# VECTOR-BASED MECANUM-DRIVE ROBOTIC SYSTEM WITH MACHINE VISION AND SENSOR FUSION FOR SOCCER TRAINING

by

Bartholomew Nathaniel Briggs

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Applied Energy & Electromechanical Systems

Charlotte

2019

Approved by:

_____
Dr. Aidan Browne

_____
Dr. Maciej Noras

_____
Dr. Wesley Williams

ABSTRACT

BARTHOLOMEW NATHANIEL BRIGGS. Vector-based Mecanum-Drive Robotic System with Machine Vision and Sensor Fusion for Soccer Training. (Under the direction of DR. AIDAN BROWNE)

This research proposes the use of a robotic system for soccer training. The purpose of this research is to devise, build, and demonstrate a vector-based mecanum-drive robot to act as an impediment to a soccer player dribbling toward a known goal target. Successful implementation of the system is defined by the robot maintaining its objectives during interaction with the human player in a specified test environment: maintaining a three-point linearity between the player and goal target; maintaining an appropriate distance from the player, known as the buffer zone, whose center is the human player and whose radius is between 0.5 and 1 meter; and maintaining orientation with the robot's front aligned tangent to the buffer zone circumference.

The robot's objectives are demonstrated in a test environment that is indoors with standard ambient light conditions and smooth, untextured concrete surface with a minimum useful area of 5 meters by 5 meters. Prior to the match, the human player begins in the corner opposite to the target, while the robot begins in the center of the test area. The game begins with the human player presses the start button; the robot indicates commencement of the match to the human player; the game ends when time expires or the player presses the stop button.

During the match, data from the robot's front and rear camera sensors are fused to create appropriate vectors defining robot strafe magnitude and direction for

maintaining linearity and buffer zone distance, as well as required rotation for maintaining buffer zone tangency. Data collected through trials is used to verify system function and optimize autonomous control.

A Proof-of-Concept Robotic System was constructed and proved ultimately successful in demonstrations against a player of reasonable ability.

This project extends the previous work of Dr. Aidan Browne and Stephen Padgett: applying a vector-based drive system (Mecanum) with a vector-based sensor (Slamtec - RPLIDAR-A1: 2D Puck LiDAR) through enhancement of its Obstacle Tracking and Avoidance System.

# DEDICATION

Without the support of my beautiful wife Sana, my loving parents Cynthia and Kendall, my siblings Joshua and Jocelyn, and my puppy dogs Bert and Beastmaster, there is no way my life would have led to my ability to complete a project of this complexity and magnitude. This thesis is dedicated to the amazing people above and to my newborn son Zain whose world is brand new and is without limit.

ACKNOWLEDGEMENTS

My colleague Vedant Raval has gone above and beyond to assist me with some of the more perplexing aspects of this project – helping to bring me back to Earth on what can be done within the short time period of the Master's program, showing me the idiosyncratic aspects of the LabVIEW programming language, and trying to get the LiDAR to obey my commands. It turns out that eating an elephant is accomplished one bite at a time and can go much faster with assistance from a clever mind full of electrical, mechanical, and programming knowledge.

I would also like to give much appreciation to my professors in the Applied Energy & Electromechanical Systems program for constantly challenging me throughout my coursework and giving me the freedom to choose projects that were incredibly fulfilling. A special depth of gratitude and appreciation is deserved by Dr. Aidan Browne who provided a great amount of help when I was intensely in need of it – helping me define a path to success and providing to me a workspace, tools, and equipment to make this project happen.

TABLE OF CONTENTS

LIST OF TABLES ..... IX

LIST OF FIGURES ..... X

LIST OF SYMBOLS AND ACRONYMS ..... XIV

1. INTRODUCTION ..... 15

1.1. Motivation [Problem Statement] ..... 15

1.2. State of the Art [Literature Review] ..... 17

1.2.1. Sensors and Vision Systems for Localization ..... 18

1.2.2. Sensor Fusion ..... 24

1.2.3. Safety ..... 25

1.2.4. Mecanum Drive System Control ..... 26

1.2.5. Measurement Uncertainty ..... 28

1.2.6 Soccer-Specific Robotic Systems ..... 30

1.3. Mecanum Drive ..... 31

1.4. Technologies Investigated ..... 35

2. SYSTEM ARCHITECTURE ..... 41

3. SENSOR FUSION ..... 51

4. CONTROL SYSTEM ..... 71

4.1 Mecanum Drive Implementation ..... 71

4.1.1. Tele-Operational Mapping ..... 78

4.2 Autonomous Control ..... 83

5. TEST PLAN ..... 91

6. RESULTS ..... 97

7. DISCUSSION AND CONCLUSIONS ..... 106

REFERENCES ..... 110

APPENDIX ..... 114

Data Fusion Lexicon Definitions per JDL: ..... 114

Component Technical Specifications: ..... 115

Mecanum Wheels ..... 115

CIM Motor ..... 116

Toughbox Mini Gearbox ..... 116

US Digital E4P Optical Encoders ..... 117

Talon SRX Motor Speed Controllers ..... 117

Battery ..... 117

Power Distribution Panel ..... 118

Voltage Regulator Module ..... 118

LIST OF TABLES

TABLE 1: Technologies Considered....40

TABLE 2: Mechanical Parameters of Robot....42

TABLE 3: Mapped Voltage Colors for LEDs....50

TABLE 4: PWM Duty Cycle Mixture....77

TABLE 5: Visibility Loss Summary Data (s)....103

TABLE 6: Motor Parameters....116

TABLE 7: Talon SRX Technical Parameters....117

TABLE 8: myRIO PWM Channel Layout....118

LIST OF FIGURES

FIGURE 1: Game Layout ..... 16

FIGURE 2: Right Mecanum Wheel ..... 31

FIGURE 3: Left Mecanum Wheel ..... 32

FIGURE 4: Mecanum Configuration with Force Vectors for Forward Movement ..... 33

FIGURE 5: Forward Movement Vector Addition ..... 33

FIGURE 6: Clockwise Rotation Vector Addition ..... 34

FIGURE 7: Lateral Movement Vector Addition ..... 34

FIGURE 8: Controllable Degrees of Freedom for Mecanum Drive (X,Y,ψ) ..... 35

FIGURE 9: LiDAR Components Considered ..... 36

FIGURE 10: LiDAR State Machine Noise ..... 37

FIGURE 11: Stereo Vision Mounting ..... 38

FIGURE 12: Thermal Imaging System ..... 39

FIGURE 13: Thermal Vision Red Dot Error ..... 39

FIGURE 14: FutBot Front View ..... 41

FIGURE 15: FutBot Rear View ..... 41

FIGURE 16: Mecanum Drive System Components ..... 43

FIGURE 17: Power System Components ..... 44

FIGURE 18: Controller and Vision Hardware ..... 45

FIGURE 19: Vision Conduit System ..... 46

FIGURE 20: Vision Mounting ..... 47

FIGURE 21: Buttons and LED Splitter ..... 48

FIGURE 22: Front-Facing Camera Sample Image Initial ..... 52

FIGURE 23: Front-Facing Camera Color Model and Threshold Values ..... 53

FIGURE 24: Front-Facing Camera Sample Image Threshold Validation ..... 53

FIGURE 25: Front-Facing Camera Sample Binary Image ..... 54

FIGURE 26: Front-Facing Camera FFT Low-Pass Filter ..... 55

FIGURE 27: Front-Facing Sample Image FFT Filter Result ..... 55

FIGURE 28: Front-Facing Camera Binary Morphology Erosion Parameters ..... 56

FIGURE 29: Front-Facing Sample Image Binary Erosion Result ..... 56

FIGURE 30: Front-Facing Camera Binary Morphology Dilation Parameters ..... 57

FIGURE 31: Front-Facing Sample Image Binary Dilation Result ..... 57

FIGURE 32: Front-Camera FFT Method Performance Analysis ..... 58

FIGURE 33: Front-Camera Binary Morphology Method Performance Analysis ..... 58

FIGURE 34: Front-Facing Camera Sample Image Centroid ..... 59

FIGURE 35: Front-Facing Camera Sample Image Centroid Coordinates ..... 60

FIGURE 36: Front-Facing Camera Sample Image Histogram Statistics ..... 61

FIGURE 37: Buffer Zone Tangency ..... 61

FIGURE 38: Goal Target Sample Image Initial ..... 62

FIGURE 39: Rear-Facing Camera Color Model and Threshold Values ..... 63

FIGURE 40: Rear-Facing Camera Sample Image Threshold Validation ..... 63

FIGURE 41: Rear-Facing Camera Sample Binary Image ..... 64

FIGURE 42: Rear-Facing Camera FFT Low-Pass Filter ..... 65

FIGURE 43: Rear-Facing Sample Image FFT Filter Result ..... 65

FIGURE 44: Rear-Facing Camera Sample Image Circle Detection Result ..... 66

FIGURE 45: Rear-Facing Camera Circle Detection Parameter Details ..... 66

FIGURE 46: Rear-Camera FFT Method Performance Analysis ..... 67

FIGURE 47: Rear-Camera Binary Morphology Method Performance Analysis ..... 67

FIGURE 48: Rear-Facing Camera Binary Morphology Erosion ..... 68

FIGURE 49: Rear-Facing Camera Sample Image Binary Morphology Erosion Result ..... 68

FIGURE 50: Rear-Facing Camera Binary Morphology Dilation ..... 69

FIGURE 51: Rear-Facing Camera Sample Image Binary Morphology Erosion Result ..... 69

FIGURE 52: Rear-Facing Camera Sample Image Centroid ..... 70

FIGURE 53: Rear-Facing Camera Sample Image Centroid Coordinates ..... 70

FIGURE 54: Front-Facing Camera Sample Image Histogram Statistics ..... 70

FIGURE 55: Mecanum Mixer Front Panel ..... 71

FIGURE 56: Duty Cycle for PWM ..... 73

FIGURE 57: Vector Magnitude ..... 74

FIGURE 58: Mecanum Mixing of PWM Duty Cycles ..... 76

FIGURE 59: Xbox One Controller Mapped Parameters ..... 79

FIGURE 60: Xbox Controller Parameters VI ..... 79

FIGURE 61: Tele-op Control Front Panel ..... 82

FIGURE 62: Goal Priority ..... 85

Figure 63: Goal Target Image Parameters ..... 85

FIGURE 64: Intended Robot Movement ..... 87

FIGURE 65: Player Priority ..... 88

FIGURE 66: Player Target Image Parameters ..... 88

FIGURE 67: Proportional Controller Diagram ..... 90

FIGURE 68: Goal Monitor Loop Test Setup ..... 92

FIGURE 69: Player Monitor Loop Test Setup ..... 94

FIGURE 70: Player and Goal Monitor Fusion Loop Test ..... 95

FIGURE 71: Goal Monitor Loop Test Front Panel ..... 97

FIGURE 72: Goal Monitor Loop Test - Proportional Control Repeatability ..... 98

FIGURE 73: Goal Monitor Loop Step Response ..... 99

FIGURE 74: Player Monitor Loop Test Front Panel ..... 99

FIGURE 75: Goal Monitor Loop Test - Proportional Control Repeatability ..... 100

FIGURE 76: Player Monitor Loop Step Response ..... 101

FIGURE 77: Goal and Player Monitor VI Front Panel ..... 102

FIGURE 78: Goal and Player Monitor Loop Test ..... 104

FIGURE 79: Distance Monitor Loop Step Response ..... 104

FIGURE 80: Full Fusion Step Response ..... 105

FIGURE 81: Futbot Reaction to Player ..... 106

FIGURE 82: CIM Motor Torque-Speed Curve (“CIM Motor—VEXpro Motors—VEX Robotics,” n.d.) ..... 116

FIGURE 83: myRIO 1900 Channel Pinouts (National Instruments) ..... 119

LIST OF SYMBOLS AND ACRONYMS

| | |
|---|---|
| VI | Virtual Instrument |
| IDE | Integrated Development Environment |
| PID | Proportional-Integral-Derivative (Controller) |
| PWM | Pulse Width Modulation |
| SOC | System On Chip |
| RGB | Red, Green, Blue |
| DM | Direct Movement |
| RM | Rotational Movement |
| LM | Lateral Movement |
| ROI | Region of Interest |

# 1. INTRODUCTION

## 1.1. Motivation [Problem Statement]

Throughout my life, the two primary subjects of interest have always remained: Soccer (Futbol) and Robots – so naturally, combining the two of them for an overwhelmingly intensive project makes sense does it not?

This project incorporates many of the ideas, strategies, theories, and applications presented during my master's degree curriculum in Applied Energy and Electromechanical Systems at the University of Charlotte, North Carolina whose primary subjects include: System Dynamics, Applied Mechatronics, Applied Statistics, Advanced Instrumentation, as well as Energy Generation and Conversion and Energy Transmission and Distribution.

Soccer is one of my greatest passions – which is greatly understating the situation. I practice nearly daily to increase agility, foot speed, dexterity, shot and pass accuracy, and dribbling technique. Developing appropriate drills for dribbling technique has proven the most difficult to adequately correspond to the dynamic environment of a live game. Using cones as obstacles can help with confidence on the ball and building muscle-memory via repetitive movement, however, the ability to perceive an opposing player's approach and create successful strategies for mitigating ball loss has only been personally resolved by having an actual defensive player attacking the ball as they would in a game scenario. To combat this problem, the thesis project presented in this paper proposes building an agile robot with machine vision processing techniques and sensor

fusion to suitably track an offensive player in order to maintain linearity between the player and the goal target, while always properly facing the player, and keeping an appropriate distance from the player.

The starting layout for the game can be seen in the following image:

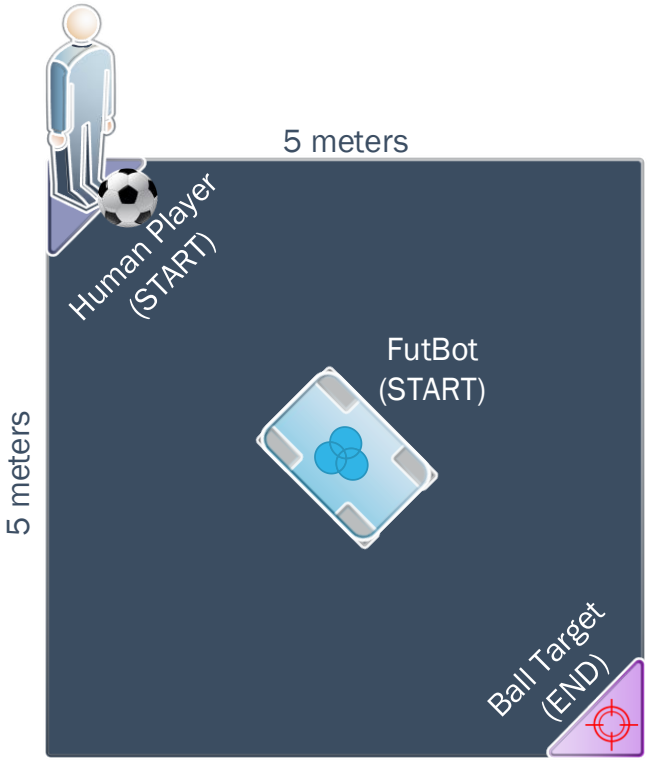



FIGURE 1: Game Layout

The objectives of the robot during the match:

- *Maintain 3-point linearity between the player and the goal target*
- *Maintain an appropriate distance from the player-defined by a circular area known as a buffer zone whose center is the human player. The human can reduce this buffer zone during approach, but the robot cannot.*
- *Maintain orientation with its Front aligned tangent to the buffer zone circumference*

In order to achieve its objectives, two cameras are used – one in the front and the other in the rear. The front-facing camera tracks the player target, while the rear-facing camera tracks the goal target. The data from these sensors are fused to create appropriate vectors defining robot strafe magnitude and direction for maintaining linearity, as well as the required robot rotation for maintaining tangency. Data collected through trials is used to verify and optimize the control system.

This project extends the previous work of Dr. Aidan Browne and Stephen Padgett: applying a vector-based drive system (Mecanum) with a vector-based sensor (Slamtec - RPLIDAR-A1: 2D Puck LiDAR) through enhancement of its Obstacle Tracking and Avoidance System applied for a specific purpose.

The thesis is demonstrated through a constructed proof-of-concept robotic system demonstrated against a player of reasonable ability and analyzed for validity with recommendations for future enhancement.

## 1.2. State of the Art [Literature Review]

To determine the best approach and verify the appropriate path for this project, many resources have been referenced to identify reliable systems and best practices as they apply to the various subject areas covered by this project:

- Sensors and Vision Systems for Localization
- Sensor Fusion
- Safety, Mecanum Drive System Control
- Vector-based Navigation

- Measurement Uncertainty
- Soccer-specific Robotic Systems

### 1.2.1. Sensors and Vision Systems for Localization

Critical to this system's implementation was determination of sensor choice and localization methods. Machine vision was the obvious choice for drive control as it is widely used for object recognition, 3D positioning, and target tracking. For this system, the robot must track a static object (the goal target) located behind the robot, and a dynamic object (the player target) located in front of the robot. The locations of the targets relative to the robot are used to create the appropriate lateral movement (left/right), direct movement (forward/backward), and rotation (clockwise/counter-clockwise).

Looking at general sensors used in locomotive systems by Patole et. al provided an in-depth overview of common technologies – including LiDAR, Ultrasound, and Cameras. Common Signal processing techniques were also provided for measuring the performance of differing telemetry parameters. This article was used as a reference and verification of sensor choice since it provided varied resources of existing research, methods, and techniques (Patole, Torlak, Wang, & Ali, 2017). Ultimately, the simplest sensors that suit the application objectives should be used, and for this robot, the simplest sensors were deemed to be two monocular cameras: one front-facing, and the other rear-facing.

Further knowledge of sensor systems specific to locomotion - specifically autonomous guided vehicles (AGVs) was my next logical step for discovery. The paper by Luettel et. al. was a survey that included several successful autonomous system approaches – highlighting commonalities and differentiators of proven AGVs. They covered many of the current technologies used to track humans and motion as well as static obstacles. For static obstacles, an occupancy map is commonly used. Since humans and other moving objects are capable of moving any direction at varying speeds, often predictive state estimation techniques are used such as recursive Bayesian Filters or most commonly the extended Kalman Filter – these estimation techniques could certainly be implemented in future work on this system.

Detecting the movement of these humans or objects can be done via optical flow – checking anomalous differences between monocular image frames. The robotic system discussed in this paper uses a simplified differential optical flow technique, where only a single point (the centroid) is compared to the image center during each calculation loop, normalized, and multiplied by a gain to exhibit wheel movement as intended. To do so the frames are turned into a binary image using color thresholding. Then for each frame, the histograms of mean pixel values and centroid calculations are used to determine the *size and location of the player target with the front-facing camera*, and *the location of the goal target with the rear-facing camera*. (Luettel, Himmelsbach, & Wuensche, 2012).

Before deciding on the two camera solution mentioned previously, I initially conceived of the system using LiDAR (Light Detection and Ranging), as well as Stereo

Vision, and Thermal Vision for my system. While considering these sensors, I consulted a paper by Song, Choi, & Kim. This paper demonstrated the use of two-dimensional LiDAR and an RGB-depth camera - providing a novel technique to determine the location of a moving target via tracking algorithms, depth of field, and extrinsic calibration. The advantage of their proposed system is redundancy to eliminate the failure mode caused by an incidental failure of a single tracker. Ultimately neither LiDAR, Stereo Vision, nor Thermal Vision was used on the final robot, so this direct approach was not useful for my final system.

According to Cadena e.t al, Simultaneous Localization and Mapping (SLAM) has been one of the most reliable methods for achieving localization in an unstructured environment research topic has been a key one for the past several decades. Several research groups have provided in-depth comparisons of existing approaches and provided open-source codebases for implementation by researchers (Cadena et al., 2016), (Mur-Artal & Tardós, 2017), (Chow, Lichti, Hol, Bellusci, & Luinge, 2014), and (Endres, Hess, Sturm, Cremers, & Burgard, 2014).

Also, prior to determining the final control system and sensors, I anticipated that the environment may need to be mapped in order for the robot to have knowledge of its surroundings and to know the difference between static and transient or dynamic obstructions – so I explored top SLAM approaches. However, upon further evaluation of the game setup and robot objectives, I determined that the environment for which this robotic system operates is certainly structured, where surroundings and setup should be known and well-controlled.

If the robot were to be used in an uncontrolled environment for future work, of these approaches, the most effective localization method seemed to be the ORB-SLAM2 system – which according to authors provided state-of-the-art accuracy by measuring the 29 most popular public sequences (known as the KITTI visual odometry benchmark). Their system applied to monocular, stereo, and LiDAR-based systems, using iterative closest point (ICP) for bundle adjustment (BA). Other approaches I studied were similar in their use of ICP – the minimization of the square of the errors between two corresponding entities. For instance, using ICP and Kalman Filters (KF) – which are the most popular filter type for robotics (Chow et al., 2014); or use of ICP and general purpose keypoint detectors with ORB keypoints providing minimum processing time at the expense of a small increase in error (Endres et al., 2014). To improve the algorithms that were used in the control system of the final robot prototype of this thesis, minimizing the square of the errors may have provided a faster response to stimuli, rather than just minimizing the error itself as was implemented.

The main advantage of the ORB-SLAM2's implementation over the others was their focus on real-time use with the ability of lightweight localization mode – visual odometry tracks for unmapped regions and keypoint matches for comparing local mapping to an underlying global calibration map. This allowed their system to quickly detect when sensors returned to existing mapped areas while correcting accumulated error and re-localizing the camera after any tracking failures – caused by sensor occlusion, aggressive motion, or system reset.

Even though SLAM was ultimately unused in this project, I knew that my control system must also process image data quickly and regain appropriate tracking of the player and target even after gross loss of visibility. Initially, I considered using an underlying global map (with the five square meter layout and known goal location) combined with a compass so that the robot would always know its heading and be able to regain location between the player and goal if either target were lost. This global map could also be considered the occupation grid that was indicated as commonly used in AGV applications by Luettel et. al. However, in the final prototype, all that was necessary in order for the robot to achieve its objectives were the two cameras – one in the front and one in the rear with no map being necessary.

Other non-SLAM approaches were also assessed. For instance, the physical principles considered by Lin and Shih including mass eccentricity and Lagrange dynamics equations proved too complicated to actually be useful. (Lin & Shih, 2013).

Tong and Chen's system was also studied for potential use. They determined effective methods for reconstructing focused images from multiple sources using Non-sub-sampled Shearlet Transforms (NSST), which decomposed images into low and high frequency components. The initial filter algorithm used to eliminate noise used a similar approach – by using a fast-Fourier transform (FFT) and removing high-frequency components above a particular threshold, however, this approach proved to be more costly in terms of processing time versus the morphological opening method (also known as shrink and grow; or erosion/dilation) that was eventually by both sensors.

Similar to the multi-focus fusion algorithm described above, Stauffer and Grimson proposed using multiple cameras to passively observe moving objects to learn patterns of activity. Motion tracking, camera coordination, activity classification, and event tracking were used to accumulate information of duplicated activities using an adaptive background subtraction method. This approach of using binary tree classification was not useful for my project (Stauffer & Grimson, 2000).

Another novel approach to vision feedback was provided by (Mezouar & Chaumette, 2002). In this approach, they created control loops for path planning and control via image-space with specific reference to targets remaining in view, calibration parameters, shape of target, and dimensions of target. The experimental results provided confirmation that their approach was successful – however, this was another paper that proved to be too complicated for legitimate use in my system.

More interesting was the approach of (Chumerin & Van Hulle, 2008) which used two comparative data streams to track independently moving objects and recognition of those objects. Their proposed system used stereo-vision, LiDAR, and optical encoders for actively determining proximity, relative speed, and relative acceleration. Similarly, for this robot, two data streams are used to track two independent objects: a single static object, and a single dynamic object however, this is accomplished using only two cameras (one for each object) in order for the system to achieve its objectives with minimum complexity.

### 1.2.2. Sensor Fusion

Sensor Fusion as defined by the Joint Directors of Laboratories (JDL) workshop [1]: "A multi-level process dealing with the association, correlation, combination of data and information from single and multiple sources to achieve refined position, identify estimates and complete and timely assessments of situations, threats, and their significance." The Joint Directors of Laboratories (JDL) Data Fusion Working Group created the process model for data fusion, and a Data Fusion Lexicon (Hall & Llinas, 1997). See the appendix for these definitions.

At its most basic definition, sensor fusion involves incorporating the information acquired from multiple sources in order to increase known information of a specific target entity surpassing what is achievable by a single sensor alone. Hall & Llinas describe the true goal of multi-sensor fusion is to make accurate and appropriate characterizations of a target entity in its surrounding environment through redundant or complementary combinations of data. The robot presented in this paper must have awareness of what is occurring in front of it in order to react to the movement of the player, as well as the awareness to the rear and its location relative to the goal target, in order to keep itself between the player and the goal target. The two sensors used: the front-facing camera and the rear-facing camera complement each other since a single camera alone would not have the same bi-directional awareness and their combined information is processed for defining action or inaction of the system.

Sources of information available for sensor fusion include *A Priori* information: the game layout, expected lighting conditions, and game surface; dynamic information:

the images from the webcams. The human-computer interface also provides useful information and includes the light indication system, the start button, the emergency stop button, and optional tele-operational control. Data from the robot's vision system is pre-processed to calculate centroids and histogram mean values

### 1.2.3. Safety

Ensuring safety of a user is undoubtedly any automated system's first priority – Wu, Bateman, Zhang, and Lind provided a framework that covered the fundamentals of applying monitoring systems to ensure safety during robotic use with the ability to flag abnormal situations including methods for determining the system process are discussed via circular cascading (cyclical) actions. They provided a use-case via simulation and demonstrated the ability to discovers system process errors using their methodology providing confidence that their approach was believable (Wu, Bateman, Zhang, & Lind, 2018). Ultimately no monitoring systems were used on this robot since a simpler system was implemented.

Zanchettin et. al provided some best practices when considering shared environments between humans and robots focusing on ones in which they are expected to work in tandem. A real-time kinematic control system was proposed for ensuring human safety as a top priority and has been validated with a dual-arm (7-DOF per arm) robot (Zanchettin, Ceriani, Rocco, Ding, & Matthias, 2016). This was an important read since my system is also real-time and discovering working solutions was critical to safety assurance. The best practice used in this robotic system was the implementation of an

emergency stop (E-Stop) button, which immediately causes the wheels to stop all rotation

Similarly, Haddadin et. al provided additional methods of creating safe robot interactions with humans. Context-independent collision events were considered to identify how a robot would safely interact with its environment while performing expected tasks (Haddadin, Luca, & Albu-Schäffer, 2017). For incidental collisions with the human player and my robot system, I added padding to the outer perimeter of the chassis and limited the top speed to one deemed suitable for safe robot interaction. Since the game layout incorporates a relatively small area (5 meters by 5 meters), and the speed of the robot is limited, incidental injury the human player is unlikely.

#### 1.2.4. Mecanum Drive System Control

Physical parameters of concern affecting the mecanum drive system were considered, but ultimately proved not useful in the development of my robot's control algorithms. The three parameters of concern identified by Conciecao et al. were viscous friction, Coulomb friction, and moment of inertia. Initially, I planned on combining those physical parameters with state-space representation of an armature-controlled DC motor to provide a robust method of linear and rotational control but ultimately chose a simpler route through a mecanum mixer model where throttle values are input for control of direct, lateral, and rotational movement(Conceição, Moreira, & Costa, n.d.).

The most useful information for developing kinematic models came from a master's thesis describing the inverse kinematic control and rotational dynamics

involved for building an appropriate control system of a KUKA youBot which incorporates a similar mecanum drive. The KUKA youBot is an industrial experimental robotic platform using mecanum wheels to achieve omnidirectional movement. In this paper, the USARSim (Unified System for Automation and Robot Simulation) was used to simulate the physical principles involved in the action of the robot. The results of the simulation verified the experimental values with a further need for optimization – illustrating how this free software may provide sufficient expectation of realistic action if set up appropriately. For my uses, a very simple kinematic model is all that was required ("A KUKA youbot simulation in USARSim. Freddy de Greef—PDF," n.d.).

To develop the control system various approaches were considered. (Vidal, Shakernia, Kim, Shim, & Sastry, 2002) used both unmanned Aerial Vehicles (UAVs) and unmanned Ground Vehicles (UGV) placed into teams of pursuers and evaders using game theory using two greedy pursuit policies: local-max and global-max. Their solution proposed places emphasis on both independent and collaborative approaches using pursuit policy, map building, communication, navigation, sensing, and control. The approach may prove useful in the future when an entire robotic soccer team is developed extending the approaches of the ones used in my project (Vidal et al., 2002).

Motion planning for general terrains using geometric paths and vehicle speeds to minimize motion time per the various system constraints: dynamics, topography, obstacles, mobility. The proposed solution by (Shiller & Gwo, 1991) used a cost function to determine a best path via spline geometry and is demonstrated on a simple dynamic model. In my system, a cost function, or tracking score is used to determine appropriate

weighting where priority on tracking the human player is placed above tracking the goal target when determining the appropriate throttle values for direct, lateral, and rotational movement for the optimum state of action during real-time activity.

### 1.2.5. Measurement Uncertainty

The true value of a measurement is never exact, so building a system that does not rely on exact measurement but still achieves objectives is desirable. In this robot's case, the interaction with the player is always dynamic, so general knowledge of direction and relative speed are all that is necessary to prompt movement of the robot to consistently aspire to maintain its objectives of facing the human player, maintaining a buffer zone, and placing itself between the player and the goal target. The centroid calculation and mean histogram value are used for this movement, where their true values have some level of uncertainty surrounding those used.

Distributed detection systems with multiple sensors compound the accumulation of measurement error. Many of these can be seen in (Tenney & Sandell, 1981)'s paper, which is decades old, but still relevant. They illustrated the issue of using multiple sensors succinctly, by providing the unknowns that can affect readings, and methods of determination are presented in detail.

Sources of uncertainty in this application are the location and speed of the human player, the orientation of the human player when the image is taken (side-facing or front-facing), the friction of the play surface, the noise from the surrounding environment, lighting conditions, battery level, mecanum wheel rolling resistance,

controller-laptop data transfer rates, and so on. There are many sources of uncertainty that come into play for this application, but the control system was created to be able to handle this uncertainty.

Another resource used to define optimal approaches, was Viswanathan and Varhay's survey of common methods including: Neyman-Pearson (NP) Criterion and Bayes Formulation for dependent architectures, and Likelihood Ratio Test (LRT) for independent architectures provided a solid foundation on which to minimize localization errors. The most helpful sections of this paper provided the tradeoffs for simplicity vs. complexity of computations and details of the most common generalized models as the robot's development progressed, the initial high complexity solution that was theorized proved unnecessary and ultimately led to the development of the simple of minimizing error between targets being tracked and their location relative to the image frame's center.

Uncertainty for robotic control systems was discussed in detail by (Medina & Hirche, 2018) with information based on biological systems. Their relation stochastic control via statistical methods provided a valid approach for minimizing rotational and translational error. Their approach used costs (weights) which were applied to sources of uncertainty to increase the robustness of the control system for both linear and non-linear settings (Medina & Hirche, 2018). As mentioned previously, the tracking weights varied between the human player and goal target. This approach in conjunction with the known map construction of the environment proved quite successful for my purposes of autonomous action. In my case, these weighted systems were termed as

priorities for a particular action and were based upon the error of the sensor measurement and the target locations within the image frame.

### 1.2.6 Soccer-Specific Robotic Systems

A conference paper from Badung State Polytechnic University in Indonesia on the topic of robotic humanoid soccer provided verification of some approaches used in this application from their analysis of related studies: namely using HSV filtration techniques to determine objects from images and rotation of the robot to maintain perpendicularity to the tracked object by keeping that object in the middle of an image. This is what was done for my robotic system. Their actual research was not useful for my project, where they extend the previous work in the field by tracking multiple objects in a single image and calculating target position with an artificial neural network without the requirement for perpendicularity since the images used in the analysis are taken by the goalkeeper. (Awaludin, Hidayatullah, Hutahaean, & Parta, 2013)

Non-uniform lighting conditions for machine vision can cause difficulty with algorithms requiring color classification. Research by (Dhanapanichkul & Chongstitvatana, 2005) provides a method of approaching shadow compensation for soccer robot teams from a programmatic perspective. They compensate by determining HSV values on a per-pixel basis, comparing each value to the field color to develop albedo ratios to the field color and other colors of interest, then adjusting the pixel-level brightness before component labeling and classification. In my application, I overcame this issue by adding LED floodlights to the front and rear of the robot. The effect of

these lights was to provide reliable HSV readings of the colors of interest: in my robot's case, those colors are red for the goal target and green for the human player.

### 1.3. Mecanum Drive

This specialized omnidirectional drive system is a relatively modern invention as it is less than 50 years old. It was originally devised in 1972 by a man named Bengt Erland Ilon who worked for the Swedish company Mecanum AB (hence the name of the drive system).

The key to this system is the mecanum wheel, which is a wheel surrounded in its circumference by sets of parabolic rollers typically angled at 45 degrees relative to the axis of rotation of the wheel. This 45-degree angle can be in either direction, clockwise or counterclockwise, providing two possible wheel configurations which are mirrored assemblies of one another – referred to as the left and right mecanum wheels. Other angles are also possible, but for this thesis, the 45-degree type wheels are used. See the following images for an illustration of both wheel configurations.

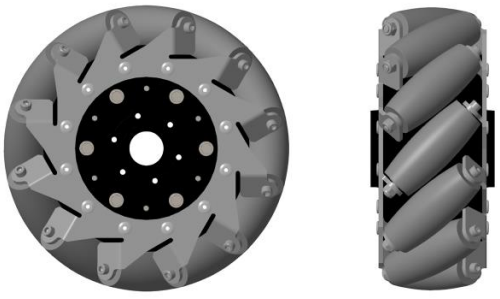


FIGURE 2: Right Mecanum Wheel

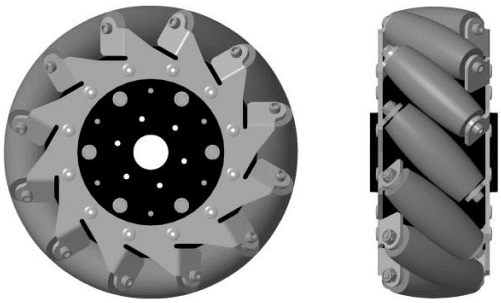


FIGURE 3: Left Mecanum Wheel

When a specific combination of the wheel configurations is used in a 4-wheeled robot, the robot inherits the special ability for strafing – moving in any direction while maintaining consistent heading orientation. This is also known as holonomic drive – where the controllable degrees of freedom equal the total degrees of freedom (x, y, ψ) where x and y describe the two-dimensional movement in the ground plan, and ψ describes the angular rotation around the z-axis running from the ground plane up through the center of the robot. Due to the angle of the rollers, the force vector created by the wheel during its rotation allows for the unique ability of the robot to move in any direction (strafe) without the need to turn its front wheels – allowing the robot to face the same direction while moving in any vector direction – since all degrees of freedom are controllable. Please see the following images for the force vector detail. ("Holonomic (robotics)," 2019)

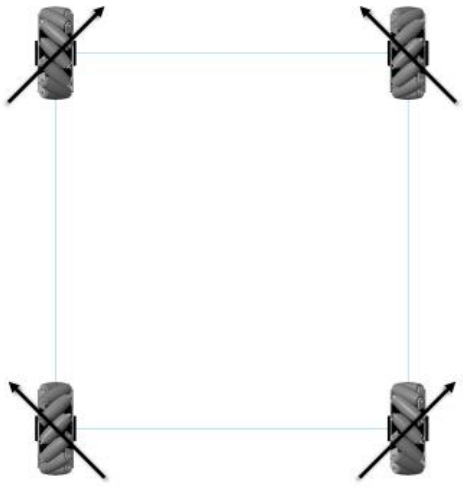


FIGURE 4: Mecanum Configuration with Force Vectors for Forward Movement

Imagine the figure above, where the wheels of the robot are all rotating forward in the same direction: combining the force vectors of the front and rear wheels for the left and right side provides the forward-facing vectors seen in the following image.

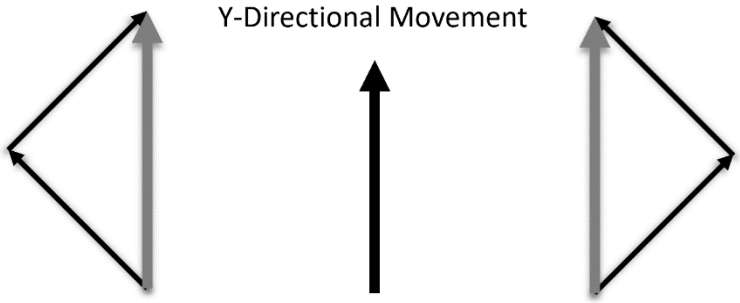



FIGURE 5: Forward Movement Vector Addition

Next, imagine that the wheels on the left side of the robot are rotating forward, but the wheels on the right side are rotating backward: combining the force vectors of the front and rear wheels for the left and right side provides clockwise rotational movement seen in the following image. Yaw angle, $\psi$, is the rotation around the Z-axis

which runs from the ground plane interacting with the wheels through the center of the robot toward the sky.

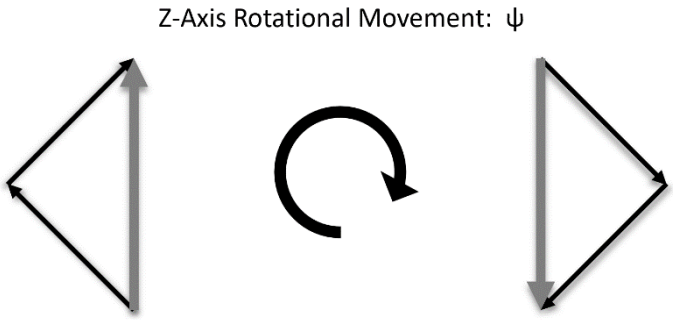



FIGURE 6: Clockwise Rotation Vector Addition

Finally, imagine the scenario where the front left and rear right wheels are rotating forward, while the front right and rear left wheels are rotating backward.

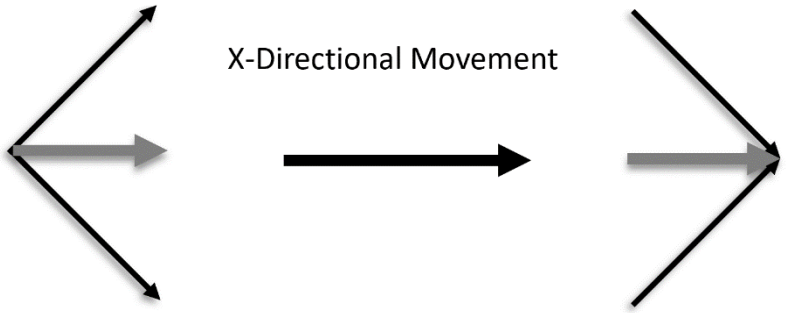



FIGURE 7: Lateral Movement Vector Addition

From the images above, it should be apparent that by controlling the forward and backward rotation of each wheel independently, the robot is able to allow for yaw rotation of any angle as well as any vector movement in the X-Y plane. Therefore, the controllable degrees of freedom are X, Y, and ψ as shown in the image below.

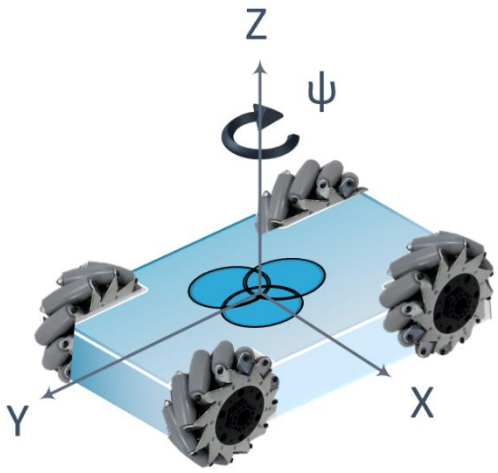



FIGURE 8: Controllable Degrees of Freedom for Mecanum Drive (X,Y,ψ)

## 1.4. Technologies Investigated

As mentioned in the Literature Review section, before simplifying the final sensors used to only two monocular cameras: one front-facing, and one rear-facing, many other technologies were investigated to determine their usefulness and validity in this application. These technologies included LiDAR, Stereo Vision, and Thermal Vision.

My initial concept for the robot system was to use the LiDAR's time-of-flight characteristic to determine the distance of the human player from the robot – in order to maintain an appropriate buffer zone per the objectives. Components tested for use were the RPLiDAR A1M1 and A3M1. These can be viewed in the following figure:


FIGURE 9: LiDAR Components Considered

The A1M1 is a belt-driven, low-cost 2D LiDAR system with an 8-meter range that was used on the original mecanum platform. I was never able to achieve communication with this system with LabVIEW code used on the previous robot platform or through the software supplied by the manufacturer.

The RPLiDAR A3M1 which is an upgraded version of the A1M1 capable of achieving up to 16000 samples per second with a 25-meter range was also explored for use. Communication with this device and the controller is performed with a USB serial TTL converter communicating at 256000 bauds per second. I was able to acquire adequate data from the software supplied by the manufacturer, however, when trying to control via LabVIEW I was unsuccessful. After many weeks of creating a LabVIEW state machine with the assistance of a colleague to handle the massive amounts of incoming data from this component, we were never able to eliminate the noise in the system to make it useful for implementation. Additionally, I was able to use the front-facing camera to determine the distance of the human player from the robotic system through estimates derived from the mean histogram value of the images – making the

LiDAR ultimately unnecessary. An example of the noisy code output is provided in the following image.

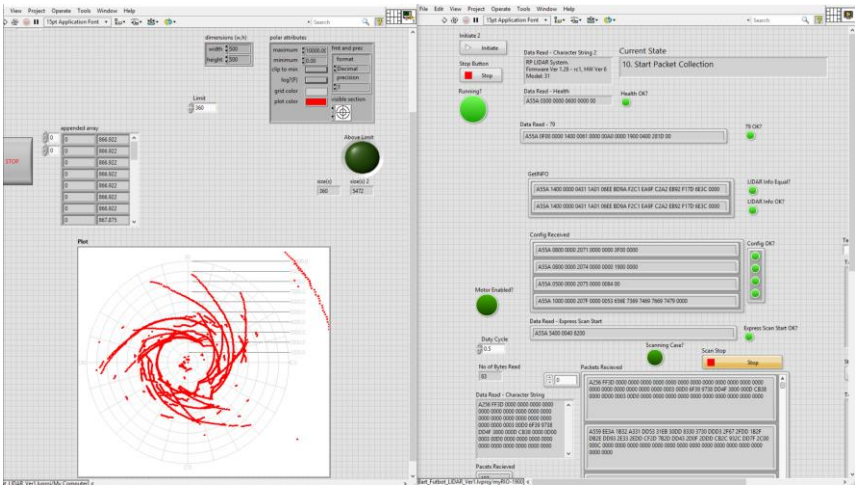


FIGURE 10: LiDAR State Machine Noise

The A3M1 LiDAR was left mounted for future implementations of the robotic system and is shown secured to a plate that fits into the inner diameter of the PVC and secured with a set screw so that internal wiring could be accessed if necessary.

Stereo Vision was also considered for use in this system. The original concept involved using images taken from two cameras spaced the same distance as the human eyes to generate the angle of the human target in relation to the robot as well as provide redundant depth analysis for the LiDAR. After mounting the cameras as seen in the following image, I was never able to communicate with them via LabVIEW.


FIGURE 11: Stereo Vision Mounting

Through much effort, I discovered that the reason for their lack of communication is that the laptop used for code development had a Windows 10 operating system and the since the cameras were an older model: LifeCam VX-3000, the drivers were never updated, not available, and would have never worked with the laptop. The difficulties experienced with these cameras actually worked to my benefit since as I was discovering new webcams for use, I realized that I could acquire the same angular information of the targets with only a single camera (one for the front of the robot and one for the rear), by comparing subsequent frames to determine change in position of the targets, rather than comparing simultaneous images of a pair of stereo cameras. The effect was the overall simplification of the robotic system.

The last technology that proved unsuccessful for use with this system was Thermal Vision. The original concept involved using the heat signatures from thermal images as pre-globbed frames for simply identifying the human target. The idea was to use the thermal image with a pre-trained neural network – either a HOG (histogram of

oriented gradients) or Haar Cascade for feature detection to identify a human's torso. A proof-of-concept system was built using a FLIR Lepton and Raspberry Pi 3 coded with Python. This system can be seen in the following image:

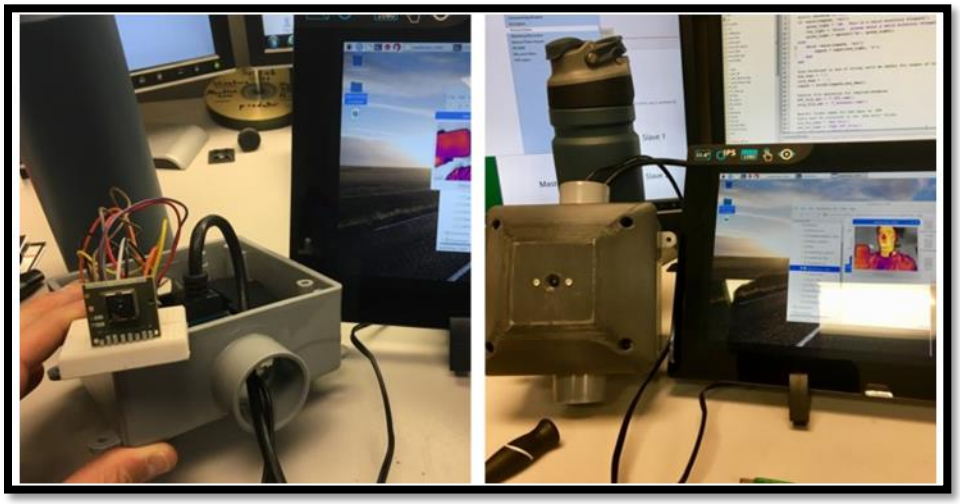


FIGURE 12: Thermal Imaging System

The unit was functional but ultimately determined to not be useful for this system since slight jostling caused the sensor to unseat from its associated PCB. This caused a red dot to appear as seen below. The fix for this error was to open the enclosure, pry off the sensor with a set small set of plyers and re-seat the sensor in its PCB. This was just not a tenable solution for a robot with dynamic movement and moderately high vibration from the mecanum wheels.

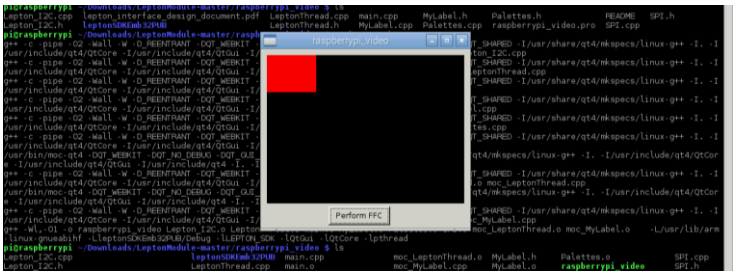



FIGURE 13: Thermal Vision Red Dot Error

A summary of the technologies investigated for this project can be seen in the following table.

TABLE 1: Technologies Considered

| ***Technology*** | ***Specific Component*** | ***Was it Used?*** | ***Why or Why not?*** |
|---|---|---|---|
| *Thermal Vision* | FLIR Lepton | No | Sensor would consistently show error - requiring hardware to be reset on the PCB. |
| *Stereo Vision* | 2 - LifeCam VX3000 | No | Cameras not compatible with Windows 10. Ultimately could achieve the same information of relative target angle via monocular vision. |
| *LiDAR* | RPLiDAR A1M1<br>RPLiDAR A3M1 | No | A1M1 could not communicate via serial connection - may have been damaged previously. A3M1 - was not able to create useful State Machine with LabVIEW to eliminate noise from the signal. Would have been easier if ROS were used. |
| *Machine Vision* | Logitech C270 HD | Yes | Cameras were compatible with Windows 10 - provided enough resolution at the required frame rate. |

## 2. SYSTEM ARCHITECTURE

The following images provide front and rear views of the robotic system, lovingly referred to as the FutBot:


FIGURE 14: FutBot Front View


FIGURE 15: FutBot Rear View

An existing mecanum chassis platform was modified to suit the goals of this application: to act as an impediment to a dribbling soccer player by seeking to keep itself between

the player and the goal, maintaining an appropriate distance from the player, and seeking to always face the player

The chassis of the robot is relatively simple and comprised of various aluminum channel – perforated C-channel, L-channel and box channel secured by steel hardware sized according to the specifications of the following table:

TABLE 2: Mechanical Parameters of Robot

| | |
|---|---|
| Base Dimensions: | 30.25"W x 32.25"L |
| Lower Platform Height: | 7" |
| Upper Platform Height: | 17" |
| Vision Height: | 46.5" |

The original base dimensions, as well as lower and upper platform heights, were not modified from the original chassis design since most of the major components remained the same. The height for vision was chosen to be suitable for a wide array of player heights. With the average player height of 71.5” according to the International Centre of Sports Studies’ analysis of the 2009 soccer labor market, 46.5” was estimated to be at the level of a standard player’s solar plexus, meaning that during dynamic movement, the camera would still have a suitable view of the center of mass of the human player.

Secured beneath the Lower platform are the mechanical components of the Mecanum Drive System, which is comprised of the following major components:

- 8” Mecanum Wheels – quantity of 4 (2-right; 2-left)
- 2.5” CIM Motor – quantity of 4
- ToughBox Mini Gearbox – quantity of 4
- US Digital E4P Optical Encoder – quantity of 4
- Talon SRX Motor Speed Controllers – quantity of 4

The Talon SRX motor speed controllers, which can also be considered part of the drive system are located between the lower and upper platforms where the majority of electrical components are placed. These drive system components were supplied with the original mecanum chassis and found to be wholly suitable for the purposes of this application. Detailed technical specifications for these components can be found in the Appendix.

The following image illustrates the major components of the Mecanum Drive System:

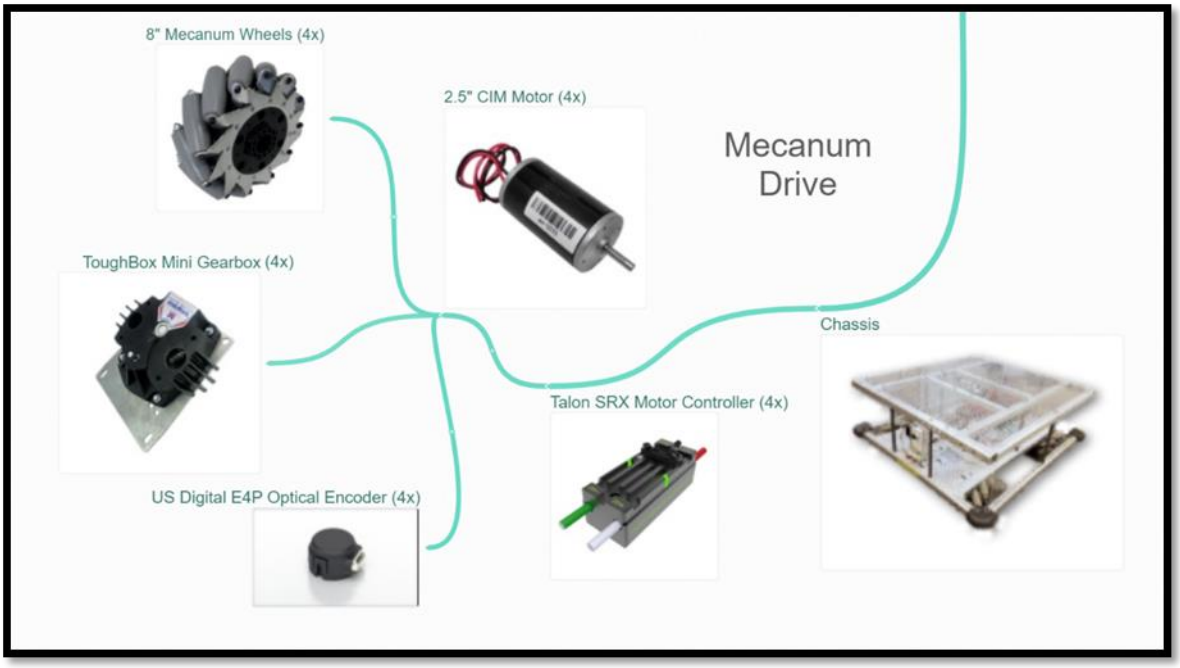



FIGURE 16: Mecanum Drive System Components

The Power System, which is housed between the lower and upper platforms is comprised of the following major components:

- Battery
- Power Distribution Panel (PDP)

- 120A Thermal Circuit Breaker
- Voltage Regulator Module (VRM)

These components were also supplied with the original mecanum chassis and found to be suitable for this application. The battery capacity of 18A-h was found to be useful for more than 30 minutes of gameplay.

The following image illustrates the major components of the Power System:

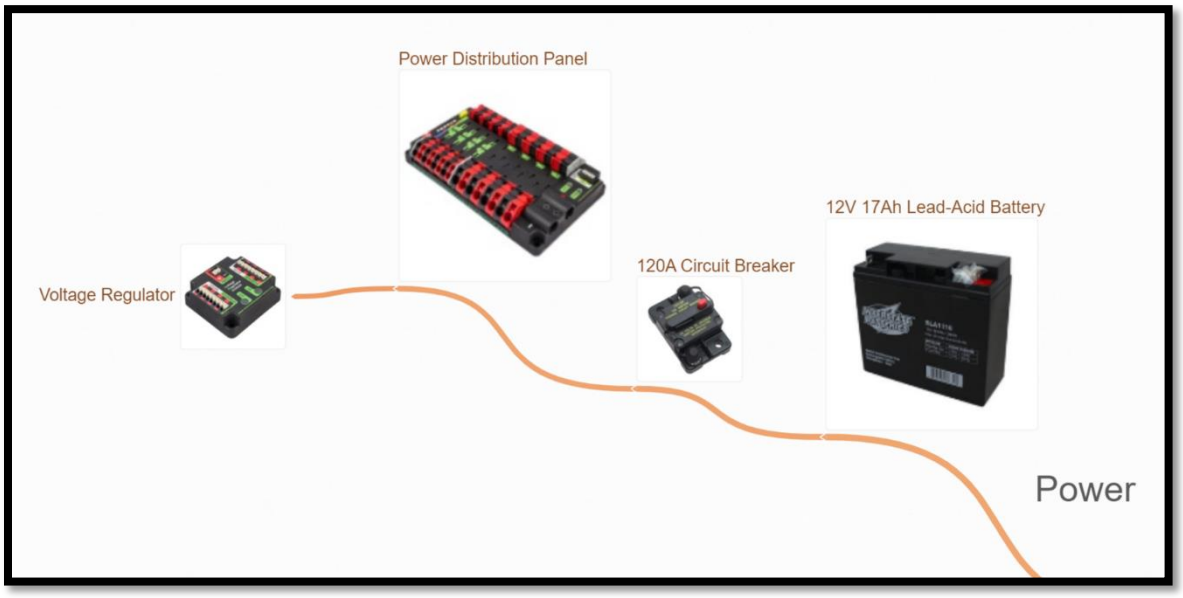



FIGURE 17: Power System Components

The main controller for this Robot is a myRIO 1900 by National Instruments. This reconfigurable Input/Output device (RIO) allows for rapid development via National Instruments LabVIEW – a visual data flow programming language. Onboard processing is performed by a dual-core ARM Cortex-A9 processor or Xilinx FPGA. The RIO has 3 separate channels (A, B, C) with 10 analog inputs, 6 analog outputs, and 40 digital input/outputs. The device also contains a single USB 2.0 port which can be expanded with a powered, compatible hub. Also of note is a 2.4GHz wireless radio that creates its

own hotspot for communication and reprogramming wirelessly by a host computer running LabVIEW. This allows some processes to be off-loaded to a host computer to reduce the burden on the controller. There are several other tools available on-board the RIO including pushbuttons, LEDs, audio output, and an accelerometer.

The cameras used for the vision processing were two Logitech C270 HD webcams that provide a 720p resolution at 30 frames per second with fixed focus and a 60-degree field of view. Communication with the controller is through a USB 2.0 hub connected to the single port available on the myRIO 1900 controller. The minimum criteria for the vision system were images of 160x120 pixels at 30 frames per second.

The following image illustrates the major components of the controller and vision systems:

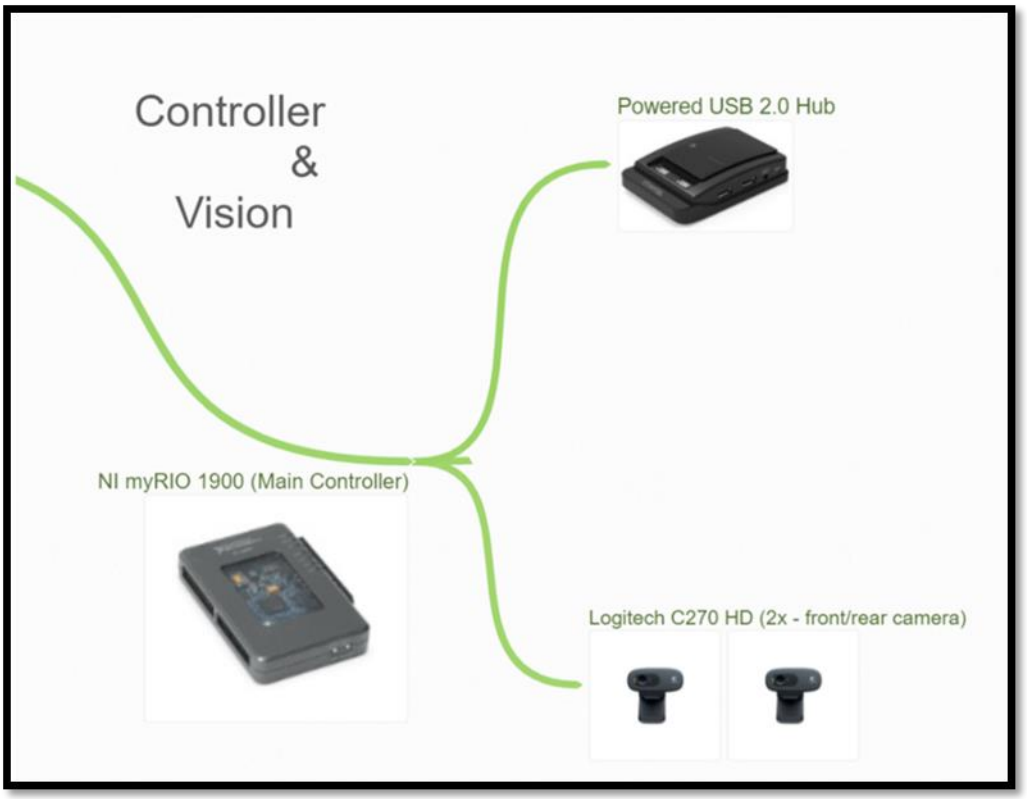



FIGURE 18: Controller and Vision Hardware

In order to reach the required heights of the Vision System outlined in Table 1, 1” PVC pipe was used to create a seamless cable path and mounting location for the hardware components. To ensure that the pipes maintained vertical mounting, the angle for stability was controlled by a turnbuckle constrained between two aluminum L-channels. As mentioned previously the primary chassis used in this robot was inherited from a previous project along with the mecanum drive and power system components, so the mounting technique was chosen to reduce the overall burden of reconfiguration. An example of this conduit system and component mounting can be seen in the following figure.

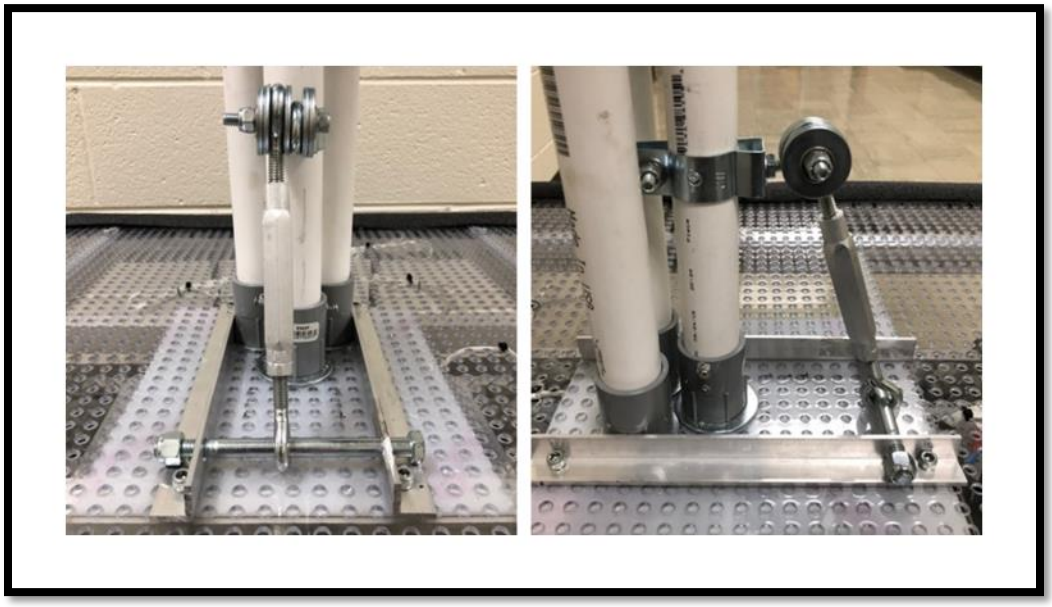


FIGURE 19: Vision Conduit System

Mounting of the cameras used a combination of off-the-shelf parts compatible with the PVC diameter, as well as a custom-designed 3D printed mounting bracket. The 3D-printed mounting bracket was created to constrain the cameras for front and rear at

the proper height and secure them during dynamic movement of the robot. This mounting technique can be seen in the following figure.

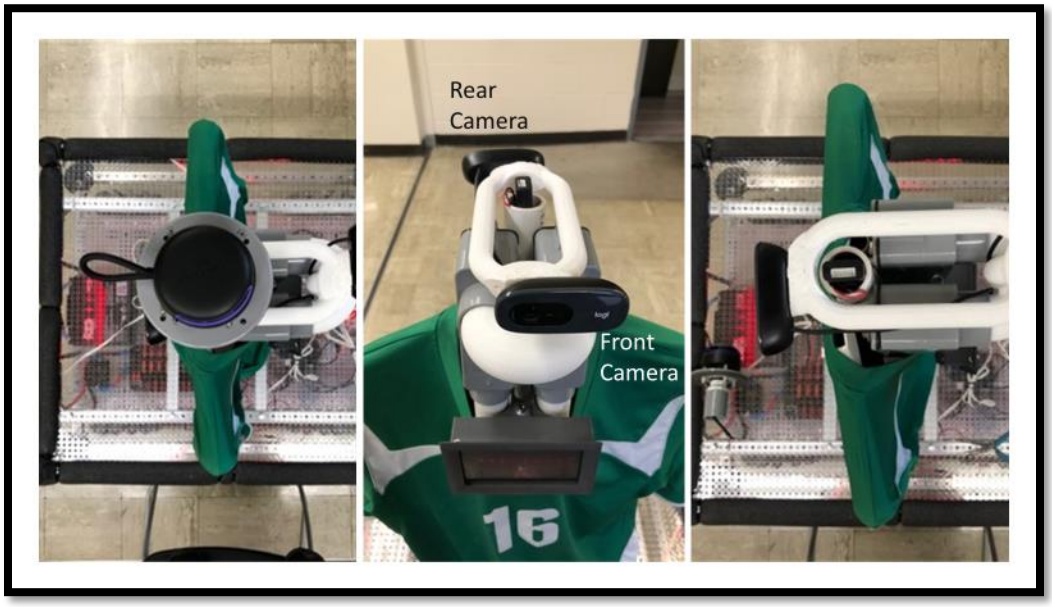



FIGURE 20: Vision Mounting

One of the objectives of the thesis to provide a means with which to communicate with the human player. This is accomplished through an LED lighting indication system and a simple human-machine interface comprised of only two buttons (Start Button, and Emergency Stop Button). The buttons and LED power supply splitter can be seen in the following image:

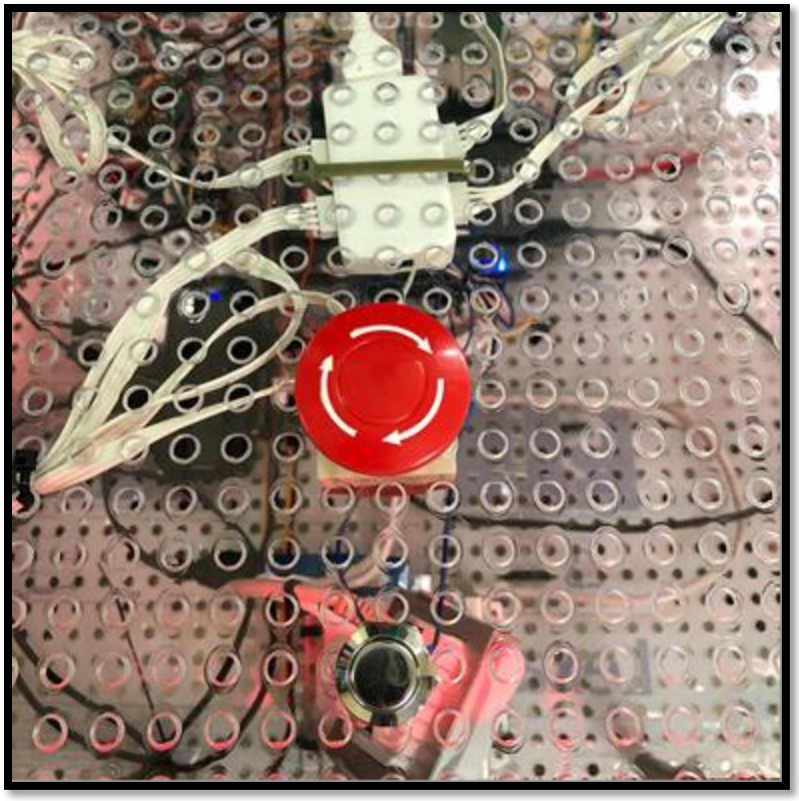


FIGURE 21: Buttons and LED Splitter

When the robot is powered on and ready to begin play, the LEDs are blue in color. The human player then presses the start button to begin play. The color of the LEDs change to yellow as the 10-second count-down to game commencement begins. With three seconds until game start, the LEDs then change their color to orange to indicate to the player that the game is about to begin. During gameplay the color of the LEDs is green. Once the timer has fully elapsed or the human player presses the stop button, the LEDs change to red to indicate that the motors are powered off.

The LED lighting system makes use of the DIODER 4-piece, multi-color lighting strip from IKEA. The original controller was removed and replaced with a NodeMCU 0.9 ESP-12 Module, which is an open-source firmware SOC (system on a chip) based on the ESP8266 by Espressif, useful as a microcontroller which can be programmed with the

Arduino IDE. This microcontroller was combined with a P9813 LED Driver to create a method of responding with appropriate RGB lighting commands via analog signals received from the myRIO controller.

A program containing the ChainableLED library header is loaded onto the NodeMCU to define the necessary variables for creating the control program: number of LEDs (1 in this case since all 4 light bars indicate the same color), data pin communicating with LED driver (digital output - D0), clock pin communicating with LED driver (digital output - D1), and signal pin communicating with myRIO (digital input – A0). A 100-millisecond delay is instantiated for loop control, as well as two values for mapping the input variable (0-1023) to analog voltage (0-3.3V). The setup function is used to initialize the LEDs, set their color to a value of zero for all RGB (Off), and begin serial communication at 115,200 bauds per second. After setup, the loop that begins upon power-up and continues until power down reads the analog signal provided by the myRIO's Channel C – analog output 0 (AO0) and the analog output express VI. The analog value received by the NodeMCU, which is between 0 and 1023 is mapped to a voltage between 0 and 3.3 by the following equation:

$$mapped\ voltage\ = analog\ value * 3.3/1023 \quad (1)$$

The following table provides the color for the given mapped voltage values:

TABLE 3: Mapped Voltage Colors for LEDs

| Mapped Voltage (MV) | Color |
|---|---|
| MV < 0.3 | Off (R=0, G=0, B=0) |
| 0.3 ≤ MV ≤ 0.9 | Red (R=255, G=0, B=0) |
| 0.9 ≤ MV ≤ 1.5 | Blue (R=0, G=0, B=255) |
| 1.5 ≤ MV ≤ 2.1 | Orange (R=255, G=60, B=0) |
| 2.1 ≤ MV ≤ 2.7 | Yellow (R=255, G=150, B=0) |
| 2.7 ≤ MV ≤ 3.3 | Green (R=0, G=255, B=0) |

The VI that controls the color on the myRIO is about as simple as it could possibly be. A ring enumerator is used for each potential color and the midpoint voltage value is sent as a single sample through the analog output express VI: off – 0, red – 0.6, blue – 1.2, orange – 1.8, yellow – 2.4, or green – 3.

The start and emergency stop button controls are also very simple. The start button is a momentary switch that is connected to Channel C – digital input 1 (DIO1) and uses a single sample of the digital input express VI to make appropriate action – i.e. indicating to the robot that the human player is ready for the game to begin. The emergency stop is a mechanically latching switch that is connected to Channel C – digital input 5 (DIO5) and also uses a single sample of the digital input express VI to make appropriate action – i.e. setting all motor throttle values to zero and changing the lighting to red. This button must be unlatched before any program can continue.

# 3. SENSOR FUSION

For this project, the depth of fusion processing does not reach far beyond Level 1. The environment A Priori, or previously known information that is used to program the object refinement parameters. The data then undergoes transformations before it is capable of being fused for robot action.

The forward and rear-facing cameras are each making feature-based inferences by thresholding specific HSV color values of a target (human or goal), and transforming that information into the target's approximate centroid location, distance from the image center, and mean value for the histogram distribution of pixel values contained within the image.

The centroid of the human player and its relative distance from the image center is used to control rotational movement. The centroid of the goal target and its relative distance from the image center is used to control lateral movement. The mean histogram value of the human player is used to control direct movement via maximum threshold. Mean histogram values for both the human player and goal target are also used with minimum threshold values to determine if the targets are within the frame. The front and rear camera information are fused at the decision level to determine the PWM signals necessary to achieve the net result of the lateral, direction, and rotational movement.

The front-facing camera is only concerned with tracking the human player. Images captured by this camera pass through an algorithm which filters the incoming

image by a pre-defined HSV threshold. The ranges of these values were derived from multiple images at various angles and lighting intensities to provide reliable identification of the unique green color of the shirt, which serves as the control variable for the player tracking model.

The output of the model is a binary image, which drastically reduces processing requirements – reducing the file size from a 32-bit (~1200 kilobyte size) to 8-bit binary (~300 kilobyte size). The following figure shows the green shirt that was used for all testing with the human player.

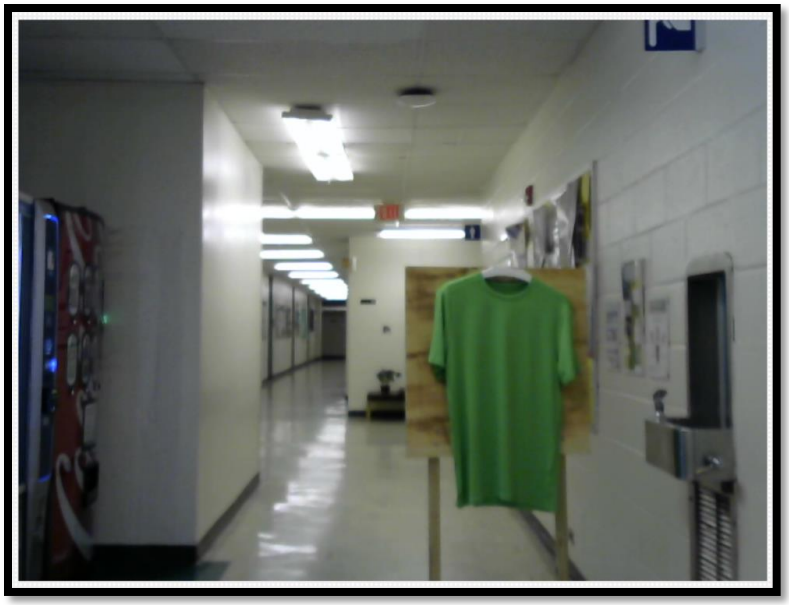


FIGURE 22: Front-Facing Camera Sample Image Initial

Optimization of algorithms involved trials of various parameter thresholds and color models. The most successful parameters for this environment are provided by the following image. For different lighting scenarios or environments, these threshold values may benefit from recalibration.

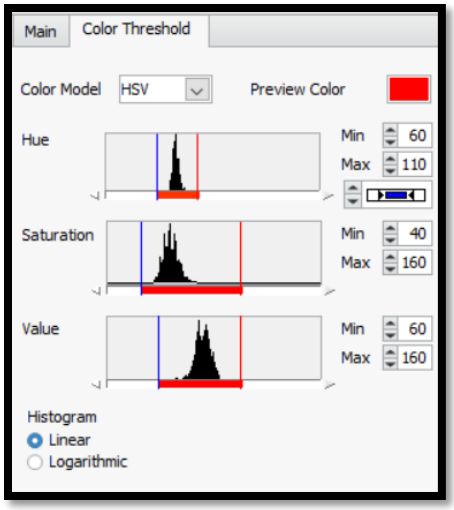



FIGURE 23: Front-Facing Camera Color Model and Threshold Values

The image is filtered to identify only the green color of the shirt and eliminate background information.

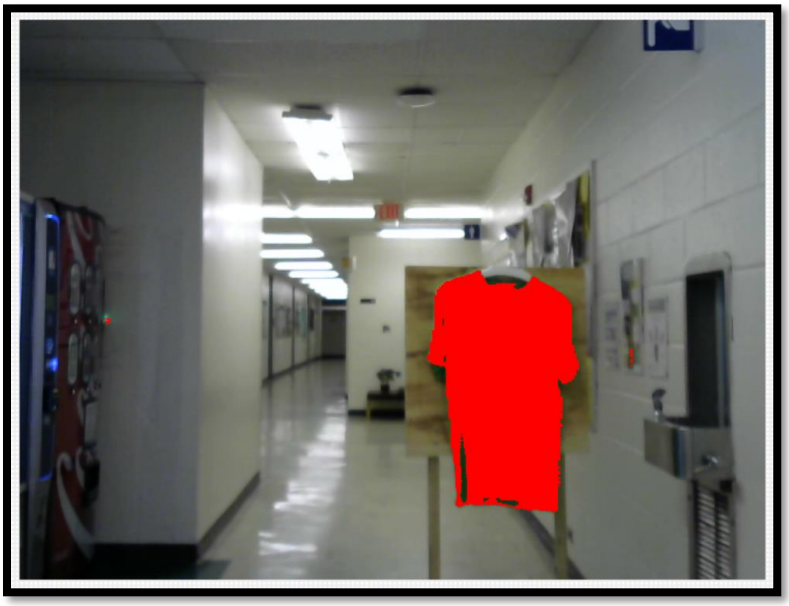


FIGURE 24: Front-Facing Camera Sample Image Threshold Validation

FIGURE 25: Front-Facing Camera Sample Binary Image

After the binary image is created, some noise still exists within the image since the threshold values will inevitably pick up some of the green colors in the background. My first effort in eliminating this noise was to use an FFT filter. The FFT filter functions by transforming the image to its frequency representation providing details about the periodicity and spatial distribution of pixel value variation. The noisy aspects of the images are attributed to higher spatial frequencies, while the ROI contains more gradual variation and is associated with lower spatial frequencies. In this case, a lowpass filter is used to truncate high frequencies. The following images detail the FFT Filter step and the resulting processed image.

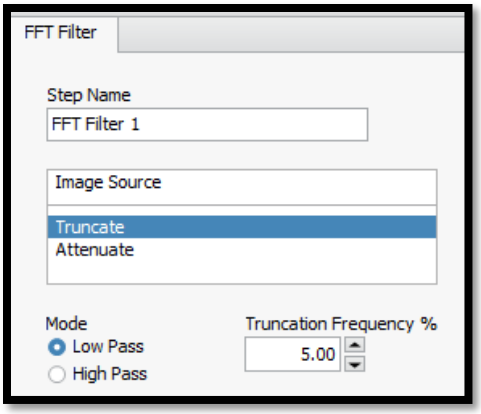



FIGURE 26: Front-Facing Camera FFT Low-Pass Filter

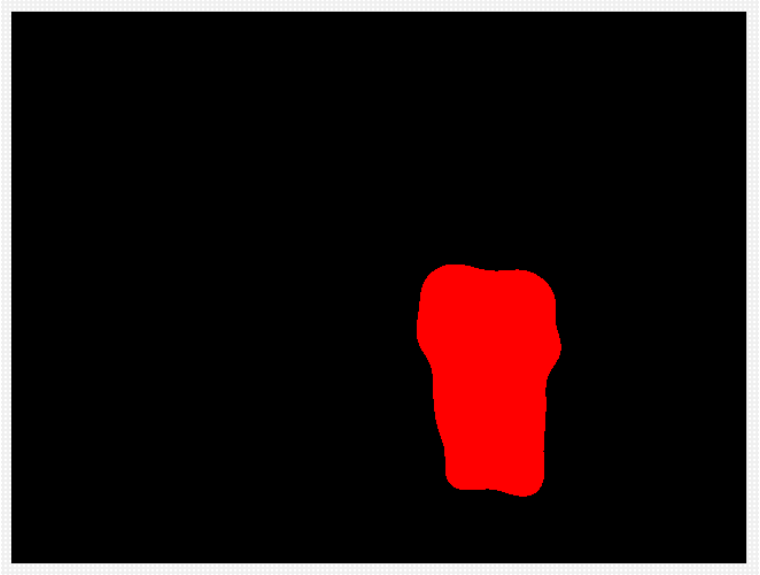


FIGURE 27: Front-Facing Sample Image FFT Filter Result

The difference between the initial sample image and the FFT filter result is a significantly smaller data sample which contains only the region of interest (the human target center of mass) with the background fully filtered.

This method works well, however, the processing cost is more onerous than what is accomplished by a binary erosion/dilation morphology method, known as morphological opening or shrink and grow technique. With the morphological opening method, the image is first eroded to eliminate smaller artifacts within the image frame

that were caught in the color threshold. Subsequently, the pixels that remain are then dilated. The morphological opening steps and results are shown below in the following images.

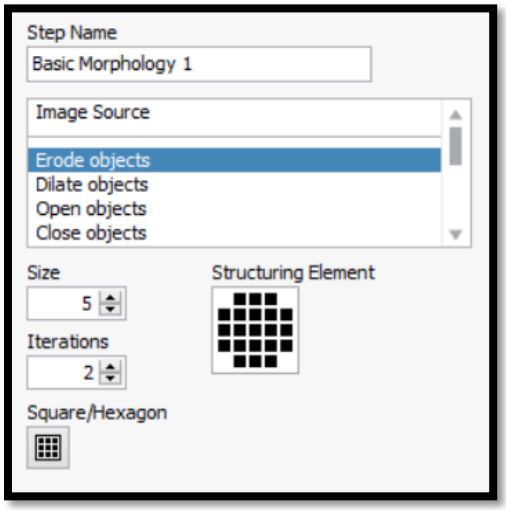



FIGURE 28: Front-Facing Camera Binary Morphology Erosion Parameters

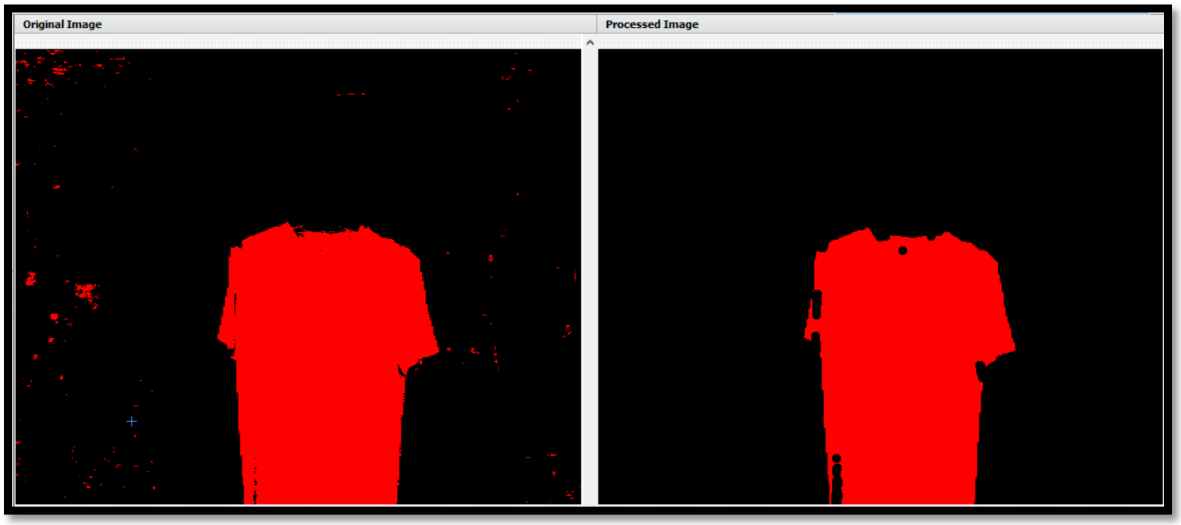



FIGURE 29: Front-Facing Sample Image Binary Erosion Result

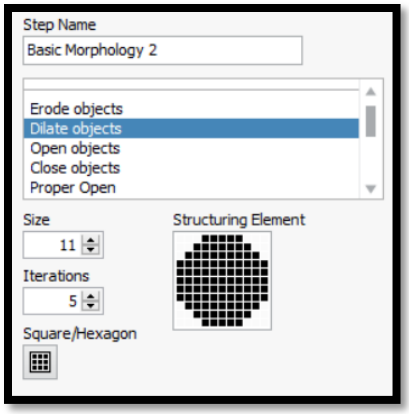



FIGURE 30: Front-Facing Camera Binary Morphology Dilation Parameters

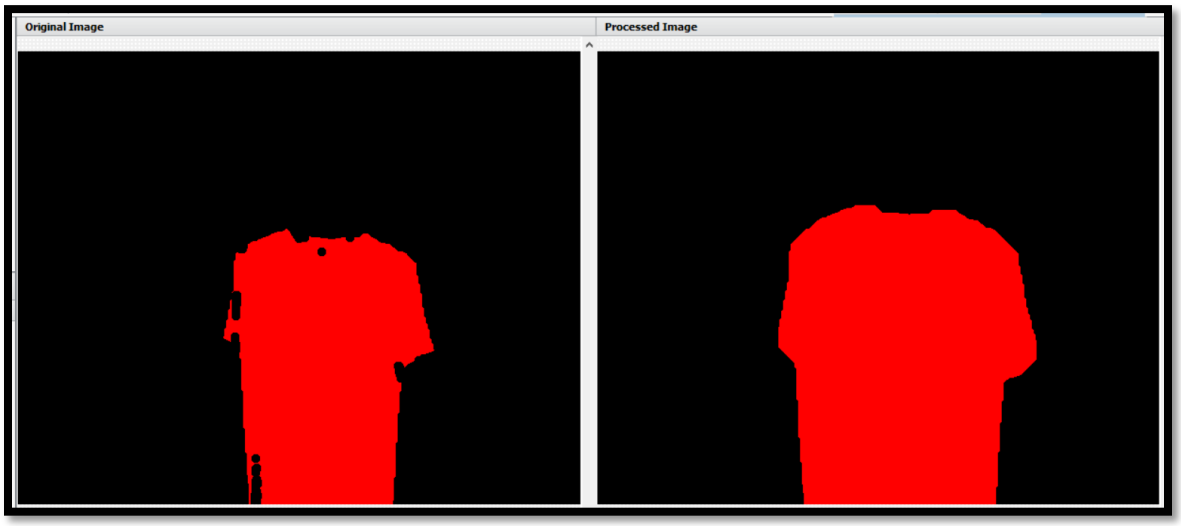



FIGURE 31: Front-Facing Sample Image Binary Dilation Result

The average inspection time for the FFT method vs. the binary morphology method is calculated with the performance meter tool within Vision Assistant. The cost of the FFT Filter alone is around 74 milliseconds, while the cost of the Binary Morphology Method, both erosion and dilation are around 13 milliseconds combined. The inspection results are shown below.

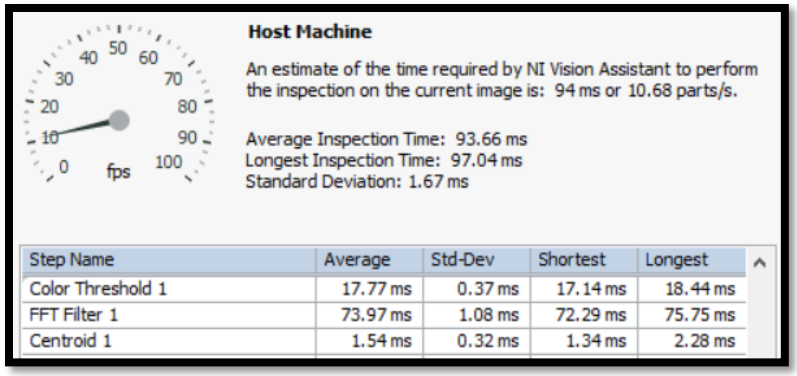



FIGURE 32: Front-Camera FFT Method Performance Analysis

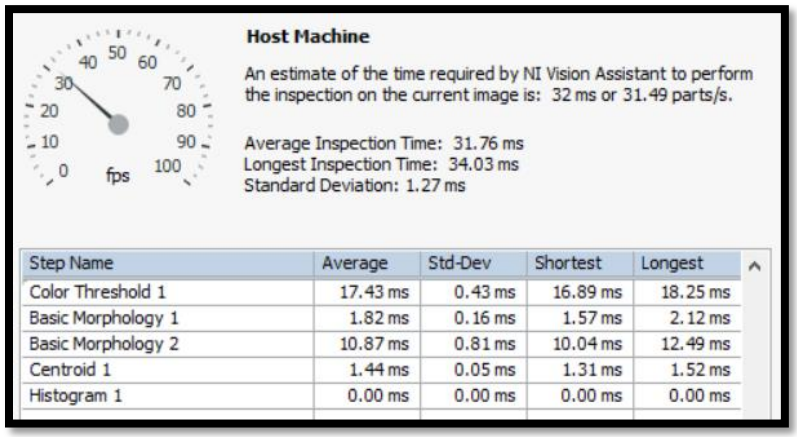



FIGURE 33: Front-Camera Binary Morphology Method Performance Analysis

The overall average processing time for the binary morphology method was around 32 milliseconds, while the FFT Filter method is around 94 milliseconds. Since these calculations must be performed for each frame that comes through the webcam, the faster they can be processed, the higher frame-rate that is possible for use, and thus the quicker reaction time of the robot. The frame rate for this technique coupled with low-resolution images allowed for real-time processing and reaction by the robot.

Another pre-processing step to make this data useful is to determine its centroid. The image centroid (X,Y) is calculated by the following equations:

$$X = \frac{\sum_{i=1}^{n} \sum (X_i * PV_i)}{\sum_{i=1}^{n} PV_i} \quad (2)$$

$$Y = \frac{\sum_{i=1}^{n} \sum (Y_i * PV_i)}{\sum_{i=1}^{n} PV_i} \quad (3)$$

Where $X_i$ is X coordinate of the $i^{th}$ pixel, $Y_i$ is Y coordinate of the $i^{th}$ pixel, and $PV_i$ is the $i^{th}$ pixel value ranging from 0 to 255, where 0 is minimum energy (black pixel) and 255 is maximum energy (white pixel).

The result of the centroid calculation can be seen in the following images. This centroid measurement is used to estimate the human player's distance from the center of the image (the error). This error measurement is then multiplied by a gain in order to create rotational movement whose objective is to keep the centroid of the target in the center of the image.

FIGURE 34: Front-Facing Camera Sample Image Centroid

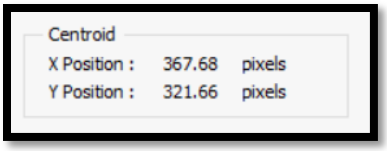



FIGURE 35: Front-Facing Camera Sample Image Centroid Coordinates

The final value necessary for calculation is one that can ascertain the visibility of the player in the frame. This is done by creating a histogram of the values within the image. Since the image is binary, any value not black should indicate that a human is present unless noise was able to make its way through the binary morphology noise filtration method. This is possible, but unlikely due to the fact that the environment is well controlled and information is A Priori. Even so, noise can be further filtered by reading the mean histogram value and using a minimum threshold value to provide a Boolean value of True or False to indicate that a human is present.

Additionally, this mean histogram value is used to estimate the human player's distance from the robot. The robot moves forward with a direct movement toward the player until a maximum threshold value threshold is met, causing the robot to stop and maintain an appropriate distance from the player. A sample of the histogram statistics can be seen in the following image.

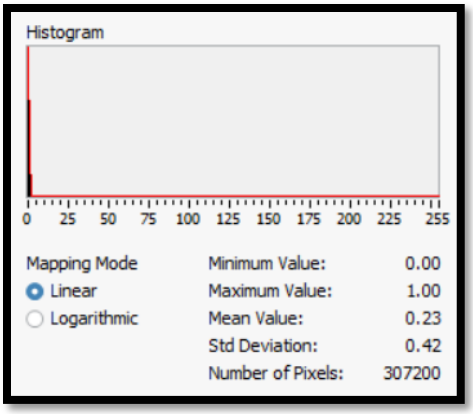



FIGURE 36: Front-Facing Camera Sample Image Histogram Statistics

This algorithm was developed through NI Vision Assistant, which can auto-generate the necessary code for creating a LabVIEW Virtual Instrument program.

Remembering the primary objectives of the robot during the match, the function of this sensor is to provide the data necessary for ensuring that the robot makes appropriate effort to maintain its front aligned tangent to the buffer zone circumference, as well as keep an appropriate distance from the human player. See the image below for a visual reference of this objective.

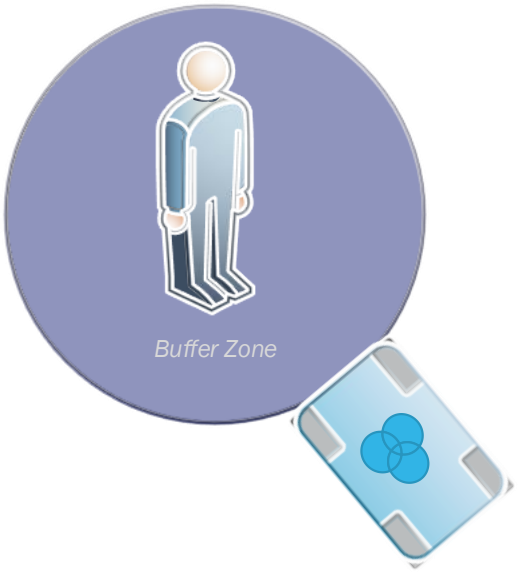



FIGURE 37: Buffer Zone Tangency

The rear-facing camera is only concerned with tracking the goal target. Similar to the front-facing camera, images captured by this camera pass through an algorithm that filters the incoming image by a pre-defined HSV threshold. The ranges of these values were derived from multiple images at various angles and lighting intensities to provide reliable identification of the unique red color of the squares on the face of the goal target, which serves as the control variable for the goal tracking model.

For this game, the goal target is a 24”x24” plywood board that has been raised to a height of 48 inches. On this target, several color swatch squares of Red Geranium paint color are placed on the target in an equilateral triangle. The goal target can be viewed in the following image acquired from the rear-facing camera:

FIGURE 38: Goal Target Sample Image Initial

Pre-processing of the image data is performed via HSV color thresholding. The filter values were determined based on several images with similar lighting conditions to that of the actual game environment. See the following images for further detail.

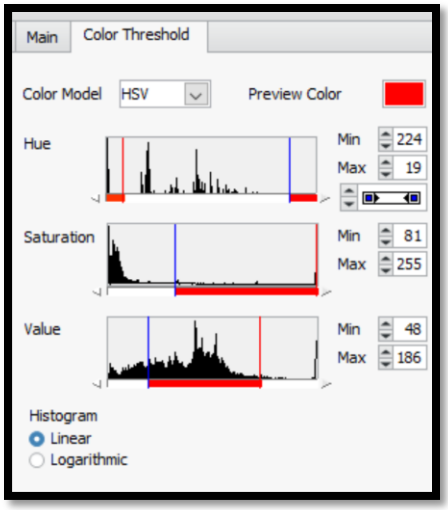



FIGURE 39: Rear-Facing Camera Color Model and Threshold Values

FIGURE 40: Rear-Facing Camera Sample Image Threshold Validation

The image is filtered to identify only the red color of the target and eliminate background information.

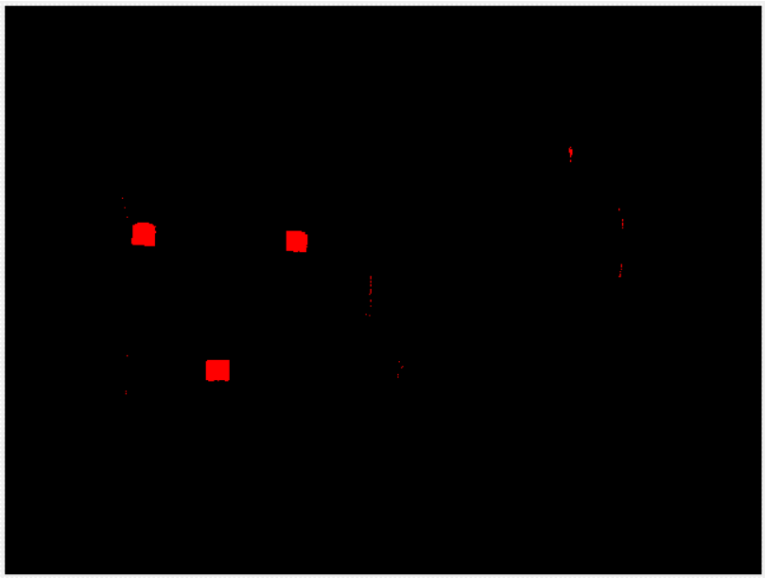


FIGURE 41: Rear-Facing Camera Sample Binary Image

After the binary image is created, inevitably some noise unassociated with the ROI will exist within the image frame. The initial algorithm for eliminating noise was the same FFT filtration method tested for use with the front-facing camera, where the higher spatial frequency artifacts are removed via lowpass filter. Note that the truncation frequency for the rear-facing camera is higher than the front-facing camera (10% vs. 5%). This is due to the smaller area of the color being used for the target vs. the large area of the green shirt. If the truncation frequency were lower, the small squares would be considered 'noise' and be eliminated. The following images detail the filter and the resulting processed image.

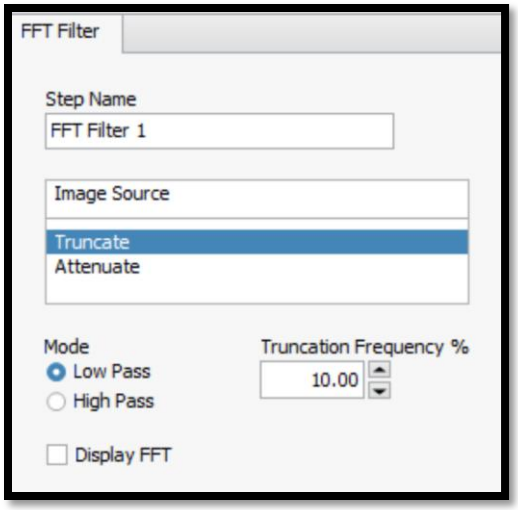



FIGURE 42: Rear-Facing Camera FFT Low-Pass Filter

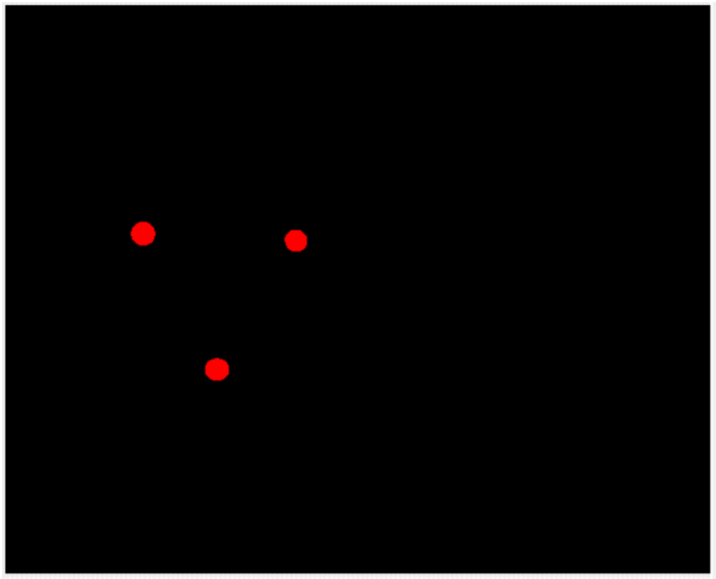


FIGURE 43: Rear-Facing Sample Image FFT Filter Result

A unique aspect of the FFT filter's effect on the squares was to turn them into circles by truncating the corners. Using this effect for purpose – a circle detection algorithm was implemented to count the number of circles that are visible. The circles were used as a count to verify that the target was actually being seen, rather than some other noise in the environment that unexpectedly remained after color thresholding and

FFT filtration. By ensuring that three circles were visible, the visibility of the goal target could be verified. See the following images for detail.

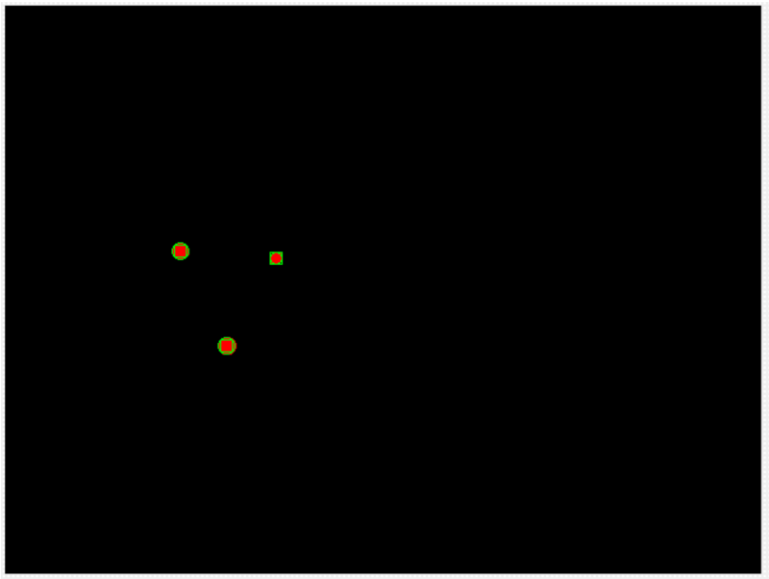


FIGURE 44: Rear-Facing Camera Sample Image Circle Detection Result

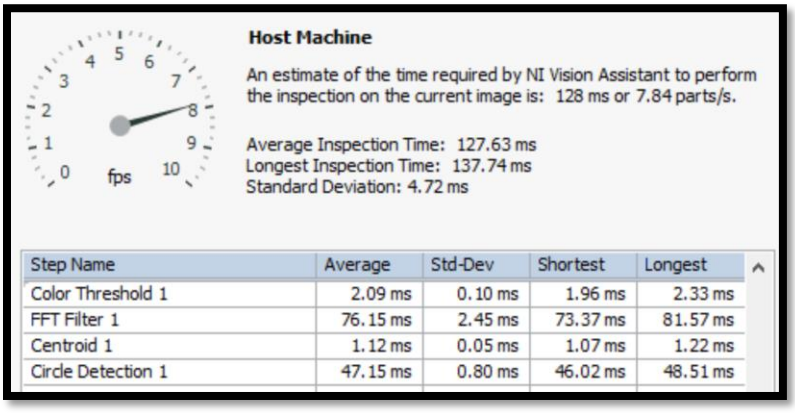
| Results ... | 1 | 2 | 3 |
|---|---|---|---|
| Center X | 148.00 | 229.00 | 187.00 |
| Center Y | 207.00 | 213.00 | 287.00 |
| Radius | 7.00 | 5.00 | 7.00 |

FIGURE 45: Rear-Facing Camera Circle Detection Parameter Details

The circle detection algorithm used a minimum radius of 4 pixels and a maximum radius of 100 pixels, for which all sample images containing the goal target provided a reliable reading. For images where not all squares were visible or were very close to the camera, the number of circles was outside of the expected range. For images where the goal target was fully visible, the results are similar to what is provided in the figure above.

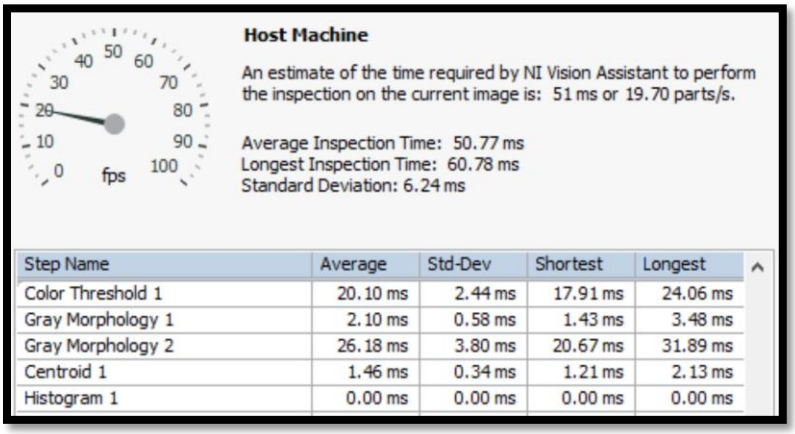
However, after determining the lower processing costs using the morphological opening technique for the front-facing camera, the same approach was used for the

rear-facing camera. Again, the erosion/dilation technique provided with superior results with faster processing times. See the performance analyses below.




| Step Name | Average | Std-Dev | Shortest | Longest |
|---|---|---|---|---|
| Color Threshold 1 | 2.09 ms | 0.10 ms | 1.96 ms | 2.33 ms |
| FFT Filter 1 | 76.15 ms | 2.45 ms | 73.37 ms | 81.57 ms |
| Centroid 1 | 1.12 ms | 0.05 ms | 1.07 ms | 1.22 ms |
| Circle Detection 1 | 47.15 ms | 0.80 ms | 46.02 ms | 48.51 ms |

FIGURE 46: Rear-Camera FFT Method Performance Analysis




| Step Name | Average | Std-Dev | Shortest | Longest |
|---|---|---|---|---|
| Color Threshold 1 | 20.10 ms | 2.44 ms | 17.91 ms | 24.06 ms |
| Gray Morphology 1 | 2.10 ms | 0.58 ms | 1.43 ms | 3.48 ms |
| Gray Morphology 2 | 26.18 ms | 3.80 ms | 20.67 ms | 31.89 ms |
| Centroid 1 | 1.46 ms | 0.34 ms | 1.21 ms | 2.13 ms |
| Histogram 1 | 0.00 ms | 0.00 ms | 0.00 ms | 0.00 ms |

FIGURE 47: Rear-Camera Binary Morphology Method Performance Analysis

The average processing cost for the morphological opening algorithm is 50.8 ms versus 127.6 ms for the FFT and circle detection algorithm. The updated filtration for the front and rear-facing cameras significantly accelerates the overall system performance and reaction time of the robot. See the following images for further detail of the Morphology setup. After receipt of the image frame, the HSV color threshold is performed to achieve a binary image. The erosion morphology step is then implemented, followed by the dilation morphology, centroid determination, and

histogram mean value calculation. The centroid of the goal target is used to estimate the goal target's distance from the center of the image (the error). This error measurement is then multiplied by a gain in order to create lateral movement whose objective is to keep the centroid of the target in the center of the image.

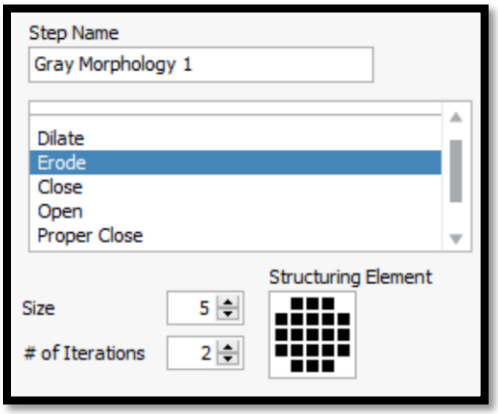



FIGURE 48: Rear-Facing Camera Binary Morphology Erosion

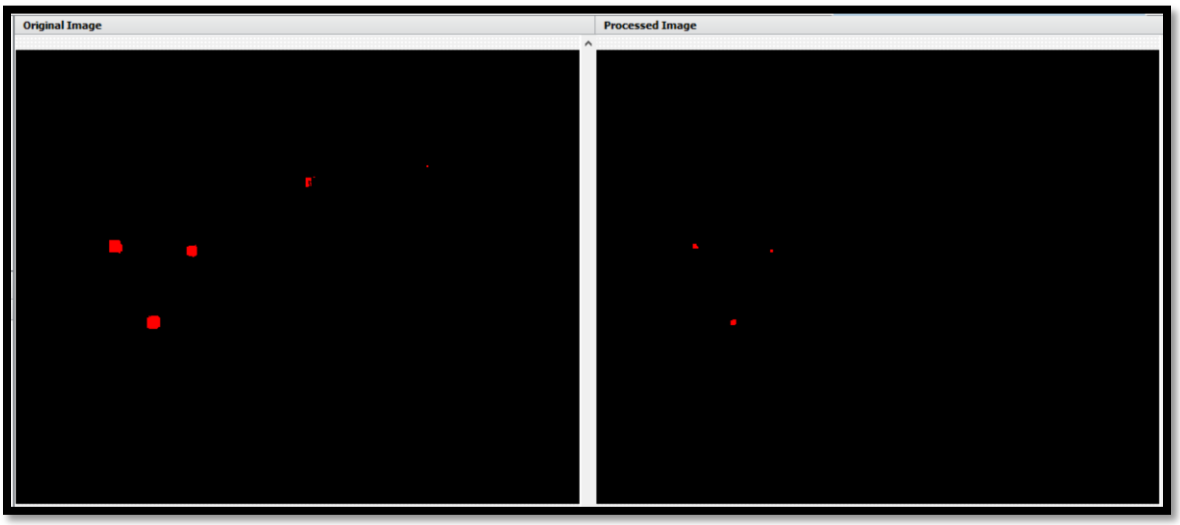



FIGURE 49: Rear-Facing Camera Sample Image Binary Morphology Erosion Result

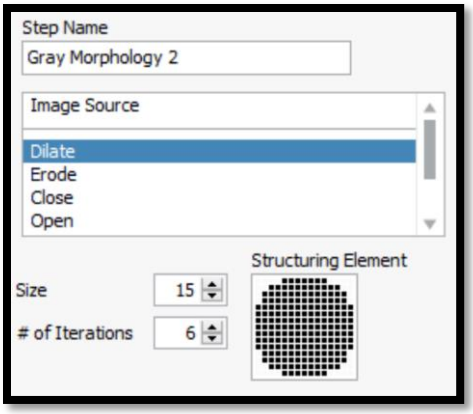



FIGURE 50: Rear-Facing Camera Binary Morphology Dilation

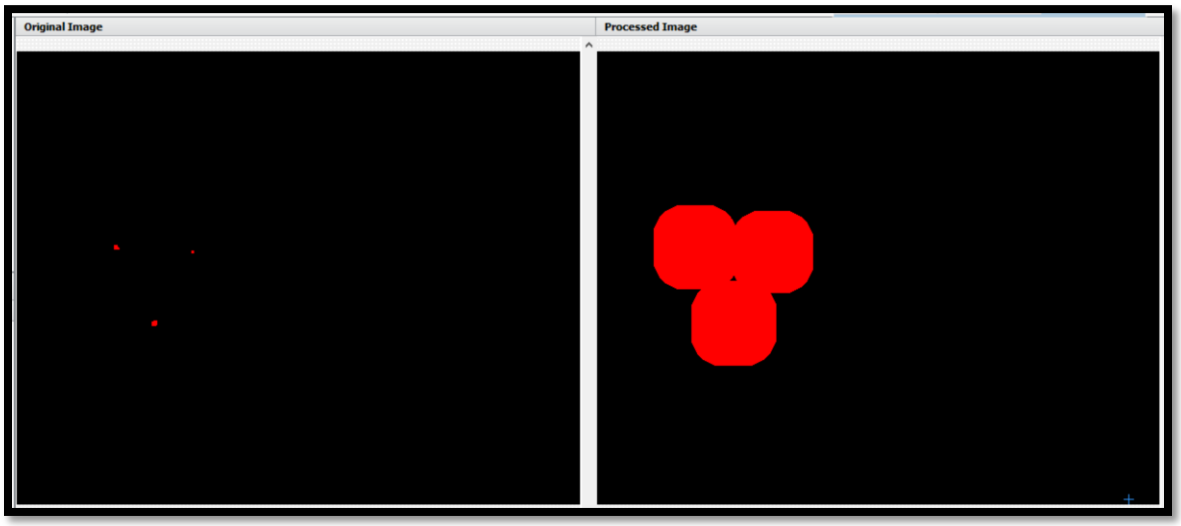



FIGURE 51: Rear-Facing Camera Sample Image Binary Morphology Erosion Result

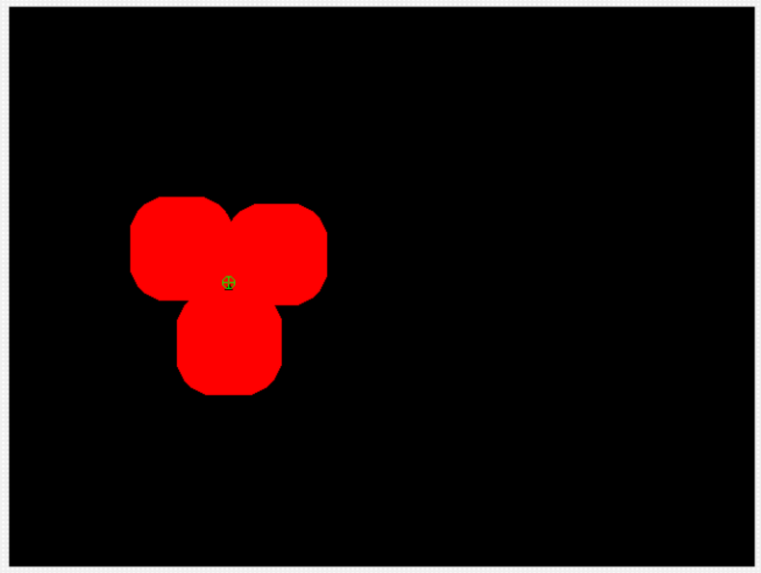


FIGURE 52: Rear-Facing Camera Sample Image Centroid

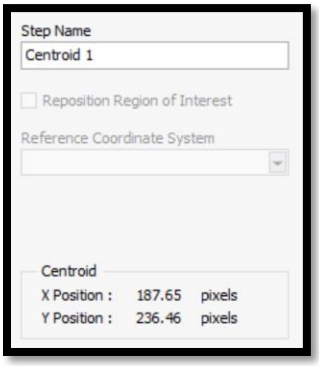



FIGURE 53: Rear-Facing Camera Sample Image Centroid Coordinates

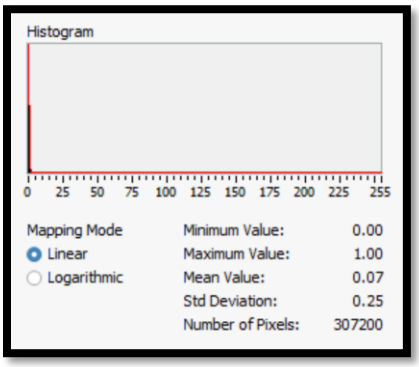



FIGURE 54: Front-Facing Camera Sample Image Histogram Statistics

# 4. CONTROL SYSTEM

## 4.1 Mecanum Drive Implementation

To build a control system capable of achieving the independent 4-wheel drive necessary for the Mecanum Drive function, a virtual instrument (VI) was created via LabVIEW for the main controller of the robot (myRIO 1900). Inputs for this program are throttle control toggle and throttle requests for Direct Movement (backward/forward) Lateral Movement (left/right), and Rotation(CCW/CW). Outputs for this program are the PWM values sent to the speed controllers (Talon SRX). The following image shows the front panel of the VI.

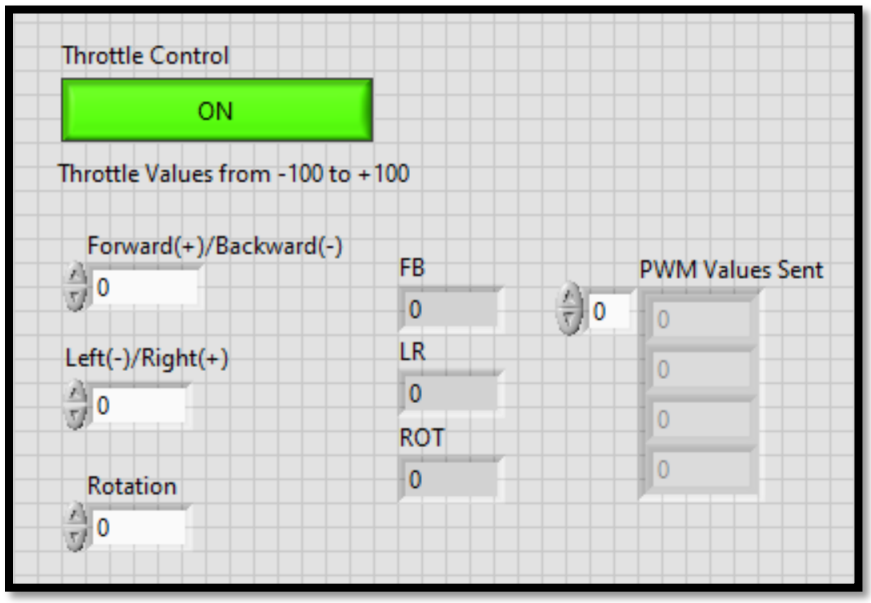



FIGURE 55: Mecanum Mixer Front Panel

The inputs for throttle control are values from -100 to +100. For Direct Movement, a positive value indicates forward movement of the robot (+Y direction), while a negative value indicates backward movement (-Y direction). For Lateral

Movement, a positive value indicates right directional movement (+X direction) and a negative value indicates left directional movement (-X direction). For Rotation, a positive value indicates clockwise rotation (+ψ direction), while a negative value indicates counterclockwise rotation (-ψ direction).

For this robot's application, the input values are each tied to a particular sensor and interpretation of the sensor's data controls the throttle response required for the associated degree of freedom. For instance, the Lateral Movement is controlled by the rear-facing camera, and the Rotation and Direct Movement is controlled by the front-facing camera. The data is fused together in this Mecanum Mixer VI to provide robot response to the targets that it is intended to track.

The Talon SRX speed controller expects a signal for a high time input pulse between 1ms (backward rotation at full throttle) and 2ms (forward rotation at full throttle) to control the wheel movement to which it is tied. There are four total speed controllers – one for each motor. A signal of 1.5ms would indicate no movement of the motor. The channel layout of speed controllers for the Left Front (LF), Right Front (RF), Left Rear (LR), and Right Rear (RR) is provided in the appendix. These channels correlate to the available inputs of the myRIO 1900.

The frequency of the PWM signal sent to the speed controller is set to a constant 50Hz. See the following image and equations for the calculation of the duty cycle.

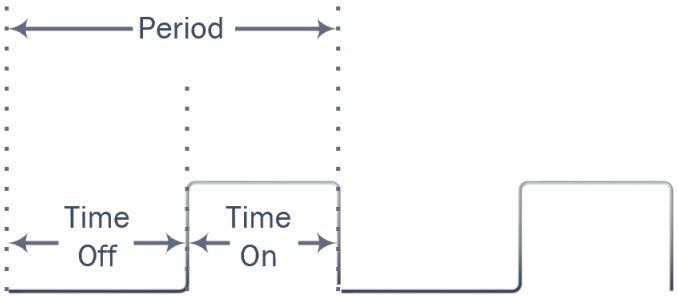



FIGURE 56: Duty Cycle for PWM

$$Duty\ Cycle = \frac{Time\ On}{Time\ On + Time\ Off} = \frac{High\ Time\ Input\ Pulse}{Period} \tag{4}$$

$$High\ Time\ Input\ Pulse\ (for\ Talon\ SRX)$$

$$= \begin{cases} 1ms & full\ backwards\ rotation \\ 1.5ms & no\ rotation \\ 2ms & full\ forward\ rotation \end{cases} \tag{5}$$

$$Period = \frac{1}{f} = \frac{1}{50Hz} = 0.02s = 20ms \tag{6}$$

$$Duty\ Cycle = \begin{cases} \frac{1ms}{20ms} = 0.05 & full\ backwards\ rotation \\ \frac{1.5ms}{20ms} = 0.075 & no\ rotation \\ \frac{2ms}{20} = 0.1 & full\ forward\ rotation \end{cases} \tag{7}$$

Creating the duty cycle for the PWM signals of the Mecanum control is accomplished by a multitude of steps. First, consider the Direct and Lateral Movement throttle values provided to the VI. The sign for each of the Direct and Lateral Movement values is extracted by the following:

$$sgn(x) = \begin{cases} -1 & if\ x < 0, \\ 0 & if\ x = 0, \\ 1 & if\ x > 0. \end{cases} \tag{8}$$

The resultant vector magnitude of these throttle values is then calculated. The resultant vector magnitude of the Direct (Ẏ) and Lateral Movement (Ẋ) throttle values are determined by the following:

$$V = \sqrt{\dot{X}^2 + \dot{Y}^2} \tag{9}$$

An image illustrating this calculation is shown below:

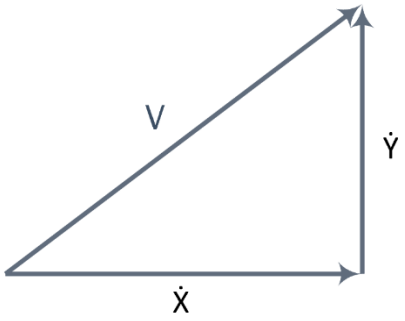



FIGURE 57: Vector Magnitude

This vector magnitude is then used as the divisor to determine the PWM duty cycle signal sent to each of the motors prior to mixture. The square of each signal (maximum of 100) is multiplied by its original sign and divided by the vector magnitude. This value is then divided by 4000 to ensure that the final value is between 0.025 and -0.025, which are used in the PWM duty cycle calculations. If the value is not a number (for the occasion where throttle values are zero), a value of 0 is recorded.

Direct Movement and Lateral Movement PWM duty cycle calculations can be seen in the following equations:

$$Premix\ Direct\ Movement\ PWM\ Duty\ Cycle$$

$$= \begin{cases} \frac{sgn(\dot{Y}) * \dot{Y}^2}{\sqrt{\dot{X}^2 + \dot{Y}^2}} * \frac{1}{4000} & \dot{X}|\dot{Y} \neq 0 \\ 0 & \dot{X}, \dot{Y} = 0 \end{cases} \quad (10)$$

$$Premix\ Lateral\ Movement\ PWM\ Duty\ Cycle$$

$$= \begin{cases} \frac{sgn(\dot{X}) * \dot{X}^2}{\sqrt{\dot{X}^2 + \dot{Y}^2}} * \frac{1}{4000} & \dot{X}|\dot{Y} \neq 0 \\ 0 & \dot{X}, \dot{Y} = 0 \end{cases} \quad (11)$$

The calculation for the Rotational Movement PWM Duty Cycle is according to the following equation:

$$Premix\ Rotational\ Movement\ PWM\ Duty\ Cycle = \frac{\dot{\psi}}{4000} \quad (12)$$

where positive values indicate clockwise (CW) rotation and negative values indicate counterclockwise (CCW) rotation.

To understand the mixing of the duty cycles for Direct Movement (DM), Lateral Movement (LM), and Rotational Movement (RM), see the following image which relates the throttle values to the wheel force vectors:

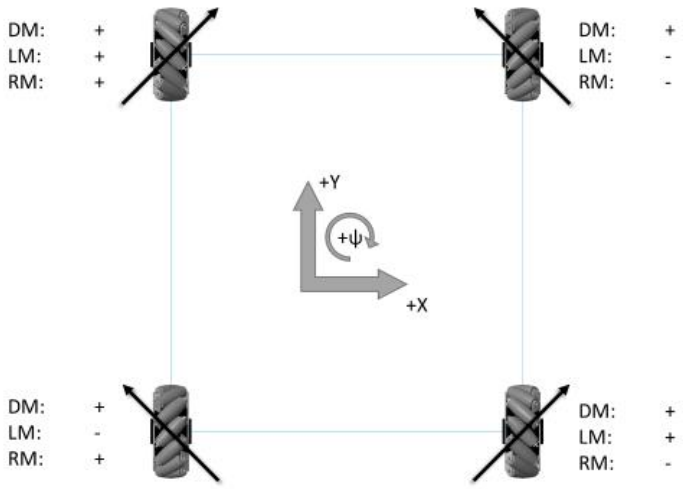



FIGURE 58: Mecanum Mixing of PWM Duty Cycles

From this image, the mixture of signals becomes much more intuitive. If the robot were requested to move forward, the Direct Movement (DM) throttle value would be positive (+) for all wheels, causing forward rotation. If the robot were requested to rotate clockwise, the Rotational Movement (RM) throttle value would be positive (+) for the left front (LF) and left rear (LR) wheels, and negative (-) for the right front (RF) and right rear (RR) wheels. If the robot were requested to move to the right, the Lateral Movement (LM) throttle value would be positive (+) for the left front (LF) and right rear (RR) wheels, and negative (-) for the left rear (LR) and right front (RF) wheels.

The following table provides the associated calculations for PWM signals provided to each wheel motor via the Talon SRX speed controller:

TABLE 4: PWM Duty Cycle Mixture

| | | | |
|---|---|---|---|
| Left Front (LF) | DM+LM+RM+0.075 | Right Front (RF) | DM-LM-RM+0.075 |
| Left Rear (LR) | DM-LM+RM+0.075 | Right Rear (RR) | DM+LM-RM+0.075 |

The values of the mixture prior to the addition of 0.075 are coerced to values within the range of -0.025 to +0.025 ensuring that no error is encountered, such as receiving a value surpassing the intended maximum or minimum. As mentioned previously, a duty cycle value of 0.075 would indicate no rotation for the motor. Taking the Left Front wheel as an example, the maximum value of DM+LM+RM would be +0.025 and the minimum value would be -0.025. At maximum value, the left front wheel would receive a value of 0.025+0.075 = 0.1, which is the expected PWM duty cycle value for full forward rotation. At minimum value, the left front wheel would receive a value of -0.025+0.075 = 0.05, which is the expected PWM duty cycle value for full backward rotation.

Consider the following scenario where $\dot{X}$ has a throttle value of -30, $\dot{Y}$ has a throttle value of +40, and $\dot{\psi}$ has a throttle value of -10. Using the table and equations above, the following calculations can be performed for determining each of the PWM duty cycle signals provided to the speed controllers:

$$Premix\ Direct\ Movement\ PWM\ Duty\ Cycle = \frac{+(40^2)}{\sqrt{30^2+40^2}} * \frac{1}{4000} = 0.008 \quad (13)$$

$$Premix\ Lateral\ Movement\ PWM\ Duty\ Cycle = \frac{-(30^2)}{\sqrt{30^2+40^2}} * \frac{1}{4000} = -0.0045 \quad (14)$$

$$Premix\ Rotational\ Movement\ PWM\ Duty\ Cycle = \frac{-10}{4000} = -0.0025 \quad (15)$$

$$Left\ Front\ (LF) = 0.008 + (-0.0045) + (-0.0025) + 0.075 = 0.076 \quad (16)$$

$$Left\ Rear\ (LR) = 0.008 - (-0.0045) + (-0.0025) + 0.075 = 0.085 \quad (17)$$

$$Right\ Front\ (RF) = 0.008 - (-0.0045) - (-0.0025) + 0.075 = 0.09 \quad (18)$$

$$Left\ Rear(LR) = 0.008 + (-0.0045) + (-0.0025) + 0.075 = 0.081 \quad (19)$$

These calculations are performed in a while loop that is controlled by the throttle control toggle and calculated values are output as a 1D array.

### 4.1.1. Tele-Operational Mapping

Verifying the Mecanum Mixer program was performed with an Xbox One controller via remote or tele-operational control. This program was written in such a way that it could be easily ported to automated control with camera sensor input after the sensor fusion algorithms were devised. The button and joysticks of the Xbox One controller were mapped in such a way that the robot could be easily and intuitively

controlled. The following images illustrate the mapped parameters.

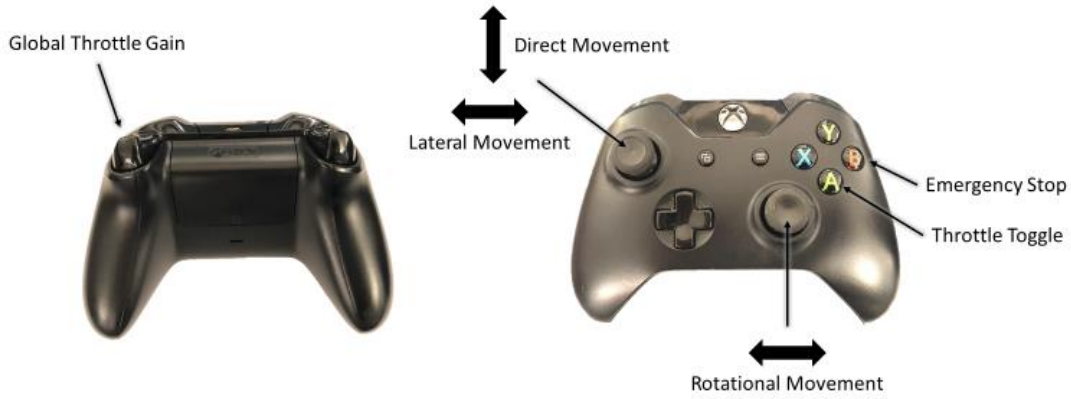



FIGURE 59: Xbox One Controller Mapped Parameters

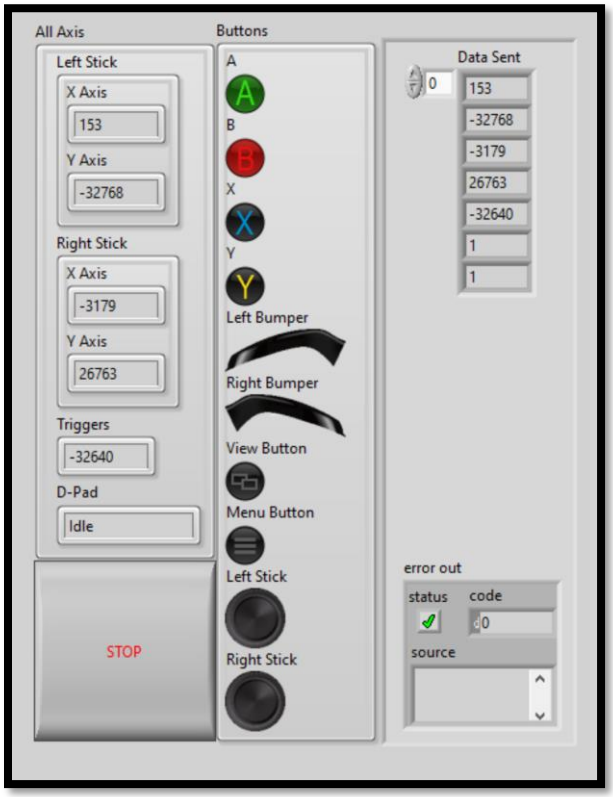



FIGURE 60: Xbox Controller Parameters VI

The Xbox Controller VI is run on the laptop used for programming the myRIO with LabVIEW. The Xbox controller parameters are each individually mapped, placed in

an array and set as a shared variable named 'Joystick Transmission.' This shared variable array is then used by the teleop program running on the myRIO which uses the Mecanum Mixer as a sub-VI to cause the associated movement.

In the teleoperation control VI, the 'A' button provides a Boolean value (0 or 1) which is used to toggle the throttle control of the Mecanum Mixer VI on or off. In this program, the robot's start button also causes the same action. The 'B' button also provides a Boolean value (0 or 1) and is used to set the Emergency Stop to true, which stops all motors and stops the loop for the motor control. In this program, the robot's Emergency Stop button also causes the same action. The right trigger provides an analog signal from 0 to -32768, which is mapped to the overall throttle coefficient. This can be considered the global gain on all throttle values. The value's sign is inverted and then divided by 65536 and then the value is coerced between 0.15 and 0.5.

The left joystick provides an analog signal of -32768 to 0 for left action on the X-axis, 0 to 32768 for right action on the X-axis. The received value for the left joystick X-axis is divided by 360 and coerced to a value between -10 and 10 which is then multiplied by the lateral multiplier before being sent to the lateral movement parameter of the Mecanum Mixer VI. Any value out of range of the coercion is forced to 0. The lateral multiplier allows for the discovery of optimum throttle values for an increasing response of movement relative to the forward/backward motion of the direct movement. This was important for discovering robot speed that was still deemed safe for use with human interaction.

The left joystick also provides an analog value of -32768 to 0 for up action on the Y-axis, and 0 to 32768 for down action on the Y-axis. The received value for the left joystick Y-axis is divided by 360 and inverted (so up direction is positive and down direction is negative), and then the value is coerced between -10 and 10 before being sent to the direct movement parameter of the Mecanum Mixer VI. Any value out of range of the coercion is forced to 0.

The right joystick provides an analog signal of 0 to 32768 for right action on the X-axis, and -32768 to 0 for left action on the X-axis. The received value for the right joystick X-axis is divided by 360 and coerced to a value between -10 and 10 before being multiplied by the rotational multiplier and being sent to the rotational movement parameter of the Mecanum Mixer VI. Any value out of range of the coercion is forced to 0.

The shared variable containing the aforementioned controller parameters is used by the Teleoperational Control VI. This VI begins by setting all movement parameters to zero using local variables for Rotation, Direct Movement, Lateral Movement, Throttle Control, and E-Stop!. The LED VI is also used to indicate the robot's current state – and during initialization is set to Blue – meaning that the robot is ready to play, but throttle control is off. The movement parameters are all sent to a special sub-VI created to determine the minimum movement threshold before being sent as PWM values to the Mecanum Mixer VI. The reason for this is that the motors need a certain torque value to overcome the inertia of the system. The thresholds are used to define this minimum value to cause movement.

The special sub-VI is called Inverted Value Threshold Coercion. Inputs to the VI are a value and a minimum value threshold. If the input value is equal to zero, the output value is equal to zero. If the absolute input value is between zero and the threshold value, the value is coerced towards the threshold value and maintains the polarity of the input value. If the absolute input value is greater than the threshold value, it is passed without affecting value or polarity.

The start button on the robot is used with the LED control sub-VI to indicate the state of the robot's throttle control – green for on, and blue for off. The emergency stop button is also used with the LED control sub-VI to indicate that the robot is in the red stopped state and when pressed also causes all throttle values to be locked at 0, the tele-op control program to end, and the throttle control to be turned off. The front panel of the Teleoperational Control VI can be seen in the following image.

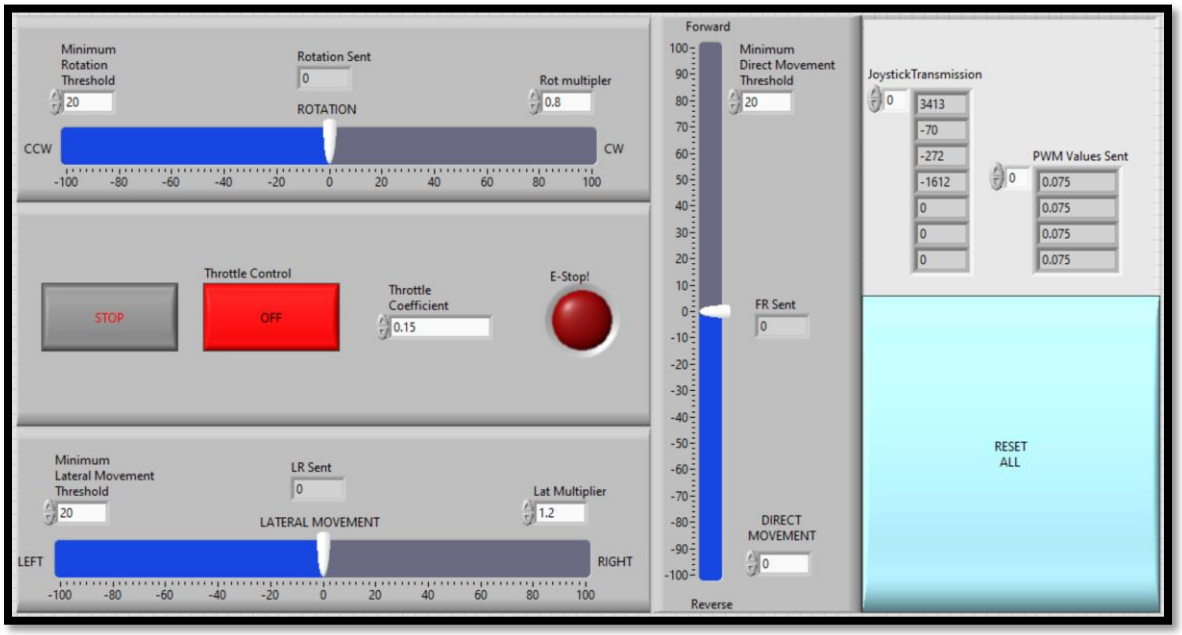



FIGURE 61: Tele-op Control Front Panel

## 4.2 Autonomous Control

Fusing the sensor data from the front-facing camera, and rear-facing camera to make sense of the environment while tracking multiple targets – static and dynamic and causing the robot to take appropriate action quick enough for suitable use in soccer training took much thought and deliberation. The following subsections describe the control system in detail.

The first functional element of the main program initializes all systems to default state. The throttle control is set to false, or off, and the lights on the robot turn blue to indicate to the user that the robot has turned on and is readying itself for gameplay. Once the human player presses the start button, the initialization function begins a 10-second countdown, which changes the LED color to yellow, and when the timer gets to three seconds until game commencement, the color changes to orange. Once the game begins, the color is set to green and the robot begins moving and reacting to the sensor data.

The Game Loop is the main program of the system which collects input from the target monitor programs: Player Monitor Loop, Goal Monitor Loop, Distance Monitor Loop and outputs the appropriate values of lateral movement, direct movement, and rotational movement to the Mecanum Mixer Sub-VI. These programs will be explained in further detail below. This loop is set to operate on time for the intended amount of game time, which is initially set to 30 seconds but can be user-controlled to any desired value.

Movement priorities are calculated as the errors between the expected state and the measured state: for rotational movement, this is the difference between the image center and the human player centroid; for lateral movement, this is the difference between the image center and the goal target; for direct movement, this is the difference between maximum threshold value and mean histogram value. The ultimate result of the priorities is to define how quickly the reactions should be for each sensor reading based on the game action. The priority number for each movement type: defend ($Pr_{defend}$), seek player ($Pr_{player}$), seek goal ($Pr_{goal}$) is a number between -1 and +1. The last known location of the target being tracked is provided as a sign where -1 indicates the last known location is in the left frame of the image for the camera sensors, or to the left of the direct and +1 is when the target is to the right. These priority values are multiplied by an adjustable gain before being input into the Mecanum Mixer VI as throttle values.

The Goal Monitor Loop outputs Goal Priority (-1 to +1), Sign of Goal Priority (-1,0,+1), and Goal Visibility (T/F). This program uses the filtering sub-VI discussed in the Rear-Facing Camera Section, which uses HSV parameters and Binary Morphology to determine the centroid of the image and the mean histogram value. If the mean histogram value is above 0.02, the image is considered to contain the goal. The centroid of the x-axis is then used in the following equation to calculate the rotation error referenced to the center of the image (multiplied by the gain value):

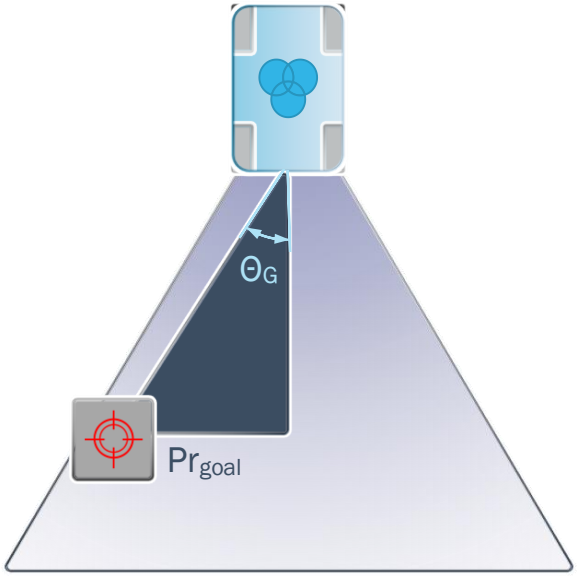



FIGURE 62: Goal Priority

$$Pr_{goal} = \frac{80 - Centroid_{goal,x}}{80} \quad (20)$$

The reasoning behind the value of 80 is that this number is the midpoint of the image being used for calculation since the incoming images are taken at 160x120 pixels. The frame rate for the incoming images is 30 frames per second. This image size was the smallest size which still provided a reliable value for a reading of the goal target. The parameters for the rear-facing webcam can be seen in the following image.


Figure 63: Goal Target Image Parameters

In the Game Loop, if Goal Visibility is True, a timer is continuously reset until Goal Visibility is False. For each subsequent iteration, the time for which the Goal Visibility is False gets passed into an array, presented and graphed on the VI front panel for data analysis.

Also, if Goal Visibility is True, the sign for goal priority is stored, and the Goal Priority value is multiplied by a constant of 0.75 before being multiplied by the Goal Gain. This value is output to the Mecanum Mixer for Lateral Movement. The case for when Goal Visibility is False is slightly more complicated. The last stored value for the Goal Sign is multiplied by 0.618 and only output if the goal timer reads a value higher than 0.25 seconds, and only reacts for a 1 second maximum before the sign for goal priority is set to zero – causing lateral movement to cease even if the goal target is not in view.

One of the robot's objectives is to create reactive movement according to the following image: maintaining 3-point linearity, where the robot tries to drive $\Theta_P$ and $\Theta_G$ to zero. The Player Monitor Loop consistently rotates the robot in order to track the player and keep the player in the center of the image, while the Goal Monitor Loop moves laterally to keep the goal target in the center of the image. In order to achieve the linearity condition, a higher tracking score on the Player Monitor Loop via a 0.25-second reaction delay on lateral movement. If this is not present, the two loops 'fight' each other by trying to maintain design rules that are slightly at odds with one another – since if the robot rotates to track the player, the goal target will accordingly move its location in-frame relative to the rear-facing camera. To combat this, the delay was built into the system so that the player was always being tracked, and then the robot would then move laterally back towards the last known position of the Goal Target.

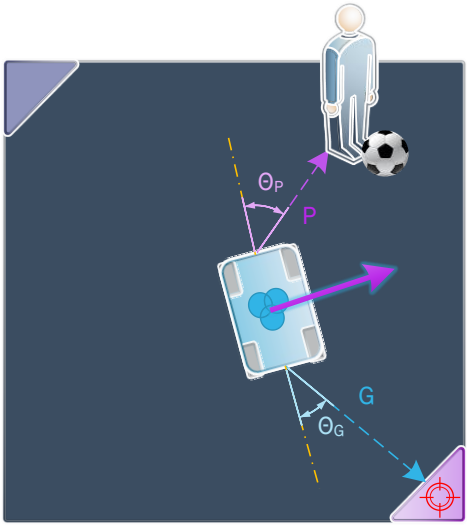


FIGURE 64: Intended Robot Movement

The Player Monitor Loop outputs the Player Priority, Sign of Player Priority, and Player Visibility. This program runs continuously during gameplay and has the primary task of monitoring the sensor data from the front-facing camera. This VI also uses filtration for HSV parameters and Binary Morphology to determine the centroid and mean histogram value outlined in the Front-Facing Camera Section – similar to the Goal Monitor Loop. If the mean histogram value is greater than 0.02, the image is considered to contain the goal. The centroid of the x-axis is then used in the following equation to calculate the rotation error referenced to the center of the image (multiplied by the gain value):

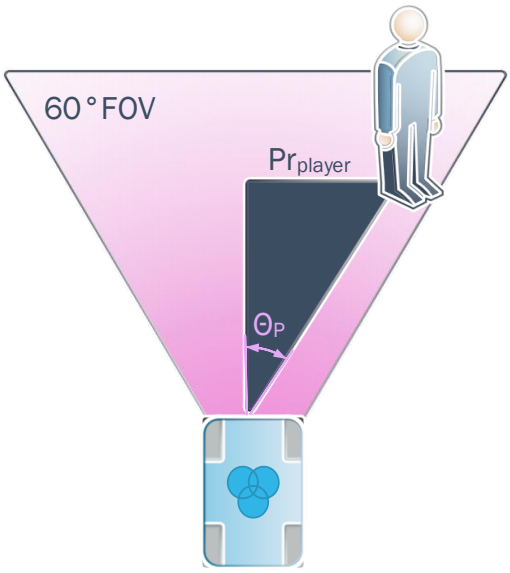



FIGURE 65: Player Priority

$$Pr_{player} = \frac{80 - Centroid_{player,x}}{80} \quad (21)$$

This image size was chosen for speed of processing and the ability to have images incoming at 30 frames per second without appreciable lag.


FIGURE 66: Player Target Image Parameters

In the main Game Loop, if Player Visibility is True, a timer is continuously reset until Player Visibility is False. This action is similar to the Goal Monitor Loop. For each subsequent iteration, the time for which the Player Visibility is False gets passed into an array, presented and graphed on the VI front panel for data analysis.

Additionally, if the Player Visibility is True, the sign for player visibility is stored and the Player Priority is passed through and multiplied by the player gain before being sent as rotational movement Mecanum Mixer Sub-VI When the Player Visibility is False, the sign of the last priority value during player visibility is multiplied by 1.125 before

being multiplied by the player gain, negated, and then sent to the Mecanum Mixer's Rotational Movement input.

The Distance Monitor Loop uses the mean histogram value acquired from the front-facing camera and subtracts this number from the maximum threshold (0.55) before multiplying by a gain value and sending that value as the direct movement throttle for the Mecanum Mixer VI. This is only valid for the condition where the mean histogram value is higher than the minimum threshold for the human player (0.2) and lower than a maximum threshold for the goal target (0.7) – to stop movement if the goal is too close.

The control systems used for the front-facing and rear-facing camera are actually a type of proportional gain controller. One of the most commonly used types of control systems is the proportional-integral-derivative (PID) controller, which uses information from the present, as well as past and anticipated future state to generate a controlled response. For this application's purpose, only the proportional feedback was needed for movement and simplification of the controller.

For proportional control, a proportional controller gain, $k_p$ is multiplied by the error term to create an output. See the following image for detail:

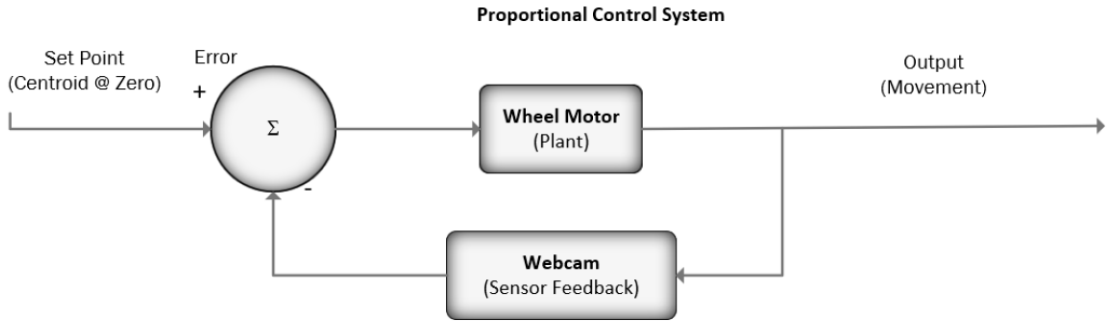



FIGURE 67: Proportional Controller Diagram

As described in the Goal and Player Monitor Loop sections, the centroid of the image is used to calculate its proportion away from the center of the image – where our set point is the pixel value of 80 and the process variable is the centroid measurement. The difference between these values is the error term. This error term is then multiplied by a gain before being output to the program which causes wheel movement. These calculations of error from centroid are continuously calculated and passed along.

The stall torque of the wheels from the system inertia actually serves to dampen any signals that would provide steady-state error, and are somewhat of a natural integral controller. In PI or PID control, the integral term accounts for historic values of error and seeks to eliminate the residual error. Since the motor needs a minimum voltage to effect movement, any imbalances of proportional control which would cause oscillations in this range are eliminated. However, this is only true for appropriate gain values. If the gain value was too high, the robot would overcompensate movements causing extreme oscillation. In this case, I could have used a Derivative term in order to control rapid changes in error. For this simple model, the P controller was effective.

# 5. TEST PLAN

The first step to verify the approach was to assess the filtering methods for centroid calculation for front and rear cameras. This verification was provided in the Sensor Fusion Section. The next step was to verify the objectives of the robot independently which cause movement of the robotic system – front-facing camera for rotational movement and direct movement, and rear-facing camera for lateral movement. After verifying rotational and lateral movement, those programs were tested for integration, which prompted the higher tracking score for the Player Monitor Loop versus the Goal Monitor Loop. The final step was to integrate the programs into a single program capable of fusing the sensor input to cause the intended reaction.

To verify the function of the Goal Monitor Loop, a simple VI was written, which calculates loop timing and also uses the outputs of the Goal Monitor Loop – Goal Priority, Sign of Goal Priority, and Goal Visibility. The front panel allows the user to start and stop the movement of the robot by either providing the calculated values to the Mecanum Mixer's Lateral Movement or sending 0 value. If the target is visible, the goal priority value is multiplied by the gain value and sent as Lateral Movement. If the target is not visible, the robot moves in the direction of the last target direction for 1 second and sets itself to zero throttle thereafter. After the output value is multiplied by a gain value, which can be adjusted by the user on the front panel, the Lateral Movement values sent are captured in a chart that illustrates these values for each loop iteration.

The speed of a player of average skill was considered to be around 1.5 m/s – which is around the same speed as a brisk walk. This speed was based on empirical observation of several colleagues attempting to dribble with the ball. To associate the robot's speed with the player's speed, the goal target was shuttled ten times between two pieces of tape 1 meter apart from one another – to calculate the time required for response, and to check if the responses showed a repeatable signal. The objective of the test was to verify that the program could read the centroid of the target and make the appropriate action without overshooting the target at the appropriate speed. The gain value was adjusted until this was repeatable. Step responses were also recorded. The pass criteria for this test was movement from the robot within a time of 2.36 seconds, which correlates to the average player dribble speed and the time it takes the player to reach the robot from starting position (3.5 meters from the robot) with repeatable result. The test setup is provided in the following image:

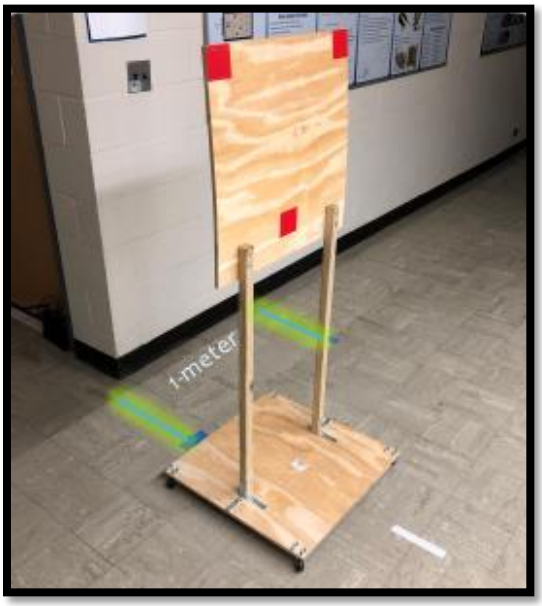



FIGURE 68: Goal Monitor Loop Test Setup

Similar to the Goal Monitor Loop Test, a simple VI was written to make use of the outputs of the Player Monitor Loop:  Player Priority, Sign of Player Priority, and Player Visibility.

The front panel allows the user to start and stop the movement of the robot by either providing the calculated values to the Mecanum Mixer's Rotational Movement or sending 0 value.  The output value is multiplied by a gain value, which can be adjusted by the user on the front panel and the values are captured in a chart that illustrates these values for each loop iteration.

Testing the algorithm was similar to the Goal Monitor Loop test, but instead of using the goal target as the moving target, I used myself and stepped 1 meter left and right ten times each second – to verify that the program could read the centroid of the target and make the appropriate action without overshooting the target at the appropriate speed.  Again, the pass criteria for this test was movement from the robot within a time of 2.36 seconds.  The gain value was adjusted until the robot's action was quick enough to track effectively, but did not overshoot its target with sustained oscillations.  Step responses were also recorded.  A view of the test setup can be seen in the following image:


FIGURE 69: Player Monitor Loop Test Setup

After both the function of the Player Monitor Loop and Goal Monitor Loop were verified and appropriate gain values were discovered, the two algorithms were combined as well as some additional functionality, including the human-machine interface (start and emergency buttons) and LED indication system.

This test uses a human subject (myself), this time stepping at a distance of √2 (1 meter to the right and 1 meter forward) within 1 second to check step response. See the following image for detail:

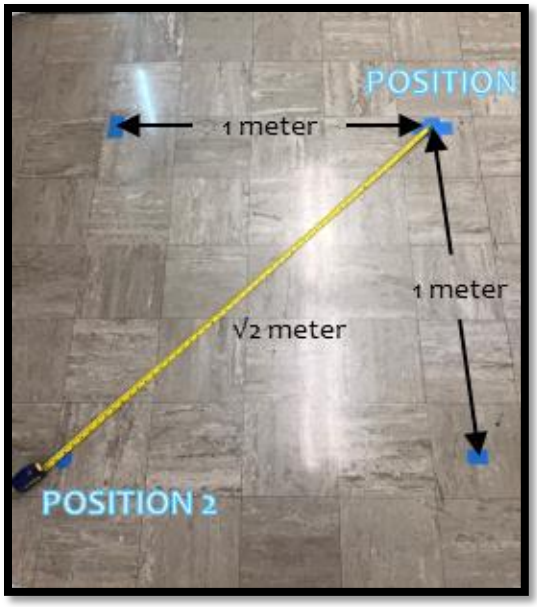



FIGURE 70: Player and Goal Monitor Fusion Loop Test

Pass criteria is movement within 2.36 seconds. Tests of dynamic movement were also performed to check consistency of tracking – pass criteria for this test was if the majority of tests lost visibility of the human player less than the goal target if the maximum visibility loss for either target was less than 15% of the total game time, and if maximum time per visibility loss for either target was less than 1 second. The front panel of the VI, loss of tracking for both player and goal are sent into arrays and the player priority charts are captured to check tracking and values sent to the motors for movement.

After verification of the Player Monitor Loop (rotational movement) and Goal Monitor Loop (lateral movement), as well as the fusion of the rotational and lateral movement, the next verification was for the Distance Monitor Loop. Using myself as the test target, a step response was recorded for 1-meter movement. If the robot responded within 2.36 seconds of movement, the test was considered passed.

The final verification of the system was the fusion of the Player Monitor Loop, the Goal Monitor Loop, and the Distance Monitor Loop, which incorporates all objectives of the thesis:  direct motion – maintaining an appropriate buffer zone, rotational motion – maintaining tangency with the buffer zone, and lateral motion – maintaining 3-point linearity with the robot between the player and goal.

This test involved reading a step response for a diagonal movement of √2 – the same movement used for verification of the Goal and Player Monitor Fusion Loop. Using the same passing criteria as the previous step responses, if the robot was able to make appropriate movement within 2.36 seconds, it was deemed successful.

# 6. RESULTS

For the Goal Monitor Loop, the Gain Value found to produce desired repeatable results without consistent overshoot and oscillation was found to be a value of 60. The following image illustrates the front panel output from the test shuttling the goal target between tape placed 1 meter apart:

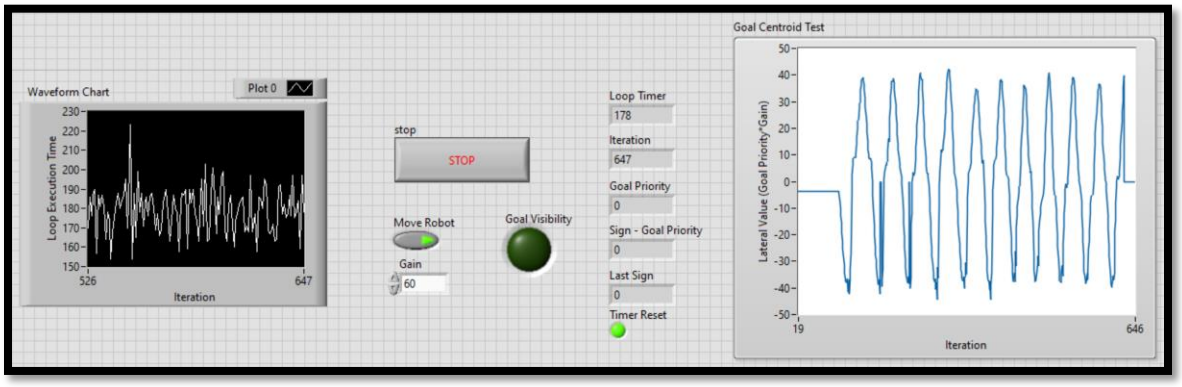



FIGURE 71: Goal Monitor Loop Test Front Panel

The left-hand side of the image provides the loop execution time for each iteration. The RMS value for this loop iteration is around 180ms.

Focusing on the chart to the right side of the front panel, we have the following figure:

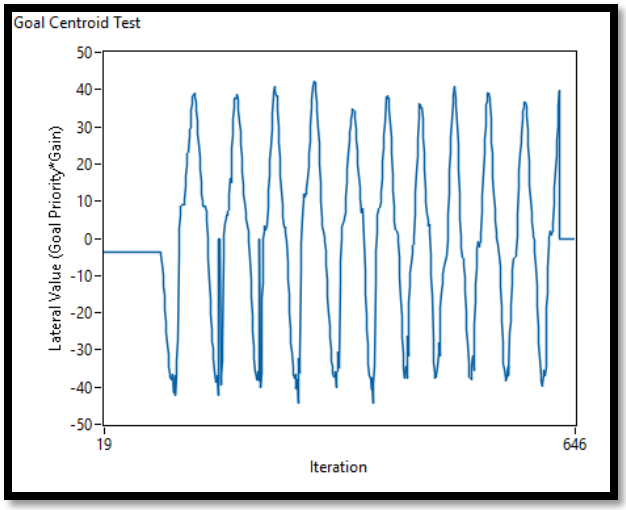



FIGURE 72: Goal Monitor Loop Test - Proportional Control Repeatability

When the target is moved to the left of the robot, the minimum lateral value sent was nearly -40 and then movement goes back to zero once the target is centered. When the target is moved to the right of the robot, the maximum lateral value sent was nearly +40 and then movement goes back to zero once the target is centered. The blips that show immediate zero value are when the tracking was lost. The case for when visibility goes to zero accounts for this by sending an output which causes movement in the last direction of the image centroid, then tracking is regained quickly. If tracking is lost for too long (above a second), the value for movement goes to zero. Otherwise, the robot would continue to one direction without any knowledge of its position relative to the target. By stopping movement, there is a chance that the target may come back into view – especially when other sensors cause different movements (i.e. rotation from front-facing camera control).

The step response for the Goal Monitor Loop is given by the following image:

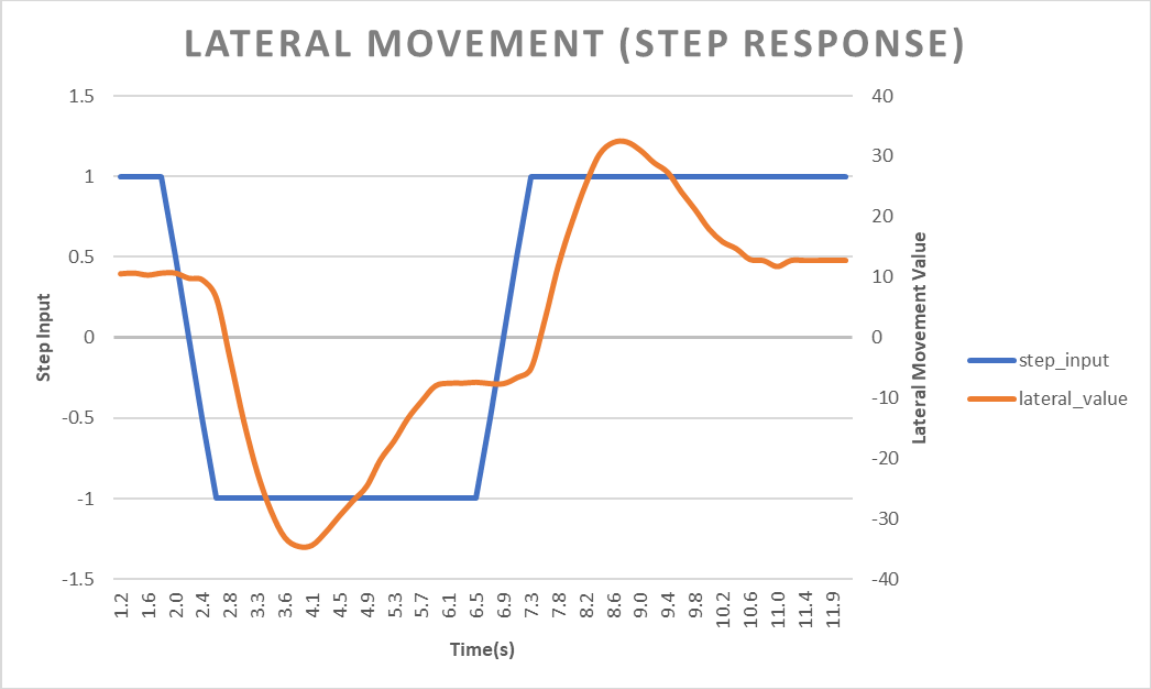



FIGURE 73: Goal Monitor Loop Step Response

The critical response for this loop, which was calculated from the end of the step input until robot steady-state was calculated to be 1.228 seconds.

For the Player Monitor Loop, the gain value found to produce desired repeatable results without sustained oscillation and overshoot was found to be around a value of 32. The following image illustrates the front panel output from the stepping 1 meter to the left, then back 1 meter to the right.

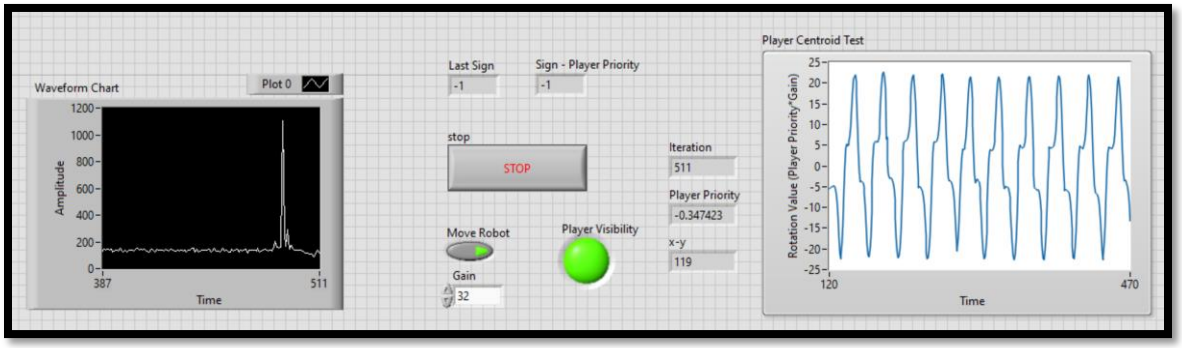



FIGURE 74: Player Monitor Loop Test Front Panel

Similar to the Goal Monitor Loop Test Front Panel, the chart on the left-hand side illustrates the loop execution time for each iteration – which again had an RMS value of nearly 180 milliseconds, however in this image, there is also a spike valued for loop hesitation.

Focusing on the chart on the right-hand side, we can see the repeatability of the control system.

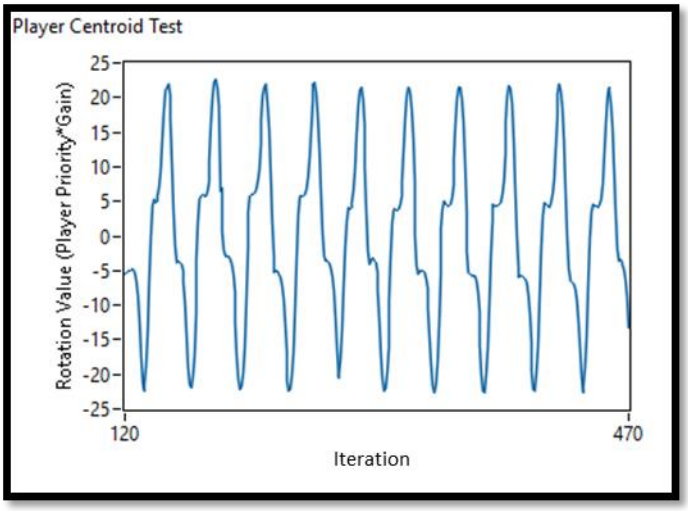



FIGURE 75: Goal Monitor Loop Test - Proportional Control Repeatability

When the player steps to the right, the maximum value sent to the rotational input of the Mecanum Mixer VI is around -22, causing counter-clockwise movement and then movement goes back to zero once the target is centered. When the player steps to the left, the value sent to the rotational input is around +22, causing clock-wise movement and then movement goes back to zero once the target is centered. In this test, tracking was not lost.

The step response for the Player Monitor Loop is given by the following image:

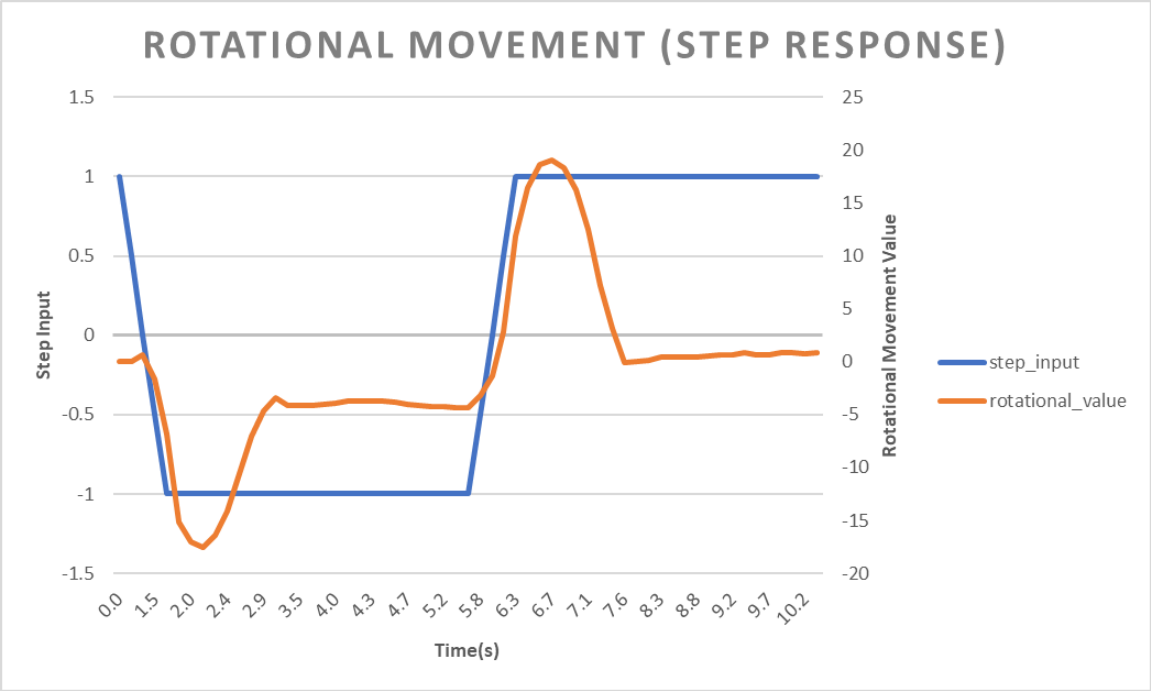



FIGURE 76: Player Monitor Loop Step Response

The critical response for this loop, which was calculated from the end of the step input until robot steady-state was calculated to be 0.807 seconds.

An example of the front panel used to illustrate the fusion result of the Goal Monitor Loop and Player Monitor Loop is given by the following image:

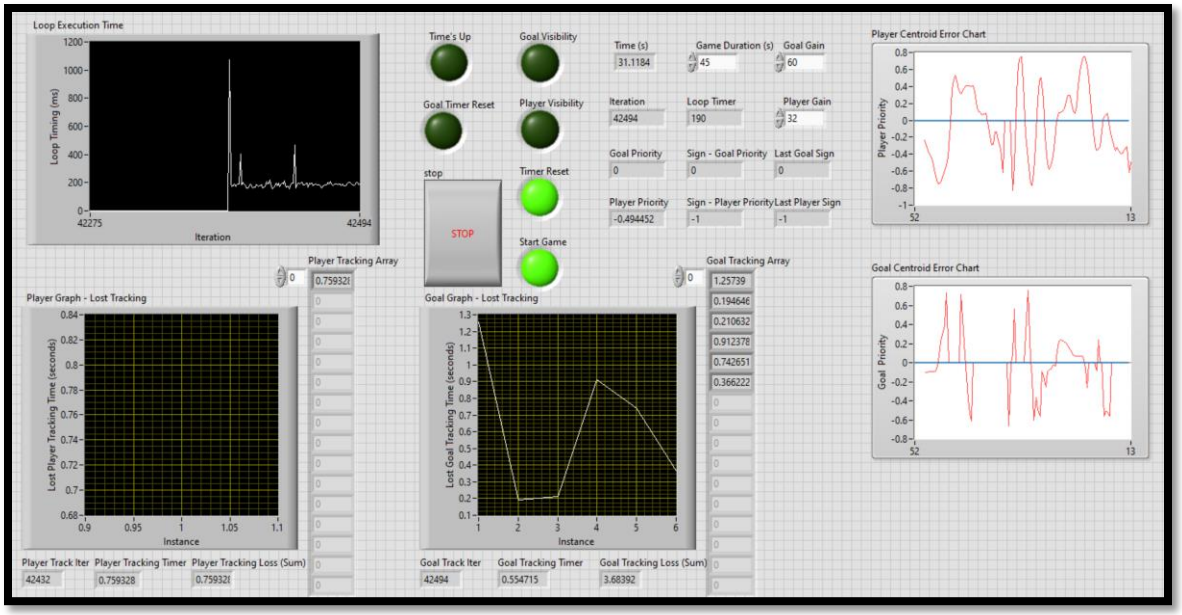



FIGURE 77: Goal and Player Monitor VI Front Panel

From the image above, we can see that for each round of play, the priority values also called the centroid error are shown in charts, as are the times for lost tracking of either the player or goal. In this illustration of a successful trial, the player tracking was only lost one time for around 0.76 seconds, while the goal target was lost in tracking several times, but none for longer than 1.26 seconds.

The sum of each array is also calculated to make a comparison between trials. The following table illustrates these values.

TABLE 5: Visibility Loss Summary Data (s)

| *Trial* | Player | # Times Lost | Goal | # Times Lost | Total Game Time |
|---|---|---|---|---|---|
| *1* | 0.56 | 1 | 2.79 | 3 | 20.27 |
| *2* | 0 | 0 | 3.39 | 4 | 20.27 |
| *3* | 0.37 | 1 | 1 | 1 | 20.36 |
| *4* | 0 | 0 | 2.22 | 3 | 20.32 |
| *5* | 0 | 0 | 2.83 | 3 | 20.34 |
| *6* | 0 | 0 | 1.21 | 2 | 20.2 |
| *7* | 0.77 | 2 | 0.19 | 1 | 20.25 |
| *8* | 0 | 0 | 2.36 | 2 | 20.33 |
| *9* | 0 | 0 | 2.23 | 4 | 20.23 |
| *10* | 0 | 0 | 2.25 | 4 | 20.24 |
| *Average* | 0.17 | | 2.25 | | 20.281 |

For an average game duration of 20.28 seconds, the player tracking was lost only 0.83% of the time, and the goal tracking was lost only 10.09% of the time. The average time per player visibility loss was 0.44 seconds, while the average time per goal visibility loss was 0.76 seconds.

The step response of the Goal Monitor Loop and Player Monitor Loop fusion VI is given by the following figure:

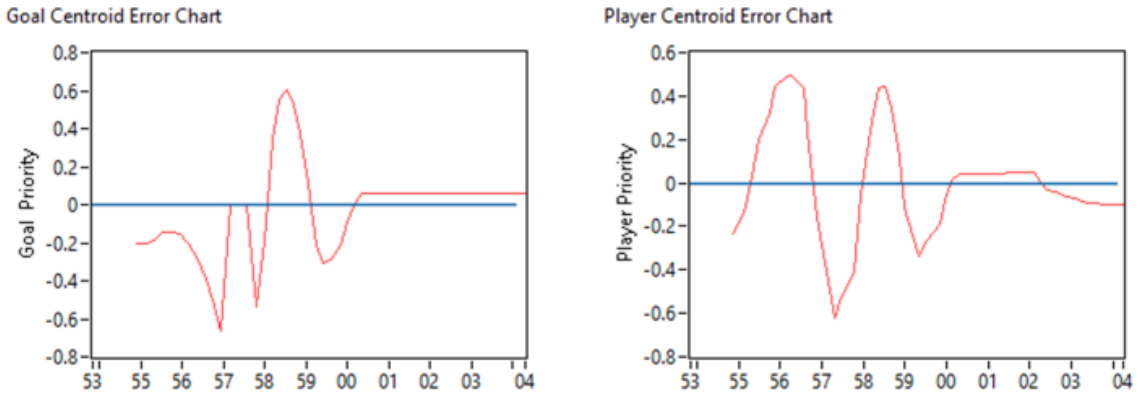



FIGURE 78: Goal and Player Monitor Loop Test

The critical response for this loop, which was calculated from the end of the step input until robot steady-state was calculated to be 2.23 seconds.

For the Distance Monitor Loop, the step response is given by the following image:

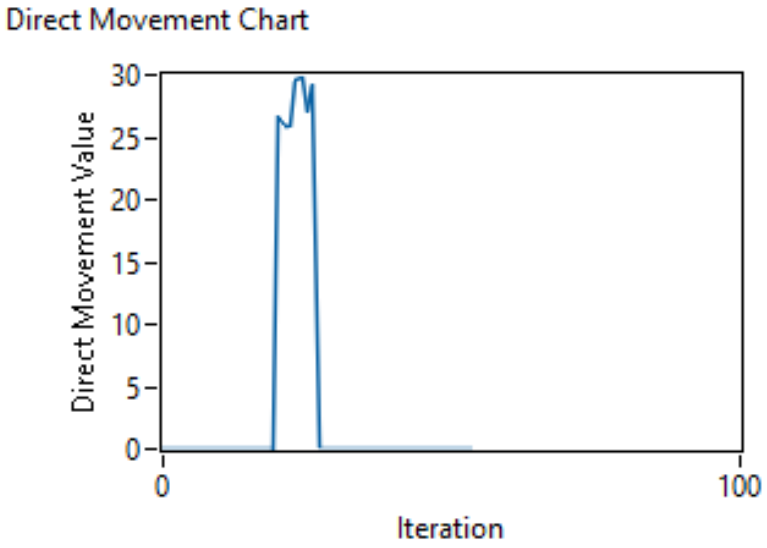



FIGURE 79: Distance Monitor Loop Step Response

The critical response for this loop, which was calculated from the end of the step input until robot steady-state was calculated to be 2.32 seconds.

For the fusion loop meeting all objectives of the thesis, combining the Player Monitor Loop, the Goal Monitor Loop, and the Distance Monitor Loop, the step response is given by the following image:

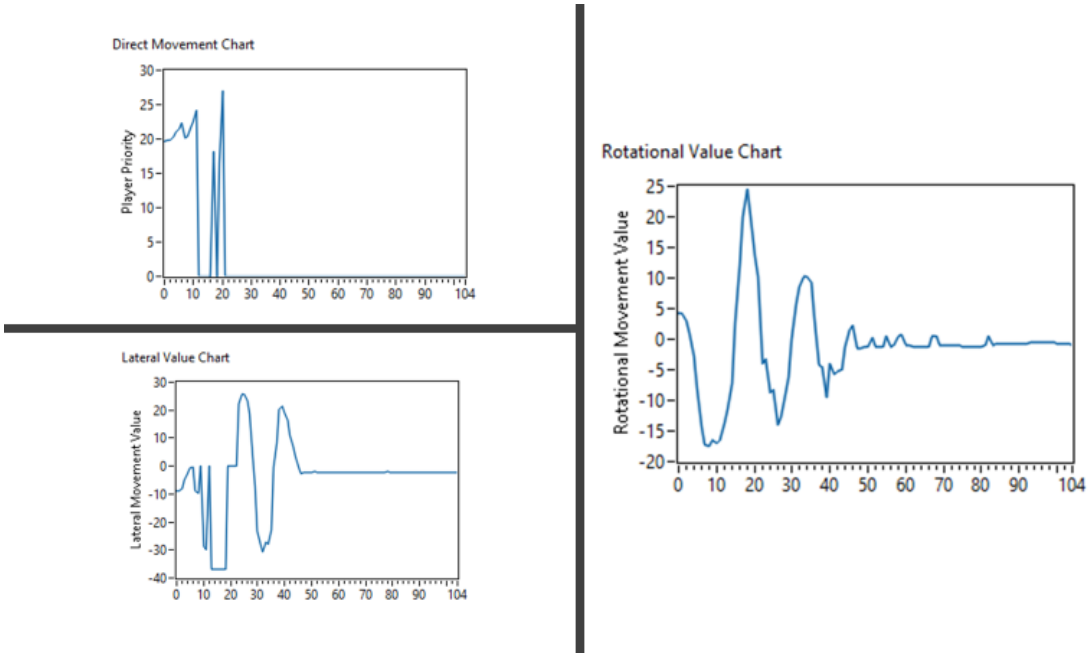



FIGURE 80: Full Fusion Step Response

The critical response for this loop, which was calculated from the end of the step input until robot steady-state was calculated to be 2.57 seconds.

# 7. DISCUSSION AND CONCLUSIONS

The primary contribution of this project is verifying that for a robot tracking a dynamic target (a dribbling soccer player) located to the front of the robot and a static target (the goal target) located to the rear of the robot, whose primary objectives are to face the player, maintain appropriate distance from the player, and place itself between the player and goal - mecanum motion could be controlled simply with just two sensors: one front-facing camera and one rear-facing camera.

The following image illustrates the realistic robot movement in reaction to the human player for the proof-of-concept vehicle created for this thesis.

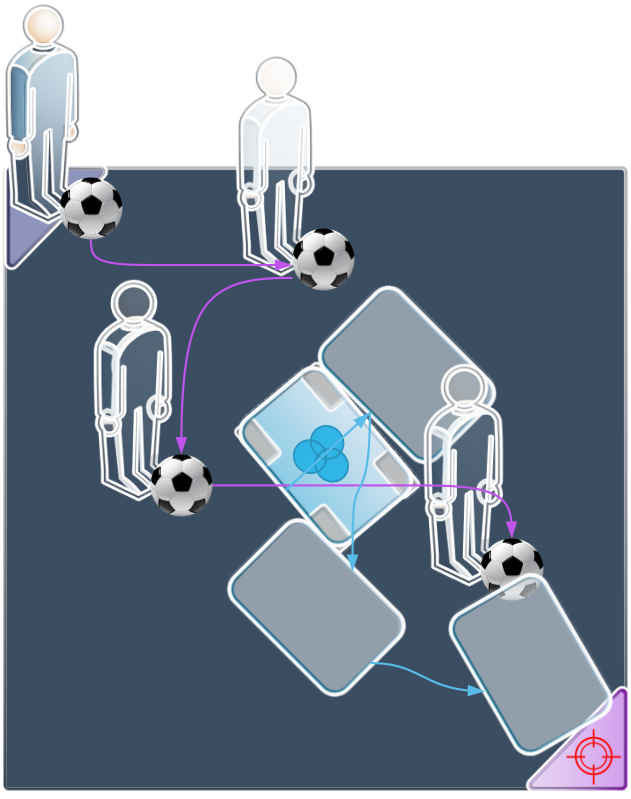


FIGURE 81: Futbot Reaction to Player

The algorithm was favored to track the human player since if the player is tracked well, even if the goal target is lost, it will commonly come back into view as the robot makes an appropriate movement to seek a position which puts itself between the player and the goal. Since the Futbot has been programmed to track the player with higher priority, the typical reaction to the human player moving to one side or the other during initial gameplay is for the Futbot to rotate and then strafe back towards the goal since visibility loss of the goal is more common. This approach naturally makes the Futbot move backward as it rotates and then moves back toward the centerline between the player and goal. The effectiveness is proven mostly heuristically through actual interaction with the robot, as well as the pass/fail criteria outlined in the Test Plan. Many videos were taken to prove success.

All tests were considered passing except for the final fusion test, which missed the 2.36-second pass criteria by 0.21 seconds. I believe that the wireless communication between the network created by the myRIO controller and the computer used for programming caused a delay that would not be necessary if the program were run as a startup on the myRIO and did not pass data back to the laptop. This wireless communication constraint is likely also the culprit of the spikes seen in the iteration loop execution times on the front panel.

To improve tracking, the goal target (24”x24” wood panel) could have been painted fully red instead of using paint swatches set up in a triangular formation – these were initially chosen for circle count and creating an equilateral triangle – which was potentially going to be used to for angular determination. After the discovery of the

faster binary morphology erosion/dilation method versus the FFT filter, the circles were no longer used for determining if the goal was visible – the histogram mean value and threshold were used instead.

The shirt tracked very well, but unfortunately, the available test environment was incredibly noisy. The vending machines in the background contained Mountain Dew bottles which just happened to be a similar color to the shirt. Also, any red that could be found in the frame was covered with white paper – this included fire extinguishers and exit signs. This actually worked to my benefit in the long run, since it required a more stringent application of thresholding values – limiting noise seen in the images after this filter was applied.

Future work of the robot could involve facial recognition for ensuring that it approaches only the player it has been tasked with defending. Also, adding LiDAR to the sensor set would help better inform of the environment and allow for some additional action beyond what was determined as this thesis' objective. Building a slam map of the environment, when the knowledge is not A Priori could allow the discovery of static vs. transient obstructions. Also, adding a Kalman Filter for error calculation could improve robot reaction and improve sensor fusion. With more time, it would have been fun to add pool noodles on a wiper motor with acoustic sensors so that they swiped at the ball when I came near.

I hope that this inspires students at UNC Charlotte to one day join the RoboCup and build robots that will one day truly be capable of beating the World Cup champions.

This project has challenged me beyond expectations and led to a result that is truly remarkable. It constantly reminded me to maintain the mantra of 'keep it simple.'

Improvements to this system could include Kalman filters, α/β filters, or multiple hypothesis trackers. In the robotic application of this thesis, the Player target moves, Goal target remains stationary – future work could involve tracking both robot proximity to either target or also the targets' proximity to one another determining which patterns are most effective. One could also apply machine learning techniques for common player approaches – using deep learning techniques based on reward structure to have the system develop predictive techniques of player target action to devise its own methods of improvement.

REFERENCES

A KUKA youbot simulation in USARSim. Freddy de Greef—PDF. (n.d.). Retrieved October 16, 2018, from https://docplayer.net/37097772-A-kuka-youbot-simulation-in-usarsim-freddy-de-greef.html

Awaludin, I., Hidayatullah, P., Hutahaean, J., & Parta, D. G. (2013). Detection and object position measurement using computer vision on humanoid soccer. *2013 International Conference on Information Technology and Electrical Engineering (ICITEE)*, 88–92. https://doi.org/10.1109/ICITEED.2013.6676217

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., … Leonard, J. J. (2016). Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. *IEEE Transactions on Robotics*, *32*(6), 1309–1332. https://doi.org/10.1109/TRO.2016.2624754

Chow, J., Lichti, D., Hol, J., Bellusci, G., & Luinge, H. (2014). IMU and Multiple RGB-D Camera Fusion for Assisting Indoor Stop-and-Go 3D Terrestrial Laser Scanning. *Robotics*, *3*(3), 247–280. https://doi.org/10.3390/robotics3030247

Chumerin, N., & Van Hulle, M. M. (2008). Cue and Sensor Fusion for Independent Moving Objects Detection and Description in Driving Scenes. In D. Mandic, M. Golz, A. Kuh, D. Obradovic, & T. Tanaka (Eds.), *Signal Processing Techniques for Knowledge Extraction and Information Fusion* (pp. 161–180). https://doi.org/10.1007/978-0-387-74367-7_9

Conceição, A. S., Moreira, A. P., & Costa, P. J. (n.d.). *MODEL IDENTIFICATION OF A FOUR WHEELED OMNI-DIRECTIONAL MOBILE*. 6.

Dhanapanichkul, S., & Chongstitvatana, P. (2005). Shadow compensation for computer vision in a robot soccer team. *IEEE International Symposium on Communications and Information Technology, 2005. ISCIT 2005.*, *1*, 13–16. https://doi.org/10.1109/ISCIT.2005.1566788

Endres, F., Hess, J., Sturm, J., Cremers, D., & Burgard, W. (2014). 3-D Mapping With an RGB-D Camera. *IEEE Transactions on Robotics*, *30*(1), 177–187. https://doi.org/10.1109/TRO.2013.2279412

Haddadin, S., Luca, A. D., & Albu-Schäffer, A. (2017). Robot Collisions: A Survey on Detection, Isolation, and Identification. *IEEE Transactions on Robotics*, *33*(6), 1292–1312. https://doi.org/10.1109/TRO.2017.2723903

Lin, L.-C., & Shih, H.-Y. (2013). Modeling and Adaptive Control of an Omni-Mecanum-Wheeled Robot. *Intelligent Control and Automation*, *04*(02), 166–179. https://doi.org/10.4236/ica.2013.42021

Luettel, T., Himmelsbach, M., & Wuensche, H. (2012). Autonomous Ground Vehicles—Concepts and a Path to the Future. *Proceedings of the IEEE*, *100*(Special Centennial Issue), 1831–1839. https://doi.org/10.1109/JPROC.2012.2189803

Medina, J. R., & Hirche, S. (2018). Considering Uncertainty in Optimal Robot Control Through High-Order Cost Statistics. *IEEE Transactions on Robotics*, *34*(4), 1068–1081. https://doi.org/10.1109/TRO.2018.2830374

Mezouar, Y., & Chaumette, F. (2002). Path planning for robust image-based control. *IEEE Transactions on Robotics and Automation*, *18*(4), 534–549. https://doi.org/10.1109/TRA.2002.802218

Mur-Artal, R., & Tardós, J. D. (2017). ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics*, *33*(5), 1255–1262. https://doi.org/10.1109/TRO.2017.2705103

Patole, S. M., Torlak, M., Wang, D., & Ali, M. (2017). Automotive radars: A review of signal processing techniques. *IEEE Signal Processing Magazine*, *34*(2), 22–35. https://doi.org/10.1109/MSP.2016.2628914

Shiller, Z., & Gwo, Y.-. (1991). Dynamic motion planning of autonomous vehicles. *IEEE Transactions on Robotics and Automation*, *7*(2), 241–249. https://doi.org/10.1109/70.75906

Stauffer, C., & Grimson, W. E. L. (2000). Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(8), 747–757. https://doi.org/10.1109/34.868677

Tenney, R. R., & Sandell, N. R. (1981). Detection with Distributed Sensors. *IEEE Transactions on Aerospace and Electronic Systems*, *AES-17*(4), 501–510. https://doi.org/10.1109/TAES.1981.309178

Vidal, R., Shakernia, O., Kim, H. J., Shim, D. H., & Sastry, S. (2002). Probabilistic pursuit-evasion games: Theory, implementation, and experimental evaluation. *IEEE Transactions on Robotics and Automation*, *18*(5), 662–669. https://doi.org/10.1109/TRA.2002.804040

Wu, H., Bateman, R., Zhang, X., & Lind, M. (2018). Functional Modeling for Monitoring of Robotic System. *Applied Artificial Intelligence*, *32*(3), 229–252. https://doi.org/10.1080/08839514.2018.1447431

Zanchettin, A. M., Ceriani, N. M., Rocco, P., Ding, H., & Matthias, B. (2016). Safety in human-robot collaborative manufacturing environments: Metrics and control. *IEEE Transactions on Automation Science and Engineering*, *13*(2), 882–893. https://doi.org/10.1109/TASE.2015.2412256

# APPENDIX

## Data Fusion Lexicon Definitions per JDL:

*Sources of Information* – local or network sensors providing input data; reference information or A Priori knowledge of the environment.

*Human-Computer Interaction (HCI)* – peripherals for human input and communication.

*Source Pre-processing (Process Assignment)* – pre-screening data to allow fusion to focus on the most relevant data.

*Level 1 Processing (Object Refinement)* – Synthesis of pre-processed data to make an assessment of object features and kinematics for discrimination and tracking via four key functions: data alignment, data/object correlation, feature association, and entity classification.

*Level 2 Processing (Situation Refinement)* – relates the entities being tracked to their environment with higher-level pattern inferences.

*Level 3 Processing (Threat Refinement)* – uses the current situational understanding to predict future events making inferences of intent and predicted opportunity.

*Level 4 Processing (Process Refinement)* – ensures optimal performance of data fusion system via four key functions: monitoring performance, identifying deficits in data quality or type, determining which sources are necessary for specific object feature

identification, and allocating and directing resources of the system to optimize application objectives.

*Data Management* – known as a support function for the various levels of fusion processing, managing data for efficient storage, retrieval, and protection is a significant undertaking due to the potentially vast types and amount of data accrued by the sensor systems.

Component Technical Specifications:

Mecanum Wheels

- Steel Side Plates riveted to black Polycarbonate Core
- Diameter: 8"
- Load Capacity: 500 lbs per wheel
- Weight: 4.58 lbs
- Width Across Middle: 3.50 in.
- Nylon Core Rollers - 80A durometer TPU over-mold
- Roller Durometer: 80A
- Rollers: 12

CIM Motor

TABLE 6: Motor Parameters

| | |
|---|---|
| Free Current: | 2.7 AMPs |
| Free Speed: | 5,310 RPM |
| Maximum Power: | 337 Watts |
| Stall Current: | 133 AMPs |
| Stall Torque: | 343.4 in-oz |
| Voltage: | 12 Volts |

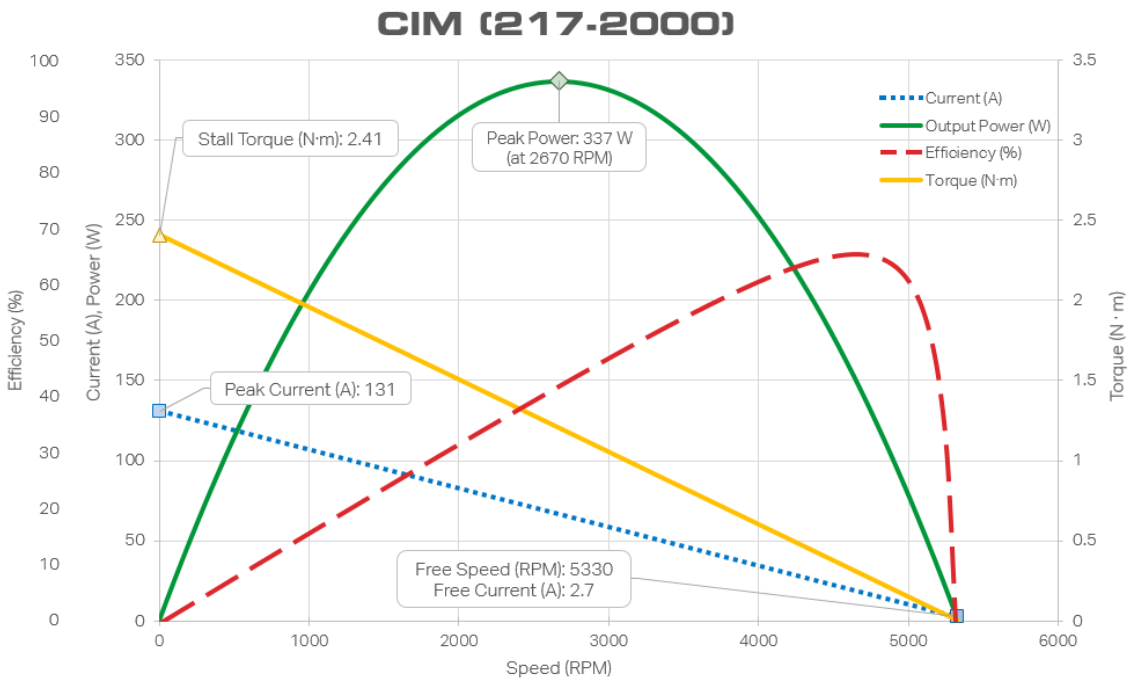



FIGURE 82: CIM Motor Torque-Speed Curve ("CIM Motor—VEXpro Motors—VEX Robotics," n.d.)

Toughbox Mini Gearbox

- Angled Plate

- 12.75:1 gear ratio
- Stall Torque:  91.22 ft-lbs
- Free Speed:  416.47 rpm
- Wheel Speed for 8” Wheel:  14.54 ft/s

US Digital E4P Optical Encoders

- Counts Per Revolution:  250
- Bore:  ¼”
- No Index
- Single-Ended Output

Talon SRX Motor Speed Controllers

TABLE 7:  Talon SRX Technical Parameters

| | | |
|---|---|---|
| Voltage: | Nominal:  12V | Min/Max:  6-28V |
| Current: | 60A continuous | 100A surge (2s) |
| PWM Input: | Pulse (high time):  1-2 ms nominal | Rate (period):  2.9-100 ms |
| PWM Output: | Chop Rate (switching frequency) | 15kHz |
| Minimum Throttle: | (Deadband) | 4% |

Battery

- Sealed Lead-Acid Type
- 12 Volts
- 18 Amp-hours

- Weight:  12.5 lbs

Power Distribution Panel

- 12V DC Input
- 12V DC Output
- 8 Channels:  40A protection
- 8 Channels:  20/30A Protection
- 1 Channel:  20A protection for Voltage Regulator Module
- 1 Channel:  10A for myRIO
- 1 CAN Bus:  feedback to myRIO

Voltage Regulator Module

- 12V DC input
- 2 Channels:  12V, 1.5A limit (2A peak)
- 2 Channels:  12V, 500mA limit
- 2 Channels:  5V, 1.5A limit (2A peak)
- 2 Channels:  5V, 500mA limit

TABLE 8:  myRIO PWM Channel Layout

| | | | |
|---|---|---|---|
| Left Front (LF) | Channel A:  PWM 0 | Right Front (RF) | Channel A:  PWM 1 |
| Left Rear (LR) | Channel B:  PWM 0 | Right Rear (RR) | Channel B:  PWM 1 |

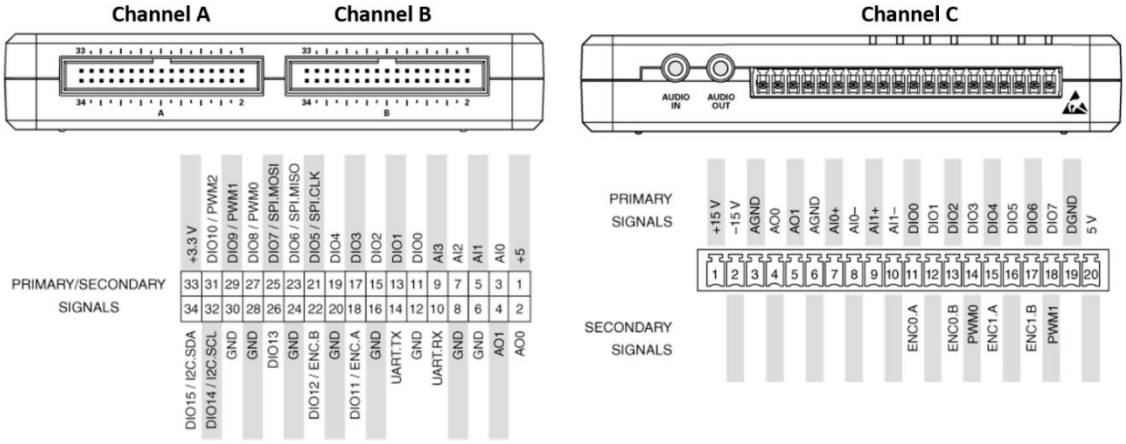



FIGURE 83: myRIO 1900 Channel Pinouts (National Instruments)