Improving the Navigation with Mecanum Wheels in Botball

Felix Steinwendter, Sebastian Munkhbat
Higher Technical Federal Teaching and Research Institute
Department of Computer Science
2700 Wiener Neustadt, Austria
Corresponding author's email: felix.steinwendter@gmail.com

***Abstract*—This paper discusses the implementation of a mecanum-wheel-based robot for the Botball tournament. The omni-directional wheel design provides a high degree of maneuverability and flexibility, making it highly advantageous in areas with limited turning space. This paper derives equations for navigating a mecanum-wheeled robot along predefined paths. To mitigate real-world trajectory deviations, software-based motor compensation techniques are implemented to correct wheel misalignment.**

***Index Terms*—mecanum wheel, Botball, robotics, omni-wheel**

I. Introduction

The mecanum wheel, also referred to as the Swedish wheel or Ilon wheel, is a form of an omni-directional wheel composed of a series of rubberized rollers mounted along its perimeter at a 45° angle. A typical mecanum-wheel configuration consists of four wheels arranged in a rectangular pattern, with their axle perpendicular to the robot's body. The wheels alternate between left- and right-handed rollers, ensuring that rollers on diagonally opposite wheels are aligned parallel to each other.

Due to this design, a wheel primarily generates force towards a diagonal in the robot's reference frame. Through careful manipulation of each wheel's rotational speed and position on an inertial reference frame, arbitrary longitudinal and lateral movement can be achieved. Furthermore, low-friction rotation relative to the robot's center point can be incorporated, allowing for smooth motion along user-defined trajectories.

These unique characteristics are useful for a wide variety of applications, ranging from extraterrestrial exploration vehicles to omnidirectional wheelchairs or transportation vehicles such as forklifts [1].

II. Controlling the robot

A. Equipment

The robot in Fig.1, powered by a robotics controller based on the Raspberry Pi 3B+, was designed based on a rectangular metal chassis symmetrically equipped with four mecanum wheels. Their velocities can be precisely and independently controlled using pulse-width modulation (PWM) driven motors. A laser-cut wooden alignment tool was used to ensure no discrepancies in initial positioning across test runs. To avoid manual intervention with the robot at runtime, test runs were initiated remotely, and optionally a light sensor serving as a start signal was integrated into the robot. Additionally, the placement of the servos, robotics controller (codenamed "wombat") and robotic claw arms were placed in a way that equally distributes weight across the platform.

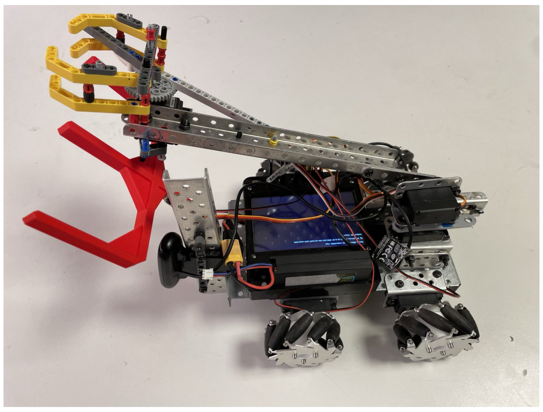


Figure 1: The robot used for testing

B. Kinematic Model

In the provided image and the robot used for the tests, both the front and rear axles are the same width. However, real-world implementations of such a robot could include different axle widths for various reasons. While these discrepancies would not affect straight and sidewards motion, diagonal or omnidirectional movements could be influenced, as asymmetric force distribution would introduce deviations from the expected path. To counteract, continuous monitoring and correction would be necessary, if the mathematical equations for calculating velocities are not adapted accordingly.

The robot's kinematic model's system velocities and configuration values as visualized in Fig. 2 for translational and rotational motion are defined as the following:

- H, Y, X: inertial frame of the reference point H in the inertial basis;
- O: robot's x and y position relative to it's inertial frame

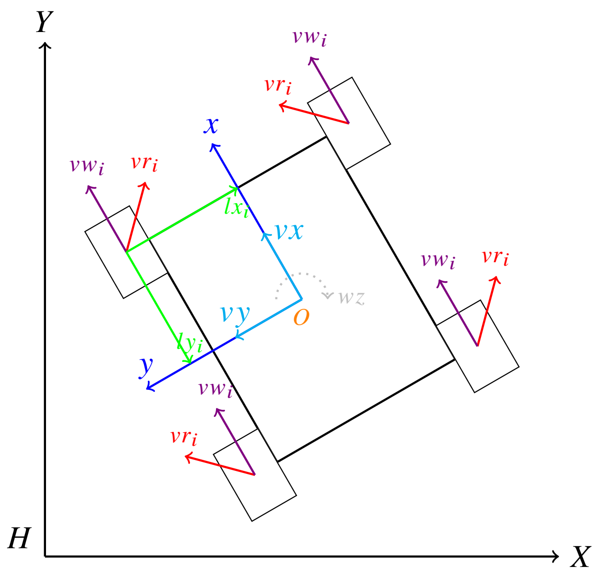


Figure 2: Schematic of the robots geometry

- x, y: coordinate system centered at the robot's body associated.
- vx, vy [m/s]: robot's linear velocity
- ωz [rad/s]: robot's angular velocity
- vw_i $(0 \leq \mathrm{i} \leq 3)$ [m/s]: velocity vector of the i-th wheel corresponding to wheel revolutions
- vr_i $(0 \leq \mathrm{i} \leq 3)$ [m/s]: velocity vector of the i-th wheel corresponding to roller revolutions
- lx_i, ly_i $(0 \leq \mathrm{i} \leq 3)$: distance between the i-th wheel and the robot's center
- dw: absolute euclidean distance between a wheel and the robot's center
- ω_i $(0 \leq \mathrm{i} \leq 3)$ [rad/s]: i-th wheel's angular velocity

The forward kinematic equations for the robot are derived as follows [2]:

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \frac{1}{r} \begin{bmatrix} 1 & -1 & -dw \\ 1 & 1 & dw \\ 1 & 1 & -dw \\ 1 & -1 & dw \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} \tag{1}$$

A function `mecanum_setwhlspd(double vx, double vy, double wz)` has been incorporated into the C++ robot library, which applies this equation to calculate the appropriate power to exert to the PWM-controlled motors of each wheel. It expects a target velocity as a vector for both the x and y axes and a target rotation around the robot's center as it's arguments.

C. Kinematics Theory

This section of the paper will focus on elaborating on the variables visualized in Fig.2, as well as explaining how they are used to mathematically derive the equation allowing for accurate control. Each wheel, denoted as the i-th wheel, has two rotary components, the wheel itself and the rollers attached to it. The variable vw_i refers to the velocity of the i-th wheel measured in m/s. vr_i is used to describe the velocity of the rollers on the wheel i in m/s. The distance between a wheel and the robots center is represented as the vector (lx_i, ly_i). Since vr_i cannot be easily measured, it is calculated using the formula

$$vr_i = \frac{1}{cos(45)} * r_r * \omega_i$$

where r_r equals the radius of the rollers and ω_i a wheel's angular velocity measured in rad/s. According to the following equations

$$v_{S_i} = vr_i * sin(\gamma_i)$$

$$v_{E_i} = vr_i * cos(\gamma_i) + \omega_i * r_i$$

in which ω_i equates to the angle between vw_i and vr_i, γ_i represents the angle between the previously calculated vr_i and vw_i, and lastly r_i is the radius of the i-th wheel, the vectors and the speed at which each wheel moves are now known. v_{E_i} and v_{S_i} represent the velocity vector of one of the mecanum wheels and the velocity vector of the rollers on the same wheel respectively. The omni-directional movement is achieved by coordinating the four wheels so that desired velocity vectors reinforce each other, while undesired forces cancel out. This principle is most evident when moving forward or backward, where all four wheels rotate at the same speed in the same direction. However, if one wheel loses traction or slows unexpectedly, the robot's movement may become unstable. This becomes most obvious when wanting to accelerate forward or backwards, since you merely need to turn all four motors at the same time, with the same speed, but should one wheel have a worse grip on the ground or suddenly decelerate, the robot will suddenly move differently.

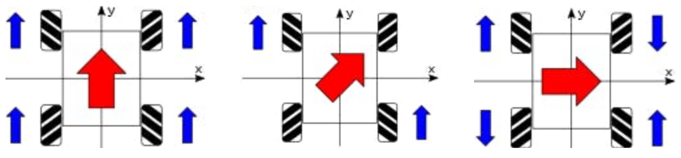
To make the robot turn on its own axis, the wheels need to be alternatively spun forward or spun backward, depending on the direction in which the robot needs to turn. To have accurate turns, measured in the degrees turned from the current position, the angle turned in a single tick and the original orientation will be taken and used to calculate the needed turn time. To ensure the accuracy in real life application, a PID controller will be used as a feedback loop for the robot. This will be explored in more detail further on.

III. Real Life Application

While omni-directional robots have a wide variety of usecases [1], the practical scope is narrowed down to use within the Botball robotics competition by KIPR (kipr.org). Thus, the experiments focus on evaluating the performance of such robots in competitive environments. For that reason, driving as fast as possible is essential. Experimental results were obtained using the robot shown in Fig. 1, a custom-designed robot made for the Botball competition. The measurements as described in the next section were conducted by placing the robot in a fixed location using a custom alignment tool to ensure consistent placement. Then, the robot was programmed to follow predefined paths or travel in a specific direction. This allowed us to collect reliable data under controlled conditions. An onboard gyroscope was used to record rotational drift, wheel encoders to collect data about the distance traveled, and a measuring tape for manual verification. Each test was repeated multiple times and results averaged to account for variability caused by external factors.

IV. Quantifying drift over time

The 45-degree angled rollers, while enhancing mobility, introduce higher friction than conventional wheels. This leads to increased lateral and rotational drift, affecting movement precision. Uneven floors and slippery surfaces further negatively impact reliability. The most common driving directions, 0°, 45° and 90°, are therefore analyzed to provide data on drift when driving in each of them as in Fig.3, respectively.



(a) 0° (b) 45° (c) 90°

Figure 3: Fundamental movement directions

A. Measurement methodology

To be capable of making meaningful predictions about drift in an isolated environment, specifically a known uniform surface and a fixed robot configuration, two system variables have to be gathered for strictly linear movement: the rotational drift constant and the lateral drift constant. There are a number of methods by which both can be obtained, but the decision was made to employ manual measurements in combination with onboard instruments in a realistic setting representing a Botball table. The rotational drift constant is obtained by placing the robot at a fixed starting position, and the `mecanum_setwhlspd()` as described in Section II is employed to steer the robot in the desired direction. After a few seconds, the motors are halted and the rotation reported by the gyroscope is read.

The sideways drift is trickier to accurately determine, as it is first necessary to reduce rotational drift to near zero. Using a closed-loop proportional-integral-derivative (PID) control system, specifically designed to stabilize the yaw of the robot, negating rotational drift, the lateral drift can be extracted effectively. The PID error feedback was taken directly from the onboard gyroscope, reporting yaw variations at a precision of $\frac{1}{100}$ of a degree. The PID control law was defined as

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt}, \qquad (2)$$

where $u(t)$ is the control output, $e(t)$ the error feedback (delta between the desired and current yaw), and K_p, K_i, and K_d are the proportional, integral, and derivative gains, respectively. These were empirically tuned to approach zero rotational drift with no visible overshoot and good reliability even over longer distances, keeping the robot within 1° of variation between the starting and end point. A proportional controller using the same feedback was tested, and achieved similarly consistent results.

The test procedures were repeated ten times, and the averaged results are displayed in table I, listing the average velocity as well rotational and lateral drift per second.

Table I: Measurement data

	Data		
	average velocity	rotational drift	lateral drift
0°	0.24 m/s	1.4°/s	0.01 m/s
45°	0.15 m/s	2.4°/s	0.03 m/s
90°	0.18 m/s	9.87°/s	0.03 m/s

This table summarizes the robot's path data for the three initial orientation angles in Fig.3. Each row represents the average results of each ten test cases, and the columns provide the following information:

- **average velocity [m/s]**: The average velocity the bot was traveling at during the test, calculated by dividing the total distance driven by the time taken.
- **rotational drift [/s]**: The average rotational drift being accumulated every second.
- **lateral drift [m/s]**: The average lateral drift being accumulated every second.

B. Trajectory deviation formula

Using the forward velocity, sidewards drift and rotational drift constant obtained, a formula for the position (x, y) of the robot after N seconds can be derived.

The paramters are defined as follows:

- ω [rad/s]: rotational drift constant converted to radians per second
- dv [m/s]: sidewards drift constant in meters per second
- df [m/s]: forward velocity constant in meters per second
- N [s]: time in seconds
- t [rad]: orientation angle in radians
- θ [rad]: orientation angle in radians

It is being assumed that the robot's origin is at $(0, 0$ in world space, with an initial orientation angle $t = 0$. The initial orientation θ is assumed to be along the positive y axis: $\theta(0) = 90 = \frac{\pi}{2}$ radians.

Since these calculations require radians instead of degrees, all constants previously denoted in degrees have to be converted using the formula

$$\omega = \frac{\omega_{\text{deg}} \pi}{180} \text{rad}$$

The first part of the equation is the orientation as a function of time. θ changes linearly with time due to the assumption of a constant rotational drift:

$$\theta(t) = \theta(0) + \omega t = \frac{\pi}{2} + \omega t$$

The second part is the velocity component, composed of a forward velocity v_f:

$$v_{f,x} = v_f \cos(\theta(t))$$
$$v_{f,y} = v_f \sin(\theta(t))$$

and a second part, the sideways drift v_s, acting perpendicular to the forward direction $(\theta(t) - \frac{\pi}{2}$:

$$v_{s,x} = v_s \cos\left(\theta(t) - \frac{\pi}{2}\right) = v_s \sin(\theta(t))$$
$$v_{s,y} = v_s \sin\left(\theta(t) - \frac{\pi}{2}\right) = -v_s \cos(\theta(t))$$

The resulting differential equation is thus:

$$\frac{dx}{dt} = v_{f,x} + v_{s,x} = v_f \cos(\theta(t)) + v_s \sin(\theta(t)),$$
$$\frac{dy}{dt} = v_{f,y} + v_{s,y} = v_f \sin(\theta(t)) - v_s \cos(\theta(t))$$

Substituting $\omega(t)$ and its trigonometric identities:

$$\cos\left(\frac{\pi}{2} + \omega t\right) = -\sin(\omega t), \quad \sin\left(\frac{\pi}{2} + \omega t\right) = \cos(\omega t)$$

The velocity component is simplified to:

$$\frac{dx}{dt} = v_f(-\sin(\omega t)) + v_s \cos(\omega t) = -v_f \sin(\omega t) + v_s \cos(\omega t)$$
$$\frac{dy}{dt} = v_f \cos(\omega t) - v_s(-\sin(\omega t)) = v_f \cos(\omega t) + v_s \sin(\omega t)$$

To get to a specific x and y position after N seconds, the equation is integrated from $t = 0$ to $t = N$:

$$x(t) = \int_0^t \left(-v_f \sin(\omega s) + v_s \cos(\omega s)\right) ds$$
$$y(t) = \int_0^t \left(v_f \cos(\omega s) + v_s \sin(\omega s)\right) ds$$

Next, the evaluation of both integrals:
For $x(t)$:

$$x(t) = \left[\frac{v_f}{\omega} \cos(\omega s) + \frac{v_s}{\omega} \sin(\omega s)\right]_0^t$$
$$x(t) = \frac{v_f}{\omega}(\cos(\omega t) - 1) + \frac{v_s}{\omega} \sin(\omega t)$$

For $y(t)$:

$$y(t) = \left[\frac{v_f}{\omega} \sin(\omega s) - \frac{v_s}{\omega} \cos(\omega s)\right]_0^t$$
$$y(t) = \frac{v_f}{\omega} \sin(\omega t) + \frac{v_s}{\omega}(1 - \cos(\omega t))$$

The final equation to calculate or position at $t = N$ is:

$$x(N) = \frac{v_f}{\omega} (\cos(\omega N) - 1) + \frac{v_s}{\omega} \sin(\omega N),$$
$$y(N) = \frac{v_f}{\omega} \sin(\omega N) + \frac{v_s}{\omega} (1 - \cos(\omega N)),$$

There is a special case in which $\omega = 0$ (no rotation all), where the motion is linear and can be reduced to:

$$x(N) = v_s N, y(N) = v_f N$$

This simplified equation is applicable for modeling the trajectory with rotational drift correction enabled, as visualized in Fig.4. Using this approximating equation, the position over time assuming no correction and with rotational correction are graphed.

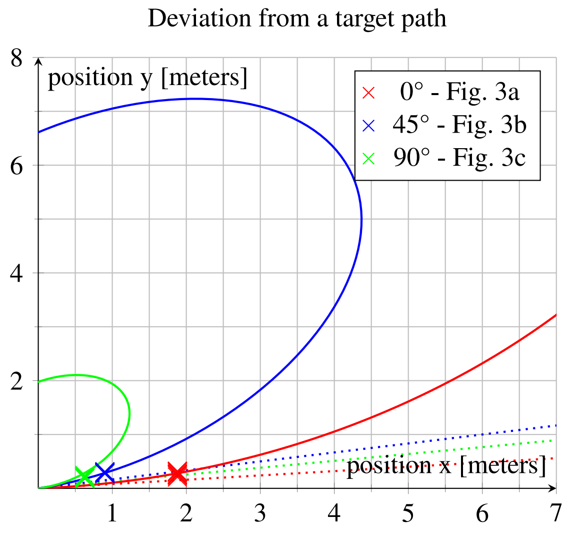


Figure 4: Physical measurements taken at certain distances are used to approximate a function correlating to the absolute position, including deviation from the straight-line path, when driving in each of the primary directions, traced as a function of time. More test runs are conducted without any trajectory control mechanisms in place, their final positions visualized as crosses, to validate the correctness of the approximations. Additionally, paths without any rotational drift are graphed (dotted lines). They can be used to loosely approximate real world paths with any of the correction methods described in IV-A applied.

V. Countering lateral drift

For certain applications, unlike the Botball competition, it may be desirable to minimize the deviation even further. Negating lateral drift to is essential to achieve that goal; theoretical solutions are conceptualized within this section.

A. Constant counter-steering

The sideways drift velocity, denoted v_s and measured in meters per second, corresponds to a velocity perpendicular to the current heading. If $v_s > 0$ corresponds to a drift to the left, $v_s < 0$ effectively induces a countering drift to the right. By selecting an appropriate value for v_s, one can counteract lateral drift. However, different drift constants as the orientations changes and possible variations across multiple runs complicate this approach, and due to it not being a viable option for curved paths either, it remains theoretical.

B. Feedback-based Control Strategies

Practical alternatives may rely on feedback control to minimize lateral displacement. A simple approach could be to employ visual indicators to derive positional data from, for example ArUco markers to calculate an accurate pose, a line to follow or other external hints. Optionally, an accelerometer-based PID could be fine-tuned to deliver the desired countersteering motion. Furthermore, predictive models could minimize deviations by predicting future lateral movement and adjusting in a preventive manner. Beyond adjustments, other mechanisms can effectively work against lateral drift. If the system allows for accurate control over each motors velocity, slower rotations at critical sections could lessen unwanted effects. These strategies are particularly effective in real-world scenarios involving external disturbances.

VI. Conclusion

The unique properties of mecanum wheels enabled omni-directional movement, allowing for motion in all directions and rotation about the z axis. Straight-line motion on a 2-dimensional plane could be achieved entirely without altering the robot's heading, while following curved trajectories is made possible by adjusting it's rotation. However, the design of mecanum wheels and their rollers introduce drift, becoming an apparent issue in precise navigation. Several experiments were conducted to assess its impact on straight-line motion. The results lead to the conclusion that maintaining a stable heading is the most effective measure to counter the drift. The material properties of the mecanum wheel rollers and the track surface have been shown to have a significant effect on course stability. Therefore, it is essential that all tests are performed on a uniform, clean surface in order to mitigate these effects. To successfully eliminate rotational drift, the effectiveness of a closed-loop PID controller based on gyroscopic readings was demonstrated. This stabilization reduces total drift down to lateral drift, for which control methods have been proposed. Their applicability depends heavily on usecase and remains largely unexplored, but it's theorized that a position-aware driving model could yield a more robust system and improved accuracy.

Acknowledgement

The authors would like to thank Dr. Michael Stifter for supporting the paper in providing both writing and technical assistance. They would also like to thank Dipl.-Ing. Harald Haberstroh, who has so kindly offered his skills and years of knowledge to robo4you.

References

[1] F. Adăscăliţei and I. Doroftei, "Practical applications for mobile robots based on mecanum wheels-a systematic survey," *The Romanian Review Precision Mechanics, Optics and Mechatronics*, vol. 40, pp. 21–29, 2011.

[2] N. G. Hamid Taheri, Bing Qiao, "Kinematic model of a four mecanum wheeled mobile robot," *International Journal of Computer Applications*, vol. 113, no. 3, pp. 6–9, March 2015. [Online]. Available: https://ijcaonline.org/archives/volume113/number3/19804-1586/

[3] A. Gfrerrer, "Geometry and kinematics of the mecanum wheel," *Computer Aided Geometric Design*, vol. 25, no. 9, pp. 784–791, 2008, classical Techniques for Applied Geometry. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167839608000770

[4] N. Tlale and M. de Villiers, "Kinematics and dynamics modelling of a mecanum wheeled mobile platform," in *2008 15th International Conference on Mechatronics and Machine Vision in Practice*, 2008, pp. 657–662.

[5] D. Xu and Y. Gao, "Modeling and analysis of mecanum wheel and its four-wheel system."

[6] L. Lin and H. Shih, "Modeling and adaptive control of an omni-mecanum-wheeled robot," *Intelligent Control and Automation*, vol. 4, pp. 166–179, 2013.

[7] H. Xu, G. Yu, Y. Wang, X. Zhao, Y. Chen, and J. Liu, "Path planning of mecanum wheel chassis based on improved a* algorithm," *Electronics*, vol. 12, no. 8, 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/8/1754

[8] S. Dickerson and B. Lapin, "Control of an omni-directional robotic vehicle with mecanum wheels," in *NTC '91 - National Telesystems Conference Proceedings*, 1991, pp. 323–328.

[9] A. Patel. (2024, oct) Omni-directional robots based on the mecanum wheel. [Online]. Available: https://nhsjs.com/2024/omni-directional-robots-based-on-the-mecanum-wheel/

[10] V. Alakshendra and S. S. Chiddarwar, "Adaptive robust control of mecanum-wheeled mobile robot with uncertainties," *Nonlinear Dynamics*, vol. 87, no. 4, pp. 2147–2169, March 2017. [Online]. Available: https://doi.org/10.1007/s11071-016-3179-1

[11] M. Alfiyan and R. D. Puriyanto, "Mecanum 4 omni wheel directional robot design system using pid method," *Journal of Fuzzy Systems and Control*, vol. 1, no. 1, p. 6–13, Mar. 2023. [Online]. Available: https://ejournal.ptti.web.id/index.php/jfsc/article/view/27