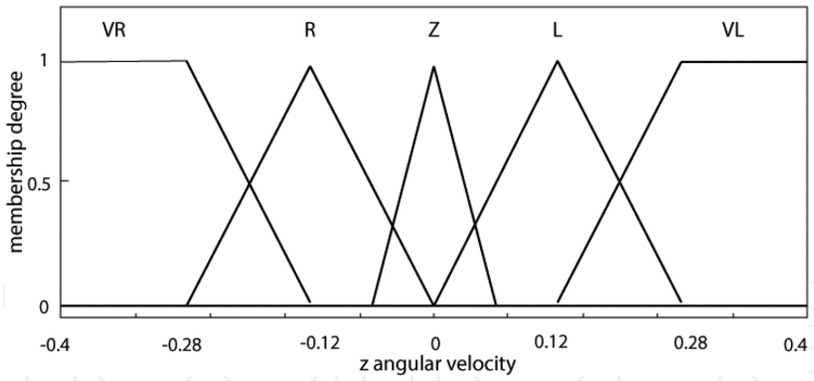

Figure 11. Membership functions of z angular velocity output.

Table 1 shows inference system that the PD fuzzy controller used, the first row encloses fuzzy sets of derivative of the error, and the first column contains the fuzzy sets of the error. The rest of the table spaces corresponds to the output decision.

$e\backslash\dot{e}$	**N**	**Z**	**P**
VN	VL	VL	VL
N	L	L	L
Z	R	Z	L
P	R	R	R
VP	VR	VR	VR

Table 1. Fuzzy rules for controller 1.

Figure 12 shows the response of the system when the PD controller is applied. The response presents oscillations around the desired value. Then, according to **Figure 8**, a change must be

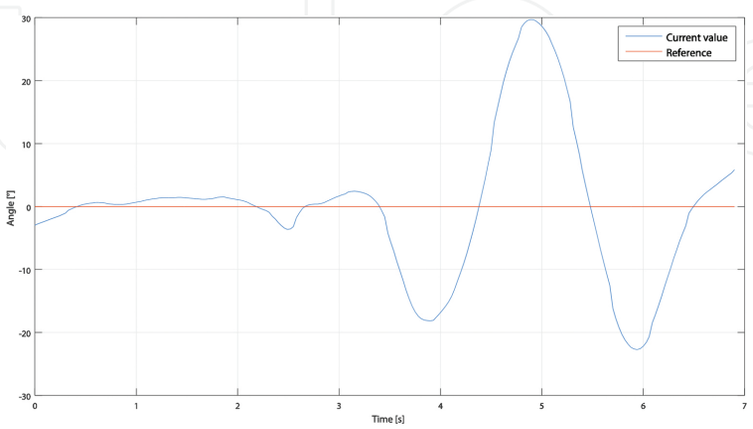

Figure 12. Output response of the first designed controller.

http://dx.doi.org/10.5772/63979

applied in membership functions, rules, or discursive universe. In agreement with the response, the evaluation of the inference system is correct, because when the current moves away from the reference, the controller tries to correct the error. However, the applied correction is more than necessary. For instance, the modification must be done in the output discursive universe or the output fuzzy sets.

After repeating the process, the final output for defuzzification is presented in **Figure 13**. As shown, the main change against **Figure 11** was the discursive universe. The output range was reduced to eliminate the oscillations in the response and the results are shown in Section 3.

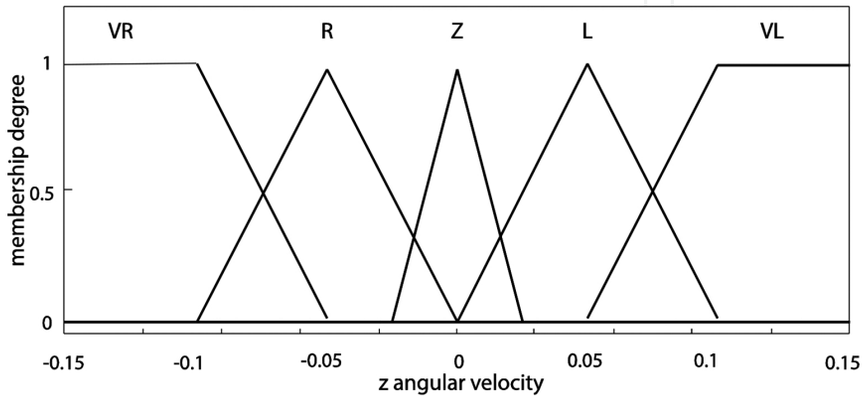

Figure 13. Membership functions of z angular velocity output for controller 1.

The second controller has the task to perform turns around its own z-axis; therefore, any force in x and y was applied. The structure and the design process of the PD fuzzy controller was the same as the first controller. The characteristics of the numbers of membership functions in each input and output and the inference system were preserved. The parameters that changed in this design was the discursive universe: the error input is established from $-\pi/2$ to $\pi/2$, the derivative of the error goes from -1 to 1, and the output goes from -0.3 to 0.3, as shown in **Figure 14**, to achieve bigger changes in reference and reduce the settled time.

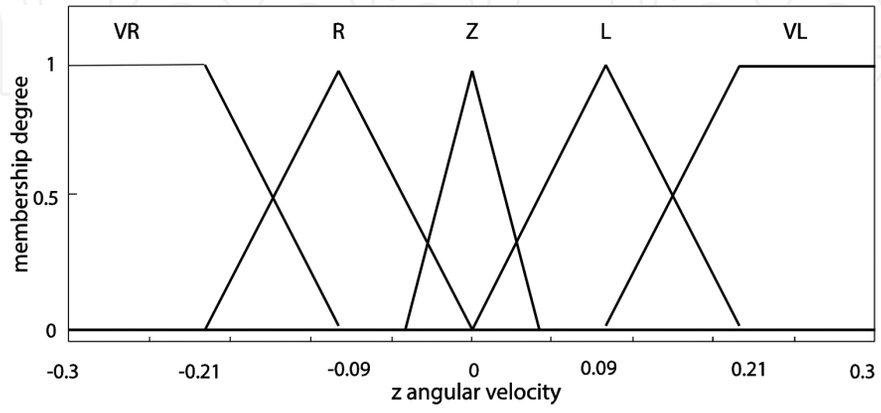

Figure 14. Membership functions of z angular velocity output for controller 2.

3. Analysis and results

Figures 15–17 depict the responses of the system in different situations. **Figure 15** shows the response when the robot makes a $\pi/2$ radians turns to the right. In this case, the robot just applies angular velocity around z-axis, while the x and y velocities stay in 0. The overshooting was of 4.09%, the settling period was achieved in 4.2 s, and the integral absolute error (IAE) was of 2.83. It is important to have a small overshooting because that is translated into less controller force and less oscillations. **Figure 16** depicts the reference of a given angle that NAO had to follow to keep a straight walking during 10 s. From that, one can analyze that without control the robot could go to its left because the effort that the controller was performed had a negative velocity in z, which is translated into turning to the right. In **Figure 17**, A straight walking is presented; the difference is that the operator changes manually the trajectory of the robot. As shown in the figure, the robot reacted and return to the given reference.

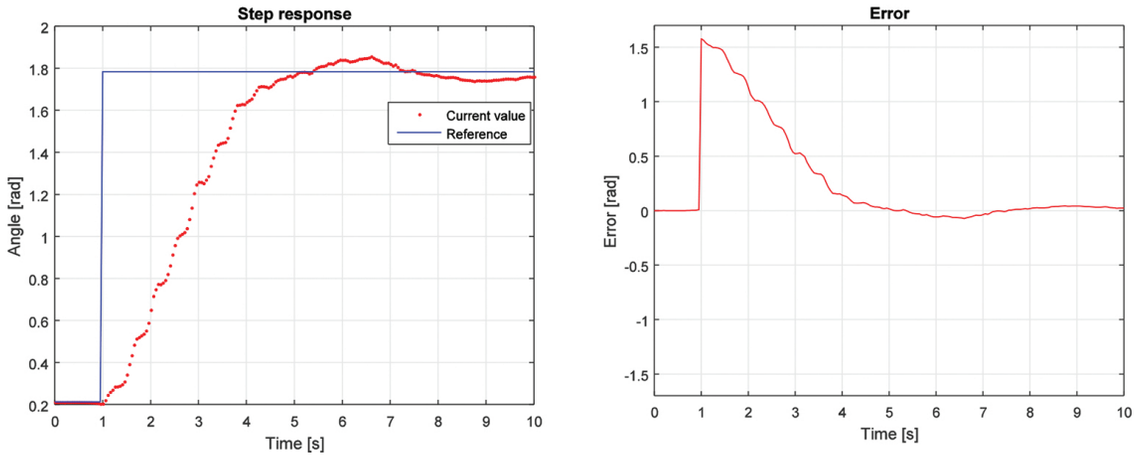

Figure 15. Step response of $\pi/2$ radians.

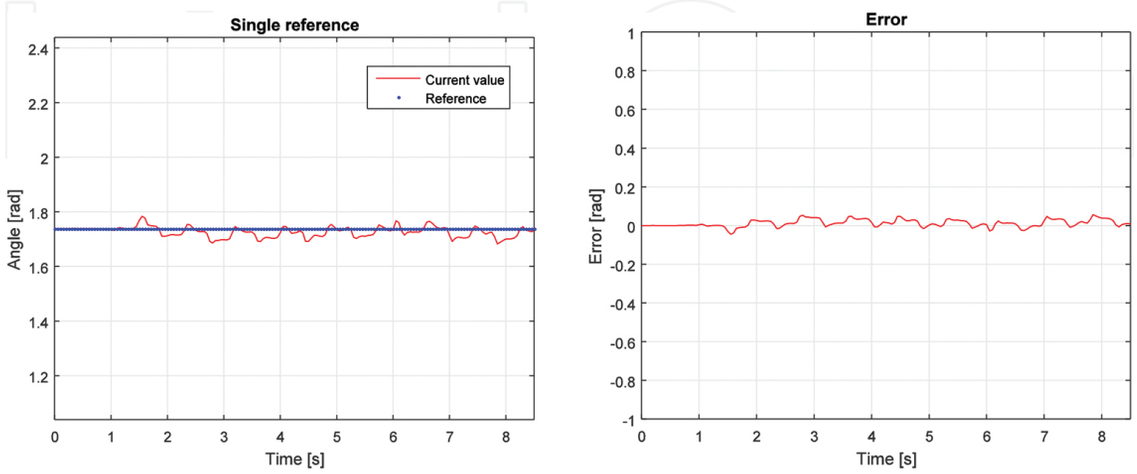

Figure 16. Following a single reference during biped walking.

http://dx.doi.org/10.5772/63979

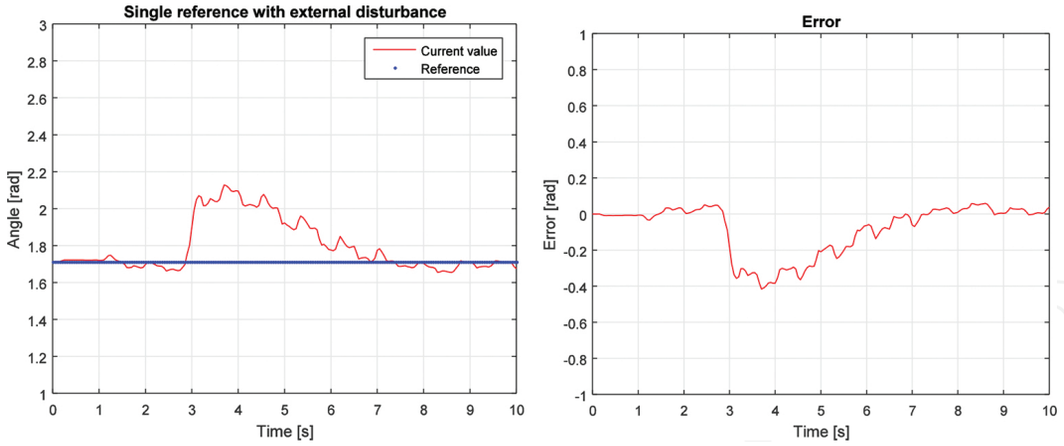

Figure 17. Disturbance response during biped walking.

Figure 18 shows the main problem of the robot during walking process. The deviation that the robot had with and without control is presented. According to the test, NAO was able to walk straight for 1.86 m and achieved the desired final position with controller 1. On the contrary, when the same target was established, taking out the control, the robot started to walk to the left and it had a displacement of 0.44 m to the left, which provoked that the robot got out from the black lines. The different deviations that the robot presents during each walking is unpredictable because it depends on the NAO's motors, the soil, and the mathematical model and that is the reason of applying a fuzzy PD controller to the plant. In Annex 1, the code is presented to build the fuzzy system controller.

The developed controller was used to take NAO to the Robotics and Artificial Intelligence National Contest 2015. In this challenge, there were different categories where the robots can

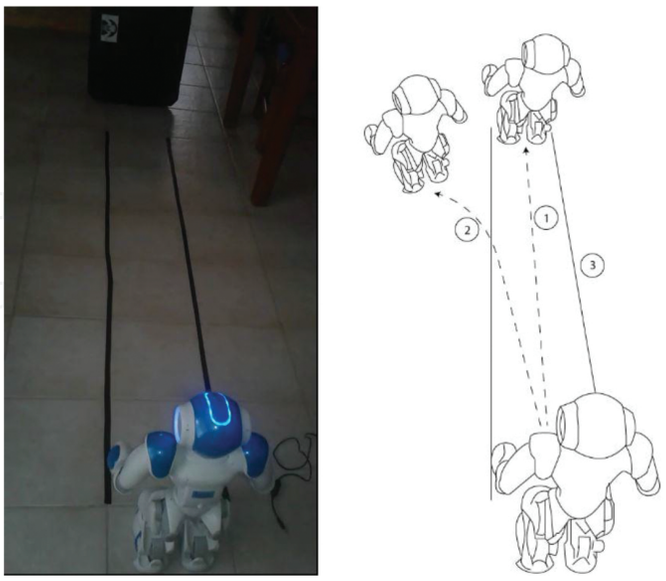

Figure 18. Comparison of biped walking with and without control.

participate. The two chosen categories were Individual Race and Resolution Maze. Individual Race consisted in walking a distance without leaving the track in the shortest time. The rules for this category are the following:

1. The robot must have an activation signal.
2. The robot must start moving within 10 s after the referee gives the whistle.
3. If the robot leaves the track, it has 3 s to return.
4. When the robot finishes the competition, the participant must deactivate it until the referee gives the signal.

In Resolution Maze, NAO had to find the exit of a given maze in which the robot was introduced. The rules for these categories are presented:

1. The robot must have an activation signal.
2. The robot must start moving within 10 s after the referee gives the whistle.
3. If the robot leaves the maze by the start entry, the robot is disqualified.
4. The robot must say the correct Naomark number when NAO finds one to accumulate points.
5. The robot has 7 min to accomplish the maze. If not, the largest displacement is registered.
6. When the robot finishes the competition, the participant must deactivate it until the referee gives the signal.

Performing straight walking and fast turns are considerations to accomplish the challenges. Therefore, the controller was tested for this contest and it allowed NAO to perform the competition. The pseudo-code for accomplish the Individual Race is the following:

Individual R ace pseudo-code

1. Touch NAO head to start routine
2. Read Z angle from IMU sensors to set the reference
3. Start walking by applying 0.8 x velocity, 0 y velocity and 0 z velocity
4. While (touch Head is FALSE)

    Get error and its derivative

    Evaluate fuzzy PD controller

    Update z velocity

5. Enter in rest mode

---

http://dx.doi.org/10.5772/63979

The pseudo-code for accomplish the Resolution Maze is the following:

Resolution Maze pseudo-code

1. Touch NAO head to start routine

2. Read Z angle from IMU sensors to set the reference

3. Thread 1:

While (touch Head is FALSE)

While (distance <0.35 m)

Walk applying 0.8 x velocity

Get error and its derivative

Evaluate fuzzy PD controller

Update z velocity

While (wall in front = true)

Calibrate reference

If (turn right = FALSE)

Turn right using PD fuzzy controller

Else

Turn left using PD fuzzy controller

4. Thread 2:

While (touch Head is FALSE)

If (Naomark detected = TRUE)

Activate speech: "#number Naomark founded"

5. Enter rest mode

Figures 19–21 are evidence of NAO participation at the contest.

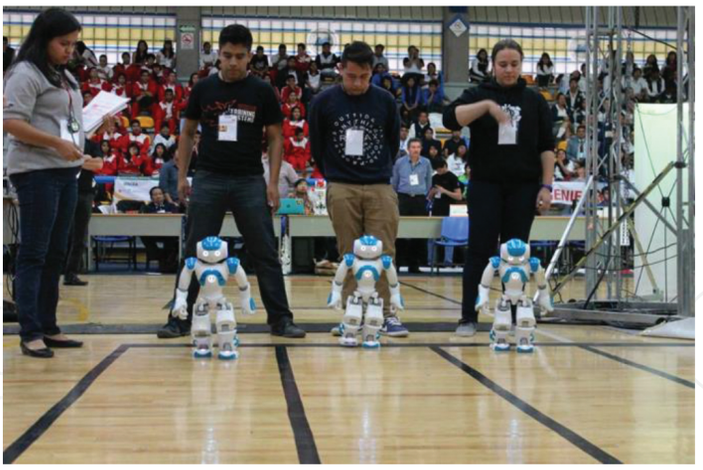

Figure 19. Individual race competition.

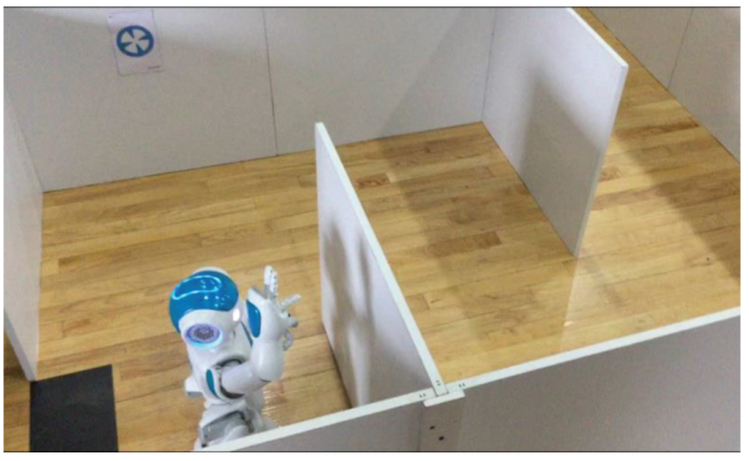

Figure 20. Maze resolution.

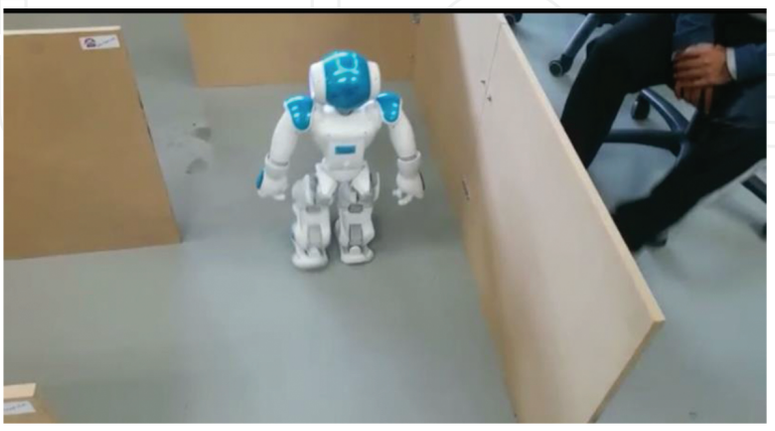

Figure 21. Testing the control algorithm.

http://dx.doi.org/10.5772/63979

4. Conclusion

The presented control approach validates that a PD fuzzy controller is enough to allow the robot following different references without having its mathematical model. Tests were executed in different environmental conditions, changing principally the soil and walking velocity that caused the robot to present different noise magnitudes during the sensor reading process. Despite that, the robot was able to achieve the trajectory, allowing to conclude that the controller was robust. Moreover, when external disturbances were applied manually, the robot also concludes its tasks. The participation of the robot in Robotics and Artificial Intelligence National Contest 2015 supports the validation of the controller.

The NAO platform is good in terms of testing different robotics algorithms. The high-level programming allows users to implement different tasks because the platform is really complete. It has different sensors such as sonar sensors, bumpers, infrared sensors, cameras, Wi-fi connectivity, DMA processes, and so on. Moreover, there are three ways of programming: Choregraphe, Python, and C++. Also provides API that includes not only very high-level functions to perform complex task such a speech recognition, simple walking task, face recognition, and so on, but also one can develop its own high complex function using Python or C++ programming.

5. Annex

The following code is written in Python Language, and it contains classes to construct Fuzzy Systems and some membership functions.

```
class FuzzyTriang:
def __init__(self,pi,pm,pf):
self.pi = pi
self.pm = pm
self.pf = pf
def alpha(self,x):
if(x>=self.pi and x<self.pm):
return (1/(self.pm-self.pi))*(x-self.pi)
elif(x>=self.pm and x<=self.pf):
return -(1/(self.pf-self.pm))*(x-self.pf)
else:
return 0
def alphaCut(self,x):
```

```
cuts = list()
cuts.append([self.pi,0])
cuts.append([x/(1/(self.pm-self.pi))+self.pi,x])
cuts.append([x/(-(1/(self.pf-self.pm)))+self.pf,x])
cuts.append([self.pf,0])
return cuts
def getCenMax(self):
return self.pm
def getPI(self):
return self.pi
def getPM(self):
return self.pm
def getPF(self):
return self.pf
def getPuntos(self):
return [self.pi,self.pm,self.pf]
class FuzzyLinSh:
def __init__(self,p1,p2,ext,side):
self.p1 = p1
self.p2 = p2
self.ext = ext
self.side = side
def alpha(self,x):
if(self.side == "l"):
if x <= self.p1:
return 1
elif (x > self.p1 and x < self.p2):
return -(1/(self.p2-self.p1))*(x-self.p2)
else:
return 0
```

http://dx.doi.org/10.5772/63979

```
else:
if x <= self.p1:
return 0
elif (x > self.p1 and x < self.p2):
return (1/(self.p2-self.p1))*(x-self.p1)
else:
return 1
def alphaCut(self,x):
cuts = list()
if(self.side == "l"):
cuts.append([self.ext,0])
cuts.append([self.ext,x])
cuts.append([x/(-(1/(self.p2-self.p1)))+self.p2,x])
cuts.append([self.p2,0])
else:
cuts.append([self.p1,0])
cuts.append([x/(1/(self.p2-self.p1))+self.p1,x])
cuts.append([self.ext,x])
cuts.append([self.ext,0])
return cuts
def getCenMax(self):
if(self.side == "l"):
return (self.ext + self.p1)/2
else:
return (self.ext + self.p2)/2
def getP1(self):
return self.p1
def getP2(self):
return self.p2
def getPoints(self):
```

```
return [self.p1,self.p2]
class FuzzySystem:
def __init__(self, input1, input2, outputs, rules):
self.input1 = input1 # error
self.input2 = input2 # derivada
self.output = outputs
self.rules = rules
self.vn = 0
def evaluar(self, x, y):
alphas = []
auxreglas = [[0, 0, 0, 0, 1], [0, 0, 0, 0, 1], [0, 0, 0, 0, 1],
[0, 0, 0, 1, 0], [0, 0, 0, 1, 0], [0, 0, 1, 0, 0],
[0, 0, 0, 1, 0], [0, 0, 1, 0, 0], [0, 1, 0, 0, 0],
[0, 0, 1, 0, 0], [0, 1, 0, 0, 0], [0, 1, 0, 0, 0],
[1, 0, 0, 0, 0], [1, 0, 0, 0, 0], [1, 0, 0, 0, 0]]
nr = len(auxreglas) #inputs
nc = len(auxreglas[0]) #outputs
nr = len(auxreglas) # inputs
nc = len(auxreglas[0]) # outputs
nE = len(self.input1)
nD = len(self.input2)
alphase = list()
alphasd = list()
# Evaluacion del sistema
for i in range(0, nE):
alphase.append(self.input1[i].alpha(x))
for i in range(0, nD):
alphasd.append(self.input2[i].alpha(y))
alphasin = list()
for i in range(0, nE): # error
```

http://dx.doi.org/10.5772/63979

```
for j in range(0, nD): # derivative
if alphase[i] < alphasd[j]:
alphasin.append(alphase[i])
else:
alphasin.append(alphasd[j])
# Rules evaluation
for i in range(0, nr):
for j in range(0, nc):
auxreglas[i][j] = auxreglas[i][j] * alphasin[i]
# Find maximums
alphasF = []
centrosM = []
for i in range(0,nc):
aux = 0
for j in range(0,nr):
if(auxreglas[j][i] > aux):
aux = auxreglas[j][i]
alphasF.append(aux)
centrosM.append(self.output[i].getCenMax())
num = 0
den = 0
for i in range(0,nc):
num = num + alphasF[i]*centrosM[i]
den = den + alphasF[i]
if (den!=0):
res = num/den
self.vn = res
return self.vn
else:
return 0
```

Acknowledgements

The contributions of several students from Tecnologico de Monterrey, Mexico City Campus, are gratefully acknowledged, specially Kennet García Oviedo and Enrique Jiménez Vázquez. This work has been supported by School of Engineering and Sciences from Tecnologico de Monterrey.

Author details

Edgar Omar López-Caudana* and César Daniel González Gutiérrez

*Address all correspondence to: edlopez@itesm.mx

Tecnológico de Monterrey, Campus Ciudad de México, Ciudad de México, México

References

[1] B. Siciliano, O. Khatib (Eds.), *Springer handbook of robotics*. Berlin: Springer, 2008.

[2] M. E. Rosheim, *Robot evolution: the development of anthrobotics*. New York, NY: Wiley, 1994.

[3] S. Behnke, Humanoid Robots—From Fiction to Reality, *KI-Zeitschrift*, vol. 4, pp. 5–9, Dec. 2008.

[4] V. Bruce, A. W. Young, *In the eye of the beholder: the science of face perception*. Oxford, England; New York: Oxford University Press, 1998.

[5] B. Adams, C. Breazeal, R. A. Brooks, B. Scassellati, Humanoid robots: a new kind of tool, *IEEE Intell. Syst.*, 2000, 15(4):25–31.

[6] K. Sreenath, A. K. Sanyal, The reaction mass biped: equations of motion, hybrid model for walking and trajectory tracking control, 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, 26–30 May 2015, IEEE; 2015, pp. 5741–5746.

[7] Y. Bar-Cohen, C. L. Breazeal, *Biologically inspired intelligent robots*. Bellingham, WA, USA: SPIE Press, 2003.

[8] D. Kastsian, E. Oertel, M. Monnigmann, Optimal parameters for stable walking of a compass-like biped robot, 2014 IEEE Conference on Control Applications (CCA), 8–10 October 2014, Juan Les Antibes, IEEE; 2014, pp. 814–819.

[9] M. Yilmaz, U. Seven, K. C. Fidan, T. Akbas, K. Erbatur, Circular arc-shaped walking trajectory generation for bipedal humanoid robots, 2012 12th IEEE International

Workshop on Advanced Motion Control (AMC), Sarajevo, 25–27 March 2012, IEEE; 2012, pp. 1–8.

[10] R. Kurazume, S. Tanaka, M. Yamashita, T. Hasegawa, K. Yoneda, Straight legged walking of a biped robot, 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2–6 Aug. 2005, IEEE; 2005, pp. 337–343.

[11] S. Kajita, K. Tani, Experimental study of biped dynamic walking in the linear inverted pendulum mode, Proceedings of the 1995 IEEE International Conference on Robotics and Automation (volume: 3), Nagoya, 21–27 May 1995, IEEE; 1995, pp. 2885–2891.

[12] P. Wieber, Trajectory free linear model predictive control for stable walking in the presence of strong perturbations, IEEE-RAS International Conference on Humanoid Robots, Genova, Italy 2006, pp. 137–142.

[13] L. A. Zadeh, R. R. Yager, Fuzzy sets and applications: selected papers. New York: Wiley, 1987.

[14] E. H. Mamdani, S. Assilian, An experiment in linguistic synthesis with a fuzzy logic controller, *Int. J. Man-Mach. Stud.*, 1975, 7(1): 1–13.

[15] J.-S. R. Jang, C.-T. Sun, E. Mizutani, Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence. Upper Saddle River, NJ: Prentice Hall, 1997.

[16] Nima Shafii, Abbas Abdolmaleki, Nuno Lau and Luis Paulo Reis. Development of an omnidirectional walk engine for soccer humanoid robots. *Int. J. Adv. Robot Syst.*, 2015, 12:193. doi:10.5772/61314.

[17] S. Wen, J. Zhu, X. Li, A.B. Rad, X. Chen. End-point contact force control with quantitative feedback theory for mobile robots. *Int. J. Adv. Robot Syst.*, 2012, 9:236. doi: 10.5772/53742.

[18] E. Al Gallaf, Transputer Neuro-Fuzzy Controlled Behaviour-Based Mobile Robotics System, Mobile Robotics, Moving Intelligence, Jonas Buchli (Ed.), InTech, 2006. doi: 10.5772/4726. Available from: http://www.intechopen.com/books/mobile_robotics_moving_intelligence/transputer_neuro-fuzzy_controlled_behaviour-based_mobile_robotics_system.

[19] W. Higgins, A comparison of complementary and Kalman filtering, *IEEE Trans. Aerosp. Electron. Syst.*, May 1975, vol. AES-11(3):321–325.

[20] Aldebaran Robotics. NAO Documentation. Retrieved from http://doc.aldebaran.com/2-1/home_nao.html.

IntechOpen