ECE 470 Project Final Report

Ayano Hiranaka(ayanoh2), Meining Wang(meining2), Weihang Liang(weihang2), Liwu Tsao(liwuwt2)

Team Name: Katsu-Don

Codebase: https://gitlab.com/weihang2/ece470_project

Video: https://www.youtube.com/watch?v=HVntnbgSMss

Abstract

Our project simulates a sumo robot competition. One of the robots use a lidar for perception, and a PID control loop to always face directly at the opponent. The second robot is equipped with 4 cameras (one viewing the front, one with 45-degree view range from front to left, one with 45-degree view range from front to right, and one in the back) to detect keypoints on each image scene. The camera robot tries to localize the object by the scene where the most keypoints appears, and take its action to get closer to that place.

Three experiments (competitions) were conducted in this project: (1) autonomous lidar-based robot vs human-controlled robot, (2) autonomous camera-based robot vs human-controlled robot, and (3) autonomous lidar-based vs camera-based robots. Both the autonomous robots developed in this project were highly competitive against human-controlled robots. Additionally, the performance of the lidar, PID controller, and image processing are evaluated.

In conclusion, although the focus of this project is an extremely specific application of robotics, the simulation provided a valuable opportunity to investigate sensing, decision-making, and control algorithms relevant to a wide range of robotics applications. In the future, this project can be extended to: real-life experiment using the algorithms developed in this project, entertainment-focused competition simulation, and simulation in irregular conditions, such as non-circular arena or random starting position.

ECE 470 Project Final Report

1. Introduction

In a sumo robot competition, two robots try to push each other out of a dohyo, a circular arena. Most sumo robots, especially the smaller ones, compete autonomously and use different types of sensors to detect opponents and avoid falling out of the arena. Even though the rules are simple, sumo robot competition is significant in the field of robotics because it is a testing ground for high-speed sensing and decision making. While most real-life sumo robots use infrared proximity sensors, we decided to use lidar and camera on our two robots to investigate new possibilities. The objective of the project is to model a sumo robot competition between two autonomous sumo robots, and to investigate new possibilities for sumo robot perception and decision-making algorithms. In this project, each of the two robots feature different sensors: lidar and camera. The two robots also use different decision-making algorithms.

Through research, we found out that most literature about the sumo robots focused on the mechanical or electrical design of the robot, both of which we could not easily modify in the V-REP environment. Additionally, research about the use of fuzzy logic focused mainly on real-life sumo robots that use infrared proximity sensors. However, we did learn from our research several potential strategies that might work well. Our lidar-based robot, Don-Bot, utilized the lidar readings and the PID control loop to always face towards the opponent. Our camera-based robot, Katsu-Bot, combines the readings from the 4 camera sensors, then use the scene with the greatest number of detected "keypoints" to make an overall decision. A visualization of the two robots is shown in Figure 1.

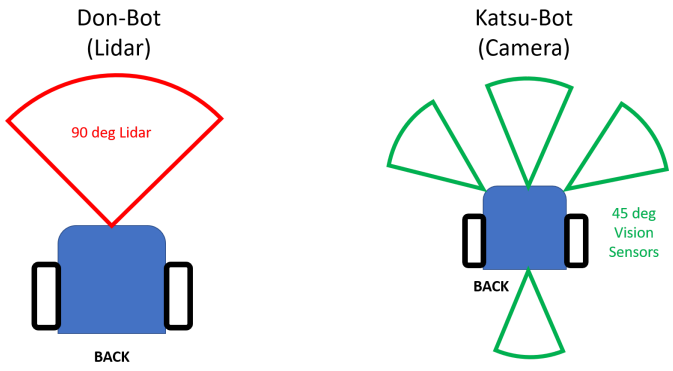

Figure 1: Two Sumo Robots (Don-Bot and Katsu-Bot)

ECE 470 Project Final Report

2. Method

2.1 Control Variables

In real sumo robot competitions, every robot has unique mechanical and electrical characteristics, and these design factors are equally important as the sensor and software strategies. However, for simplicity and for effective comparison of the two sensor types, we ensure that the two robots are identical in their mechanical and electrical characteristics by using the DR12 mobile robot from the V-REP model library as both of the competing robots. In addition, two identical "line" sensors (small binary color sensor that reads positive when the robot is on the white arena outline) are added to each robot to help preventing them from driving out of the arena by themselves. Since both robots shared the DR12 mobile base, we could use a generalized model of DR12 in Python with forward and inverse kinematics. Then using inheritance, we could easily write additional functions corresponding to the two types of sensors while sharing most of the base functions. The following sections describe the two robots' perception, decision-making, and control algorithms.

2.2.1 Don-Bot Perception

The lidar-based Don-Bot had a 90-degree proximity sensor in the front. In V-REP, the proximity sensor simply returns the closest point inside the defined angular sector. This simulates choosing the closest point in a lidar scan. The closest point read from the sensor is assumed to be the location of the opponent, and the Don-Bot moves towards the opponent in an attempt to push it off of the arena.

2.2.2 Don-Bot Decision-Making Algorithm

Don-Bot's decision-making algorithm is shown in Figure 2. There are two pieces of information the Don-Bot can obtain from its sensors: the opponent's location with respect to the front of itself (from its lidar readings) and the arena border (from its line sensors). When making an action decision, Don-Bot considers three cases: (1) the opponent is observed and the arena line is not observed, (2) the opponent and the line is observed, and (3) neither the opponent nor the line is observed. In the first case, Don-Bot simply moves towards the opponent to push it because the risk of falling off the arena is low. The second case is less trivial. Although the line sensor

reading suggest a high risk of falling off the arena, it may beneficial to continue pushing the opponent if the opponent is almost falling off the arena as well. If the angle between the front of the Don-Bot and the opponent is small, the opponent is considered to be in front of Don-Bot (small angle condition). In this situation, Don-Bot continues to move towards the opponent to push it off the arena. In our simulation, the small angle condition occurs when the angle between the front of Don-Bot and the opponent is less than 5 degrees. On the other hand, if both the line and the opponent is detected and the small angle condition is not met, continuing to move towards the opponent is a high-risk action (Don-Bot will most likely fall off the arena before pushing the opponent off). In such case, Don-Bot stops pushing the opponent, and takes an action to move away from the line. Specifically, it will back up, and turn in the direction opposite from the line sensor detecting the line. If both line sensors are detecting the line, Don-Bot simply moves straight back. The final condition occurs when neither the opponent nor the line is detected. In such case, Don-Bot quickly turns in the direction where the opponent was last detected to relocate the opponent.

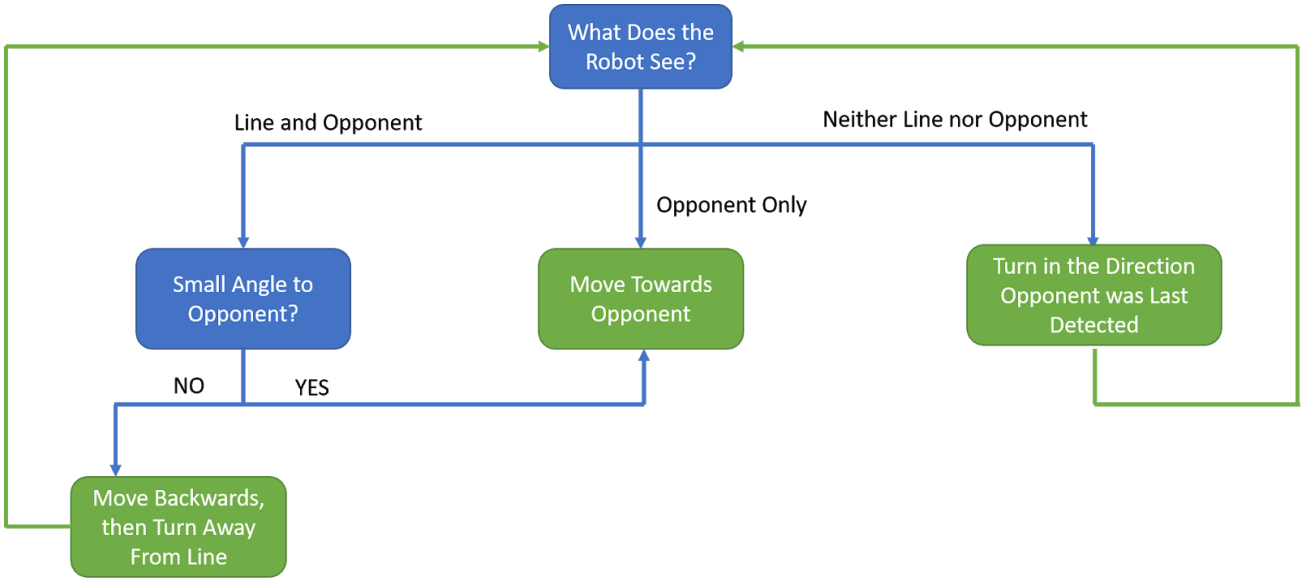

Figure 2: Decision-making Algorithm of Don-Bot

2.3 Katsu-Bot

The Katsu-Bot, uses four camera sensors (3 in the front and 1 in the back) to detect its opponent. Katsu-Bot makes an action decision by detecting "keypoints" within each camera reading, and combining information from all four sensors. The following sections describe Katsu-Bot's perception and decision-making algorithms.

2.3.1 Katsu-Bot Perception

The vision-sensor-based Katsu-Bot utilizes four vision sensors around the body of the robot: to the front, 45-degree front-left, 45-degree front-right, and back, to detect the key points of the surroundings. In V-REP, the vision sensor returns different resolutions of square images and can be processed by Python API into a stream of data inputs. The Katsu-Bot may process a decision-making algorithm and makes a better attempt to win the game.

2.3.2 Katsu-Bot Decision-Making Algorithm

Katsu-Bot's decision-making algorithm is shown in Figure 4. The robot can receive an image that has a resolution of 256 (shown in Figure 3). The image is of RGB standard and can be processed into arrays. Due to that fact, we generate a decision-making algorithm using image flows from four vision sensors. The inputs are the key points of the images detected by OpenCV (shown in Figure 3), and the outputs are the right and left wheel velocity. The algorithm detects which of the image has the most key points among all the four results. For example, if the front-left sensor detects most key points, then the Katsu-Bot will turn left, if the front-right sensor detects most key points, then the Katsu-Bot will turn right. It will also choose to move forward or backward when detecting of the front sensor or the back. If any vision sensor among the four has no key point detected, it will move forward or backward immediately in order not to fall off of the arena.

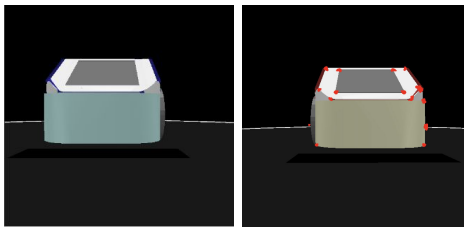

Figure 3: Examples of Vision Sensor and Key Points detect

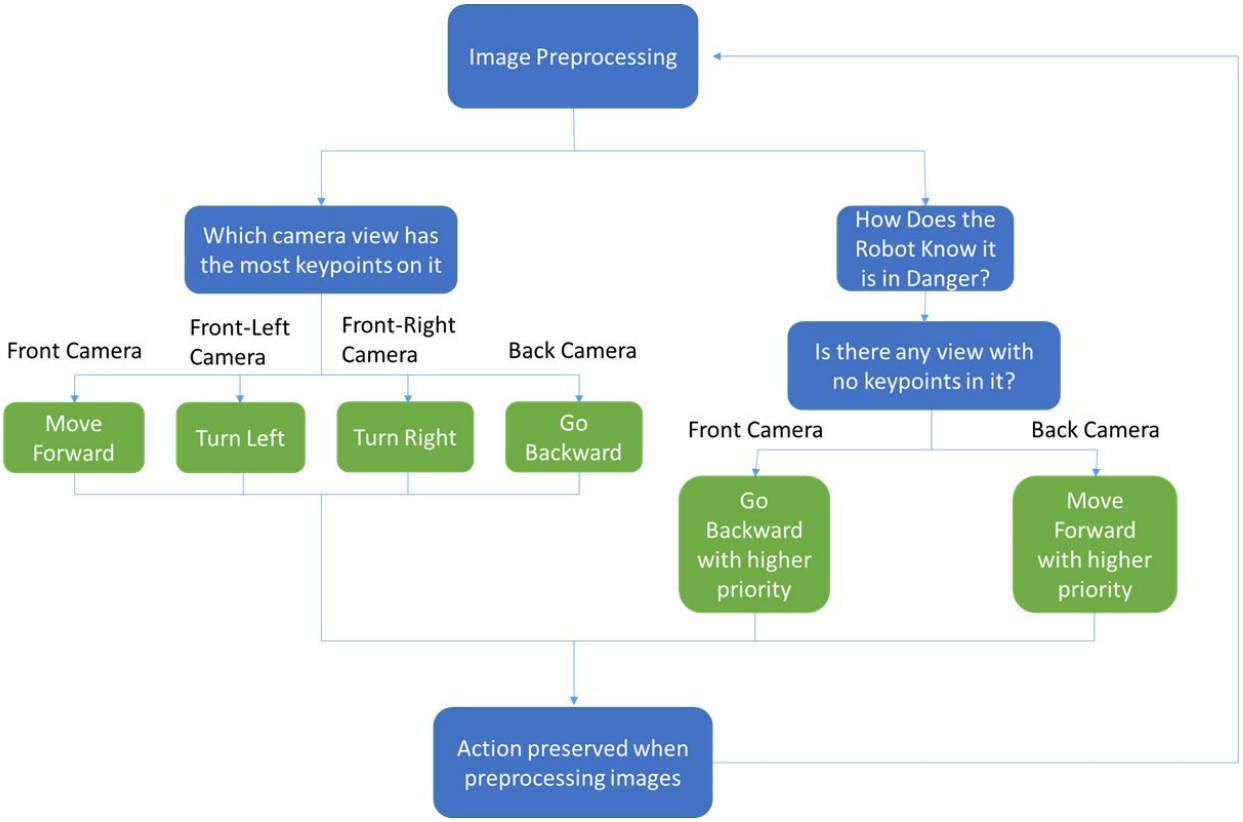

Figure 4: Decision-making Algorithm of Katsu-Bot

2.4 Forward and Inverse Kinematics

Once the action decision is made, the next step is to control the sumo robots. A common forward and inverse kinematics model is developed for the two robots. The origin of the robot's frame is located midway between the centers of its two wheels, and the direction of axes are defined as shown in Figure 5. The robot's motion can be viewed as a rotation about an instantaneous center of rotation (ICC), which is related to the origin of the robot frame by vector R.

The following are the robot's parameters:

- Wheel diameter: $R_{wheel} = 43\ mm$
- Distance between wheels: $L = 164\ mm$
- Angular velocity of wheels: ω_L , ω_R = *controlled parameter*

ECE 470 Project Final Report

The following equations describe the robots' forward kinematics. Given any input rotational velocities on each wheel, linear and angular velocities of the robot is calculated as:

- ICC vector R: $R = [0.5L(\omega_L + \omega_R)/(\omega_R - \omega_L), 0, 0]$
- Angular velocity: $\omega = [0, 0, R_{wheel}(\omega_R - \omega_L)/L]$
- Linear Velocity: $V = \omega \times R$

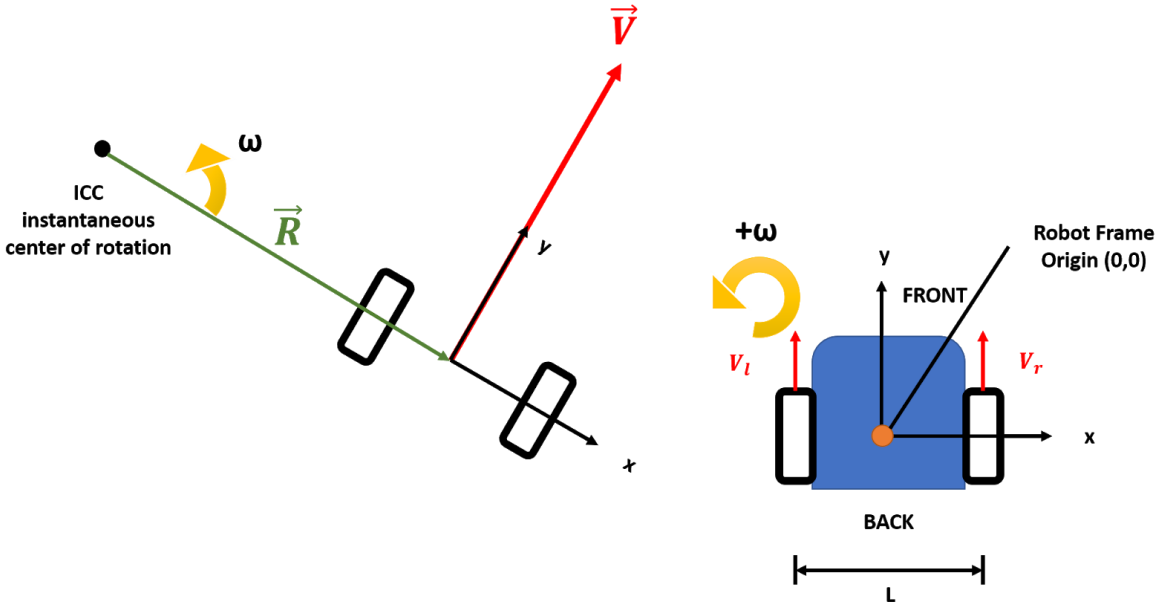

Figure 5: Forward Kinematics

The only controllable parameters for the robots are the angular velocities of each of their wheels. However, being able to control the motion of the robot itself, rather than the motion of the individual wheels, is more useful in our simulation. Here we use the forward kinematics model to define an inverse kinematic model that calculates the required wheel angular velocities, given the desired angular velocity of the robot:

- Angular velocity of left wheel: $\omega_L = (l\omega + 2v_{lim})/(2R_{wheel})$
- Angular velocity of right wheel: $\omega_R = (2v_{lim} - l\omega)/(2R_{wheel})$

where v_{lim} is the user-defined maximum allowed linear velocity of the robot.

This model allows automatic calculation and setting of individual wheel velocities given the desired robot angular and linear velocities.

2.5 PID Controller

We used a PID controller in the proximity sensor based Don-Bot for minimizing the angular error between the robot and the opponent while going forward. We wrapped the PID algorithm into a class and added a sliding window for the integral term to prevent integral windup. In a sumo robot competition, since the robots are moving relatively fast, any old data is irrelevant. Our PID controller worked really well in the simulated competitions, and the results are in Section 4.

3. Experimental Setup

The experimental setup in our project simulated the actual sumo robot competition. Since there were two robots in our project, we needed to have a way to update and sync their motion. While it was possible to put them in two threads, we used a better design which was to have a common clock that updated both robots together at a frequency of 20 Hz. Every 50 milliseconds, the two agents that controlled the robots read from the sensors and updated the robots' actions. The clock also triggered the competition judge. When a robot fell out of the arena, the judge would terminate the competition and output the winner. If there was no winner within a certain time, the competition would also end and output that the result was a draw.

While in the end, the experiment ran automatically without human control, it was nice to have control of one of the robots to test out the performance of the other. Thanks to our design that separated the agent and the robot, we could easily replace the agent that controlled the robot with human.

In this project, three different experiments (competitions) are conducted: (1) Don-Bot (lidar-based) vs human agent, (2) Katsu-Bot (camera-based) vs human agent, and (3) Don-Bot vs Katsu-Bot. The human agent is developed using PyGame, and is controlled using the "WASD" keys on a keyboard. The setup of the competition simulation is found in Figure 6 and example

matches for each of the above conditions can be found in the video (link at the beginning of report).

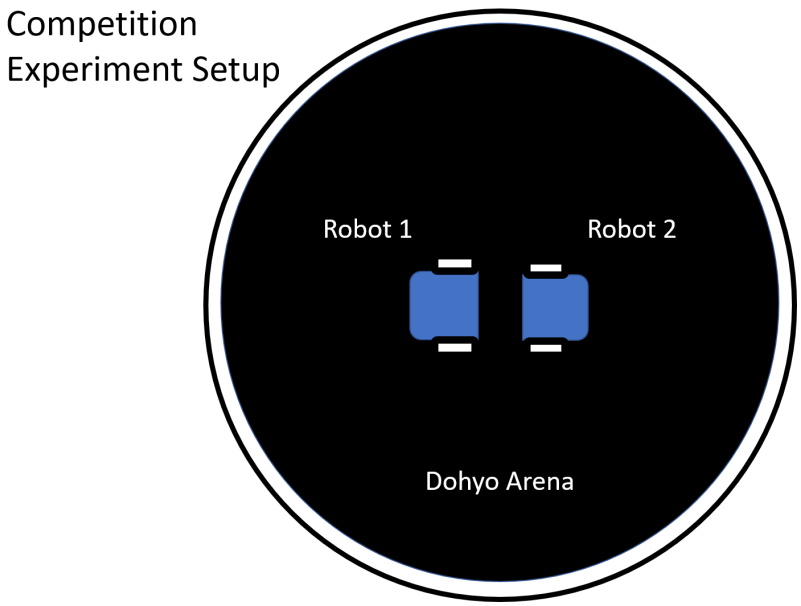

Figure 6: Competition Experimental Setup

ECE 470 Project Final Report

4. Data and Results

4.1 Don-Bot PID Error Data

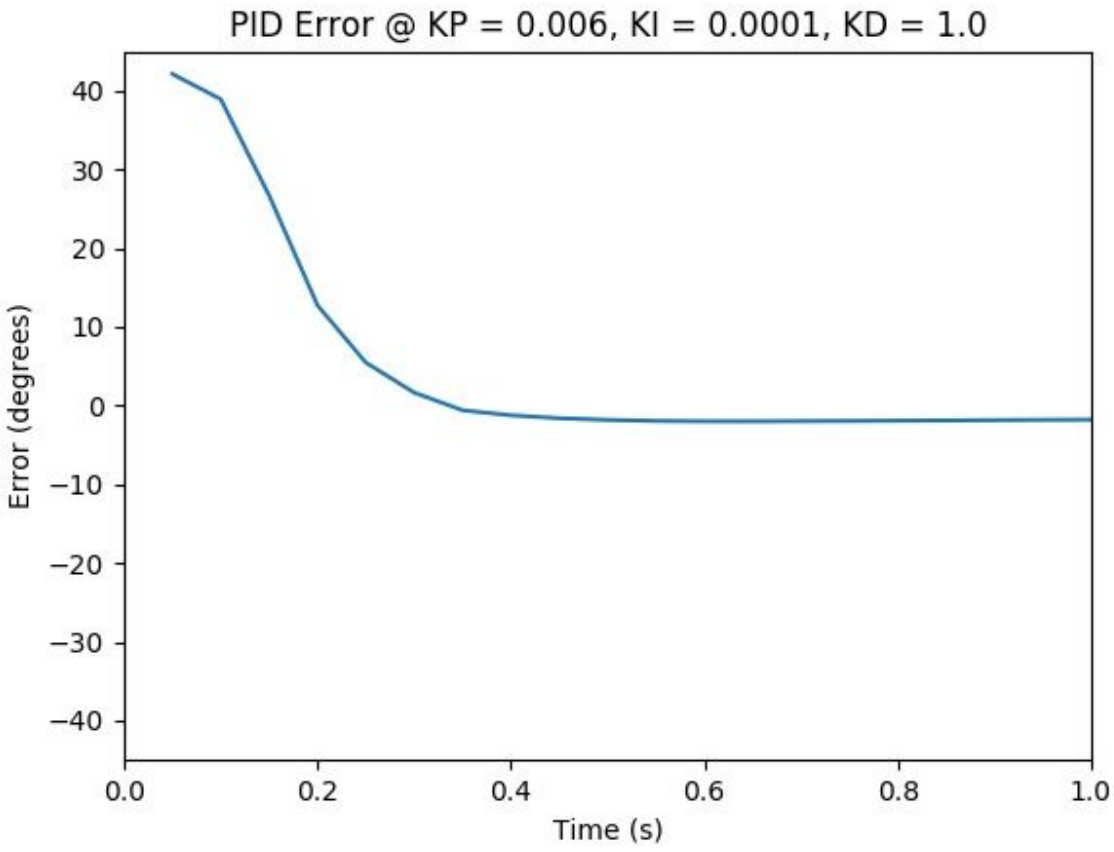

Figure 7: PID Error Data as Opponent Move in Front of Don-Bot

The first test was conducted to test the performance of the PID control algorithm. After tuning the gain variables, our PID controller use by Don-Bot performed exceptionally well in the simulator. Figure 7 shows the PID error plot as a human-controlled opponent moved in front of Don-Bot. As observed in the error plot, the PID controller had a settling time less than 0.4 seconds in simulator time and had minimal overshoot when the gains were tuned appropriately. When gains were poorly tuned, the Don-Bot struggled to keep the opponent in the front by overshooting the opponent or by moving too slowly. The successfully tuned PID controller

allowed Don-Bot to always face the opponent and use full power of its drive system to push the opponent off the arena.

4.2 Katsu-Bot tradeoff on Time Delay and Keypoint Detection

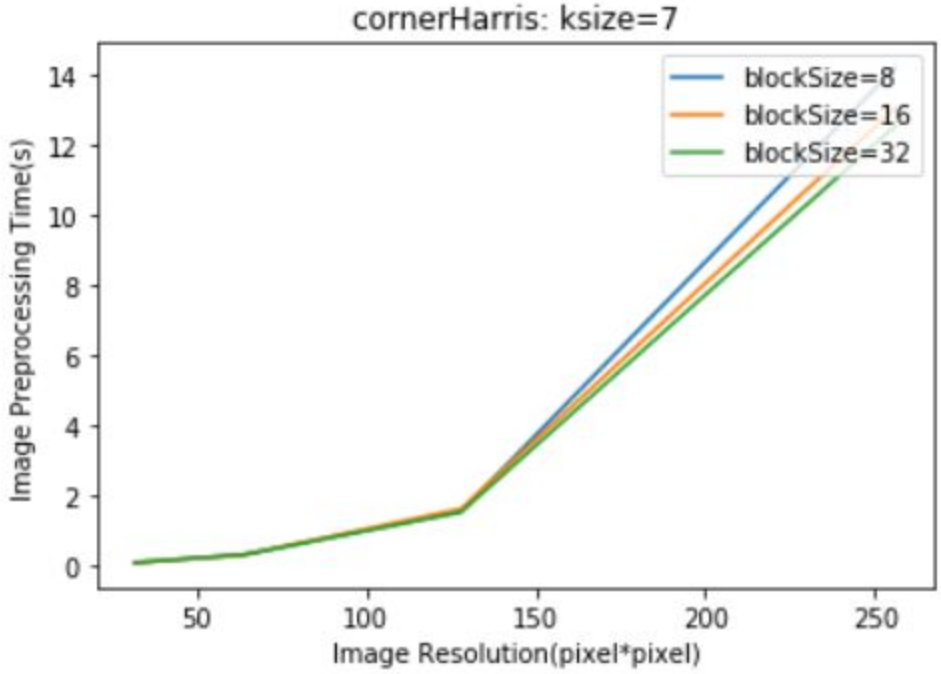

Figure 8: Katsu-Bot Time Delay on Image Processing

For several experiments on changing the Image Resolution and tuning the blockSize, we discover some interesting fact in our build-up environment. At first, we are setting our camera-based agent with (blockSize = 8, Image Resolution = 32). However, it doesn't work well due to the keypoint will take those white border line into consideration in such a low resolution. Then, when we try the setting with (blockSize = 8, Image Resolution = 256), the agent works perfectly, the white line no longer matters. But, the time delay on preprocessing those images into keypoints are too high, we can't make decision every 14 seconds, that's not useful for a real-time competition. When we tried the setting with (blockSize = 32, Image Resolution = 64), the time delay is low enough, and it can get the accurate direction of the most keypoint when another robot is in the near view. It still needs to solve the problem in low Image Resolution have a problem that it can't get enough information when another robot is far away.

ECE 470 Project Final Report

5. Conclusion

Both the camera-based Katsu-Bot and the lidar-based Don-Bot were very competitive and could beat human players easily. The final competition was intense, and both robots competed well. We saw our final strategies work during the competition, and it was interesting to test other different strategies during development.

Sumo robots and the competition were fun to build and watch, both in real life and in the V-REP simulator. Though sumo robots do not have a direct application, they are good testing platforms for robot sensors and algorithms. For example, the strategy/algorithm used to detect and push the opponent could be used in obstacle avoidance in self-driving cars. We learned a lot from the project, especially because we used two very common and important sensors in the field of robotics: lidar and camera. Through working with the sensors, we understood more about sensor-driven intelligent robots.

6. Recommendations

Interesting problems to investigate in the future include, but not limited to:

- Simulation of sumo robot competition on an irregularly shaped dohyo, random starting positions, etc.: Line-avoidance algorithm and control precision gains more importance, leading to a more challenging problem.
- “Predator-Prey” Simulation, where a predator robot (programmed only to push the prey robot out of the arena) and a prey robot (programmed only to avoid the predator and falling off the arena)
- Real-life experimentation of the perception, decision-making, and control algorithms developed in this project to investigate the accuracy of the simulated results.
- Introduce more randomness in the two robots’ decision-making algorithms to make the competition simulations more entertainment-oriented.
- Applying reinforcement learning methods such as Q-learning for the camera-based robot.

ECE 470 Project Final Report

7. References

[1] "Proximity Sensors." *Proximity Sensors*, Coppeliar Robotics, www.coppeliarobotics.com/helpFiles/en/proximitySensors.htm.

[2] Torrico, César R.c., et al. "Modeling and Supervisory Control of Mobile Robots: A Case of a Sumo Robot." *IFAC-PapersOnLine*, vol. 49, no. 32, 2016, pp. 240–245., doi:10.1016/j.ifacol.2016.12.221.

[3] "Unified Sumo Robot Rules." *Unified Sumo Robot Rules*, Robotics Society of America, Inc (RSA), 2004, robogames.net/rules/all-sumo.php.

[4] "Vision Sensors." *Vision Sensors*, Coppeliar Robotics, www.coppeliarobotics.com/helpFiles/en/visionSensors.htm.

ECE 470 Project Final Report

Item	Points	Purpose
GitHub link	5	Is the codebase well documented and easy to follow?
YouTube link	5	Does the video effectively demonstrate the outcomes of the project?
Style and Formatting	5	Is the report consistent with the lab report guidelines?
Abstract	5	Is the abstract well written and concise?
Introduction, Background, Problem Statement, and References	10	Did you effectively convey why your project is interesting? Did you do some background research to figure out how to solve your problem? Did you give people credit for inspiration and/or code used?
Experimental Setup	10	Did you effectively describe your task and experimental protocol?
Methods	20	Did you visualize the block diagram and are the modules well described? Is the course material connected to the project and did you convey understanding of the topics?
Data and Results	20	Did you have a good quantitative measure of performance? Did you provide an insightful qualitative example (of success and/or failure)? Did you analyze your systems failure cases?
Conclusions	10	Did you effectively summarize your project? Did you provide a thoughtful reflection of the project?
Project Difficulty	10	Does the project scope and outcomes match the size of the team?
TOTAL	**100**	