Autonomous Hovercraft Using Potential Based Methods

Mitchell Kain[1] and George Knight[2]

***Abstract*— The project consists of designing and building a functional autonomous hovercraft prototype, Bert, based loosely on the work of Davis et. al. The high level goal is to create the hardware and software for a hovercraft to navigate a hallway without colliding with obstacles. To achieve this goal the robot will measure its surroundings using a 360 degree LiDAR sensor and a potential based planner to to calculate a target acceleration and the corresponding control-state. The robot will move to the center of the hallway and proceed to traverse the hallway down the center line, maintaining that line until it reaches a set distance from the end of the hall.**

Future goals for the robot are to, to navigate corners, turn optimization, and navigating to a point. Future work could include loop closure and mapping, implementing a PID, and added machine learning capabilities.

The motivation for this project is largely the desire to see algorithms we have been working with used in an actual robot and gain first hand experience designing and constructing such a robot. A hovercraft was chosen because of the resources made available by [1] and the challenge of hovercraft motion planning. Success will be measured by the robot's ability to navigate down a hallway without colliding with the walls.

I. Prior Work

A. Albert

The design for this hovercraft is loosely based on the work done by Davis et. al. [1]. Their work included a component list and a detailed instruction for building a hovercraft including wiring diagrams, skirt construction, prop mounts, and servo controlled directional vanes.

While Albert 2.0 informed many of the design choices in Bert, Bert's hardware design differs in several important ways. Bert's processor has been upgraded from an Arduino Uno to a Raspberry pi for increased processing power and functionality. Bert also uses a LiDAR scanner instead of ultrasonic sensors. This allows Bert to better localize in the hallway. The most significant difference is that Bert has two drive vane controlled drive fans facing opposite directions. The second fan theoretically allows Bert independent control of translation and rotation.

B. Hardware Fabrication

A large portion of the work on this project was devoted to Bert's hardware. Some academic papers have been used, but the vast majority of resources used by the authors come from robotics tutorials and blogs. These covered a vast range of hardware topics, many of them introductions to the challenges of robotics hardware, including: ROS, motor controls [2], raspberry Pi configuration, servo controls, accelerometers [3] and LiDAR.

This work was not supported by any organization

[1]Mitchell Kain is with the Department of Mechanical Engineering, Boston University, 110 Cummington Mall, MA 02215, United States mkain@bu.edu

[2]George Knight is with the Department of Mechanical Engineering, Boston University, 110 Cummington Mall, MA 02215, United States gknight@bu.edu

II. Operation

Bert will operate on an Observe→Orient→Decide→Act loop (OODA) as described in [4].

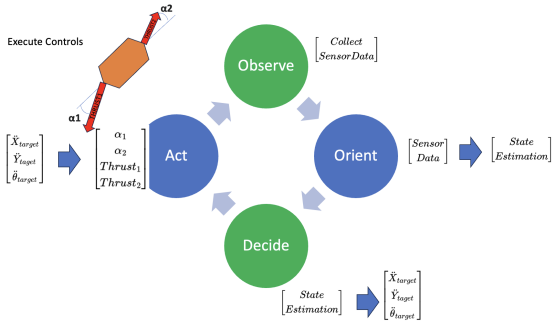

Fig. 1: OODA Loop

This loop, updates controls as the Bert moves and is the authors answer to having limited experience in designing a control scheme. The OODA Loop is a decision making method taught to pilots and seemed like a good place to start as it has been used in robotics before [4] Initial ideas of using a Proportional Integral Derivative (PID) based planning system as described in [5], were found to be excessive for the task and a simple potential planner was more computationally efficient. In simulation this proved to be an effective tool, running at roughly 40Hz on an Apple Macbook Pro. As of writing the paper this has yet to be tested on a hardware Bert, but the method should prove adaptive to different levels of computing power.

III. Orient

In order for Bert to navigate the hallway he must first estimate his position in the workspace. Given X as the direction perpendicular to the wall with X = 0 being the center, Y as the direction parallel to the wall, and θ as the angle of Bert measured from the center line of the hall, parallel to the wall. The state variables are $X, \dot{X}, Y, \dot{Y}, \theta, \dot{\theta}$. The state variables can be measured with a combination of the LiDAR data and the onboard accelerometer. Where sensing overlaps exist averages can be taken to reduce noise.

A. LiDAR measurements

To measure θ and X Bert simply finds the minimum sum of opposing angles from the LiDAR. If the minimum lengths are L_{right} and L_{left} the width of the hall $W = L_{left} + L_{right}$ and $X = L_{left} - L_{right}$ where X is measured from the center of the hallway. $\theta = 90° - \alpha$ where α is whatever angle the minimum length is measured at. Due to the self similar nature of hallways the LiDAR will not be able to measure Y or $\dot{Y}$ if both ends of the hallways are beyond the range of the sensor (six meters). Therefore the LiDAR will not calculate Y or $\dot{Y}$ and Bert will instead rely on the onboard accelerometer for those values. The values for velocity can be calculated as the change in the measured position and angle divided by the time increment (.11 seconds).

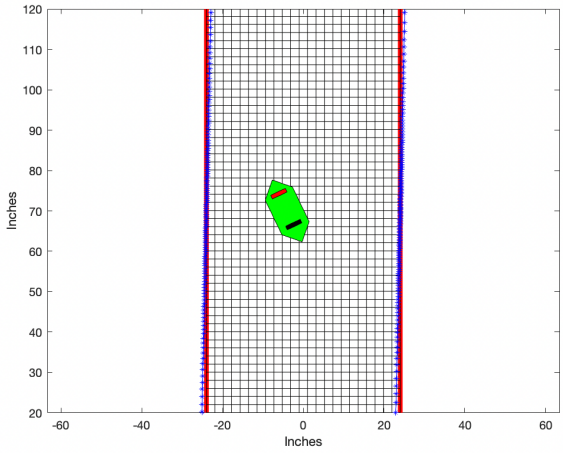

Fig. 2: Lidar Scan Walls are red, LiDAR scans are blue

The walls of the hallway are in red and the lidar scan points are in blue.

B. Accelerometers

Part of Bert's state estimation is done using the onboard accelerometer placed on his center of mass. By integrating the local $\dot{X}, \ddot{X}, \dot{Y}, \ddot{Y}, \dot{\theta}$, and $\ddot{\theta}$ we can calculate a position and velocity for all state variables. While calculations should provide accurate information, they are prone to drift and Bert's erratic movement may lead to accumulated error in the position estimations from this source. Because of this Bert will only rely on the accelerometer to measure Y and $\dot{Y}$. Even though Y is not used in the control planner, $\dot{Y}$ is used to place an upper limit on Bert's speed.

C. Initial Orientation

When placed in a hallway Bert will decide to travel in the direction for which he has the least information in order to facilitate future SLAM applications. Therefore, he will head in the direction of obstacles that are beyond his LiDAR range. If he lacks information in more than one direction he will move in the direction most in line with his initial placement, requiring the smallest angle correction. This should work whether or not the initial placement is at the beginning of the hallway or in the middle where both ends of the hallway are beyond the range of the LiDAR.

The authors assumed that Bert would have to create a coordinate scheme of the hallway before beginning to travel, but as the orientation returns all of the variables required for path planning a separate step to initialize Bert was deemed unnecessary.

Adapting the orientation step for curving environment or corners would be an interesting future development. In a curving hallway, Bert's θ and X measured from the LiDAR would still return the correct angle for Bert as the shortest line segment connecting the walls would still define the X location and this line could still be used as a reference for θ. This would require a more advanced integration of the accelerometers in LiDAR to measure $\dot{Y}$ as Y turns with the hallway. Testing this is not currently in scope, but would make Bert suitable for commercial uses in complicated environments such as navigating a maze or a home.

D. Goal Achievement

Upon successfully navigating the hallway, Bert will eventually reach the end of the hallway and stop by arresting the lift fan. This condition is considered met if the distance measured by the LiDAR within $10°$ of the nose is measured to be below four feet. Based on the results in VI-C a range of $10°$ is well within Bert's simulated accuracy which recorded a maximum θ of $< 1°$ after the first fifteen seconds with simulated noise.

IV. Decide

A. Target Acceleration

Bert uses a conic-quadratic hybrid potential based function to derive a target acceleration. While Bert cannot control acceleration directly, generating the desired accelerations and then later converting them into controls simplifies the problem and reduces the computational load. The "goal" is to reach the state space $X = \dot{X} = \theta = \dot{\theta} = 0$ and $\dot{Y} = 15 inches/second$ The potential function will return a target $\ddot{X}, \ddot{Y}, \ddot{\theta}$. By comparing these to the state variables derived in III Bert determines the controls. For example using X to find a target acceleration $\ddot{X}$:

$$\ddot{X}_{target} = \phi(-\dot{X} + X^2 * \frac{-|X|}{X}) \tag{1}$$

$$\ddot{Y}_{target} = \phi(\dot{Y} - \dot{Y}_{target})^2 * \frac{-|\dot{Y} - \dot{Y}_{target}|}{\dot{Y} - \dot{Y}_{target}} \tag{2}$$

$$\ddot{\theta}_{target} = \phi(\dot{\theta} - (\theta^2 * \frac{-|\theta|}{\theta})) \tag{3}$$

where ϕ is a positive tuning parameter unique for each target. This is also subject to a maximum acceleration for each target.

B. Collision Avoidance

The movement of a hovercraft like Bert is extremely complex. In an attempt to prevent collisions that may damage Bert, the system will also check the state variables to see if a collision is likely. If any of the thresholds are exceeded Bert will simply stop the lift fan to quickly arrest motion, start the fan again, and use the initialization algorithm to re-center and start traveling again. Thresholds include:

- Velocity
- X
- $\dot{X}$
- θ
- $\dot{\theta}$

If a calculated value for any of these variables exceeds the threshold on either the LiDAR reading or the accelerometers the lift fan will stop.

V. Act

Running the high level planning algorithm with $\ddot{X}, \ddot{Y}$, and $\ddot{\theta}$ saves compute, but in order for Bert to use them, the accelerations must be transformed into controls (vane direction and thrust).

A. Bert's Dynamics

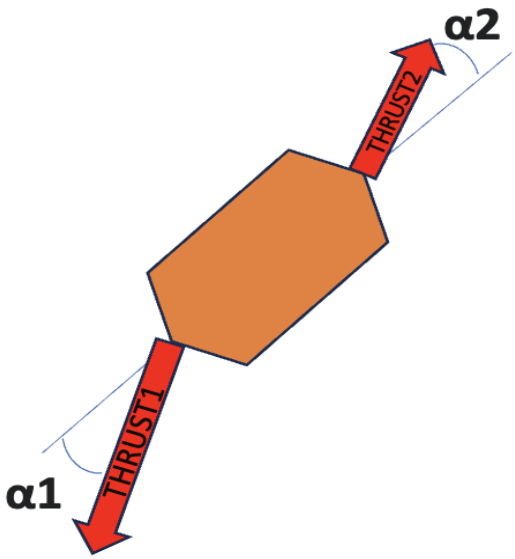

Fig. 3: Free Body Diagram

Using a free body diagram we can find the values for $\ddot{X}, \ddot{Y}, \ddot{\theta}$ given variables: T_1 (thrust of rear fan), T_2 (thrust of front fan), α_1 (angle of rear fan), α_2 (angle of front fan), θ_o, and R (distance from fans to center of mass). These controls are in the Bert's local reference frame

$$\begin{aligned} \ddot{Y} &= \frac{T_1 * cos(\alpha_1) - T_2 * cos(\alpha_2)}{mass} \\ \ddot{X} &= \frac{T_1 * sin(\alpha_1) - T_2 sin(\alpha_2)}{mass} \\ \ddot{\theta} &= \frac{(-T_1 sin(\alpha_1) + T_2 sin(\alpha_2))R}{Im} \end{aligned} \tag{4}$$

To calculate the controls that best correspond to the target accelerations including the adaptive weights, these must be converted to the global reference frame before passing to the minimization function.

B. Converting Acceleration to Controls

From here the controls are a non-linear optimization problem. Bert finds a control scheme that minimizes error:

$$Error = |\ddot{X}_t - \ddot{X}_c, \ddot{Y}_t - \ddot{Y}_c, \ddot{\theta}_t - \ddot{\theta}_c| \tag{5}$$

Given target accelerations $\ddot{X}_t, \ddot{Y}_t$, and $\ddot{\theta}_t$ and control accelerations $\ddot{X}_c, \ddot{Y}_c$, and $\ddot{\theta}_c$, which are the accelerations that result from a given control 4 after conversion to the global reference frame.

Finally, Bert passes the previous controls as an approximation for the local minima. This reduces compute and discourages drastic control changes over iterations.

VI. Simulation

A simulation of Bert was conducted in MATLAB soft wear as a proof of concept for this control scheme. This facilitated troubleshooting and the fine tuning of variables without the danger of breaking any of the physical components. As time passed it became likely that Bert could not be delivered on the initial schedule. The simulation is meant to show the effectiveness of the above control scheme.

A. Sensor in Simulation

The method in III-A was able to perform extremely well. Running on a Macbook Pro Bert was consistantly able to estimate his state in 7 milliseconds to within .01 inches and .02 radians. This error seems was mostly the result of a rounding error in the psuedo-scan generator. The accelerometer data was not included in the simulations.

B. Insights from Simulation Testing

1) Tuning Error Evaluation: Because of the orientation of the thrust control vanes he can actually accelerate faster in θ and Y if he is sideways to the hallway, rather than pointing his "nose" in the direction he is heading. Additionally the error for θ was measured in radians while the position errors were measured in inches. The inches tended to be much higher so the control scheme de-prioritized Bert's orientation. To prevent Bert from spinning the authors initially added a much higher tuning constant ϕ for theta. As a short term solution we tried limiting the angle of the vanes, but quickly rejected this idea as it reduced Bert's mobility.
The authors also experimented with an alternate control scheme that minimizes θ Bert aims to minimize $\dot{\theta}$ where $\ddot{\theta}_{target} = -\dot{\theta}\phi$ dubbed Spinny Bert. We have decided to include this version of Bert in the simulation. This also provide an interesting question of creating an iteration of Bert capable of maintaining a stable path while spinning or other dynamic movements.

After some digging it became clear that the target accelerations were well beyond the scope of what Bert could achieve. This was causing an enormous error when comparing

the target accelerations and caused erratic movement. After reducing the maximum value for the target accelerations Bert's movement became much more predictable.

2) Adaptive weights: An easy solution to the problem of θ error being measured in radians while location errors were measured in inches and tended to be higher was to multiply any error in θ by a high tuning constant, but this opened the door for a more interesting approach. The Bert simulation would still crash into the walls when it was within his capabilities to avoid the collision. We decided to use an adaptive weight to emphasize $\ddot{X}$ when Bert was close to a wall or $\ddot{\theta}$ when Bert was spinning out of control. After some tuning Bert seemed to be most stable given the adaptive weights:

$$Weights = \begin{bmatrix} X^2/\alpha \\ 1 \\ \\ theta^2\alpha + 1 \end{bmatrix} \tag{6}$$

The total control error can then be calculated as

$$Error_{total} = [(X_t - X_c)^2, (Y_t - Y_c)^2, (\theta_t - \theta_c)^2] \times Weights \tag{7}$$

C. Simulation Results

The simulation was run on a MacBook Pro laptop with an Apple M1 Pro chip. A typical OODA loop calculation took roughly .025 seconds, well below the period of the LiDAR (.11 s). Actually running this on the pi is still not tested, but we hopeful are it would work. For all of the tests Bert was given an extra random acceleration (up to ± 2 inches/sec, $\pm .2$ radian/sec) that changed every five seconds to simulate drift and test the overall robustness of the control scheme. The hallway depicted is marked by two inch tiles, is 4 feet wide and 100 feet long. Video of the simulation can be found at `https://www.youtube.com/playlist?list=PLQG6q-hFOpJaUhR-wqjivPP_yO_Z090Gd`.

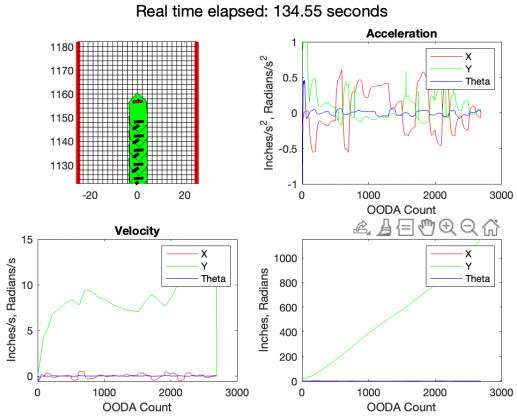

Fig. 4: Simulation

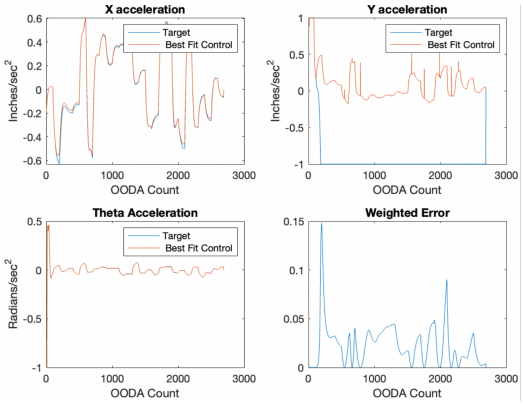

Fig. 5: Control Performance

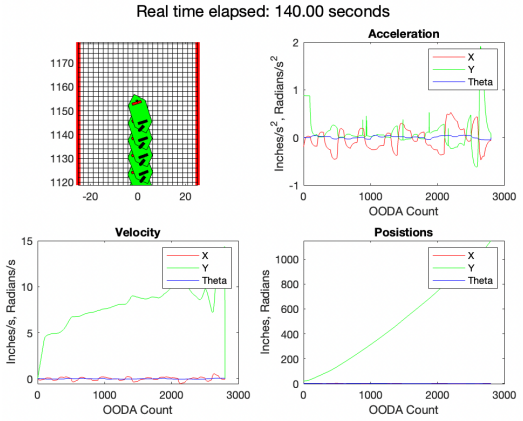

Fig. 6: Minimize $\dot{\theta}$ Simulation

D. Simulation Performance Analysis

Both Versions of Bert were able to successfully navigate the hallway without a collision, which we consider a success as far as the simulation. A four foot wide and 100 foot long hallway with random acceleration noise does not provide much of any room for error and is a serious challenge for the robot. Looking at figures 4,6 we can see how the acceleration changes every five seconds (100 loops) as the noise changes and Bert has to correct for a new error. Before deciding which scheme to implement we ran tests to see which scheme performed better in a variety of scenarios. First we decided to have the two models race, each Bert was given ten runs with noise in the Y direction turned off. The difference in average time and X displacement were minimal.

	Mean Time	Mean $\lvert X \rvert$ Inches
Bert	137.64	.76
Spinny Bert	135.55	.82

TABLE I: Control Comparison

Since the race did not show a clear winner, we decided to test how they performed with higher levels of noise. The regular Bert proved much more resilient to noise and could successfully navigate the hallway despite roughly twice the noise. Trying to understand precisely why the scheme leads

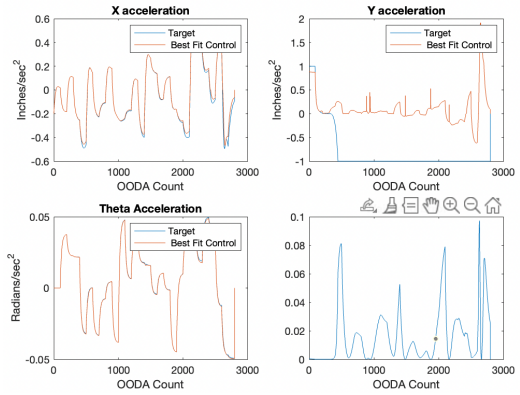

Fig. 7: Minimize $\dot{\theta}$ Control Performance

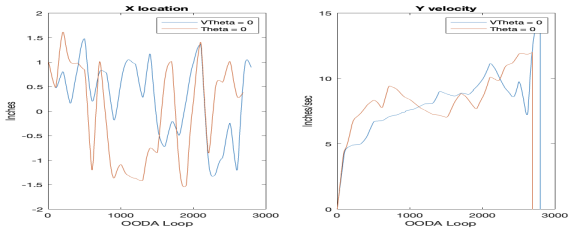

Fig. 8: Example Comparison Run

to particular outcomes is difficult but it is possibly the result of tuning variables mainly being tested on the original Bert model. Comparing the error in the weights is difficult as the weights change on each loop, but it seems that if the hovercraft state exceeds a threshold the weight error climbs and the robot behaves erratically. This may be due to the quadratic weight or the previous control not being a close enough approximation to find the minima. In situations where rapidly changing controls are required this could lead to larger errors in the gap between controls accelerations and targets. As to why this is more pronounced in Spinny Bert, it seems possible that as Bert's θ changes the ideal control would change more from loop to loop. In the future a hybrid quadratic and conic weight may be able to mitigate this problem.

After assessing the performance of both models in simulation we decided to use the original Bert as it is more robust to noise. Finally a version of the simulation that uses simulated LiDAR data was added. The LiDAR simulation has a much more jittery θ, but was able to navigate the hallway successfully. Compared to the other simulations the average $|X|$ is similar. In terms of compute, processing the lidar scans only added an average of seven milliseconds per loop, well below what was required for Bert's performance. As of writing this paper we are unsure as to what is causing this additional rapid oscillation. It may be tied to the code that generates the simulated LiDAR scan which can give a rounding error of up to 1.5 degrees, or it may just be the result of Bert's perceived location rapidly changing based on LiDAR noise.

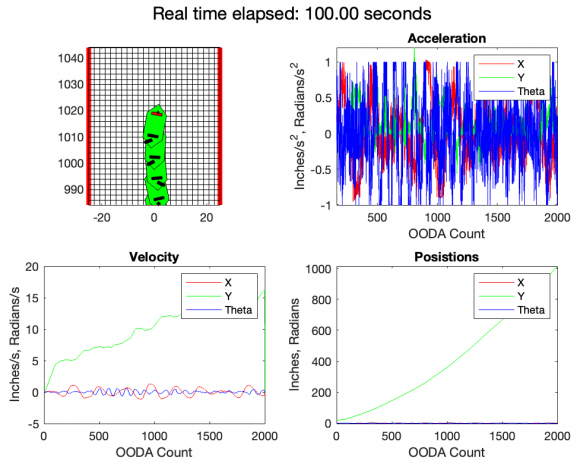

Fig. 9: Lidar Simulation

VII. Hardware

Bert has several sensing and propulsion systems including a LiDAR, an Accelerometer, Lift, and Directional Fans. The integration of each system presented its own set of challenges. Additionally they were all connected via a Raspberry Pi 4B as a central processing unit.

A. LiDAR

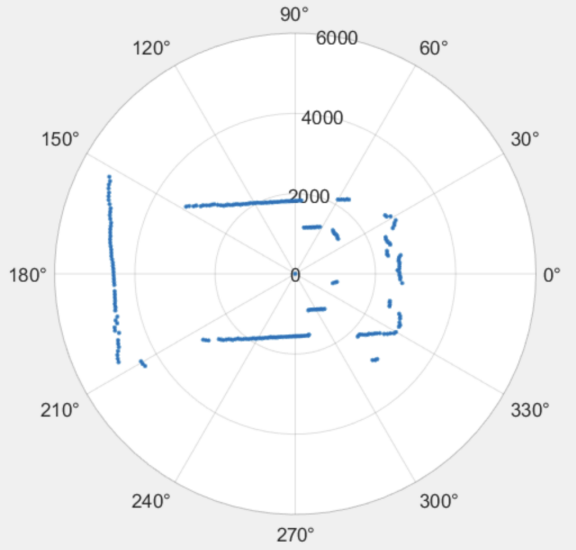

Fig. 10: The Author's Living Room millimeter scale

The RPLiDAR from Slamtec is the cheapest 2D 360° LiDAR system we could find. It was ideal for our purposes but for the majority of the project we could not make it work. This difficulty was due largely to the lack of support from the parent company who operates out of China. Most of the documentation was in Chinese, and much of the support on online forums was out of date. After a long period of trial and error and a helpful tutorial from adafruit [6] it turned out that the included software is unusable after the last few updates (despite many driver updates) and they no longer support python.

Additionally although the LiDAR device will start spinning when plugged in, it is very sensitive to the specific USB cable used in the connection and will not transmit information.The included USB did not meet the devices standards and needed to be switched out for another. The streaming data from the LiDAR was turned into a generator in order to provide on demand information to the control algorithm. It provided 360 distances (one for each degree) in a list several times a second.

B. Accelerometer

The trickiest part of placing the accelerometer was getting it to be approximately in the center of mass. This turned out to be impossible in all three axes as it would need to be suspended inside (or just above) a moving fan blades path. Since the Z axis isn't used we just needed to center it in the X and Y directions.

The data streaming from the accelerometer was also at several times per second, but at a different rate than the LiDAR. Its data stream was also turned into a generator to provide on demand info to the control algorithm.

C. Fans

Electronic Speed Controllers (ESCs) and brush-less motors were paired for the propulsive elements of the hovercraft. Servos were used to steer control surfaces (vaines) to the desired angle as determined by α_1 and α_2 inputs from the control algorithm. The $Thrust_1$ and $Thrust_2$ are controlled by the ESCs and the output of the control algorithm. Finding a way to arm and calibrate the ESCs without a transmitter was a challenge. Eventually a simulated transmitter was written that output pulse width modulation (pwm) signals to from the raspberry pi via Arduino.

D. Vanes

Each drive fan had a servo which could direct the force of the vanes. This is what allowed Bert to rotate and translate independently. The basic design was largely copied from [1].

Fig. 11: Control Vanes

Fig. 12: Bert in his natural Habitat

E. Raspberry Pi

The Raspberry Pi was Chosen over the Arduino due to the ease with which it could be interacted with. It was coded in python rather than C or C++ as would have been needed for the Arduino. Additionally the Pi was believed to be a better choice since as fully fledged computer it could perform all the required calculations and could more easily connect to the LiDAR. However, as a full computer control of the hovercraft is not its only process and it therefore has occasional lag. Additionally it required an additional power source to operate.

VIII. Conclusion

The intent of the authors for this project was to implement the motion planning techniques learned over the course of the semester in a hardware robot. The challenges included building from the class materials and executing them in reality. The software and motion planning algorithm is functioning well, but work is still continuing to fully integrate the Bert model into the physical Bert platform.

Although Bert was completed at the time this paper was published several integration challenges still loom. Bert is not executing the algorithm with full functionality and a lot of hardware tuning remains to be completed, but for now he mostly manages to avoid crashing into walls.

References

[1] D. Davis, Z. Marhoon, N. Lai, C. Chen, and C. McRae, "Albert 2.0: Smart mini-hovercraft," 2021.

[2] lobodol. (Year of the latest update or creation) Esc-calibration. Accessed on 12/10/23. [Online]. Available: https://github.com/lobodol/ESC-calibration?tab=readme-ov-file

[3] Arijit1080. (Year of the latest update or creation) mpu6050-with-raspberry-pi. Accessed on 12/10/23. [Online]. Available: https://github.com/Arijit1080/mpu6050-with-Raspberry-Pi/blob/master/mpu_6050.py

[4] OODA loop. [Online]. Available: https://imarcai.com/ooda-loop-new/
[5] A. O'Dwyer, "Control of open-loop unstable processes with time delay using PI/PID controllers specified using tuning rules: An outline survey," in *24th IET Irish Signals and Systems Conference (ISSC 2013)*, pp. 1–7. [Online]. Available: https://ieeexplore.ieee.org/document/6621221
[6] Adafruit. (2019) Using the slamtec rplidar on raspberry pi. Accessed on 12/16/23. [Online]. Available: https://learn.adafruit.com/slamtec-rplidar-on-pi/using-the-slamtec-rplidar