Enhancement of Accuracy in Botball Navigation

Bernhard Klauninger*, Konstantin Lindorfer, Sebastian Kawicher
Höhere Technische Bundes Lehr- und Versuchsanstalt Wiener Neustadt
(Federal Technical Secondary College)
Department of Computer Science
2700 Wiener Neustadt, Austria
*Corresponding author's email: bernhard@klauninger.at

***Abstract*—Accurate and consistent positioning of a robot is a key factor to success in Botball. However, repeated execution of a program that moves the robot will result in a slightly different outcome every time. With the many physical factors involved and the limited number as well as types of sensors allowed, reducing the variation in motion to negligible levels has always been a challenge.**

This paper explores methods to enhance the accuracy of a Botball robot's movements by exploiting the untapped potential of the gyroscope built into the Wombat controller. Through a series of experiments, such as driving in a straight line or turning an angle, the authors compare the effectiveness of different techniques and show that using the Wombat's inertial sensors can greatly improve the accuracy and consistency of a robot's movements. It also eliminates the need for frequent recalibration of correction constants, allowing for much faster development of Botball strategy programs.

I. Introduction

An autonomous robot is designed and programmed to perform a specific task accurately and reliably. Achieving this can be challenging because the physical world adds countless factors that cause the actual outcome to deviate from the target outcome. Based on previous research conducted by the authors of [1], this paper lays the foundation for future studies to explore the potential of underutilized sensors. Numerous methods exist to improve the consistency of autonomous robots, including the enhancement of their structural robustness, the application of advanced sensors, and the optimization of their operating environment. However, in the context of the Botball competition, the opportunities for the use of these methods are severely limited, necessitating the optimal use of the available resources.

Commonly used methods for correcting the movements of a Botball robot involve the application of Back-EMF to create a mechanism that minimizes deviations during motion. In contrast, despite their potential benefits, the Wombat controller's other inertial sensors, such as the gyroscope, remain largely unexplored and thus underutilized in Botball. Therefore, the authors decided to investigate the effectiveness of gyroscope-based methods for correcting the motions of Botball robots and compare their effects with those of conventional methods to further improve the accuracy and consistency of robots in Botball.

II. The Given Environment

To design a robot that can successfully operate autonomously in a given environment, it is necessary to understand the static and dynamic components of that environment and how they interact. This includes factors such as physical characteristics, available resources, and temporal and spatial dimensions. Furthermore, it is important to consider the real-time measurements and corrections needed to accurately respond to dynamic changes in the environment. By considering the above factors, a robot can be designed and developed to effectively operate in a given environment.

A. Botball

In our given environment of Botball, robots must perform various tasks on a game table to score points. If the area in which a team wishes to score has a reference point, such as a black tape line on the ground, a sensor can be used to find that point, navigate to it consistently, and position the robot to perform its task. However, if there is no such reference, the team must find another way to position the robot. This could include driving a fixed distance from a more distant landmark, such as the border of the game table or a black tape line elsewhere on the ground, which is where the research conducted in this paper should prove useful.

B. The Wombat Controller

The Wombat represents the current state-of-the-art controller iteration for the Botball competition, incorporating advanced technological features built upon the Raspberry Pi platform [2]. The device provides access to most of the standard Raspberry Pi ports, including Ethernet and USB, as well as a range of analog and digital input and output ports for connecting motors, servos, and sensors. Additionally, the Wombat features a touchscreen with a graphical user interface, facilitating ease of use for peripheral testing and program execution. Power is supplied through an external lithium iron phosphate ($LiFePO_4$ or LiFe) battery pack. The chip extending the Raspberry Pi comes with some more notable features:

1) Back-EMF: To determine the current state of a Botball motor, the Wombat incorporates a technique that measures the motor's counter-electromotive force, commonly referred to as Back-EMF [3]. It leverages the principle that an electric motor is also an electric generator. By briefly interrupting the power to the motor during operation, the amount of energy produced

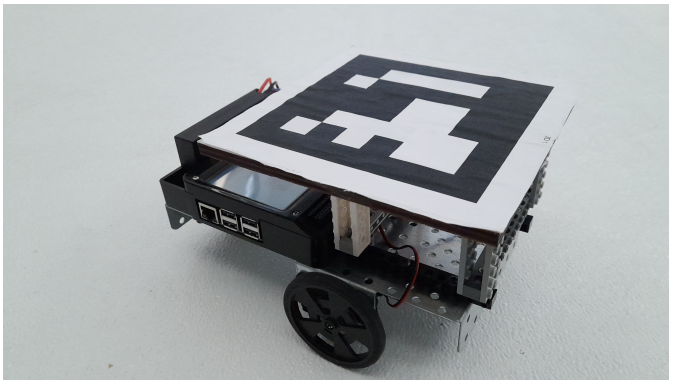

Fig. 1: The robot used for experiments in this paper

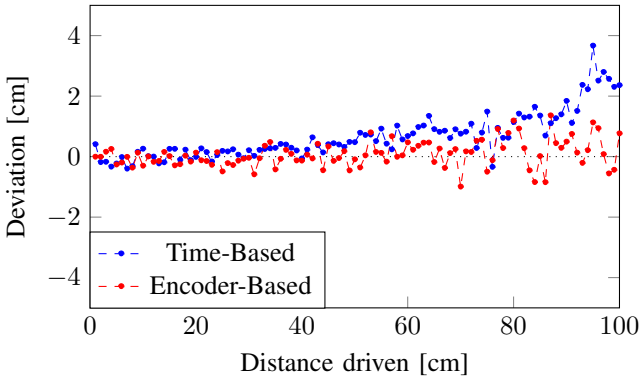

Fig. 2: Deviation from the intended distance when driving in a straight line

by the motor can be measured, which is directly proportional to its rotational velocity. The controller then approximates the rotational position of the motor using this information and provides it in a unit called ticks, where 1820 ticks are roughly equivalent to one revolution of the motor [4].

2) IMU: The positional state of a robot can be effectively measured using an Inertial Measurement Unit (IMU). An IMU typically consists of several sensors, including but not limited to an accelerometer, a gyroscope, and a magnetometer. The accelerometer measures acceleration, the gyroscope measures angular velocity, and the magnetometer can determine cardinal direction.

The Wombat is equipped with an MPU-9250 [5], a System in Package (SiP) combining the MPU-6500 and the AK8963. The MPU-6500 includes a 3-axis gyroscope, a 3-axis accelerometer, and an onboard Digital Motion Processor (DMP) capable of integrating data from multiple sensors through complex sensor fusion algorithms [6]. The added AK8963 is a 3-axis digital compass.

C. Mechanical Structure

To minimize inaccuracies in movement, a rigid robot structure is desirable. In this experiment, we use a very minimal construction based on the Wombat Robot Build Guide by KIPR [7]. As shown in Fig. 1, the robot is built using a metal chassis with two wheels directly mounted to motors on the front axis, a ball caster on the rear, the Wombat controller, and an ArUco marker [8] centered on the rotational axis.

D. Experimental Setup

To accurately determine the position and rotation of the robot during our experiments, we applied a downward-facing camera within a confined area. An ArUco marker is placed in the upper left corner of the camera's field of view, representing the origin of a Cartesian coordinate system along the ground. Using this marker in combination with the marker mounted on the robot, the position of the robot in centimeters and its rotation in degrees relative to this point can be obtained at any time by calling an API endpoint. This fully automated setup not only speeds up the measurement process compared to manual measurement but also eliminates the potential for human error.

III. Driving for a Specific Distance

One of the challenges in the navigation of an autonomous robot is driving forward or backward a certain distance. When there are no external points of reference, the robot needs a way to measure the traversed space.

A. Time-based Distance Estimation

A straightforward approach to measuring the traveled length is to record the elapsed time since the initiation of movement and halt after a certain amount of time has passed. Assuming that the robot is always driving at a constant speed, the time required to reach a certain distance can be calculated by multiplying the desired distance by a conversion factor, C_A. This factor is the slope of the linear relation between the desired distance and the time required to reach it and can be calculated as:

$$C_A = \frac{\Delta time[sec]}{\Delta distance[cm]}$$

To evaluate the effectiveness of this method, we conducted experiments in which the robot traversed distances ranging from 1 to 100 centimeters in 1-centimeter increments. The deviation from the intended distances is shown by the blue graph in Fig. 2. At first glance, this method seems fairly accurate for shorter distances and still viable for longer distances, but any changes in the environment can easily affect the conversion factor needed to drive the correct distance. These environmental factors include surface conditions, motor performance, the force required to move the robot, and many more. Since most of these factors naturally change over time, it is not feasible to calculate the correct conversion constant by taking them all into account, making distance estimation based on time impractical.

B. Measuring the Motor Position using Back-EMF

Our next attempt to approximate the distance the robot has traveled is to utilize the built-in Back-EMF measurement. By

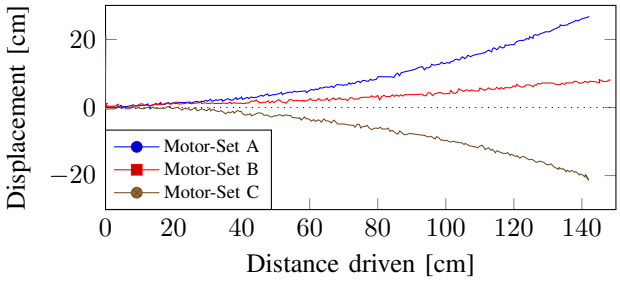

Fig. 3: Path of the robot when driving straight without any correction with three different sets of motors

multiplying the arithmetic mean of the left motor position, pos_L, and the right motor position, pos_R, with a new conversion factor, C_B, the current distance traveled can be calculated:

$$distance[cm] = C_B \cdot \frac{pos_L[ticks] + pos_R[ticks]}{2}$$

where C_B, similar to C_A, is the ratio of the distance traveled in centimeters to the position change of each motor in ticks:

$$C_B = \frac{\Delta distance[cm]}{\Delta position[ticks]}$$

This information can then be used to stop the robot when it reaches a certain distance in centimeters.

Driving distances ranging from 1 to 100 centimeters in 1-centimeter increments using this method shows slightly better accuracy than the time-based method, as shown by the red graph in Fig. 2. However, using a sensor-based measurement such as this is significantly less susceptible to external factors than relying on the elapsed time to estimate the distance traveled, especially to the speed at which the motors are being driven.

IV. Driving in a Straight Line

Another challenge faced in the navigation on a Botball game table is driving in a straight line. When no external points of orientation are provided, the robot has to rely on internal sensors to maintain its heading.

A. Without Any Correction

The simplest method for driving in a straight line is to activate both motors and continue driving until the desired distance is reached. However, as shown in Fig. 3, this approach results in an arcuate path rather than a straight line. Since using other motors of the same model results in a different arcuate path, it can be inferred that this deviation is due to the difference in motor strength.

While this difference in motor strength could be compensated for by reducing the power supplied to the stronger motor, this approach requires frequent manual calibration as the motors wear down, and is prone to human error. This would lead to inconsistent performance and is not ideal for navigation in Botball.

loop
$error \leftarrow motorTicks_A - motorTicks_B$
$integral \leftarrow integral + error$
$derivative \leftarrow error - lastError$

$correction \leftarrow (K_p \cdot error) + (K_i \cdot integral) + (K_d \cdot derivative)$

set motor A to $targetPower - correction$
set motor B to $targetPower + correction$

$lastError \leftarrow error$
end loop

Fig. 4: An algorithm to balance the position ticks of two motors using a PID controller [9]

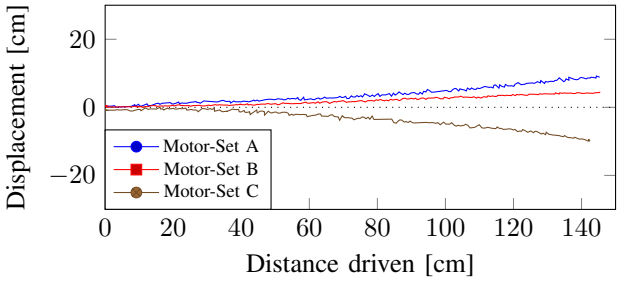

Fig. 5: Path of the robot when driving straight using Back-EMF balancing with three different sets of motors

B. Back-EMF Balancing

Theoretically, if the position ticks of each motor increase equally as the robot moves, it is moving in a straight line. This principle can be utilized to reduce the power of a motor if it has progressed too far. To optimize this correction, a proportional-integral-derivative (PID) controller, such as the one shown in Fig. 4, can be implemented [9]. This algorithm continuously monitors the difference between the position ticks of both motors and adjusts the power supplied to each motor proportionally, with integral and derivative terms, to minimize this error and maintain near-equal position ticks. To find ideal constants for each term of the PID controller, the Ziegler-Nichols tuning formula [10] can be employed.

Fig. 5 shows the path traced by the robot when the left and right motors are balanced according to their tick position using a PID controller. Compared to the path followed without any correction, the actual path of the robot is much closer to the ideal, straight path. The circular curvature, however, remains present. Since Back-EMF position measurement depends on the strength of each motor, and because the difference in strength between two motors can vary, it is not feasible to consistently achieve a perfect balance between two motors using this method.

C. Gyroscope

The final method discussed in this paper for achieving straight-line motion uses the gyroscope built into the Wombat controller to correct for angular motion. By continuously adjusting the power supplied to each motor based on the difference between the current gyroscope ticks and the initial

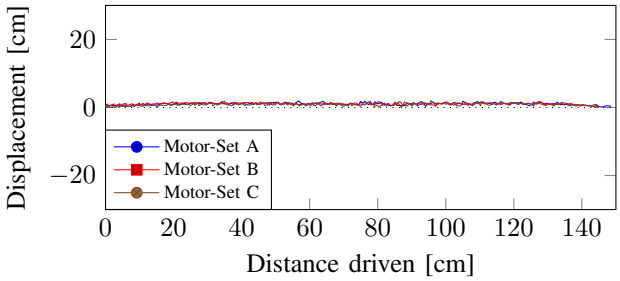

Fig. 6: Path of the robot when driving straight using the gyroscope to correct lateral movement with three different sets of motors

value, the heading of the robot should remain constant. For this purpose, the PID controller used previously is adapted to use the gyroscope ticks as the error value.

As shown in Fig. 6, the displacement of the robot using this method is negligible, making it far more successful than the other methods discussed. It also has the added benefit of using more exact data about the physical motion of the robot, eliminating inaccuracies caused by variations in motor strength and enabling the correction of the robot's heading in the event of an accidental collision.

V. Turning an Angle

The last challenge addressed in this publication is the rotation of a Botball robot around its Z-axis. The algorithm for this type of motion is generally very similar to the algorithm for driving in a straight line, so the methods developed for driving a certain distance and maintaining a straight heading can be adapted for this purpose.

A. Reaching the Desired Angle

Since rotating around the Z-axis is the same as driving a distance with one motor reversed, the angle a robot has currently turned can be calculated using the current motor position and a conversion factor, C_C, as such:

$$angle[deg] = C_C \cdot \frac{pos_A[ticks] - pos_B[ticks]}{2}$$

where pos_A is the position of the forward-rotating motor in ticks, pos_B is the position of the backward-rotating motor in ticks, and C_C is the ratio of the distance in ticks driven by each motor to the angle turned by the robot:

$$C_C = \frac{\Delta angle[deg]}{\Delta position[ticks]}$$

Fig. 7a indicates that this method is already consistent and thus feasible. However, since the motion is now rotational, the Wombat's built-in gyroscope can replace the motor position ticks for determining the turned angle. Using the reading of the gyroscope in ticks, $gyro$, and a new linear conversion factor, C_D, the angle a robot has currently turned can be calculated as:

$$angle[deg] = C_D \cdot gyro[ticks]$$

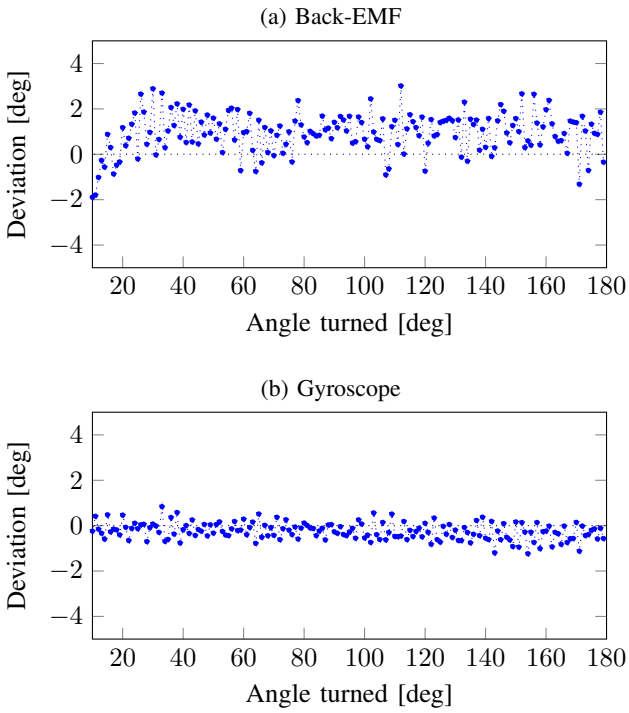

Fig. 7: Deviation from the target angle using different measurement methods

where C_D represents the ratio of the gyroscope ticks to the angle turned by the robot:

$$C_D = \frac{\Delta angle[deg]}{\Delta gyro[ticks]}$$

Fig. 7b shows that using the gyroscope to measure the angle greatly improves accuracy and consistency compared to using motor position ticks. Again, using the gyroscope has the advantage of measuring the angular velocity of the robot, which is used to approximate the physical rotation of the robot very accurately. This allows for highly consistent rotation by a desired angle regardless of the motors used.

B. Eliminating Lateral Movement

For a rotational movement to be accurate, it should also not cause any lateral movement of the robot's Z-axis. Fig. 8a shows the horizontal displacement of the robot when turning by different angles. Similar to the displacement observed in a straight motion, this is caused by the variation in motor strength. Therefore, it can be improved by balancing the motor position ticks using a PID controller. This approach effectively minimizes the displacement of the robot when turning, as illustrated in Fig. 8b.

VI. Conclusion

Accurate positioning of a robot on a Botball game table is not an easy task, but with appropriate sensors and the means to process their output, it is possible to reduce navigational inaccuracies to a level where they are no longer a concern.

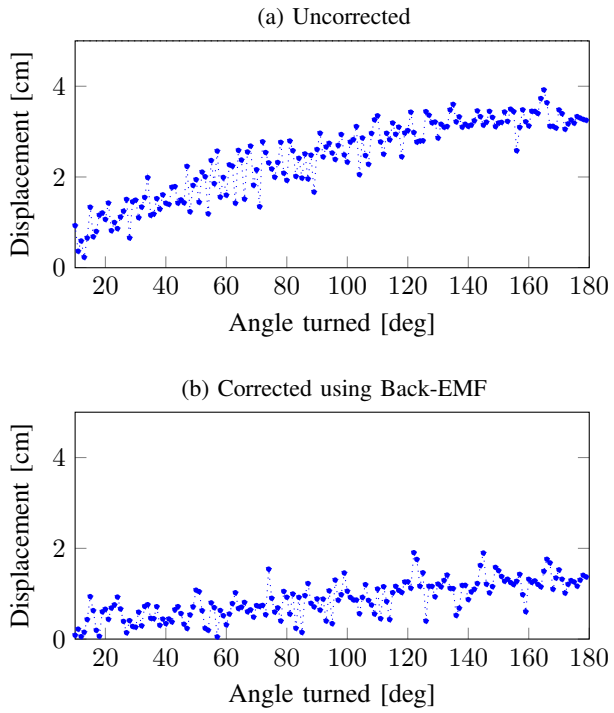

Fig. 8: Lateral displacement of the robot when turning by a certain angle

Simply powering the motors on and off based on time may cause the actual outcome to differ greatly from the intended outcome due to the random nature of the physical world. To mitigate these errors, the robot's actual position must be determined by some kind of sensor.

For measuring the distance traveled, the use of Back-EMF to determine the position of a motor has proven to be the most successful method. It can also be used to approximate the angle turned in a rotation, however, the gyroscope built into the Wombat controller offers far superior accuracy for measuring the angular position of the robot, allowing the deviation from the intended orientation to be negligible.

Overall, for the autonomous operation of a robot in an environment such as Botball, it is important to always use the most accurate measurement available to achieve the highest possible consistency.

Acknowledgements

The authors would like to thank Dr. Michael Stifter, the teachers of the HTBLuVA Wiener Neustadt, and the members of robo4you for their kind support in producing this publication. We would also like to express our special thanks to Joel Klimont for providing us with the ArUco marker detection system.

References

[1] F. Böhler, C. Fellner, V. Griesmayer, S. Groiß and S. Kawicher
"Driving in a Straight Line accurately"
(unpublished)

[2] Raspberry Pi Foundation
"Raspberry Pi"
www.raspberrypi.com
(accessed at 03.03.2023)

[3] Acroname
"All About Back-EMF Motion Control"
acroname.com/blog/all-about-back-emf-motion-control
(accessed at 03.03.2023)

[4] KISS Institute for Practical Robotics
"Wombat Curriculum – Motor Position Counter"
www.kipr.org/wombat-curriculum/motor-position-counter
(accessed 03.03.2023)

[5] TDK
"MPU-9250"
invensense.tdk.com/products/motion-tracking/9-axis/mpu-9250/
(accessed 03.03.2023)

[6] C. Atwell
"What is sensor fusion?"
www.fierceelectronics.com/sensors/what-sensor-fusion
(accessed 03.03.2023)

[7] KISS Institute for Practical Robotics
"Wombat Curriculum – Wombat Robot Build Guide"
kipr.org/curriculum-bb/wombat-robot-build-guide
(accessed at 23.01.2023)

[8] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, M. J. Marín-Jiménez
"Automatic generation and detection of highly reliable fiducial markers under occlusion"
doi.org/10.1016/j.patcog.2014.01.005
(accessed at 23.01.2023)

[9] K. H. Ang, G. Chong and Y. Li
"PID control system analysis, design, and technology"
doi.org/10.1109/TCST.2005.847331
(accessed at 23.01.2023)

[10] C. C. Hang, K. J. Åström and W. K. Ho
"Refinements of the Ziegler–Nichols tuning formula"
doi.org/10.1049/ip-d.1991.0015
(accessed at 01.03.2023)