10%/10% Motor Control
28%/30% Balance Control -- no separate block diagram for pitch angle control; body velocity step response plot does not contain other states (like pitch angle)
12%/10% Turn Control
15%/20% Odometry -- gyrodometry equations missing (required in checklist). Little to no description of how odometry model was validated.
25%/30% Path planning -- "force field" algorithm is poorly described. Supporting plots are hard to follow.
-3% required plot (odometry square) was put in appendix.
TOTAL: 87%

***Abstract*—We propose a cascaded PID controller to balance and steer a differential drive cart-pole robot. We also present planning algorithms for the robot's autonomous navigation in the presence of obstacles. The controller gains are analyzed for sensitivity and the planning algorithms are evaluated using an Optitrack motion capture setup.**

I. Introduction

IN the ”Balancebot” lab we were tasked with building a two-wheeled robot, capable of autonomous navigation through gates [1]. The robot built here resembles a differential drive cart-pole system in its construction as well as its inherent instability [2]. To facilitate control of the robot we mounted a Seeed Beaglebone Green single-board computer with which to perform onboard calculations. We also attach a custom expansion board ”ROB550 Mobile Robotics Cape” to the Beaglebone furnished with a 9DOF IMU to measure heading angle and body angle of the robot. The Cape also contains software used calibrate the IMU as well as interface to the motors. We used a 11.1 V LiPo battery to power the motors and the Beaglebone. We provide a detailed BOM for the robot in Appendix D.

We propose a cascaded proportional-integral-derivative (PID) [3] control scheme to control the robot, wherein the output of each outer PID layer is provided as an input to the next inner PID layer. Fig. 2 depicts the architecture of the 4-layer cascaded PID controller used. The innermost PID layer controls the motor speed, the next PID layer enables the robot to balance itself by controlling the body angle, the third layer controls the robot velocity, while the outermost layer controls the distance error from the desired position. A separate PID controller is implemented to turn the robot to desired headings. In Section II-D we detail the odometry used to provide feedback to the distance and turning PIDs, and in Section II-E we detail planners used to provide the position and heading inputs to the cascaded PID.

II. Methodology

The orientation in which the robot can balance itself corresponds to its unstable equilibrium. By trial and error, we found that the unstable equilibrium for the robot corresponds to body angle, $\alpha = 1.5^o$ (Fig. 1). For all of our experiments, we use a sampling rate of 100Hz for our control loop, which corresponds to a timestep of 0.01s.

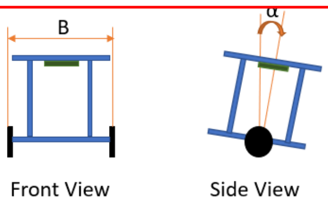


Fig. 1: A crude drawing of the robot.

In Section II we detail the filtering of sensor data, the control scheme and the path planning algorithms used. We follow with results in Section III and conclude with discussion in Section IV.

A. Data Filtering

We filter wheel encoder counts to remove outliers using both a median filter and a threshold. The median filter stores the most recent k counts including the current timestep and outputs the median of the stored values. Before inserting each value into the median filter, we compare its magnitude with the threshold. Values above that are replaced with the last value inserted into the median filter. For our implementation, by trial and error, we determined that $k = 5$ produces best results.

B. PID Tuning

Here we outline our procedure for tuning each PID controller. We start with all gains at zero and first aim to find a K_p value, say K_p^*, such that responses for higher K_p values produce oscillations and underdamped behavior, while lower K_p values produce overdamped behavior. The response for such a K_p for the body angle PID is shown in Fig. 3(a). Note that the response produced by this K_p could be critically damped, but disturbances to the system may make the response appear otherwise. We next aim to find a K_p and a K_i term jointly to eliminate steady state error, if any exists. To do this we pick a K_p in the range of 20% to 70% of K_p^*, and attempt to adjust the K_i until the PI controller's response converges with no steady state error. This step is repeated until the K_p and K_i have been found. An example of K_p and K_i found here are shown in Fig. 3(b). Lastly we find a K_d term to remove possible oscillations in the response. Keeping the found K_p and K_i fixed, we search among values of K_d smaller than the K_p term,

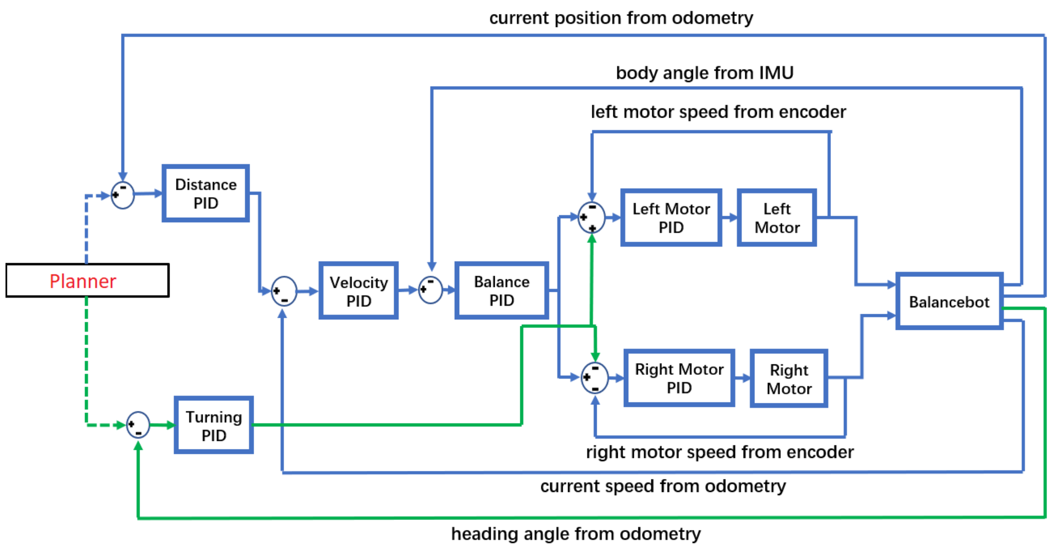

current position from odometry
body angle from IMU
left motor speed from encoder
Distance PID
Planner
Velocity PID
Balance PID
Left Motor PID
Left Motor
Balancebot
Right Motor PID
Right Motor
Turning PID
right motor speed from encoder
current speed from odometry
heading angle from odometry

Fig. 2: Design structure of the cascade PID controller

stopping when the oscillations have been decreased an arbitrary amount. Fig. 3(c) shows a full PID for body angle control found with this method.

C. Controllers

1) Motor Control: Motor speed is controlled by adjusting the duty cycle specified for each motor, with the duty cycle ranging from -1 to 1. The motor PID for each motor then controls motor speed by taking as input the difference between desired motor speed and current motor speed calculated for the attached wheel. Wheel speed is calculated from median filtered motor encoder measurements by multiplying it with a conversion factor.

2) Balance Control: The robot's body angle is controlled by setting the same desired speed for each motor controller. The idea is to counter the unbalancing torque caused due to gravitational pull on the robot by the frictional force accelerating it. The net angular acceleration generated reduces the error input to the body angle PID.

3) Velocity Control: A PID is used to control velocity by setting the robot's desired body angle. We add the output of the velocity PID to the equilibrium body angle so that there is no offset between the attained velocity and the desired velocity.

4) Turning Control: Turning control is implemented making use of the robot's differential drive. Using the separately controlled motors, turning in place can be achieved by setting opposite duty cycles for each motor to effect equal torque in opposite directions about the robot's center. We control the robot heading using another PID controller.

This turning PID takes as input the difference between desired and current heading calculated via odometry

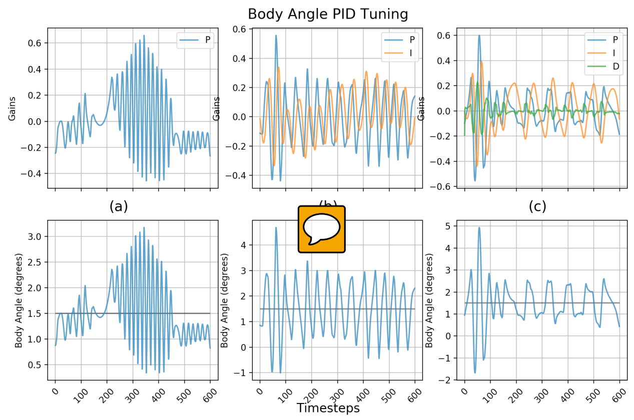


Fig. 3: Steps for manual tuning of PID gains. Finding a K_p^* value *(a)*, jointly finding the final K_p and K_i values *(b)*, and finding the K_d value *(c)*. See text for details.

(Sec. II-D) and outputs the desired motor speed for the left motor. The desired right motor speed is set to be the negative of the left. These desired motor speeds are added to the speeds set by other controllers (Fig. 2).

5) Distance Control: We implement a distance PID to position the robot at a given point in the world using odometry measurements. This controller controls the robot's velocity and not heading, so the distance to the position is minimized only along the line specified by the robot's heading vector. As such, we account for positions that do not lie along that line by projecting the point to the line and minimizing the resulting distance.

D. Odometry

Odometry refers to the use of data from motion sensors to estimate change in position over time. In this project, we used the rotary encoders attached to the motors to implement dead-reckoning, a form of odometry in which the robots current position is determined solely from the previously recorded position.

Equation 1 describes the discrete time vehicle model utilized to implement dead-reckoning when the robot is moving in a straight trajectory, while equation 2 is used when the robot is turning as well [4]. Here, $[X_k, Y_k, \Theta_k]^T$ refers to the pose of the robot relative to its initial pose after k^{th} timestep, L_k and R_k are the distances traveled by the left wheel and the right wheel respectively between timesteps $k-1$ and k, rk is the instantaneous radius of rotation computed using Eq. 3, and B refers to the track width of the robot (Fig. 1). If Lk and Rk were equal, we inferred that the robot was moving in a straight line in the previous time interval, else we used equations pertaining to the turning case.

$$\begin{bmatrix}\Theta_k\\X_k\\Y_k\end{bmatrix} = \begin{bmatrix}\Theta_{k-1}\\X_{k-1}\\Y_{k-1}\end{bmatrix} + \begin{bmatrix}0\\Lk\cos(\Theta_{k-1})\\Lk\sin(\Theta_{k-1})\end{bmatrix} \tag{1}$$

$$\begin{bmatrix}\Theta_k\\X_k\\Y_k\end{bmatrix} = \begin{bmatrix}\Theta_{k-1}\\X_{k-1}\\Y_{k-1}\end{bmatrix} + \begin{bmatrix}(Rk-Lk)/B\\rk[sin(\Theta_{k-1})-sin(\Theta_k)]\\rk[cos(\Theta_{k-1})-cos(\Theta_k)]\end{bmatrix} \tag{2}$$

$$rk = \frac{B}{2}\Big(\frac{Lk+Rk}{Lk-Rk}\Big) \tag{3}$$

Because dead reckoning is susceptible to systematic errors caused by kinematic imperfections of the robot, we implemented gyrodometry [5] to improve the odometry. Gyrodometry is a 'model-free' algorithm for fusing wheel encoder and IMU measurements. In our implementation, we compare the robot heading angle computed using dead-reckoning with the heading angle determined from IMU 'yaw' values after each timestep. If this deviation is larger than a pre-determined threshold, we replace the dead-reckoning heading angle with the IMU heading angle.

However, even the heading angle data from the IMU is not always accurate and accumulates a drift error over time, especially in presence of temperature variations [6]. To ensure gyro drift is not substantial, before every task, we calibrated the IMU using the Cape software library program `rc_calibrate_gyro.c` with the robot at rest. Also, we chose the deviation threshold such that the heading angle correction is done only when the dead-reckoning heading angle is less accurate than the IMU heading angle.

We set the threshold for the gyrodometry algorithm by measuring the IMU's average drift over 40s in $n = 6$ trials, and found it to be $1.3°$+/-$0.8°$. We chose a threshold of $5°$, such that it is outside the 99% confidence interval.

E. Local Planners

To enable autonomous navigation through arbitrarily placed gates (Event 4 of competition), we describe two planners implemented: a rotate-translate-rotate (RTR) planner, and a potential fields planner.

1) RTR Planner: Given a set of waypoints in the workspace, the RTR planner follows the waypoints in sequence by first rotating to face the current waypoint, translating towards it, and finally rotating to face the next waypoint. This algorithm relies on the 'careful' placement of waypoints to ensure that it does not collide with obstacles, as there is no checking along the segments connecting waypoints for collisions.

2) Position Control: This planner is different from RTR in that it controls the robot's error in distance and heading simultaneously. This amounts to providing the error in heading to the turning PID and projected distance error to the distance PID at the same time. If the robot is facing opposite the destination position however, the heading error will be large and the possible heading adjustment to correct it may conflict with the controls to reduce the distance error. This motivates us to use the symmetry in the robot to redefine the robot's current heading for the timestep to be its reversed heading. Then the robot may move 'forward' while reducing the smaller heading error.

3) Potential Fields Planner: In the potential fields planner, artificial potential fields act upon the robot to attract it to the final configuration while repelling it from obstacles (ref Spong). The final configuration is thus made the global minimum of the potential field, with an attractive field whose origin is that configuration. Here we model both obstacles and the robot as points in the workspace, so the final configuration is a point in $\mathbb{R}^2$ corresponding to location in the workspace. Obstacles are modeled as repulsive fields such that the artificial force exerted is infinite at the center of the obstacle and decreases with distance from the obstacle.

We model the attractive field as a combination of parabolic and conic well potentials. Letting d denote the Euclidean distance between the point x and the origin of the potential, the attractive field is given by:

$$U_A(x) = \begin{cases} -\frac{1}{2}\eta d^2, & \text{if } d \le \rho \\ -\eta\rho d + \frac{1}{2}\eta d^2, & \text{otherwise} \end{cases}$$

The repulsive field for each obstacle is modeled as:

$$U_R(x) = \begin{cases} \frac{1}{2}\eta(\frac{1}{d} - \frac{1}{\rho}), & \text{if } d \le \rho \\ 0, & \text{otherwise} \end{cases}$$

The current force acting on the robot is determined by the sum of the forces from the nearest repulsive field and attractive forces, which are found by taking the derivative of the above formulas with respect to x. At every timestep, the robot's movement is then determined by finding the gradient of the potential fields and moving a distance according to a specified step size (possibly using a local planner such as RTR), with heading given by the direction of the gradient. The potential fields algorithm for reaching a single destination is summarized in Algorithm 1.

Algorithm 1 Potential Fields Planner

```
Require: attractive potentials $a_1, \ldots, a_n$, repulsive potentials $r_1, \ldots, r_m$, initial position $x_0$, step size $\alpha$, destination $w$, tolerance $t$
  $x = x_0$
  Find closest repulsive potential $r^*$
  while $\|x - w\| > t$ do
      Calculate $F = \frac{dr^*}{dx}|_x + \sum_{i=1}^{n} \frac{da_i}{dx}|_x$
      $x = x + \alpha\frac{F}{\|F\|}$
```

F. Path Planning

Each of the local path planners in the previous section path from an initial position to a provided waypoint. Here we detail methods for autonomous navigation through arbitrarily placed gates. We describe a method for placing waypoints, and a method that builds upon that for making a potential fields path planner.

1) Gate Waypoint Placing: One way of passing through gates is to first consider the navigation without obstacles. To pass through a given gate, waypoints placed on both sides of the gate's midpoint can be used to navigate to the gate's entrance on one side, then to the waypoint through the gate on the opposite side. In this fashion, each gate can be traversed in sequence, using some planner to navigate between each waypoint.

2) Potential Fields Planner: If the positional control or RTR local planners are used to follow the waypoints placed according to the previous section, they may collide with obstacles, as they do not model them. We apply the potential fields method for planning using the previous section's waypoint placing and two types of fields to aid in passing through the gates. A global field is used when pathing to the entry waypoint of any gate, with each gate represented as a single repulsive field. After reaching the entry waypoint, a local field is used to path through the gate. Here the poles of the gate are each represented as a repulsive field, and an attractive field is placed at the gate's exit waypoint. Upon reaching the exit waypoint, the robot returns to the global field and repeats this process until all waypoints have been reached. The relevant code is found in file `planning_structs.c`.

PIDs	K_p	K_i	K_d
Left Motor Speed	0.5	1.0	0.0
Right Motor Speed	0.5	1.0	0.0
Balance	10.0	120.0	0.2
Velocity	0.05	0.01	0.0
Turning	1.8	0.0	0.0
Distance	0.7	0.0	0.0

TABLE I: Parameters for the proportional, integral, and derivative terms of each PID controllers

III. Results

Each PID was tuned using the procedure described in Section II-B, and the resulting gains are shown in Table I. We evaluated the inner PIDs, namely, the PIDs for the left motor, body angle, velocity, and turning, using a step input. For each of those PIDs aside from the motor PID we also perform an analysis of system sensitivity to the chosen gains. To avoid biases introduced by changes in how to robot was initially released, for each test, the robot was set to balance autonomously for 2 seconds before applying the step input. We used an arbitrarily chosen threshold of 60 to filter wheel encoder counts. This threshold value was determined by maximum wheel encoder count in a timestep with no load placed on the motor.

A. Sensitivity Analysis

We subject the system to step inputs under various gain settings in order to determine the system's sensitivity to the choice of gains. We evaluate the system response generated by each set of gains by first determining whether or not the system has diverged, reached a steady state error, or otherwise reached a steady state possibly with oscillations. In the last case the performance is characterized by the first time the response comes within a specified distance of the step value, and the difference between the highest and lowest peaks thereafter.

B. Controllers

1) Motor Control: We found that the gains for the left motor were also suitable for the right motor, so each motor PID uses the same gains. For a step input of 0.8m/s, Fig 4 shows the responses for the chosen gains, gains producing an underdamped response, and those producing an overdamped response.

2) Balance Control: The process used find the body angle PID's gains and the resulting responses are shown in Fig 3. In evaluating the body angle PID with step inputs of 2°, in all cases the robot would eventually fall while attempting to maintain the given body angle. For these divergent responses, we evaluate sensitivity for the chosen gains by finding the first time response

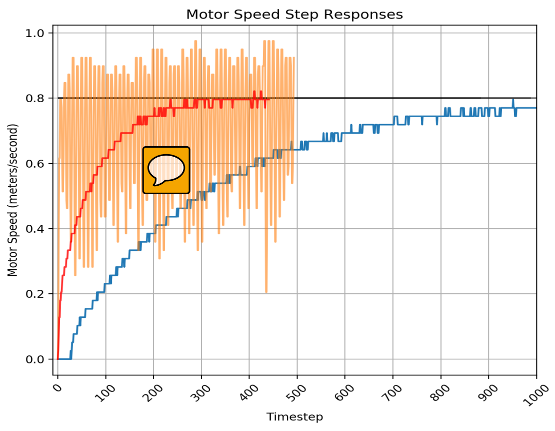


Fig. 4: Response of the chosen motor PID gains *(red)*, gains making an underdamped response *(orange)*, and those making an overdamped response *(blue)*, to a step input of 0.8m/s.

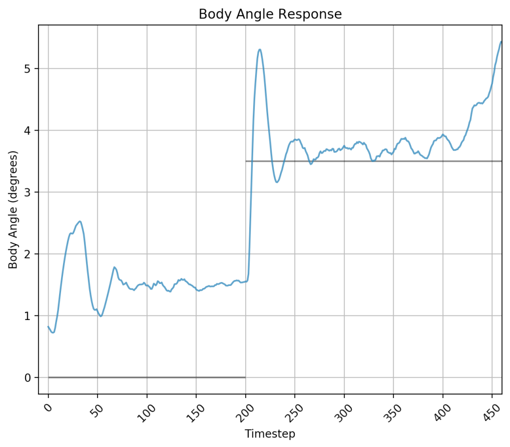


Fig. 5: Body angle response when a step input of 2° is applied after 2 s, with chosen gains (Table I)

'converges' to the step input value, and then determining the maximum amplitude (peak to peak) between that time and the time the response diverges. The body angle PID was evaluated by supplying a step input with the equilibium angle initial value and a step value of 2°, with response shown in Fig 5 and sensitivity scores in Table II.

3) Velocity Control: The velocity PID was evaluated by supplying a step input of 0.5m/s after 2 seconds at 0m/s. The controller's response is shown in Fig 6 and the sensitivity scores in Table III.

4) Turning Control: The step response generated by the chosen turning PID gains for a 180° turn is shown in Fig. 7. Sensitivity scores were collected for the turning PID with the proportional gain at ±{10%, 40%} of the chosen gains, shown in Table IV.

K_p	Min t	Amplitude
18 (+80%)	18	3.90
14 (+40%)	38	0.40
11 (+10%)	43	0.36
10 (0%)	40	0.43
9 (−10%)	78	0.52
6 (−40%)	86	2.64
2 (−80%)	DIV	-

TABLE II: Sensitivity scores for the body angle PID at different values of K_p, keeping K_I & K_D constant. The response remains comparable for K_p within ±10% of the chosen value, but degrades with further change.

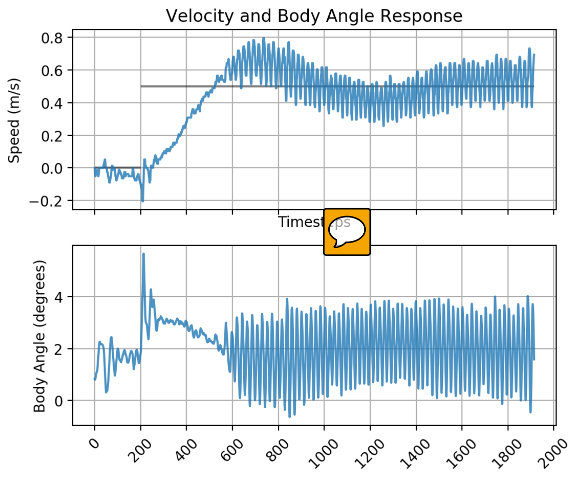


Fig. 6: Velocity response when a step input of 0.5 m/s is applied after 2 s, with chosen gains (Table I) *(top)*. Body angle response for the same input *(bottom)*.

K_p	Min t	Amplitude
0.130 (+160%)	DIV	-
0.110 (+120%)	58	0.83
0.090 (+80%)	83	0.36
0.070 (+40%)	212	0.30
0.055 (+10%)	251	0.14
0.050 (0%)	203	0.20
0.045 (−10%)	198	0.11
0.030 (−40%)	293	0.09
0.020 (−80%)	DIV	-

TABLE III: Sensitivity scores for the velocity PID at different values of K_p, keeping K_I & K_D constant. The response remains comparable for K_p within ±10% of the chosen value, but degrades with further change.

K_p	Min t	Amplitude
2.52 (+40%)	180	12.48
1.98 (+10%)	162	11.58
1.80 (0%)	169	14.77
1.62 (−10%)	163	14.08
1.08 (−40%)	168	19.78

TABLE IV: Sensitivity scores for the turning PID at different values of K_p, keeping K_I & K_D constant. The response remains similar for the range of K_p evaluated.

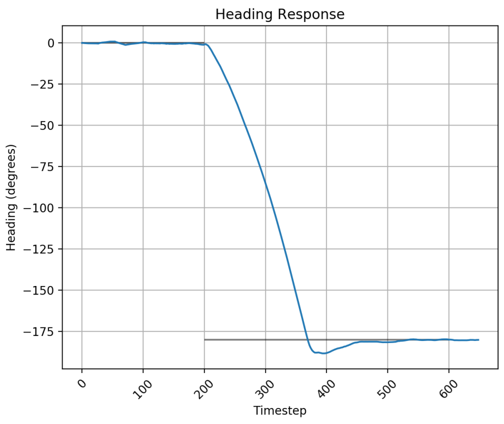


Fig. 7: Heading angle response when a step input of $-180°$ is applied after 2 s, with chosen gains (Table I)

C. Odometry

We evaluated the odometry algorithms by controlling the robot to a series of waypoints along a $1m \times 1m$ square using the RTR planner, recording the robot pose estimated with odometry along with ground truth acquired using a motion capture setup [7], which provides positional accuracy to 1mm and the robot's heading to the degree. The path followed using dead reckoning is shown in Figure 8a and the the path using gyrodometry is shown in Figure 8b. The robot was controlled to the same position before beginning each lap, so pose error was calculated as the difference between the previous and current lap's motion capture pose at the lap end position. The initial position of the robot is the same as its lap finish position unlike heading, so $n = 4$ differences were used to calculate positional error and $n = 3$ were used to calculate heading error. The pose error accumulated each lap is shown in Table V.

Odometry	Distance (meters)	Heading (degrees)
Dead Reckoning	0.20+/-0.06	0.1+/-0.1
Gyrodometry	0.02+/-0.02	0.0+/-0.0

TABLE V: Mean and standard deviation for error in robot pose estimates accumulated each lap of the square. The calculated heading error values for gyrodometry were smaller than that of measurement resolution.

D. Path Planning

For navigating the course in Event 4 we used an offset of 0.2m for placing gate waypoints and a distance threshold of 0.1m was used to consider a waypoint reached. Attractive potentials in both the global and local potential fields were represented with $\rho = .5, \eta = 1$. Repulsive potentials in the global field were represented with $\rho = .2, \eta = .1$, choosing sufficiently large ρ to cover each gate's poles, and those in local fields with $\rho = .1, \eta = .1$ with ρ chosen to avoid overlap between the poles' potentials. RTR was used as the local planner, moving with step size of 0.05m. During the event's trial runs the robot froze at the exit to the first gate, before reaching the waypoint on the opposite side at the entrance. To investigate this behavior further the same gate placement was used in simulation, and the same behavior was replicated. Hypothesizing that the trapped behavior was due to a saddle point, we again simulated the scenario with adding white noise having variance a tenth of the step size to the robot's calculated gradient. The simulation results displayed in Appendix C show that with added noise the simulated robot was able to navigate to the opposite side of the gate.

IV. Discussion

From the sensitivity scores recorded in Tables II-IV, we can conclude that the performance of the body angle, velocity and turning PIDs remains similar to the one with chosen gains (Table I) if K_p is varied within a band of $\pm 10\%$. Outside of this band, we observe larger changes in the body angle PID and the velocity PID responses, with the response eventually diverging as we keep on moving K_p further from the central value. However, the heading angle PID still performs comparably, suggesting that it is less sensitive to changes in K_p than the other two PIDs. We also observed that at all step responses not equal to the equilibrium body angle, the body angle response eventually diverges causing the robot to fall. This could be because any non-equilibrium body angle can only be maintained if the robot undergoes constant acceleration, which is not possible as the maximum motor RPM is limited. Such a phenomenon was not observed for velocity and turning PIDs.

From Table V and Fig. 8a & 8b, it is evident that gyrodometry accumulates significantly less error per lap.

The failure of the potential fields based path planning at the gates was possibly due to a saddle point, as random movements as suggested in [8] enabled the simulated robot to continue its course. A surer way of knowing, however, would be to record with fine discretization the portion of the potential field surrounding the robot at the frozen position.

References

[1] "Rob 550 balancebot lab manual," accessed: 2017-11-13.

[2] I. Fantoni and R. Lozano, *The cart-pole system*. London: Springer London, 2002, pp. 21–42. [Online]. Available: https://doi.org/10.1007/978-1-4471-0177-2_3

[3] K. Ogata, *Modern Control Engineering*, 4th ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

[4] K. Seng Chong and L. Kleeman, *Accurate odometry and error modelling for a mobile robot*, 05 1997.

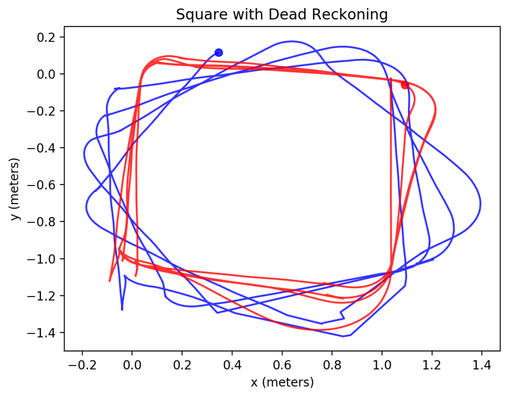
[5] J. Borenstein and L. Feng, “Gyrodometry: A new method for combining data from gyros and odometry in mobile robots,” in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 1. IEEE, 1996, pp. 423–428.

[6] “Mpu-9250,” https://cdn.sparkfun.com/assets/learn_tutorials/5/5/0/MPU9250REV1.0.pdf, accessed: 2017-11-18.

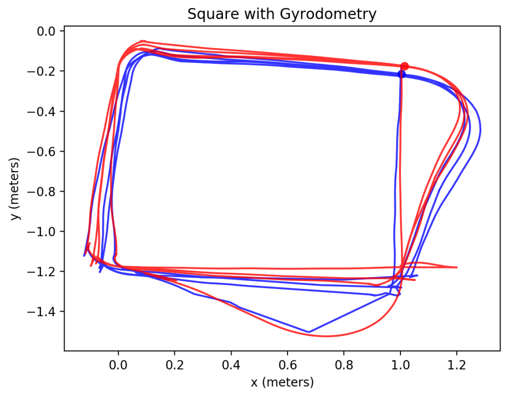
[7] “Optitrack prime13 sensor,” http://optitrack.com/products/prime-13/, accessed: 2017-11-18.

[8] M. W. Spong, *Robot Dynamics and Control Second Edition*, 2004.

APPENDIX A
ODOMETRY VALIDATION



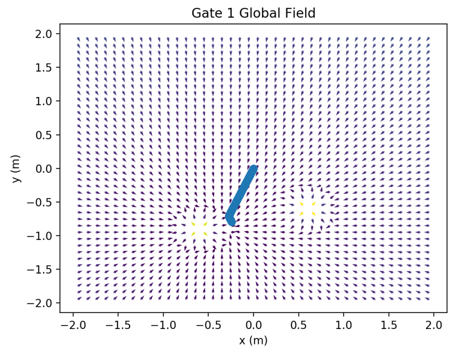
(a)



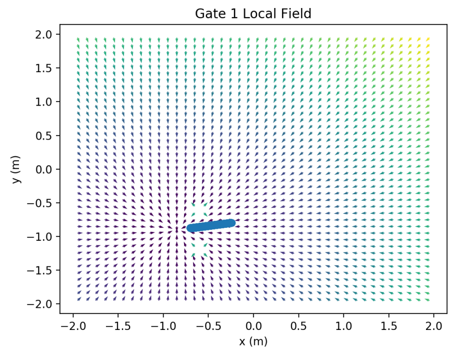
(b)

Fig. 8: Navigating a 1×1 square, with estimated robot positions *(red)* and ground truth provided by the (Prime13) Optitrack motion capture system *(blue)*. Odometry calculated using *(a)* dead-reckoning and *(b)* gyrodometry.

Appendix B
Potential Fields Planner Validation



(a)



(b)

Fig. 9: Pathing using the potential fields algorithm with robot track shown in blue. *(a)* Global potential field used for approaching Gate 1 and *(b)* the local potential field for passing through the gate.

Appendix C
Random Movement Potential Fields

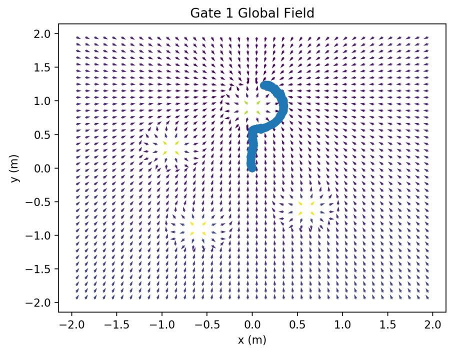


Fig. 10: With randomness added to its movements the robot is able to path to the opposite side of the gate.

Appendix D
Bill Of Materials

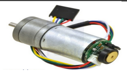
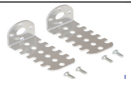
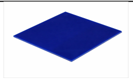
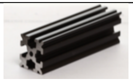
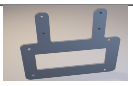
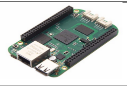
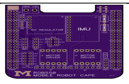
Part Name	Thumbnail	Quantity	Link
Toy Wheels		2	-
12 V DC Motors		2	https://www.pololu.com/product/3215
Motor Brackets		2	https://www.pololu.com/product/1138
0.188" thk. Acrylic Sheet		1	-
OpenBeam 1515 Black Anodized		4x150 mm 2x120 mm 2x180 mm	http://www.openbeamusa.com/purchase/amazon-listings/
Plastic L Bracket		8	http://www.openbeamusa.com/purchase/amazon-listings/
3D Printed Battery Mount		1	-
11.1V LiPo Battery Pack		1	http://www.floureon.com/product-g-267.html
Seeed BeagleBone		1	https://www.seeedstudio.com/SeeedStudio-BeagleBone-Green-p-2504.html
ROB550 Mobile Robotics Cape		1	-
Motion Capture Markers		3	http://optitrack.com/products/motion-capture-markers/
DX6e DSM Receiver		1	-
Nylon Standoffs	-	4	-
Aluminum Standoffs	-	4	-
Electrical Tape	-	1 roll	-
Wires	-	-	-

TABLE VI: Bill Of Materials