A red and black logo

Description automatically generated

Eindhoven University of Technology

Department of Mathematics and Computer Science

Mechatronics System Design

**Technical Report on Assistant Referee System**

**21-March-2024**

**TABLE OF CONTENTS**

**SNO DESCRIPTION PAGE NO**

[**1** **PURPOSE** 3](#_Toc162202519)

[**2** **REFERENCE DOCUMENTS** 3](#_Toc162202520)

[**3** **INTRODUCTION** 3](#_Toc162202521)

[**4** **FEASIBILITY STUDY** 3](#_Toc162202522)

[**5** **SYSTEM ARCHITECTURE** 4](#_Toc162202523)

[**6** **SCOPE OF THE PROJECT** 4](#_Toc162202524)

[**7** **PROCEDURES** 5](#_Toc162202525)

[**8** **IMPLEMENTATION** 5](#_Toc162202526)

[8.1 Ball Out of Play 5](#_Toc162202527)

[8.1.1 Verification of BOOP 7](#_Toc162202528)

[8.2 Last Touch 10](#_Toc162202529)

[8.3 User Interface 11](#_Toc162202530)

[**9** **VALIDATION** 11](#_Toc162202531)

[**10** **SUGGESTIONS FOR IMPROVEMENT** 11](#_Toc162202532)

[**11** **RECOMMENDATIONS** 12](#_Toc162202533)

1. PURPOSE

The purpose of this document is to provide a technical report for the project “*Assistant Referee System*” done by EngD Trainees MSD 2023 at TU/e.

1. REFERENCE DOCUMENTS

|  |  |
| --- | --- |
| Document Number | Document Name |
| MSD/FS | Feasibility Study |
| MSD/SA | System Architecture |
| MSD/VR | Validation Report |

1. INTRODUCTION

The aim of the project is to assist human referees in soccer robot games by developing an assistant referee system which is more reliable and accountable in critical decision-making process. This project has been started by Eindhoven University of Technology for the RoboCup Middle Sized League (MSL). Different technologies have been used since then including computer vision, Machine Learning and Drone Based systems to enable monitoring of the soccer game. By leveraging smart heuristics along with these technologies such as tracking cameras and machine learning, we wish to deliver a software application that conveys the violation of rules to the referee who may then use the inputs to pause the game.

The rule that we choose to focus on here is ball out of play and detecting collision of the ball with robots and ruling out the robot that touched it last before going out of play.

1. FEASIBILITY STUDY

In order to develop this particular rule, we considered multiple programming languages including:

* C++
* MATLAB
* Python

We also considered multiple sensors to observe the game:

* Sound sensors
* Temperature sensors
* Force sensor (integrated with the ball)
* RADAR
* Ultra-wide band sensor
* CCTV cameras
* Stereo cameras
* Time-of-flight cameras
* Infrared Cameras + Passive Retroreflective Markers

Detailed feasibility studies on the choice of programming languages and sensors may be

found by following the links “[GitHub Feasibility Study](https://github.com/KareemGhedan/MSD-AutoRef-2023/blob/main/Reports/General/Feasibility%20Study.docx)”

Bases on the scope of our project, we opt for Opti track motion capture system as a best option to achieve the goals of accuracy and robustness. Opti Track tracking systems are the world’s choice for low latency and precision 6DoF tracking.

1. SYSTEM ARCHITECTURE

The System Architecture document can be found in “[GitHub System Architecture Document](https://github.com/KareemGhedan/MSD-AutoRef-2023/tree/main/Reports/System%20Architecture)”. This document aims to offer a thorough insight into the design of the system, enabling stakeholders to grasp the system’s functionality, needs and requirements and system context diagram. Starting from the stakeholder’s needs and translating those to requirements and specifications, the document outlines how the project proceeds to solution domain.

To develop the system architecture, the system's thinking process was employed. The flowchart of the process is depicted:

A diagram of a diagram

Description automatically generated

1. SCOPE OF THE PROJECT

To ensure precise data acquisition and real-time tracking of collisions involving turtles/robots and the ball during game proceedings, our approach prioritizes the utilization of advanced sensor technology, such as OptiTrack systems. These sensors are strategically positioned to capture the dynamic interactions across the playing field. Our algorithm's primary objective is to devise a robust mechanism for detecting collisions with the ball and determining the last entity to make contact, crucial for decision-making in soccer gameplay. We particularly focus on scenarios involving collisions between turtles/robots and the ball, which are fundamental aspects influencing game dynamics. By centering our validation efforts on collision detection and tracking through OptiTrack sensors, we can ensure the reliability and effectiveness of our methodology across diverse game situations. The rationale behind selecting this approach can be outlined as follows:

* Complexity Simplification: Implementing OptiTrack sensors offers a simplified approach compared to traditional methods, reducing technological complexity while ensuring accurate collision detection.
* Accessibility of Technology: OptiTrack sensor technology is readily available and accessible, facilitating seamless integration into existing stadium infrastructure without significant hardware modifications.
* Versatility and Integration: The chosen approach does not directly correlate with specific rule violations, enabling effortless integration with past and future implementations while ensuring adaptability across various game scenarios.

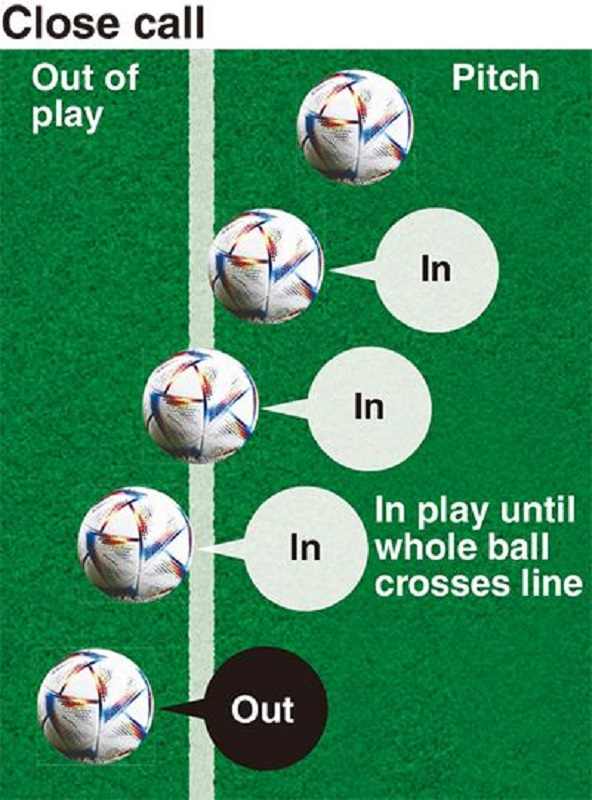
1. PROCEDURES

Utilizing Opti Track technology for ball and robot tracking, our collision detection system employs three distinctive methodologies:

* The cameras are used to acquire the position of the ball center and robot position. The data was then filtered to remove noise.
* A robust machine learning approach is embraced, employing an ensemble of 80 estimators within a random forest classifier. This model utilizes features including vectorized distances between robots and the ball, along with the Euclidean distance. This machine learning method is favored for its resilience and ability to handle intricate interactions. The Implementation methodology will be explained later in next section.

1. IMPLEMENTATION
   1. Ball Out of Play

Th**e**  requirement **for BOOP (Ball out of play)**  stated as follows: “AssistRef must have a distance error between the ball and the line of ±1.5 cm at max.”

First of all, the ball is out of play is when the whole ball is over the side line of the soccer field. An example is seen in Figure 1.

*Figure 1: Different example of the ball in and out of play.*

The OptiTrack system gives the center of the ball for each measurement. Our solution involves comparing this value to the outside lines of the field. In order for this to work, OptiTrack needs to know the coordinates of the lines. This is done using the OptiTrack CS-400, a calibration tool. The four corners of the field are calibrated using this tool. Which gives the following coordinates ([X, Z]) [ [-0.052, 4.109], [6.099, 4.136], [-0.024, -4.095], [6.127, -4.080] ] in m. A linear line is drawn between the four corners, covering the full side lines of the field.

The radius of the ball also has to be taken into account as the measurement from OptiTrack is the center of the ball and the rule says the whole ball needs to be over the side line. The radius of the ball is taken to be 11 cm, the average of the RoboCup MSL rules[[1]](#footnote-2).

Now we can calculate if the ball is out of play. If the center of the ball measurements plus the radius of the ball is outside of the calibrated lines, the ball is out of play. If it is within these calibrated lines, the ball is in play. The code for ball out of play is given in “[GitHub Link for Code](https://github.com/KareemGhedan/MSD-AutoRef-2023/tree/main/Code%20Base/BOOP)”

* + 1. Verification of BOOP

A black box on a green carpet

Description automatically generatedThe method to verify our BOOP system is as follows. First, a box is placed at the side of the outer lines of the field, as seen in Figure 2.

Figure 2: Box at the side of the field lines.

Secondly, the ball is kicked against this box and the trajectory of the ball is recorded. This way, we have the ground truth of where the ball should be when hitting the box and the position of the ball from our system. Comparing these two values with each other will give the distance error.

A football field with a yellow ball

Description automatically generatedFour different box positions are considered as shown in Figure 3 with an example trajectory of the ball. Only the right half of the field is considered as only that part of the field is covered by the Opti Track system.

Figure 3: RoboCup MSL soccer field with four black squares representing the different positions of the boxes. The yellow circle represents the ball and the dotted line the trajectory of the ball. The origin is placed in the middle of the field.

As the boxes are located along the axis of the origin, we only have to look at one coordinate. In this way, the minimal or maximal position along a certain axis of the ball is the position of the ball where the ball hits the box. Using this, the radius of the ball and the calibrated positions of the lines with respect to the origin, the error can be calculated. An example calculation of the distance error is shown below of the example trajectory shown in Figure 3.

where are all the Z-coordinate measurements of the middle of the ball, the radius of the ball and the Z-coordinate of plane of the box that is being hit by the ball. The box is placed in four different positions and for every positions, eight recordings are taken at different velocities of the ball hitting the box. This means that 32 error samples are gathered. Figure 4 shows the distance error results in a box plot. The shape suggests a normal distribution. The Chi-squared test is used to test the data for normality. The test does not reject the null hypothesis that the data is normally distributed as the p value is larger than 0.05, 0.0513. This means that a normal distribution can be assumed for this data.

A graph with blue squares

Description automatically generatedWhile assuming a normal distribution, the 95% confidence intervals of the parameters of the normal distribution is as follows: μ = [-1.26·10-2 3.6·10-3] and σ = [1.80·10-2 2.98·10-2], where μ is the mean and σ is the standard deviation in meters. Taking the highest values of both and 2σ of the normal distribution, the error is -1.26·10-2 ± 5.96 ·10-2 m.

Figure 4: Histogram of distance error of all 32 samples of the verification test.

This means that we did not meet the requirement of maximal distance error of ± 1.5cm.

*A graph of a ball position

Description automatically generated*Figure 5 and 6 show the Opti Track measurements of the middle of the ball plus the radius of the ball and the field line where the ball is being rebounded.

A graph with lines and numbers

Description automatically generated*Figure 5: Ball positions measurements with the field lines.*

*Figure 6: Ball positions measurements with the field lines*.

Figure 6 shows good measurements, while Figure 5 shows missing measurements when the ball is just rebounded by the box. This miss in measurements causes the unexpected high error margin that is verified. The exact reason is unknown to us, but one way to increase the measurement accuracy is to add more markers to the ball and make the markers more round. The more round the markers are, the better the measurements. Furthermore, in Figure 5 it is seen that there are some outlier measurements. In order to mitigate this, the ball is only considered to be out of play when five consecutive measurements are outside of the line.

In conclusion, the solution for the ball out of play is comparing the middle of the ball plus the radius with the calibrated outside lines of the field. The error that came out of the verification is -1.26·10-2 ± 5.96 ·10-2 m, which does not meet our requirement of being within ± 1.5 ·10-3 m. This is due to missed measurements of the Opti Track system. To mitigate outlier measurements, the ball is out of play when five consecutive measurements are outside of the line.

* 1. Last Touch

**Touch Detection**

To enable touch detection, a binary classifier has been employed, following this pipeline:

**Data Acquisition:** Utilizing Opti Track as our input sensor, data was collected from various scenarios involving touch and no-touch interactions between a robot and a ball.

**Data Preprocessing:** several preprocessing steps were then performed on the collected data, including the removal of missing or duplicated values.

**Feature Engineering:** Through feature engineering, dataset was refined to focus on four key features:

* **X\_dist:** The difference between the X-coordinate of the ball and the X-coordinate of the robot (X\_Turtle).
* **Y\_dist:** The difference between the Y-coordinate of the ball and the Y-coordinate of the robot.
* **Z\_dist:** The difference between the Z-coordinate of the ball and the Z-coordinate of the robot.
* **Distance:** The Euclidean distance between the ball and the robot.

These features were selected for their ability to capture relevant spatial relationships between the robot and the ball, crucial for touch detection.

**Model Selection:** Four different machine learning algorithms were trained, including Multi-Layer Perceptron (MLP), Random Forest, Decision Tree, and Support Vector Machine on the dataset. Each algorithm was evaluated using metrics including Accuracy, Precision, Recall, and F1-score. Following this comprehensive evaluation the model which exhibited the best performance across these metrics was selected. The analysis identified **Random Forest** as the optimal choice for touch detection task, demonstrating superior performance compared to the other models.

**Performance Evaluation:** Finally, the performance was evaluated of the chosen model to assess its efficacy in detecting touch accurately in **real-time data**. Evaluation metrics such as accuracy, precision, recall, and F1-score, as well as the ROC Curve and Area Under Curve (AUC), were employed to comprehensively gauge the effectiveness of the model. The code is present here “[GitHub Link for Code](https://github.com/KareemGhedan/MSD-AutoRef-2023/tree/main/Code%20Base/Last%20Touch)”

* 1. User Interface

The communication system use simple UI to communicate the decision to the referee and received input of the game state from the referee. The code for the UI is given in this link “[GitHub Code for UI](https://github.com/KareemGhedan/MSD-AutoRef-2023/tree/main/Code%20Base/MainProgram_withGameStates/helper_functions)”.

1. VALIDATION

The validation process aims to ensure the accuracy, reliability, and adaptability of the assistant referee system in various game scenarios, validating the functional requirements of the system. It is important to note that the main concern is the real-world performance and meeting user expectations.

* Real-World Testing: Preparation involves ensuring seamless integration and functionality of the Assistant Referee System, including system setup, input data validation, independence verification, and collaborative testing. The system's ability to receive sensor data and interpret live game data accurately is validated.
* Implementation: An example match is played to evaluate the system's ability to distinguish the last team to touch the ball, make real-time decisions, and communicate effectively.
* Edge Cases and Failure Analysis: Attention is given to challenging scenarios, identifying edge cases, and testing the system's resilience in difficult conditions.
* Statistical Metrics and Benchmarking: Data analysis techniques are employed to evaluate decision-making accuracy and performance, benchmarking against human referee’s decisions.

The detailed validation report is present here: [GitHub Validation Report](https://github.com/KareemGhedan/MSD-AutoRef-2023/blob/main/Reports/Verification%20and%20Validation/Validation%20Methods.pdf)

1. SUGGESTIONS FOR IMPROVEMENT

The main contribution of the project this year is the code for last touch, and certain issues can be identified with the approaches used. For regression and clamping, the trajectory of the ball wasn't as expected. This could either be due to some inherent issues with OptiTrack or the changes to its codebase made by the Tech United team to detect the ball. For the machine learning approach, the current method only looks at vectorized and Euclidean distances between the ball and the robots, but to make the method more robust, we could incorporate the quaternions of the robots to the models.

1. RECOMMENDATIONS

For continuing the work of the current year, there are a couple of recommendations that could be worked upon. One is to incorporate quaternions into the machine learning model. The other is to look into the code for the Opti Track that tracks the ball to make sure that the data obtained is cleaner.

For future work, multiple rules could be looked at including throw-in, goal kick and corner kick. These would be a continuation to the current work.

1. ASADA, Minoru, et al. Middle Size Robot League Rules and Regulations for 2024. *RoboCup MSL Technical Committee*, 2024. [↑](#footnote-ref-2)