Received February 20, 2019, accepted February 24, 2019, date of publication February 28, 2019, date of current version April 5, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2902219

Kinematic Analysis and Control Algorithm for the Ballbot

CHENGTAO CAI[1], JIAXIN LU[1], AND ZUOYONG LI[2,3]
[1]College of Automation, Harbin Engineering University, Harbin 150001, China
[2]Fujian Provincial Key Laboratory of Information Processing and Intelligent Control, Minjiang University, Fuzhou 350121, China
[3]Institute of Internet Innovation, Fujian College's Research Base of Humanities and Social Science, Minjiang University, Fuzhou 350121, China

Corresponding author: Jiaxin Lu (lujiaxin@hrbeu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61673129 and Grant 51674109.

ABSTRACT The ballbot is a dynamically stable mobile robot designed to balance on a single ball, whose dynamic stability enables improved navigability in narrow, crowded, and dynamic environments. Through its single contact point with the ground, ballbot is omnidirectional and exceptionally agile, maneuverable, and organic in motion compared to other ground vehicles. For dealing with the challenging and imperative issues about ballbot, such as balancing, yaw, position control algorithms as well as the mathematical kinematic model, a novel model employing the Lagrange Equation is derived and control algorithm based on the similar principle as the second-order inverted pendulum is proposed, which is used for tackling the balancing, yaw, and position control. A cascade fuzzy proportional derivative (PD) controller and another controller with proportional integral (PI) control and PD feedback are designed for position and speed, respectively. Yaw control is presented, including head-hold mode and head-free mode. Some experiments are carried out to validate the effectiveness of the mathematical model and control algorithm.

INDEX TERMS Ballbot, Lagrange equation, cascade fuzzy control.

I. INTRODUCTION

With the rapid development of robotics, all kinds of robots are gradually entering people's lives, especially dynamic balancing robot [1]. One of the most popular dynamic balancing robots is the two-wheeled Segway Balancing robot [2]. As a Vehicle, one-wheeled Balancing robot also won a huge market. However, these balancing robots have turn limits, only move forward and backward, human can't move in accordance with owns wishes, in other words, these robots can only do one dimensional motion [3], [4]. The ballbot improves the mobile efficiency result of moving in any direction without a radius of rotation. It is supported by a ball that move in all directions, which means that it is not necessary to turn around like a wheeled or multi-legged robot to change the way forward. Moving omnidirectionality and the characteristics of single contact point with the ground makes it more suitable for navigating in the limited space [5]–[8].

In 2005, the first ballbot was invented at Carnegie Mellon University, which used the inverse mouse-ball drive mechanism to drive the ball to achieve a dynamic balance, but this structure is more cumbersome, and cannot rotate along the vertical axis. It used linear quadratic regulator (LQR) controller for full state feedback [9]. 'BallIP' was developed in 2008, the first ballbot used the structure of three Omnidirectional wheels. It achieves rotation around the vertical axis, which is a breakthrough in the ballbot. However, the paper does not describe the modeling method in detail. In terms of control algorithms, 'BallIP' used classical PD controller [10]. Ballbot with a manipulator has been proposed by Asgari et al in 2013. It has been proved that the dynamics equations of the assumed mobile robot. At the same time, a control algorithm is proposed to realize the stable motion control of the system. Finally, the execution of the simulation program is to verify the advantages of the algorithm [11]. Aiming at the ballbot is an underactuated, nonholonomically constrained system (there are more degrees of freedom than independent control inputs), Aykut et al presented the linearized equations of motion and the controllability analysis is carried out. They consider that ballbot was a linear system except for rotating about the vertical axis of the ballbot. They provided a PD controller and presented simulation results. Unfortunately, they did not implement the algorithm on a real robot [12]. There is some recent work on the application of the ball robot in a specific environment, for example, paper focuses on the control methods of the ball robot climbing and proposes

The associate editor coordinating the review of this manuscript and approving it for publication was Zhen Li.

ballbot with an annular support leg, which can keep statically stable when powered off [13]. Paper presents the equations of motion for the ballbot system on a sloped surface with a center-of-mass offset [14].

In this paper, we propose the control method and analyze the kinetics of the ballbot. Firstly, we derive the Lagrange equation of the plane model of ballbot and the mathematical relationship between the speed of the omnidirectional wheel and the speed of the plane model. The only contact point between the ballbot and the ground is the ball, thus it only relies on the inertial force of the body to maintain the dynamic stability. Cascade fuzzy PD control used for dynamic balancing in high angle non-linear areas. Head-free model and head-hold model are presented for yaw controller with little attention in previous work. Outer-loop position controller and inner-loop speed controller outputs desired attitude angle, which couples balancing and position control. The key is to achieve position and yaw control while maintaining its own balance. Finally, the ballbot is implemented shown in Fig. 1 and the experiments are carried out, including self-balancing, station-keeping, anti-interference and position-moving. The paper is organized as follows: section II describes the ballbot system construction and dynamic model, section III introduces the ballbot control method, including balancing control and position control, in addition, kinetics of ballbot was analyzed. Section IV shows the result and the Experimental results are analyzed. Section V concludes the paper.

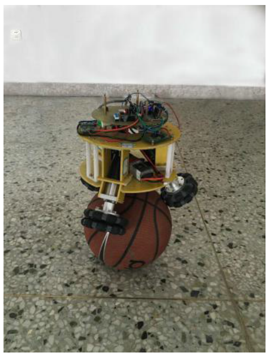

FIGURE 1. Ballbot balancing.

II. SYSTEM DESCRIPTION AND MODELING

A. SYSTEM DESCRIPTION

Fig. 2 shows the structure of the system. The ballbot is divided into two parts: ball and body. The ballbot body included processor, sensor, driver, motor, omnidirectional wheel. The processor adopts Freescale Kinetis K60 series MCU, whose hardware interfaces is wealthy and operating speed can be overclocked up to 150MHz. Robot measures the triaxial angle velocity and acceleration of the ballbot body through the Inertial measurement unit (IMU) MPU-6050. The measurement range can be controlled by a program write register. In order to adapt to the fast-dynamic movement of the ballbot,

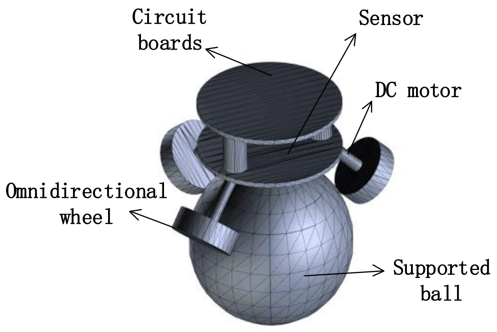

FIGURE 2. Structure of system.

we set the measurable range of the gyroscope to be 2000°/s, and the measurable range of acceleration to be 8g. Triaxial magnetic field intensity is gauged by three axis magnetometer HMC5883L measurement. Event-triggered sampling method is used to save the power energy [15]. The measurement ranges from milli Gauss to 8Gauss, then the Euler angle of the robot is calculated by quaternion algorithm. To get a more accurate Euler angle, we used a sliding average filter for angle velocity. Roll and pitch are input of the balancing controller, because the ballbot body balances on the ball dynamically. the yaw angle used to achieve the ballbot body rotate around a vertical axis. At the same time, ZigBee is used for wireless data transmission between ballbot and computer, including real-time sensor data, system configuration parameters. We select the direct current gear motor to provide enough torque and the Motor reduction ratio is 1: 16. Robomasters RM35 motor is a dedicated 5 $\sim$ 20kg robot customized power motor. The encoder feedback the motor speed integrated in the back end of motor. The three motors are 120° on the xoy plane, and the angle between the ball and plane of omnidirectional wheel α is 45°, if α is too large, the ballbot body is easy to fall; on the contrary, the mobile robot will be subject to certain restrictions. The omnidirectional wheel is used on this robot. Two alternating driven wheels ensure that the omnidirectional wheel has no sliding friction when moving parallel to the center axis, and there is no twitching phenomenon of the ballbot body. The diameter of the omnidirectional wheel is 100mm and the weight are 290g [16].

B. DYNAMIC MODELING

The dynamic analysis of the system used the Lagrange equation which is an important method for dynamic modeling. It simplifies the modeling process obviously when the degree of freedom of the system is less, because it only needs to analyze the system kinetic energy, potential energy and generalized force, rather than the complicated force analysis [17], [18]. To facilitate the analysis, the ballbot can be regarded as a uniform rigid body and a uniform rigid ball, and we decompose the three-dimensional dynamic system into three planes. It is worth noting that modeling is based on the following two assumptions:

(i)The mechanical structure is completely symmetric, and the motion of the xoz plane and the yoz plane is uncoupled.

(ii)There is rolling, no sliding and spin between the supporting ball and the ground.

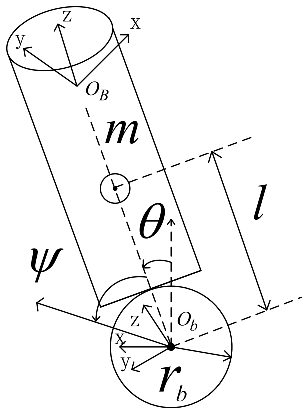

FIGURE 3. The plane model of the ballbot.

The ballbot has 5 degrees of freedom, including 2 degrees of freedom in the plane position and 3 degrees of freedom of the Euler angle. The plane model is shown in the Fig. 3. $\theta_{x,y,z}$, $\psi_{x,y,z}$ represent the rotate angle of the ballbot body and the supporting ball around each axis. The xoz plane is the same as the yoz plane model. The kinetic energy K_b and mass energy V_b of the supporting ball are given below.

$$K_{b,XOZ} = \underbrace{\frac{1}{2}I_b\dot{\psi}_y^{\,2}}_{\text{Rotation}} + \underbrace{\frac{1}{2}m_b(r_b\dot{\psi}_y)^2}_{\text{Translation}} \tag{1}$$

$$V_{b,XOZ} = 0 \tag{2}$$

Kinetic energy K_b includes rotation parts and translation parts. As the origin of the coordinate system is in the center of the ball, the mass-energy $V_b = 0$. I_b, m_b, r_b is the moment of inertia, the mass and the radius of the supporting ball.

The kinetic energy of the support ball xoy plane only includes the rotation part, and the mass energy is 0.

$$K_{b,XOY} = \frac{1}{2}m_b(r_b\dot{\psi}_z)^2 \tag{3}$$

The kinetic energy and mass energy of the ballbot body are as follows:

$$K_{B,XOZ} = \frac{1}{2}m_B(r_b^2\dot{\psi}_y^2 + l^2\dot{\theta}_y^2) + \frac{1}{2}I_B\dot{\theta}_y^2 \tag{4}$$

$$V_{B,XOZ} = m_Bgl\cos\theta_y \tag{5}$$

$$K_{B,XOY} = \frac{1}{2}I_B\theta_z^2 \tag{6}$$

I_B, m_B, r_B represent the moment of inertia, mass, radius of the ballbot body respectively.

The angle velocity of the omnidirectional wheel is like that of the support ball and the robot body. However, the moment of inertia of the ballbot body and the support ball is far greater than the moment of inertia of the drive wheel around the shaft of the motor, so the kinetic energy of the system can ignore the kinetic energy generated by the rotation of the drive wheel. Using the Euler Lagrange equations, the dynamic equations of the planar motion of the ballbot can be written in matrix form, as follows.

$$M(q)\ddot{q} + C(q,\dot{q}) + G(q) + D(\dot{q}) = \begin{bmatrix} 0 \\ \tau \end{bmatrix} \tag{7}$$

Define the system configuration vector $q = [\theta, \psi]^T$, $M(q)$ is the mass matrix, $(q, \dot{q})$ is the Coriolis force vector. $G(q)$ represents gravity vector, $D(\dot{q})$ is the friction torque, τ is the torque between the body and the support ball in the direction normal to the plane. The expressions of the above terms are as follows.

$$M(q) = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \tag{8}$$

$$M_{11} = I_b + I_B + m_br_b^2 + m_B(r_b^2 + l^2) \tag{9}$$

$$M_{12} = M_{11} + m_Br_bl\cos(\theta + \psi) \tag{10}$$

$$M_{21} = M_{12} \tag{11}$$

$$M_{22} = M_{11} + 2m_Br_bl\cos(\theta + \psi) + m_Bl^2 + I_B \tag{12}$$

$$C(q,\dot{q}) = \begin{bmatrix} -m_Br_bl\sin(\theta+\psi)(\dot{\theta}+\dot{\psi})^2 \\ -m_Br_bl\sin(\theta+\psi)(\dot{\theta}+\dot{\psi})^2 \end{bmatrix} \tag{13}$$

$$D(\dot{q}) = \begin{bmatrix} 0 \\ -\frac{gm_Br_bl\sin(\theta+\psi)}{I_B+m_Bl^2} \end{bmatrix} \tag{14}$$

C. SPEED CONVERSION BETWEEN WHEEL AND BODY

The ballbot's movement is achieved through the rotation of the omnidirectional wheel that drives the ball to roll. The relationship between the speed of the robot and the rotational speed of the omnidirectional wheel are obtained by the following methods. We define $\vec{\omega}$ as the angle velocity of the rotation of the ball, $\vec{P_i}$ as the coordinate vector of contact point of the support ball.

$$\begin{aligned} &P_1(r_b\cos\alpha, 0, r_b\sin\alpha) \\ &P_2(-\frac{1}{2}r_b\cos\alpha, \frac{\sqrt{3}}{2}r_b\cos\alpha, r_b\sin\alpha) \\ &P_3(-\frac{1}{2}r_b\cos\alpha, -\frac{\sqrt{3}}{2}r_b\cos\alpha, r_b\sin\alpha) \end{aligned} \tag{15}$$

$\vec{v_i}$ is the speed of a point P_i on the ball, we obtain the following Equation.

$$\vec{v_i} = \vec{\omega} \times \vec{P_i} = \begin{bmatrix} \omega_yP_{iz} - \omega_zP_{iy} \\ \omega_zP_{ix} - \omega_xP_{iz} \\ \omega_xP_{iy} - \omega_yP_{ix} \end{bmatrix} \tag{16}$$

$\vec{s_i}$ represents the driving direction of the three omnidirectional wheels shown in Fig. 4, their value are as follows:

$$\begin{aligned} &s_1(0, -1, 0) \\ &s_2(\frac{\sqrt{3}}{2}, \frac{1}{2}, 0) \\ &s_3(-\frac{\sqrt{3}}{2}, \frac{1}{2}, 0) \end{aligned} \tag{17}$$

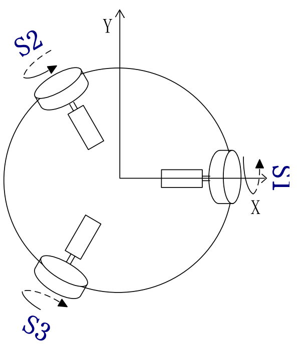

FIGURE 4. Driving direction of the three omnidirectional wheels.

The relationship between the speed of three omnidirectional wheels and the body shown as Equation (18). $r_b \cos\alpha\omega_z$ is the speed of rotation around the z axis and appears at the speed of the three omnidirectional wheels. It is not in the balancing control of x or y direction.

$$
\begin{aligned}
v_{s1} &= -v_y \sin\alpha - r_b \cos\alpha\omega_z \\
v_{s2} &= \frac{\sqrt{3}}{2} v_x \sin\alpha + \frac{1}{2} v_y \sin\alpha - r_b \cos\alpha\omega_z \\
v_{s3} &= -\frac{\sqrt{3}}{2} v_x \sin\alpha + \frac{1}{2} v_y \sin\alpha - r_b \cos\alpha\omega_z
\end{aligned} \tag{18}
$$

III. BALLBOT CONTROL

This section describes the various controllers used on the ballbot. The complete control block diagram of the ballbot is shown as Fig. 5, including position control of x, y axes and yaw control. The ballbot feedback roll, pitch, roll rate, pitch rate, speed and position to position control and yaw, yaw rate to yaw control. The speed of the three wheels is determined by the superposition of the outputs of the yaw control and position control.

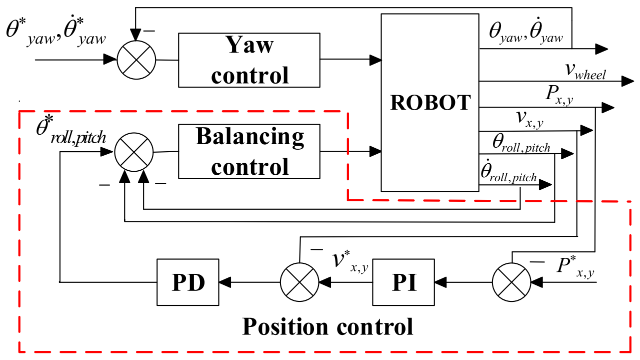

FIGURE 5. The complete control block diagram of the ballbot.

A. BALANCING CONTROL

It was proposed to use the control methods of inverted pendulum to control the robot. For the balance of ballbot, it holds the balance around the unstable point by controlling the rotation of three omnidirectional wheels. The ballbot drive balls to roll forward by controlling three motors and the roll speed of the ball is faster than the dumping speed of the body. It is possible to use the similar principle of the inverted pendulum to simulate the ballbot. Balancing control can be equivalent to two inverted pendulum of x, y direction, which is a typical nonlinear, multivariate, strong coupling and unstable system. We use the cascade PD controller, and use fuzzy control in the outer-loop angle control. The inner loop of the cascade control system is a follow-up control system whose set value varies with the output of the outer loop controller. The outer loop controller can continuously adjust the set value of the inner loop controller according to the operating conditions and load changes, to ensure that the control system still has better control effects under the condition that the operating conditions and load change. The fuzzy PD control automatically adjusts the PD value according to the preset parameter table according to the input error. Different PD values act on nonlinear systems to accelerate the response speed of nonlinear systems.

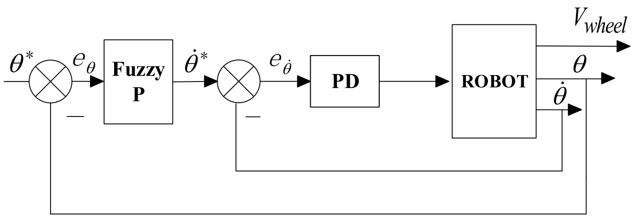

FIGURE 6. The block diagram of balancing control.

The block diagram of balancing control is shown as Fig. 6, Equation (19) shows the error between the desired angle and the Euler angle, which is considered as the input of the outer-loop angle controller. While there is only the balancing controller, the desired angle is 0deg. Desired angle velocity is output of the outer-loop angle controller. The angle velocity interference is more likely to cause instability of the ballbot than the angle, cascade controller plays the role of anticipatory control and effectively suppresses the inner loop interference. The inner-loop angle velocity is controlled by PD. As shown in Equation (20), the output of the inner-loop is the acceleration in the direction of x and y. The perform frequency of the inner-loop is usually 2 $\sim$ 5 times of the outer-loop frequency in cascade control. We choose 2 times according to the experience of the experiment. The controller in xoz plane is the same as yoz plane, the parameters setting is the same. The fuzzy control rule table is shown in Table 1, While the angle and angle velocity are in the same direction, the system becomes more and more unstable, and the error should be eliminated quickly, Kp should be the largest one. On the contrary, if the system has a trend of gradual stabilization, and Kp will be set to be small to avoid overshoot. In addition to this, the adjustment of Kp is also related to the magnitude of the angle and angle velocity. The speed is obtained by integrating acceleration in sampling time, as show in Equation (21). The speed in the direction

TABLE 1. Attitude control fuzzy rule.

θ / $\dot\theta$	PB	PM	PS	NS	NM	NB
PB	PB	PB	PB	PB	PM	PS
PM	PB	PM	PM	PM	PS	PS
PS	PM	PS	PS	0	0	0
NS	0	0	0	PS	PS	PM
NM	PS	PS	PM	PM	PM	PB
NB	PS	PM	PB	PB	PB	PB

of x and y can be divided into three directions of motors according to the Equation (18) so that the ballbot body can be dynamically balanced on the ball.

$$\dot\theta^* = kp_{Angle}(\theta^* - \theta) \tag{19}$$

$$F = kp_{Angle\_rate}(\dot\theta^* - \dot\theta) + kd_{Angle\_rate}\ddot\theta \tag{20}$$

$$v = v + \frac{F}{m}t \tag{21}$$

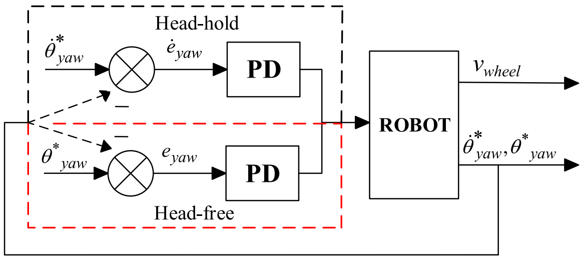

FIGURE 7. The block diagram of yaw control.

B. YAW CONTROL

The biggest innovation of the ballbot is that there is no turning radius, which is closely related to the control of the yaw angle. Yaw control is an independent control, decoupling of balancing control. Yaw control includes head-hold mode and head-free mode and the control block diagram is shown in Fig. 7. In the head-free mode, the ballbot's reference coordinate system is the geographic coordinate system. When initialized, the ballbot will calculate an initial yaw angle Yaw_{init} through a magnetic sensor, then the yaw angle is controlled by PD controller.

$$F_{yaw} = kp_{free} \cdot e_{yaw} + kd_{free} \cdot \dot e_{yaw} \tag{22}$$

The error e_{yaw} is the difference between the yaw angle and the initial yaw angle during the movement. $\dot e_{yaw}$ represents the yaw angle rate measured by IMU. The output is limited to avoid excessive output. In the head-hold mode, the ballbot's reference coordinate system is its body coordinate system. The desired yaw angle of the robot is artificially locked in the positive direction of the x, the control equation is as follows.

$$F_{yaw} = kp_{free} \cdot \dot e_{yaw} + kd_{free} \cdot \ddot e_{yaw} \tag{23}$$

$\ddot e_{yaw}$ is the difference between the current sampling value $\dot e_{yaw}$ and the last sampling value. It is worth noting that in head-hold mode, the desired angle of balancing control is affected by the yaw angle. The specific steps are to rotate the desired angle and transform the geographic coordinate system into its own coordinate system by using the rotation matrix $R \in SO(2)$.

$$\begin{bmatrix} \theta^*_{Roll} \\ \theta^*_{Pitch} \end{bmatrix} = \begin{bmatrix} \cos e_{Yaw} & -\sin e_{Yaw} \\ \sin e_{Yaw} & \cos e_{Yaw} \end{bmatrix} \begin{bmatrix} \theta^*_{Roll} \\ \theta^*_{Pitch} \end{bmatrix} \tag{24}$$

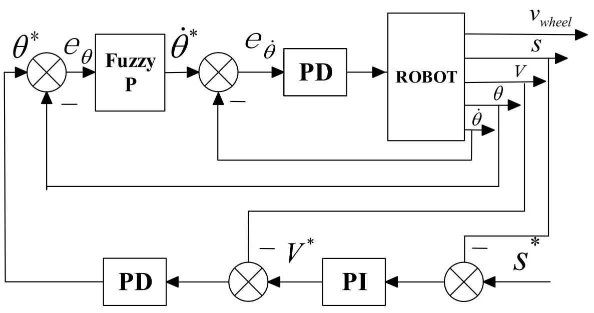

FIGURE 8. The block diagram of position control.

C. POSITION CONTROL

The ballbot will move erratically at equilibrium despite the balancing controller, which is still incomplete to a robot used for people. Position control and balancing control are mutually coupled because the ballbot tilts the body to move in the floor just like balancing robot. As shown in Fig. 8, position cascade controller includes outer-loop position controller and inner-loop speed controller. the outer-loop position controller is PI controller that outputs the desired speed which depending on the error between desired position and real position. The PI controller effectively reduces the difference of the position point during the movement. The inner-loop speed controller is a manually tuned PD controller that can reach the desired speed quickly. The output of the speed controller is desired angle determined by the error between the output of the position controller and the actual speed. The actual speed of the robot is measured by the encoder, and the equations for calculation are as follows.

$$\begin{aligned} v_x &= speed_{m1} - 0.5speed_{m2} - 0.5speed_{m3} \\ v_y &= 0.86(speed_{m2} - speed_{m3}) \end{aligned} \tag{25}$$

To decrease the error caused by the mechanical gap of encoder, the real speed is determined by converging the speeds in Equations (21) and (25). Limiting the output amplitude of position control prevents the ballbot from falling due to excessive tilt. The size of the limit determines the maximum speed of ballbot movement. When the desired position is (0,0), the ballbot keeps the station by position controller.

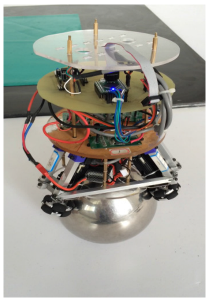

FIGURE 9. First version of the ballbot.

IV. RESULT

Fig. 9 is the first version of ballbot, we used hollow cup reducer as drive motor. After the system modeling analysis and the design of the control system, the ballbot was debugged and the dynamic balance was reached. Meanwhile, when we tested the anti-interference, the ballbot body was easy to fall from the support ball. Then, the ballbot can only balance in situ, be unable to move. At the same time, the driven wheel of rubber material is easy to be damaged, and it cannot ensure that there is no sliding friction when the omnidirectional wheels rotate parallel to the center axis of the motor. We make the second version. The supporting ball is changed into a basketball, so there is enough friction between the wheel and the ball and between the ball and the floor.

We carry out several experiments to verify the stability and robustness of revision of the ballbot. Firstly, we only retain the balancing control and set the desired attitude angle to $(0°, 0°)$. Attitude angle of the ballbot is updated with the frequency of 400Hz. The balancing control period is 10ms. As shown in the Fig. 10, the roll and pitch of the ballbot can be kept in the range of $(-0.4° \sim 0.4°)$. Due to the lack of feedback on the position, the ballbot moves wildly.

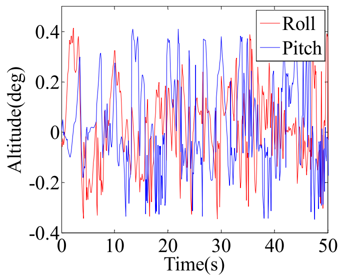

FIGURE 10. The attitude angle of the ballbot under balancing control.

To verify anti-interference of the ballbot, we took a violent impact experiment by using volleyball. Because the robot is "riding" on the ball, there is no other mechanical constraint, the interference can only be applied to the ballbot body. As shown in Fig. 11, we smash the volleyball into the robot body. The attitude angle of the ballbot increases obviously, and the dynamic balance is resumed after moving for a period along the direction be smashed. The Fig. 12 shows the roll and pitch, roll rate and pitch rate of the ballbot smashed by

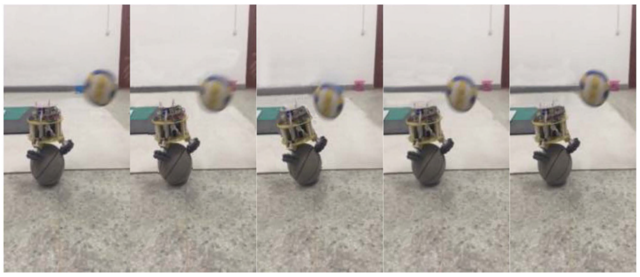

FIGURE 11. A series of picture of the ballbot smashed by volleyball.

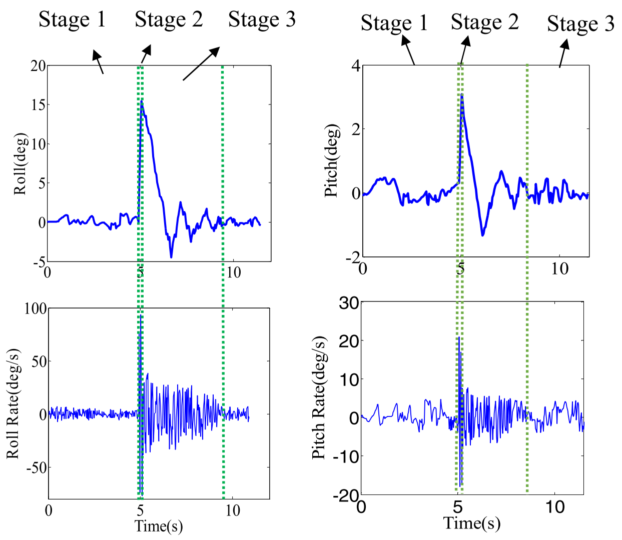

FIGURE 12. The roll, roll rate. pitch and pitch rate of the ballbot smashed by volleyball.

volleyball. Obviously, the whole process is divided into three stages. The first stage is that before the impact, the roll of the ballbot is balanced near 0 deg, the roll rate is between (-4deg/s,4deg/s); The second is that the roll rate becomes 15deg and the roll rate reaches 93.5deg/s drastically at the impact moment; Finally the roll and roll rate is restored to the same level as the first stage after about 4 seconds.

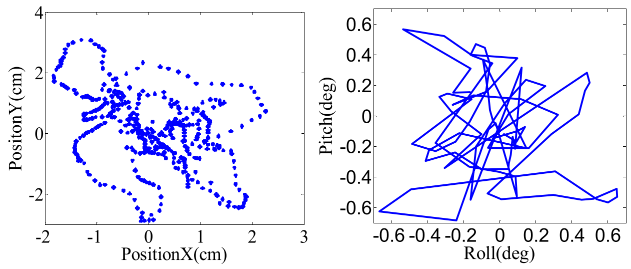

FIGURE 13. Position and attitude angle of the ballbot during station keeping.

As shown in Fig. 13, the ballbot moves between (-2cm, -2.6cm) and (2.5cm, 3cm) after combining with the position closed loop and balancing controller with setting a desired position (0cm, 0cm). The diameter of support ball is 24.5cm and the ball contacted with the ground through is a very small plane. Therefore, the range of motion error is acceptable. When keeping the original station, the range of attitude angle is ($-0.7^\circ \sim 0.6^\circ$), which is slightly larger than $-0.4^\circ \sim 0.4^\circ$ under the balancing control. The reason is that the desired angle is the output of the position control, and is no longer the 0 under the balancing control.

Table 2 shows the comparison of the position and attitude angle error with other control methods when the ballbot is keeping station, which are currently the state-of-the-art LQR control method respectively [19]. The red numbers in the table represent that our method is superior to LQR. Our position control method makes the ballbot dynamically balance in a smaller range.

TABLE 2. The comparison of the position and attitude angle error with other control methods.

	LQR [19]	Our method
PositionX Min (cm)	-10	-2
PositionX Max (cm)	7.5	2.5
PositionY Min (cm)	-8	-2.6
PositionY Max (cm)	3.5	3
Roll Min (°)	-1	-0.7
Roll Max (°)	0.8	0.6
Pitch Min (°)	-0.1	-0.7
Pitch Max (°)	0.1	0.6

The trajectory of the ballbot is as shown in the case where the preset trajectory is a square of 150cm. Because of the omnidirectionality of the ballbot, there are two methods to move along the preset trajectory: one is to set the desired values of positionX and positionY in headless mode; the other is to set the expected values of yaw and positionX in headless mode. Because experiments show that instability sometimes occurs when the first method is used, the yaw control is in the head-hold mode during the entire process, which means that the ballbot always moves in the direction of its head. The ballbot is instructed to move to the desired position (150cm, 0), (150cm,150cm), (0,150cm), (0,0) at a speed of 0.15m/s under position control and yaw control. As shown in Fig. 14, Because PI controller is used for position control,

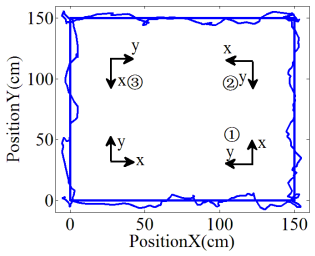

FIGURE 14. Position of the ballbot where the preset trajectory is a square of 150cm.

the position tracking has little hysteresis. After reaching the desired position, the ballbot balances within a narrow range of oscillations. The biggest error between desired and actual trajectory is 10.8cm. Yaw angle shown in Fig. 15, we rotate the desired yaw angle by -90deg after each vertex of the square is reached. Overshoot is the key factor of large error at the vertex of the final trajectory.

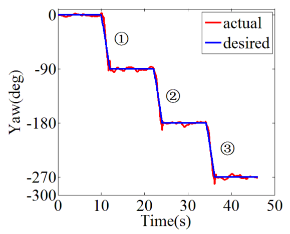

FIGURE 15. Yaw angle of the ballbot where the preset trajectory is a square of 150cm.

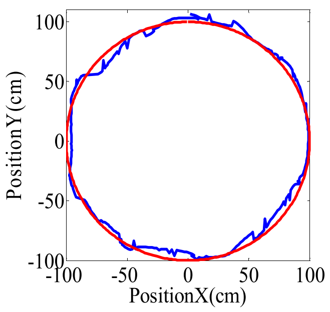

FIGURE 16. position of the ballbot where the preset trajectory is a circle with a radius of 100cm.

The position of the ballbot is shown in Fig. 16 where the preset trajectory is a circle with a radius of 100cm. The head-hold mode is used in yaw control as above. The ballbot starts from (0,100) at a speed of 0.15m/s and returns to the end (0,100). Throughout the process, the expected yaw angle is always 0, so the maximum error is less than the case where the preset trajectory is square.

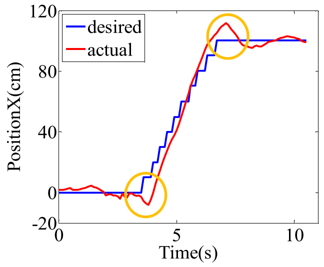

FIGURE 17. Reverse movement of the ballbot when starts and stops.

As the ballbot balancing control is an inverted pendulum problem, it must incline toward the direction of the movement to accelerate. At the same time, the inclination of the main body causes the ball to retreat in the opposite direction. Just as shown in Fig. 17, there is a large reverse movement when the ballbot starts and stops. The recoil movement will cause instability of the ballbot. We reduce the parameter Kp and the integral time appropriately to reduce overshoot, the overshoot of system decreases by strengthening the integral function.

V. CONCLUSIONS

Cascade fuzzy control method is designed to solve the balance of the ballbot which moves in any direction flexibly and turn more efficiently than traditional wheel-supported robot. Position control uses cascade PI and PD controller and yaw control divided into head-hold and head-free modes are proposed. The ballbot is completed according to the mathematical model and the designed controller. Using the cascade fuzzy control, the ballbot achieves a dynamic balance of $-0.4^\circ \sim 0.4^\circ$ and completes anti-interference experiments with amazing performance. The ballbot walking along the preset trajectory with an error of 7.2% under the joint operation of position control and yaw control. Due to the excellent omnidirectional movement of the ballbot, it moves freely in a narrow environment. An interesting topic for future research would be to improve control accuracy by applying deep learning to control methods.

REFERENCES

[1] U. Nagarajan, G. Kantor, and R. L. Hollis, "Trajectory planning and control of an underactuated dynamically stable single spherical wheeled mobile robot," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2009, pp. 3743–3748.

[2] H. G. Nguyen *et al.*, "Segway robotic mobility platform," *Proc. SPIE*, vol. 5609, pp. 207–220, Dec. 2004.

[3] A. Lotfiani, M. Keshmiri, and M. Danesh, "Dynamic analysis and control synthesis of a spherical wheeled robot (Ballbot)," in *Proc. 1st RSI/ISM Int. Conf. Robot. Mechatron. (ICRoM)*, Feb. 2013, pp. 481–486.

[4] U. Nagarajan *et al.*, "State transition, balancing, station keeping, and yaw control for a dynamically stable single spherical wheel mobile robot," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2009, pp. 998–1003.

[5] M. Shomin and R. Hollis, "Fast, dynamic trajectory planning for a dynamically stable mobile robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 3636–3641.

[6] A. Mampetta, "Automatic transition of ballbot from statically stable state to dynamically stable state," Robot. Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-RI-TR-06-41, Sep. 2006.

[7] L. Hertig, D. Schindler, M. Bloesch, C. D. Remy, and R. Siegwart, "Unified state estimation for a ballbot," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2013, pp. 2471–2476.

[8] G. Seyfarth, A. Bhatia, O. Sassnick, M. Shomin, M. Kumagai, and R. Hollis, "Initial results for a ballbot driven with a spherical induction motor," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2016, pp. 3771–3776.

[9] T. B. Lauwers, G. A. Kantor, and R. L. Hollis, "A dynamically stable single-wheeled mobile robot with inverse mouse-ball drive," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2006, pp. 2884–2889.

[10] M. Kumaga and T. Ochiai, "Development of a robot balanced on a ball—Application of passive motion to transport–," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2009, pp. 916–921.

[11] P. Asgari *et al.*, "Dynamics modelling and stable motion control of a Ballbot equipped with a manipulator," in *Proc. 1st RSI/ISM Int. Conf. Robot. Mechatron.*, Feb. 2013, pp. 21–36.

[12] A. C. Satici, F. Ruggiero, and V. Lippiello, "Intrinsic Euler-Lagrange dynamics and control analysis of the ballbot," in *Proc. IEEE Amer. Control Conf.*, Jul. 2016, pp. 5685–5690.

[13] J. Jian *et al.*, "Standing-up control and ramp-climbing control of a spherical wheeled robot," in *Proc. Int. Conf. Control Automat. Robot. Vis.*, Dec. 2014, pp. 1386–1391.

[14] B. Vaidya *et al.*, "Operation of the ballbot on slopes and with center-of-mass offsets," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2015, pp. 2383–2388.

[15] S. Li *et al.*, "Event-trigger heterogeneous nonlinear filter for wide-area measurement systems in power grid," *IEEE Trans. Smart Grid*, to be published.

[16] A. Bonci *et al.*, "Embedded system for a Ballbot robot," in *Proc. Int. Workshop Intell. Solutions Embedded Syst.*, Oct. 2015, pp. 157–161.

[17] H. Zabihi, H. A. Talebi, and A. A. Suratgar, "Open-loop trajectory planning and nonlinear control for underactuated spherical wheel mobile robot (Ballbot)," in *Proc. 24th Iranian Conf. Elect. Eng. (ICEE)*, May 2016, pp. 1–6.

[18] G. Seyfarth *et al.*, "Initial results for a ballbot driven with a spherical induction motor," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2016, pp. 3771–3776.

[19] C. C. Tsai, C. K. Chan, and L. C. Kuo, "LQR motion control of a ball-riding robot," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatron.*, Jul. 2012, pp. 861–866.

CHENGTAO CAI received the Ph.D. degree from Harbin Engineering University, Harbin, China, in 2008, where he was a Postdoctoral Research Scientist in shipbuilding and oceanography engineering, from 2009 to 2011. From 2006 to 2007, he was a Visiting Scholar with the National Research Council Canada. He is currently a Professor with the College of Automation, Harbin Engineering University. His research interests include robot intelligent control and panoramic vision applications.

JIAXIN LU was born in Inner Mongolia, China, in 1996. He received the B.S. degree in engineering from Harbin Engineering University, in 2017, where he is currently pursuing the M.S. degree in control science and engineering. His research interests include robot and computer vision.

ZUOYONG LI received the B.S. and M.S. degrees in computer science and technology from Fuzhou University, Fuzhou, China, in 2002 and 2006, respectively, and the Ph.D. degree from the School of Computer Science and Technology, Nanjing University of Science and Technology (NUST), Nanjing, China, in 2010. He is currently a Professor with the College of Computer and Control Engineering, Minjiang University, Fuzhou. He has published over 60 papers in international/national journals. His current research interests include image processing, pattern recognition, and machine learning.

• • •