Advances in Machinery, Materials Science and Engineering Application IX
M. Chen et al. (Eds.)

doi:10.3233/ATDE230517

Design of Self-Balancing Bike System Based on Autonomous Driving Technology

Anjie TU [a,1], Xiaotian FAN [a], Zhisheng WU [a], Xiai CHEN [a] and Tao ZENG [a]
[a] *School of Mechanical and Electrical Engineering, China Jiliang University, Hangzhou, China, 310018*

Abstract. In this work, we present a self-balancing bike system based on the open-source Robot Operating System (ROS) and the momentum flywheel, and has designed a bicycle control system that can achieve autonomous balance and autonomous driving. This bike uses flywheel to maintain balance and uses lidar to achieve autonomous driving. The drive part control core is TC377 microcontroller. The motor driver is Odriver, which directly control the flywheel motor. The corresponding SLAM algorithms are developed and designed in Linux based on the ROS distributed operating system framework. It collects position information via gyroscopes, commands body balance by rotating the flywheel with Cascade PID controllers and FOC controller, and uses lidar and depth cameras to localize the position and posture of the bicycle body in real-time. It generate an optimal path based on the given target point to automatically identify and avoid surrounding obstacles. And Qt interface is used to adjust parameters and display data.

Keywords. ROS, autonomous navigation, PID, momentum flywheel, inverted pendulum, bicycle

1. Introduction

With the new development of scientific and technological revolution and the increase in industrial transformation, smart cars have become the strategic focus of the future automotive industry. Simultaneously, the realization of conditional autonomous smart cars to achieve large scale production, the realization of highly autonomous driving intelligence.

The control system is intended to be a staged replication based on the practical applications of driverless bicycles, particularly in unmanned environments, in order to achieve a combination of navigation, computer vision, radar, artificial intelligence, automatic control, and motor control. Through the exploration of self-driving bicycles to develop the practical application of intelligent control, information communication, electronic engineering, control theory, sensor technology and other multi-domain technologies.

[1] Anjie TU, Corresponding author, School of Mechanical and Electrical Engineering, China Jiliang University, Hangzhou, China, 310018; E-mail: 3112977684@qq.com.

2. System Structure Design

The bicycle control system consists of two parts: host and server computers. Jetson Nano was developed as a host with a robotic ROS system and runs on Linux, including environmental perception, path planning and movement decision-making. As a server, microcontroller TC377 is responsible for data acquisition, motion direction control, and balancing control. The angle and angular rate values from the CH110 gyroscope acquisition and the rate value from the encoder's feedback are fed to the Cascade PID controller, which is then output to the brushless FOC drive plate to control the momentum flywheel, which controls the body balance [1]. The data is passed between the two via serial ports, and the next machine sends the gyroscope and process values required for the host machine to construct the map, and the host machine sends out the direction and speed required for the next machine to move.

In addition, a QT master can be used to adjust the parameters of the balance PID and to draw a real-time dynamic wave diagram of the posture. The system structure is shown in figure 1.

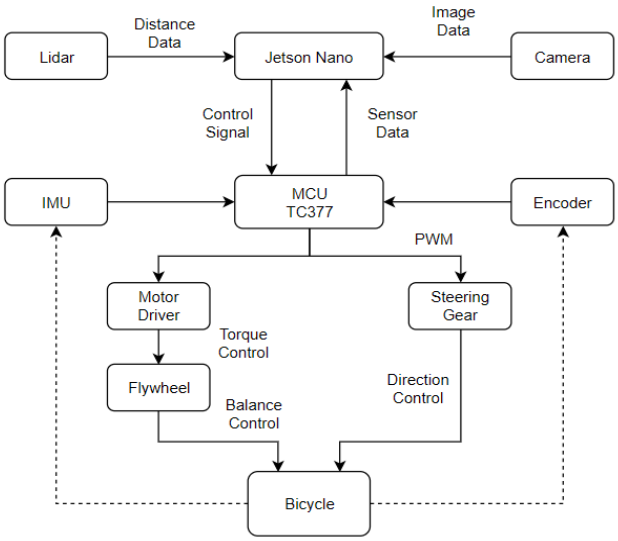

Figure 1. System structure.

3. Implementation of Balance System

3.1. Mathematical Modeling of Balance System

This system uses Langange energy method to dynamic model inverted pendulum system. (Shown in figure 2) Ignore the air resistance and simplify the flywheel inverted pendulum system to flywheel and uniform fine rod [2]. The flywheel can produce torque in the opposite direction of acceleration in the vertical plane so that the pendulum finally stabilizes near the equilibrium position.

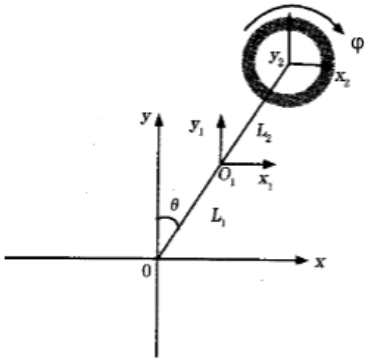

Figure 2. Flywheel Inverted Pendulum.

In this paper, the following notations are used. Establishing inertial coordinate system on moving plane $O-XY$. The simples O_1 and O_2 represents the center of gravity of rod and flywheel. (x_1, y_1) and (x_2, y_2) write the coordinate of the center of mass of rod and the center of mass of flywheel.

Table 1 shows the parameters of the inverted pendulum.

Table 1. System parameter description.

Parameter	Notation
Mass of rod	m_1 / kg
Mass of flywheel	m_2 / kg
Moment of inertia of rod	$I_1 / kg \cdot m^2$
Moment of inertia of flywheel	$I_2 / kg \cdot m^2$
Angle of flywheel	φ / rad
Angle of rod	θ / rad
Distance from O to O_1	L_1 / m
Distance from the center of mass of rod to the center of flywheel	L_1 / m

Firstly, based on Lagrangian energy method, the system dynamics model is established.

The total kinetic energy of the system is:

$$T = 0.5\left(m_1 v_{o1}^T v_{o1} + I_1 \dot{\theta}^2 \right) + 0.5\left(m_2 v_{o2}^T v_{o2} + I_2 \dot{\theta}^2 + I_2 \dot{\varphi}^2 \right) \tag{1}$$

The total potential energy of the system is:

$$V = m_1 g L_1 \cos\theta + m_2 g (L_1 + L_2) \cos\theta \tag{2}$$

The Lagrange initialization equation is introduced:

$$L(q, \dot{q}) = T(q, \dot{q}) - V(q, \dot{q}) \tag{3}$$

To make $L_1 = L_2$, the Lagrangian operator is:

$$L = \frac{1}{2}\left[m_1 L_1^2 + 4m_2 L_1^2 + I_1 + I_2\right]\dot{\theta}^2 + \frac{1}{2} I_1 \dot{\varphi}^2 - (m_1 + 2m_2) g L_1 \cos\theta \quad (4)$$

Generic coordinates q_i and L are substituted into the Lagrange equation:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_\iota}\right) - \frac{\partial L}{\partial q_i} - \frac{\partial D}{\partial \dot{q}_\iota} = f_i \quad (5)$$

In this function, $i = 1,2,\cdots,n$. The system chooses θ and φ as the two broad coordinates of the system. Find the generalized torque of the system as:

$$f = \left[-I_2 \ddot{\varphi} u - I_2 \ddot{\theta}\right]^T \quad (6)$$

Where u is the driving torque of the flywheel.

The dissipation functions of the system include:
dissipation force of rod:

$$\tau_1 = -c_1 \dot{\theta} \quad (7)$$

dissipation force of flywheel:

$$\tau_2 = -c_1 \dot{\theta} \quad (8)$$

Where c_1 is the friction torque coefficient of rod as it rotates around a rotating shaft and c_2 is the friction torque coefficient when the flywheel rotates around the motor shaft. Therefore, we can get:

$$(a + I_2)\ddot{\theta} + I_2 \ddot{\varphi} = b \sin\theta - c_1 \dot{\theta} \quad (9)$$

$$I_2\left(\ddot{\theta} + \ddot{\varphi}\right) = u - c_2 \dot{\varphi} \quad (10)$$

Formulas (9) and (10) are the kinematics expressions of a system that ignores air resistance.

3.2. Control Design of Balance System

The primary function of this system is to control the bike's balance. Once the system is powered, all devices are initialised first, and once initialisation is complete, the CH110 gyroscope data, momentum wheel revs and rear wheel revs are read every 5 msec. Either by dialling the dial code or by receiving the signal from the master machine to start the car's balance control.

The balance of the body is controlled by cascade PID controller [3] (Shown in Figure 3). When the bike leans to one side, the body can be powered back to balance by the reverse torque of the output of the momentum wheel and the rotation.

What is needed to maintain equilibrium is to control the angle to the mechanical zero angle. So, starting with a PD closed-loop control system with a single angle loop, calculated the output of motor torque and controlled it every 10ms. The input is the deviation of the car's distance to balance point and the angular velocity at this point, which take place of the differential equation of PID. The output is the motor's current being torque. Experiments showed that it can maintain a few seconds of balance, but when it converges to the mechanical zero angle the car lurched sideways and the momentum flywheel were too fast that can't output enough torque to maintain balance.

To address this problem. We add the speed loop of flywheel to be its outer loop and provide its output to the angle loop input. So that the balance point will be corrected in real time by speed feedback. The speed loop uses PI controller, and according to cascade control theory, the frequency of the outer loop needs to be slower than the inner loop, running every 40ms to give the angular inner loop enough time to reach the current balance point. Experiments have shown that the system stabilizes well, quickly regains balance when pulse signal is applied. And it can adjust to a new point to maintain balance rapidly when persistent disturbance is placed on the cycle. The effect is aided by the integrating element of the speed loop to give the bike system its current equilibrium angle.

After testing this balance system holds strong dynamic response and the Robustness is excellent.

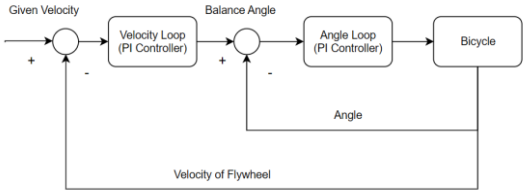

Figure 3. Cascade balance control system.

4. Implementation of Self-Driving System

The ROS-based Robotics Control host, Jetson Nano, runs on Linux systems. It uses the ROS distributed framework to control the bike as a whole, obtains IMU and odometer data from the next machine through serial port. Then integrates the data to obtain relatively accurate body posture and position information of the bike, which is fused through the extended Kalman filter program. And obtains environmental information by using radar and depth camera to realize the SLAM location map [4]. Finally, navigation controller module subscribes to odometer data and global path, uses Pure Pursuit algorithm for global path tracking, and uses bicycle kinematics model for motion calculation, uses PI control algorithm to achieve speed closed-loop control, and outputs data to the serial communication module, and then output to the bottom single-chip control module to realize the movement control of the robot [5].

In a nutshell, it is primarily designed by the C++ language, which uses the Pure Pursuit algorithm under the joint action of the communication node, navigation node, Odom processing node, and build diagram, and ultimately realizes a series of obstacle avoidance and path planning functions.

5. Qt Debugging Interface Design

Qt interface (shown in figure 4) based on Python development, using PYQT to build the interface. And the TC377 microcontroller using UDP protocol communication, through the microcomputer Flash modification save function, easy to adjust the bike parameter.

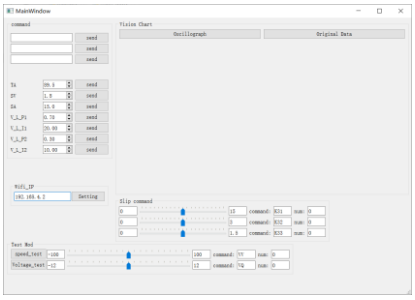

Figure 4. Qt interface.

6. Conclusions

The self-balancing bike system in this paper provides an effective driverless bicycle control scheme, which can achieve the balance and automatic navigation of the bicycle system. The combination of self-balancing and self-driving technologies can be used in specific environments to achieve automatic bicycle system driving and handle special events, replacing manual labor and mechanical operations. In addition, this project has reference significance for the future unmanned bicycle.

Acknowledgment

This research work is supported by National Natural Science Foundation of China (Grant No. 52005472) and Natural Science Foundation of Zhejiang Province (Grant No. LQ20E050015).

References

[1] Astrom KJ, Klein RE, Lennartsson A, et al. Bicycle dynamics and control: Adapted bicycles for education and research. IEEE Control Systems Magazine. 2005; 25: 26-27.
[2] Muskinja N and Tovornik B. Swinging up and stabilization of a real inverted pendulum. IEEE Transactions on Industrial Electronics. 2006; 53: 631-639.
[3] Ping L, Shi Y, Wang XF, et al. Equivalence analysis of cascade control for a class of cascade integral systems. IEEE Access. 2023;11: 12237-12248.
[4] Tian CJ, Liu HB, Liu Z, et al. Research on multi-sensor fusion SLAM algorithm based on improved gmapping. IEEE Access. 2023; 11: 13690-13703.
[5] F. Ugalde Pereira, P. Medeiros de Assis Brasil, M. A. de Souza Leite Cuadros, et al. Analysis of local trajectory planners for mobile robot with robot operating system. IEEE Latin America Transactions. 2022; 20: 92-99.