Design and Experimental Validation of a Ballbot: A Self-balancing Omnidirectional Robot

Pedro Henrique de Jesus*, Cairo Lúcio Nascimento Júnior†,
Douglas Soares dos Santos†, Sérgio Ronaldo Barros dos Santos‡
* Serviço Nacional de Aprendizagem Industrial (SENAI), Pindamonhangaba-SP, Brazil
Email: pedro.jesus@sp.senai.br
† Instituto Tecnológico de Aeronaútica (ITA), São José dos Campos-SP, Brazil
Emails: {cairo, dsoares}@ita.br
‡ Universidade Federal de São Paulo (UNIFESP), São José dos Campos-SP, Brazil
Email: sergio.ronaldo@unifesp.br

2024 IEEE International Systems Conference (SysCon) | 979-8-3503-5880-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/SysCon61195.2024.10553418

***Abstract*—This article presents the design, modelling and control of a Self-balancing Omnidirecional Robot, also known as ballbot, for performing autonomous navigation (path following) and object tracking in structured environments using vision feedback. A real ballbot has unstable and non-linear dynamics with second-order constraints, and is classified as an underactuated system. These features require the implementation of a robust and efficient embedded control system for allowing the realization of attitude and navigation control. In this context, the development of this work was divided into six stages: 1) design and mechanical construction of the omnidirectional wheels and chassis using a 3D printer; 2) mathematical modeling of the prototype developed in Matlab/Simulink and creation of a realistic simulated model in CoppeliaSim simulator; 3) design of LQR and P/PI controllers to control X and Y position and orientation respectively using the mathematical model; 4) evaluation of designed controllers using the realistic model developed; 5) implementation of the controllers in the embedded system and integration with inertial sensors and actuators; 6) experimental validation of the proposed control system for autonomous navigation and object tracking. The experimental results have shown that the developed ballbot is able to perform path following and object tracking efficiently.**

I. Introduction

A Self-balacing Omnidirectional Robot (Ballbot) is a mobile robot that incorporates the characteristics observed in an inverted pendulum which is balanced and moved on top of a sphere [1].

A Ballbot has numerous potential applications in several sectors of our society, such as working cooperatively in domestic and industrial environments transporting loads, acting as an information assistant in malls and hospitals, assisting senior citizens in sit-to-stand and walking movements, guiding leading visually impaired people in an unknown environments, autonomously mapping outdoor and indoor environments, performing mobile target tracking, and remote monitoring of large facilities.

One of the first ballbots was created in 2005 specifically to study the problem of dynamically unstable robots with a single point of contact with the ground [2]. The ballbot might carry out agile movements due to the ball rolling action and has inherently nonlinear unstable dynamics caused by the inverted pendulum. Unlike statically stable robots, dynamically unstable robots can accelerate and decelerate quickly and suffer a greater influence of gravity. This feature requires that the control system should effectively deal with the instability and nonlinearities that exist in this type of robot [3].

However, since a ballbot has unstable and nonlinear dynamics with second-order constraints and is classified as an underactuated system, designing effective attitude and navigation controllers for it is definitelynot a trivial task [4], [5].

This work aims to describe the development phases for designing, modeling and control of a ballbot used to perform autonomous navigation and object tracking in structured environments using inertial and vision sensors.

Initially, the robot chassis and its omnidirectional wheels were built using a 3D printer. Considering the robot physical parameters, including the power limitation of the motor drive power, 2 simulation models were developed: a mathematical model for Matlab/Simulink and a more realistic model for the CopelliaSim robot simulator. Using the mathematical model, the LQR and P/PI controllers were designed and tuned to control X and Y robot speed/position and orientation. Then they were evaluated using the realistic model in CoppeliaSim.

For autonomous navigation, the ballbot receives its 3D position computed by an external vision system allowing it to follow the desired path defined by a set of waypoints positioned in its environment. For object tracking the ballbot uses an embedded camera to detect and locate the target object in relation to its own position. Then the vision based controller uses this informationto generate the control signals to the wheel actuators.

The main contribution of this work is to present a methodology for the construction of a ballbot, from the mechanical aspects to the implementation of the autonomous navigation and object tracking algorithms, including the experimental validation. The algorithms developed for this work and some demonstration videos are publicly available at https://github.com/pedrophj/ballbot_LMI.

This article is organized as follows. Section II details the 2D model of the ballbot, as well as the parameters of the 3D simulation. Section III describes the methodology, constructive

aspects and the control architecture. Simulation results for autonomous navigation and object tracking are presented in Section IV. And finally, conclusion and future works are described in Section V .

A. Related Works

In the last years, several studies were carried out for improving the stabilization control design of the self-balancing omnidirectional robot; however some few works present the complete stages of development of a ballbot, that goes from its mechanical structure to the implementation of the embedded control and guidance systems, allowing a full autonomous navigation and object tracking in indoor environments.

Considering the existing literature about the path tracking and navigation, in the work of [6], the authors have only presented an attitude control for generating the velocity commands for the three wheels. In [7], a trajectory tracking approach based on waypoints for executing maneuver and avoiding collision with static and dynamic obstacles in cluttered environments is proposed. In [8], the authors have described an experimental trial to validate the control policies derived from the dynamic model and Dijsktra's algorithm for surveillance and monitoring applications. In [9], a Fuzzy-based trajectory controller is proposed for performing the straight and circular motions in simulation. In [10], a Sliding Model Control (SMC) approach for trajectory tracking in simulation considering uncertainty and disturbance is proposed. In the [11], the authors have presented the design and implementation of a Fuzzy/PD controllers for controlling the attitude and tracking trajectories.

Regarding the mechanical and control system aspects, in the work of [12], the authors have presented the mechanical construction of a ballbot and modelling procedure using the Euler–Lagrange equations to derive the dynamic model. A Linear Quadratic Regulator with Integral (LQR + I) controller is proposed to keep the location around a point. In other work [5], the ballbot modeling and LQR controllers design to control the angular and X-Y position is proposed. However, the experimental validation is conducted using previously developed prototype.

II. Background

In this section, we introduce all general aspects and some concepts employed throughout this work such as the used environment.

A. Ballbot Model

Figure 1 shows the simplified free body diagram of the ballbot represented by identical XZ and YZ planes. The free body diagram of the system is composed of a wheel (in orange), a rigid body (in blue) and a rod of length L used to connect the center of mass to the wheel. Note that X, Y and Z axes belongs to the inertial coordinate system whereas X_B, Y_B and Z_B axes composes the body coordinate system of the ballbot.

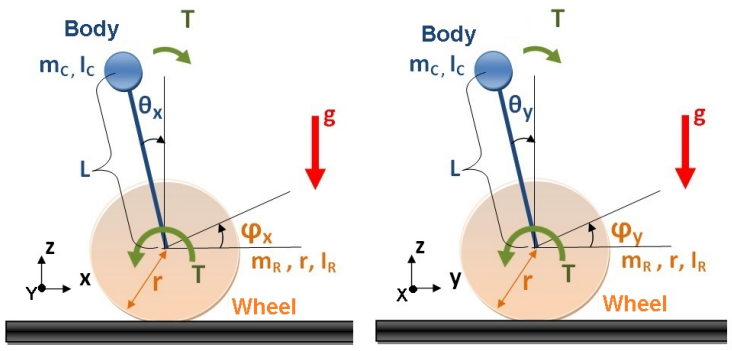


Fig. 1. Simplified ballbot planar model

Let be consider the angular acceleration on the sphere, wheel angle (φ_x) with respect to an initial horizontal reference point, and wheel angular velocity ($\dot{\varphi}_x$) as input of the simplified ballbot planar model and pitch (θ_x) angle measured in relation to the Y axis, angular velocity ($\dot{\theta}_x$) around the Y axis as output of the system. For describing the mathematical model, we use the parameters shown in Table I.

The nonlinear differential equations of the system is described in the form of a matrix as presented in eq. (1).

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} x_2 \\ \dfrac{m_c g L sin(x_1) + c_1 u + m_c r L sin(x_1) x_2^2}{I_c + m_c L^2 + m_c r L cos(x_1)} \\ x_4 \\ -ru \end{bmatrix} \quad (1)$$

where $c_1 = -I_r - (m_r + m_c)r^2 - m_c r L cos(x_1)$, $u = \ddot{\varphi}$, $x_1 = \theta_x$, $x_2 = \dot{\theta}$, $x_3 = x$, and $x_4 = \dot{x}$.

It is assumed that the sphere is a rigid body and to be on contact with the ground without friction and slipping, and the wheel dynamics are disregarded due to their operation occurs at low angular speed.

The linearization of the nonlinear model is performed for an operating point that sets the angle change around the X and Y axes to be approximately zero. Thus, we can present the nonlinear ballbot plane model as a linear model in space states form.

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 24.84 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ -0.73 \\ 0 \\ -0.12 \end{bmatrix} u \quad (2)$$

where $\dot{x_1}$, $\dot{x_2}$, $\dot{x_3}$ and $\dot{x_4}$ are the angular velocity observed on the body of the ballbot and u is the control signal.

B. Virtual Ballbot Model in CoppeliaSim

The simulator is an important tool for the control system design during the evaluation stage, since the system instability due to incorrect adjustment of the controllers might cause the ballbot to fall on the ground and consequently damage

TABLE I
PARAMETERS USED FOR MODELLING THE VIRTUAL BALLBOT

Parameter	Value	Unit
Acceleration of gravity	9.81	m/s^2
Ball radius (r)	0.12	m
Ball mass (Mc)	0.58	kg
Moment of inertia of the sphere in X	0.0134	kgm^2
Moment of inertia of the sphere in Y	0.0134	kgm^2
Moment of inertia of the sphere in Z	0.0134	kgm^2
Diameter of the cylindrical body	0.2	m
Height of the body (L)	0.3	m
body mass	1.6	kg
Moment of inertia of the body at X	0.0413	kgm^2
Moment of inertia of the body in Y	0.0413	kgm^2
Moment of inertia of the body in Z	0.0720	kgm^2
Omniwheel diameter	7	cm
Omniwheel thickness	1.3	cm
Omniwheel mass	50	g
Omniwheel moment of inertia at X	6.16×10^{-6}	kgm^2
Omniwheel moment of inertia in Y	6.16×10^{-6}	kgm^2
Omniwheel moment of inertia in Z	1.22×10^{-4}	kgm^2

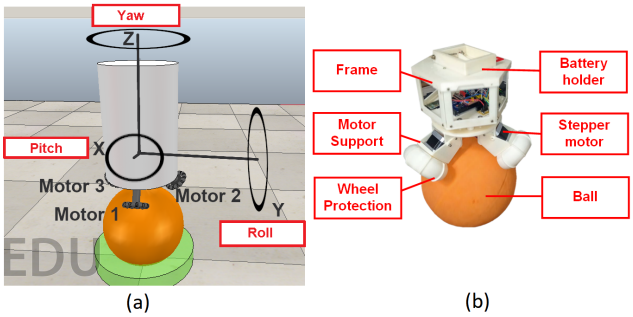
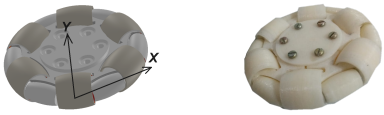


Fig. 2. Proposed actual and simulated Ballbot



(a) Model (b) 3D printed

Fig. 3. Designed Omnidirecional Wheels

the actual prototype. In this work, we developed a virtual 3D model of the ballbot using the CoppeliaSim simulator to evaluate the performance of the stabilization and trajectory tracking controllers. This model reproduces the behavior of the dynamics observed in the real prototype with reliability and precision, taking into account disturbances such as wind and irregularities on the ground. Figure 2(a) shows the virtual ballbot developed for the CoppeliaSim simulator.

The attitude and position controllers are designed using the mathematical model described in Section II and are implemented in a script using the LUA language. This language allows to execute the control algorithms implemented in a script directly in CopelliaSim without using an external API.

All the parameters shown in Table I are derived from the actual ballbot and are used to build the virtual model executed in CoppeliaSim. In CoppeliaSim each object in the scene can be edited separately, so that the dynamic properties of each part of the ballbot can be adjusted based on the values obtained from the prototype.

III. Development of the Self-Balancing Omnidirectional Robot

A. Mechanical and Motor Aspects

The actual ballbot frame (Fig. 2(b)) is composed of a set of elements manufactured using a 3D printer and fixed using screws. The designed frame has a hexagonal shape with two layers for installing the on-board electronics, camera and battery. Each actuator is composed of a set of motor and omnidirectional wheel, positioned at 120° isometric angles and fixed to the frame using specifically designed supports. The positioning defined by the actuators allows eliminating non-holomonic movement restrictions, allowing the ballbot to execute displacement in X and Y directions (longitudinal and lateral movements) and around their own vertical axis.

The motors are to be chosen to overcome the inertial imposed by the sphere, i.e., they must have enough torque to rotate the sphere producing an angular velocity that is translated into a linear velocity. The designed omnidirectional wheels (Fig. 3) have twelve 90 degree rollers symmetrically arranged to allow the ballbot to move along the Y axis and reduce body shake whereas conventional wheel rotation produces movement on the X axis.

The 4-wire bipolar stepper (NEMA 17, model 17HS4401) has 1.8 degree per step for smooth motion and requires a maximum current of 1.7A/phase for operation. This motor needs of 200 steps to complete a full 360 degree per turn and has a maximum load of 4.28 kg/cm. The angular speed of the motor depends on the frequency of the command generated by the drive in which affects the response of the platform to perform stabilization and holonomic displacement.

B. Embedded Electronics Systems

The ballbot uses a microcontroller (Arduino Mega) which it receives the acceleration and angular velocity measurements provided by the Inertial Measurement Unit (IMU) and the orientation given by the compass, and also generates the control signals of each of the motors through the drive DVR8825 composed of MOSFET transistors configured in two H bridges. The IMU and compass transmits the measurements performed using a I2C serial communication bus with a frequency of 100 kHz. Figure 4 presents the block diagram of the embedded electronic system implemented in the ballbot.

Using bluetooth communication, the ballbot receives the relative position data calculated using the localization system and also the desired position references (waypoints) from the ground station computer for carrying out the navigation. On the other hand, the ballbot transmits the measurements realized by the IMU and compass, and also control signals generated by

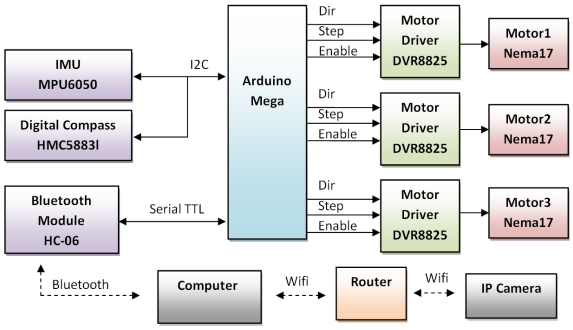


Fig. 4. Embedded system diagram

the motor drives using bluetooth communication (telemetry) to a ground station computer for user visualization.

The localization system uses the captured and processed images to estimate the position and velocity of the ballbot. An external camera captures the images from the environment and transmits the images for the ground station computer using a bluetooth connection.

C. Control System Architecture

The position control system denoted as XZ and YZ consists of two LQR controllers (Fig. 5) that are used to control the ballbot movements in relation to the X and Y axes on the global reference frame. The XZ controller takes the desired position (X_{ref}) or desired speed ($\dot{X}_{ref}$) of the ballbot as input signals whereas receiving the body angle (θ_x), angular speed of the body ($\dot{\theta}_x$), actual position (X), and ball speed ($\dot{X}$) in the X direction. In turn, the YZ controller has as input signal (reference) the desired position (Y_{ref}) or desired speed ($\dot{Y}_{ref}$) whereas receiving the body angle (θ_y), angular speed of the body ($\dot{\theta}_y$), actual position (Y), and ball speed ($\dot{Y}$) in the Y direction.

The XZ and YZ controllers (Fig. 5) provide the control signals to command three ballbot actuators. The control outputs of the XZ and YZ controllers refer to the angular acceleration on the ball (A_x and A_y) and are computed using (3) and (4).

$$A_x = \theta_x K_1 + \dot{\theta}_x K_2 + (\dot{X} - \dot{X}_{Ref})K_3 + (X - X_{Ref})K_4 \quad (3)$$

$$A_y = \theta_y K_1 + \dot{\theta}_y K_2 + (\dot{Y} - \dot{Y}_{Ref})K_3 + (Y - Y_{Ref})K_4 \quad (4)$$

where $K = [K_1, K_2, K_3, K_4]$ is the gain vector for the LQR controllers.

The body orientation controller (yaw controller) consists of P/PI controllers that can be individually selected by the user to perform the rotation movement considering the response time and the expected accuracy for the desired angular position. The P/PI controller uses the yaw angle (Ψ_{ref}) or angular speed of the body ($\dot{\Psi}_{ref}$) as the desired reference whereas receiving the measured Yaw (ω) and Yaw rate ($\dot{\omega}$) signals of the ballbot. From the defined P/PI control law, the controller computes the

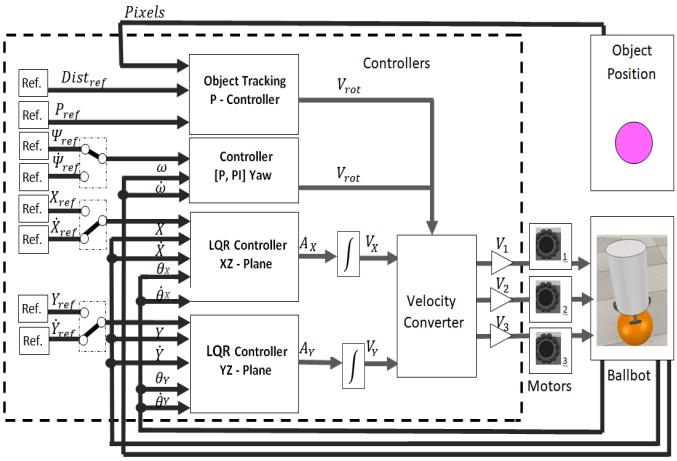


Fig. 5. Ballbot control system

control signal V_{rot} in which refers to the command of angular speed of the ballbot.

The actuation commands (V_x) and (V_y) are obtained from the output provided by the XZ and YZ controllers using a numerical integration method. A conversion module combines the actuation commands (V_x and V_y) relative to X and Y axes and also the body angular speed command V_{rot} to generate the rotation speed commands (V_1,V_2 and V_3) for each of the engines. Using (5) , (6) and (7), the rotation speed commands are calculated as described in [6]. Note that the conversion module takes into account the angle and isometric position of the actuators.

$$V_1 = -V_x cos(\beta) + V_{rot} \quad (5)$$

$$V_2 = (0,86V_x + 0.5V_y)cos(\beta) + V_{rot} \quad (6)$$

$$V_3 = (-0,86V_x + 0.5V_y)cos(\beta) + V_{rot} \quad (7)$$

where V_1, V_2 and V_3 are the commands for driving each one of the engines and V_{rot} is the body angular speed command and V_x and V_y are the actuation commands relative to X and Y axes provided by the XZ and YZ controllers.

1) Complementary Filter and Computer Vision: The attitude of the ballbot (θ_x and θ_y) is computed using an Inertial Measurement Unit (IMU) consisting of a three-axis accelerometer, and a three-axis gyrometer. Through a complementary filter, the Pich and Roll angles may be calculated from the measurements performed by the accelerometer and gyrometer. This method allows eliminate the problem of bias error caused by numerical integration. In addition to Pitch e Roll angles, the yaw angle of the ballbot can be measured through a digital compass.

The complementary filter (Eq. 8 and 9) consists of a combination of a low-pass filter and a high-pass filter which are used to obtain the measurements of θ_x and θ_y of the ballbat. The low-pass filter uses the angle measurements obtained by the accelerometer whereas the high-pass filter receives

the measurements obtained by the gyrometer. Note that the variables (weights) defined for each one of the filters were empirically adjusted. The coefficient α must be set with a value in the range of 0.5 to 1. In this work, due to the high vibration produced by the omnidirectional wheels the variable α was adjust to (i.e, 99% for the measurements obtained by the gyrometer and 1% for measurements provided by the accelerometer).

$$\theta_x \leftarrow \alpha(\theta + \theta_{gyrox}) + (1-\alpha)\theta_{accx} \tag{8}$$

$$\theta_y \leftarrow \alpha(\theta + \theta_{gyroy}) + (1-\alpha)\theta_{accy} \tag{9}$$

where θ_x and θ_y are the filtered Pitch and Roll angles, respectively, θ_{gyrox} and θ_{gyroy} are the angles obtained by the gyrometer using a numerical integration method, θ_{accx} and θ_{accy} are the angles obtained by the accelerometer using a trigonometry approach and α is filter coefficient.

To estimate the position of the ballbot during navigation and object tracking, a computer vision approach was implemented. Using the OpenCV library, we create a function to define the color threshold by keeping only the color of interest. The captured image of the environment is distorted, thus it is necessary to apply the homography matrix to transform a trapezoidal image into a two-dimensional rectangular image. After filtering, the contour detection is perform to estimate the position by converting pixels to centimeters.

2) Tuning: To adjust the gains of the LQR controllers, we need to assign weights (10 and 11) to the matrices Q and R. The Q matrix were chosen to give more importance to the Pitch and Roll angles, aiming to keep the robot balanced under the ball. The R matrix was tuned to consider the control action, so that the actuator responds quickly with less effort as possible. Controller gains are shown in Table II.

$$Q = \begin{bmatrix} 50 & 0 & 0 & 0 \\ 0 & 20 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 20 \end{bmatrix}_{4X4} \tag{10}$$

$$R = [50]_{1X1} \tag{11}$$

where Q matrix is the weights on the states and R matrix is the weights on the control input.

D. Guidance Control System

Waypoint navigation may be understood as the robot movement based on a list of points. Whenever the robot gets close enough to one waypoint (reaching a waypoint radius), the desired point changes to the next ones until reaching the last point, making the the robot completes the navigation.

The coordinate of each one of the waypoints is defined by (W_X and W_Y), and the waypoint radius is estabilished by (R_W). Thus, the Euclidean distance between the position of the robot and the next waypoint can be computed using (12) and the orientation (Yaw angle - ψ_W) may be calculated by (13).

TABLE II
BALLBOT PARAMETERS

Control	Gain
LQR (XZ, YZ) Simulation	K = $\begin{bmatrix}-70.51 & -14.56 & 0.45 & 2.98\end{bmatrix}$
LQR (YZ,YZ) Experimental	K = $\begin{bmatrix}-50 & -25 & 1.5 & 2.5\end{bmatrix}$
Yaw Proportional Control	Kp = 0.3
Yaw Velocity PI Control	Kp = 1, Ki = 0.05

$$d = \sqrt{(W_X + X_B)^2 + (W_Y + Y_B)^2} \tag{12}$$

$$\psi_W = tan^{-1}\left(\frac{W_Y - Y_B}{W_X - X_B}\right) \tag{13}$$

Algorithm 1 Waypoint navigation algorithm

```
// Initial setting
Rw ← 10 // Radius in cm of the waypoint
waypoints ← listRead('mission.txt')// Load waypoint list
i ← 1 // Counter to read waypoints
yawRef ← compassRead() // Read robot guidance
yawWrite(yawRef) // Send via bluetooth
while True do
    Xb, Yb ← estimatePosition('camera')
    Wx, Wy ← waypoints[i]
    Erro_X ← Wx − Xb
    Erro_Y ← Wy − Yb
    d ← sqrt(ErrorX * Erro_X + ErrorY * ErrorY)
    if abs(ErrorY) > Rw then
        if abs(ErrorY) > 0 then
            move('right')
        end
        if abs(ErrorY) < 0 then
            move('left')
        end
    end
    else
        if abs(ErrorX) > Rw then
            if abs(ErrorX) > 0 then
                move('forward')
            end
            if abs(ErrorX) < 0 then
                move('back')
            end
        end
        else
            move('stop')
        end
    end
    if abs(d) <= R_W then
        move('stop')
        if length(waypoints) == i then
            break
        end
        else
            i ← i + 1
            delaySecond(1)
        end
    end
end
```

Algorithm 2 Algorithm for object tracking

```
// Initial setting
Pref ← width/2 // Position reference (pixels)
Kg ← 0.02 // Yaw gains
distRef ← 50 // Distance threshold [cm]
while True do
    // Returns the position in X (pixels)
    posX ← getXObject('camera') // Returns the distance of the object
    dist ← getDistance('camera') // Calculates error for yaw control
    errorPosX ← posX − Pref // Calculate the distance error
    errorDist ← dist − distRef // Calculates yaw proportional controller
    Vrot ← errorPosX ∗ Kg // Send output controller via bluetooth
    sendOutput(Vrot) if abs(errorPosX) < 10 then
        if errorDist > 0 then
            move('start')
        if errorDist < 0 then
            move('stop')
```

The implemented Algorithm 1 perform the guidance control of the ballbot by executing the forward, backward, left and right movements with constant speed during the navigation by the environment. The objective of the algorithm is to correct the position in relation to the Y axis and then X axis, until reaching the minimum distance of the waypoint radius.

E. Object Motion Tracking

The object tracking problem consists of keeping the robot pointing to a certain object detected using an on-board camera. Thus, if the object moves to the right, the robot must make the Yaw correction in order to keep the object in the center of the image. If the object moves away (reducing the size of the captured image), the robot must move forward, reducing the position error and keeping the ballbot at a certain distance from the object.

The ballbot detects the object and tracks their movement using the images processed through a color filter. The used camera was adapted of the cell phone model Samsung Galaxy J1 and attached to the top of the robot, and their connection to the Wireless network was made using the IP WebCam application.

Knowing the diameter of the colored marker (object to be tracked) and determining the diameter in *pixels* from the image, it can estimate the distance between the colored marker and the ballbot. The distance (14) between the object and the ballbot may be estimated from the polynomial approximation obtained by the least squares method.

$$D(x) = 0,001x^3 + x,171x^2 - 0,928x + 201 \qquad (14)$$

where x is the distance defined by the *pixels* and D is the distance between the colored marker and the ballbot.

The object tracking is performed using the Algorithm 2, in which incorporates the control system presented in (Fig. 5) to perform the tracking. This tracking control is formed by the Yaw controller connected to the on-off position controller. The purpose of Yaw controller is to keep the colored object in the center of the image. Thus, this controller receive the pixels and the desired distance as input, and provides the output (rotation speed - V_{rot}) for the ballbot.

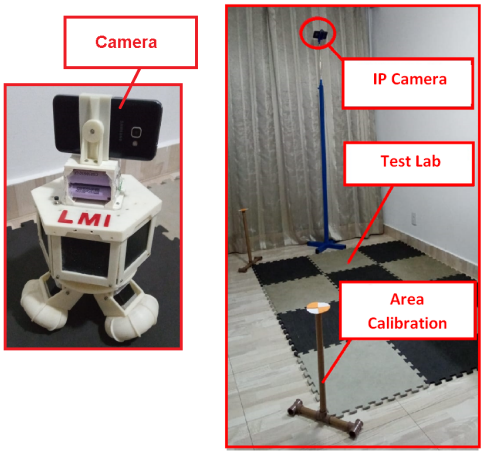


Fig. 6. Experimental Environment

IV. Results and Discussion

Figure 6 shows the proposed environment for performing the autonomous navigation using waypoints and tracking objects. A external IP camera attached on the tripod visualizes the navigation area delimited by the checkered floor.

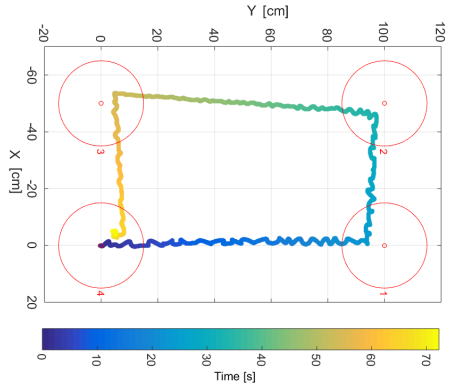
In object tracking mode, the ballbot is equipped with a colored marker, with a different tone from the colors of the environment, to estimate the position of the robot. In object tracking mode, only a camera attached to the top of the robot is used. Using this camera, the ballbot can estimate the distance of the object and compute the object position on the global reference frame.

A. Evaluation in CoppeliaSim Simulator

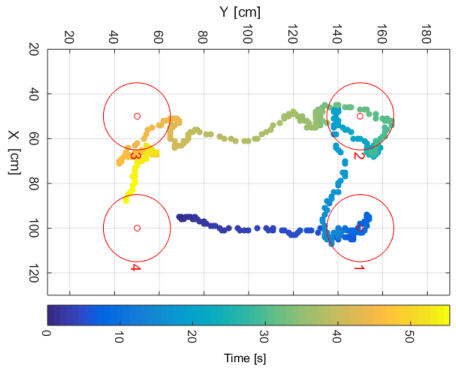
The simulation of autonomous navigation performed in CoppeliaSim aims to validate the guidance algorithm, observing the attitude, X-Y position and angular speed of the actuators. Four points (waypoints) were defined to establish the path, given by the coordination [0, 100];[-50, 100], [-50, 0] and [0, 0]. In the simulation, the waypoints are defined in the script written in LUA by the user.

In the trajectory with 4 waypoints, the simulated robot completes the mission correctly (see Fig. 7(a)), where the waypoints 1, 2, 3 and 4 were reached in the time of 22 s, 32 s, 54 s and 65 s, respectively. Note that the ballbot arrives to the points 1 and 4 executing movements only in Y axis as seen in Fig. 8(a) (blue curve). At points 2 and 3, only the translation in X axis occurs, as seen in Fig. 8(a) (red curve). In both cause the robot navigates with constant speed.

It is important observe that the orientation of the ballbot was controlled to be 0, however the response presents some

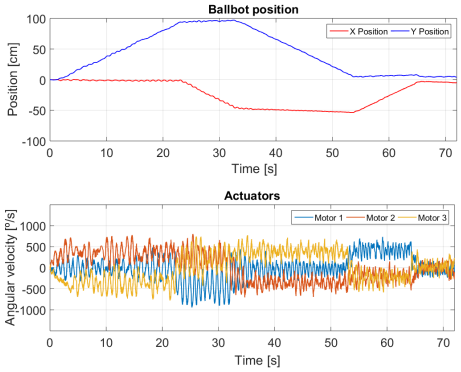


(a) CoppeliaSim simulation

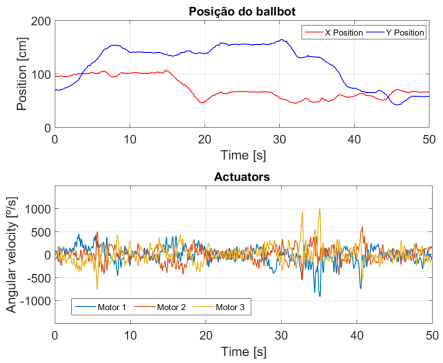


(b) Actual experimentation

Fig. 7. Ballbot navigation responses



(a) CoppeliaSim simulation



(b) Actual experimentation

Fig. 8. Ballbot position and actuators speed responses

small oscillations of 0.6 degrees around the desired reference. On the other hand, the pitch and roll angles present a larger oscillation due to the translational movements executed by the ballbot.

B. Experimental Validation

After evaluating the control and guidance algorithms in Simulation, we repeat the same assay using the experimental setup shown in Fig. 6. In the actual validation, the waypoints are defined by the user interface executed in the ground station and are sent to the actual ballbot using bluetooth communication.

The robot traveled the path to reach the four waypoints defined in the environment in 45 s as shown in Fig.7(b). This assay demonstrates the robustness of the control system design implemented to track the waypoints, and validates the methodology to construct the mechanical structure and actuators/omnidiretional wheels, including the localization algorithm using the adopted IMU and camera and also embedded firmware.

Figure 8(b) shows the X-Y position and actuator responses observed during the guidance execution to reach the four waypoints. Note that the speed spikes occur exactly during changing of the waypoints. This behavior occurs because of the ballbot needs to switch to the next waypoint at the moment that the ballbot arrives to the desired waypoint.

During experimental validation, it was observed that the Yaw angle oscillated around 10 degrees. This behavior is caused by the vibration generated by the contact between the omnidirectional wheels and the irregular surface of the ball

C. Object Tracking

Figure 9 shows the experiment performed for tracking a colored object. In this test, the object is put at three different positions to validate the detection procedure and also the ballbot orientation control. The response of the Yaw (orientation) controller is obtained moving the object among three different positions with distance of 30 cm among them. The ballbot was positioned to keep the distance of 107 cm of the colored object. Initially, the ballbot points to the object in which is positioned at the central position (position 2, see Fig. 9), and then moving the object to the position 1 and 3, the ballbot must detect this object changing their orientation.

Figure 10 shows the experimental result obtained for tracking the colored object positioned in front of the ballbot. The object was kept 6 s at the position 1, and then it was placed by 13 s and 19 s at the position 2 and 3, respectively. Even proving an external disturbance on the ballbot, it can be observed that the ballbot can keep the object tracking (see 10), demonstrating that the designed orientation controller is robust and efficient.

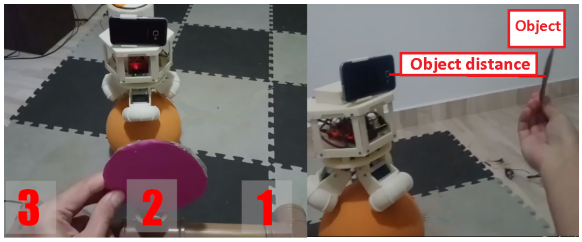


Fig. 9. Object tracking experimentation

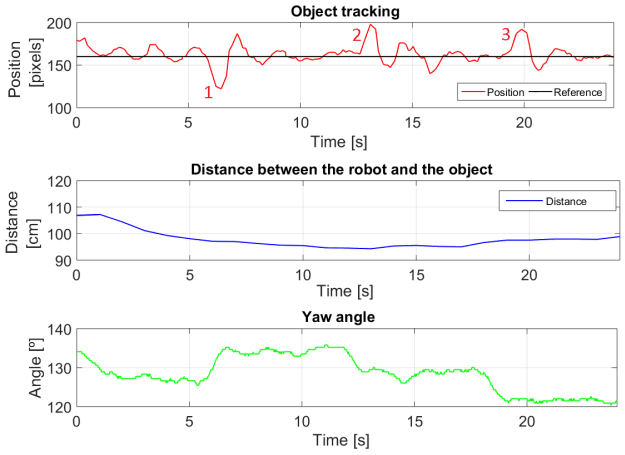


Fig. 10. Object tracking responses

D. Comparison Between Simulation and Experimental Results

Regarding to the experimental results, it was observed that the ballbot responses presented greated oscilation whenever compared with the simulation. Note that the ballbot completes the mission in a shorter time in simulation compared to the actual trial. The observed actuators speed responses has similar behaviour in both tests, around 500º/s, but with different frequencies. Besides, in both testes, the small alteration of the moment of inertia affect the ballbot performance during the navigation, which must be compensated by the designed controllers.

The attitude response also presented similar small variation in simulation and experimental trials, reaching peaks of 3 to 8 degrees. In both, the observed variation is generated during the execution of the translational movement. Specifically to the Yaw response, the robot presented a small oscillation of 0.6 degrees in simulation whereas the maximum oscillation of 18 degree was noted in the actual trial.

V. Conclusion

A low-cost ballbot capable of performing the translation and rotation motions and executing different mission by navigating through waypoints was presented. The proposed methodology enables that the mechanical structure and omnidirectional wheels to be replicated using 3D printers. During the research development, it was observerd that the high friction produced by the rollers of the wheels hamper to control the the ballbot attitude. Another observed problem is relate to the range limitation of the camera to detect the colored marker. This restriction limits the size of the navigation area used to perform the experimental validation.

The simulation and experimental tests for autonomous navigation showed that the designed controllers are able to control the ballbot during the execution of their mission, navigating by the waypoints defined in the environment to by the user. The proposed solution to track the object based on the distance between the ballbot and the object have presented a reasonable performance for distances up to 2.5 meters. In addition, the position controllers demonstrated good performance, proving the ballbot's ability to follow different path to reach the waypoints of interest or to detect a colored object.

Acknowledgements

The authors gratefully acknowledge the support for this research provided by FAPESP (Research Grants 2006/06005-0, 2016/04992-6 and 2024/01627-1), SENAI, ITA and UNIFESP.

References

[1] T. Lauwers, G. Kantor, and R. Hollis, "A dynamically stable single-wheeled mobile robot with inverse mouse-ball drive," in *2006 IEEE International Conference on Robotics and Automation (ICRA 2006)*, 2006, pp. 2884–2889. [Online]. Available: https://dx.doi.org/10.1109/ROBOT.2006.1642139

[2] T. Lauwers, G. A. Kantor, and R. Hollis, "One is enough!" in *12th Proc. Int'l. Symp. on Robotics Research*. San Francisco, CA, USA: IEEE, Oct. 2005, pp. 1–10. [Online]. Available: http://www.msl.ri.cmu.edu/publications/pdfs/isrr05.pdf

[3] C.-W. Liao, C.-C. Tsai, Y. Y. Li, and C.-K. Chan, "Dynamic modeling and sliding-mode control of a ball robot with inverse mouse-ball drive," in *2008 SICE Annual Conference*, 2008, pp. 2951–2955. [Online]. Available: https://dx.doi.org/10.1109/SICE.2008.4655168

[4] J. W. van der Blonk, "Modeling and control of a ball-balancing robot," Master's thesis, University of Twente, 2014. [Online]. Available: http://essay.utwente.nl/65559/

[5] P. Fankhauser and C. Gwerder, "Modeling and control of a ballbot," Bachelor Thesis, ETH, Swiss Federal Institute of Technology Zurich, 2010. [Online]. Available: https://doi.org/10.3929/ethz-a-010056685

[6] M. Kumagai and T. Ochiai, "Development of a robot balancing on a ball," in *2008 International Conference on Control, Automation and Systems*, 2008, pp. 433–438. [Online]. Available: https://doi.org/10.1109/ICCAS.2008.4694680

[7] M. Shomin, "Navigation and physical interaction with balancing robots," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, USA, 2016. [Online]. Available: https://doi.org/10.1184/R1/6720809.v1

[8] U. Nagarajan, G. Kantor, and R. Hollis, "Integrated planning and control for graceful navigation of shape-accelerated underactuated balancing mobile robots," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 136–141. [Online]. Available: https://doi.org/10.1109/ICRA.2012.6224970

[9] A. Lotfiani, M. Keshmiri, and M. Danesh, "Dynamic analysis and control synthesis of a spherical wheeled robot (ballbot)," in *2013 First RSI/ISM International Conference on Robotics and Mechatronics (ICRoM)*, 2013, pp. 481–486. [Online]. Available: https://doi.org/10.1109/ICRoM.2013.6510154

[10] S.-M. Lee and B. S. Park, "Robust control for trajectory tracking and balancing of a ballbot," *IEEE Access*, vol. 8, pp. 159 324–159 330, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3020091

[11] C. Cai, J. Lu, and Z. Li, "Kinematic analysis and control algorithm for the ballbot," *IEEE Access*, vol. 7, pp. 38 314–38 321, 2019. [Online]. Available: https://doi.org/10.1109/ACCESS.2019.2902219

[12] K. Sukvichai and M. Parnichkun, "Double-level ball-riding robot balancing: From system design, modeling, controller synthesis, to performance evaluation," *Mechatronics*, vol. 24, no. 5, pp. 519–532, 2014. [Online]. Available: https://doi.org/10.1016/j.mechatronics.2014.06.003