FH | JOANNEUM
Data and Information Science

Data and Information Science

FH JOANNEUM – University of Applied Sciences,
Department für angewandte Informatik,
Eckertstraße 30i, 8020 Graz, Austria

Projektarbeit (DAT_WS2020_3)

Technische Dokumentation

Team Fußball

Danninger Philipp

Hasenbichler Timo

Lagger Christian

Pützl Mathias

Inhaltsverzeichnis

Setup 4

Projektutensilien 4

Roboter 4

Spielfeld 5

Spielball 7

Overhead-Kamera & Beleuchtung 8

Router & Datenbank Raspberry Pi 9

Technische Umsetzung 10

Softwarearchitektur – „Big Picture“ 10

Objektstruktur 11

Bilderkennung 13

ArUco-Marker und Koordinatensystem 13

HSV 13

Kommunikation 15

Datenbank & Logging 16

Vordefinierter Entscheidungsbaum 18

Data Science 23

Data Labeling für Supervised Learning 23

Regression mit XGBoost 23

Random Forest Regression 24

Lasso Regression 24

Empfehlungen 25

Quellen 25

Abbildungsverzeichnis

Abbildung 1: Roboter mit ArUco-Marker 5
Abbildung 2: Roboter mit ArUco-Marker gesehen von oben 5
Abbildung 3: Roboter gesehen von Vorne 5
Abbildung 4: Geeignete Akkus der Roboter mit entsprechenden Ladegeräten 5
Abbildung 5: Tor des Spielfelds mit aufgeklebten ArUco-Markern 6
Abbildung 6: Zusätzliche Banden Elemente, um Deadlocks in Spielfeldecken zu verhindern 6
Abbildung 7: Zusätzliche Holzleisten entlang der Bande, um das Abrollen des Balles zu verbessern 6
Abbildung 8: „Nullpunkt-Turm“ 6
Abbildung 9: Verwendete Quelldübel 6
Abbildung 10: Funktionsweise der Holzleisten entlang der Spielfeldbanden 6
Abbildung 11: Der mitgelieferte Ball des Spielfeldes erwies sich auf Grund seiner Farbgebung als ungeeignet 7
Abbildung 12: Tennisball diente zum ersten Test der Objekterkennung eines Balles in einer einheitlichen Farbe 7
Abbildung 13: Die finale Wahl: Drei Schaumstoffbälle in den Farben Rot, Blau und Gelb 7
Abbildung 14: Initialgelieferte Overhead-Kamera 8
Abbildung 16: Stativ für Overhead-Kamera, positioniert auf Höhe der Mittellinie des Spielfelds 8
Abbildung 17: "Flutlicht" - Fotostudioleuchten 8
Abbildung 18: Aufbau des Stativs für die Befestigung der Overhead-Kamera 8
Abbildung 19: Router für lokales Netzwerk, in welchem die Roboter agieren 9
Abbildung 20: Raspberry Pi als Host für die Datenbank der übertragenen Spieldaten 9
Abbildung 21: Softwarearchitektur als „Big Picture“ des Projektes 10
Abbildung 22: HSV-Filterraum 14
Abbildung 23: HSV-Regler in Default-Einstellungen 15
Abbildung 24: HSV-Einstellungen, um roten Ball zu erkennen 15
Code Snippet 1 - Setup DB Client/Server 17
Code Snippet 2 - Datenbank, Tabellen, User und Berechtigung Erstellung 18
Abbildung 25: Spielsituationen in eigener Hälfte 21
Abbildung 26:Spielsituationen in gegnerischer Hälfte 21
Abbildung 27:Vordefinierter Entscheidungsbaum, basierend auf den identifizierten Spielsituationen 22

Setup

Projektutensilien

Das Projektsetup besteht allgemein aus den folgenden Gegenständen:

Ressourcen-/Kostenart	Mengeneinheit
• Roboter	• 4
• Raspberry Pi	• 6
• Spielfeld (Tore und Banden)	• 1
• Webcam (Overhead)	• 2
• Holzbanden	• 4
• Holzleisten	• 7
• 256 GB SDCard	• 6
• Stativ	• 1
• Router	• 1
• Lichtquellen	• 2

Roboter

Für das Projekt wurden dem Team vier Roboter der Marke *Adeept AWR 4WD WiFi Smart Robot Car Kit for Raspberry Pi 3 Model* zur Verfügung gestellt. Die Roboter sind mit vier gleichstarken Motoren, einer Frontkamera sowie einer Ultraschallmessung ausgestattet. Zusätzlich sorgen Lichtquellen im vorderen Bereich des Roboters für eine visuelle Unterscheidung, da mehrere Farben ausgewählt werden können. Der Aufbau dieser Roboter erwies sich als herausfordernd, da die Anleitung einige fehlerhafte Bildfolgen enthielt. Anzumerken ist allerdings, dass sich die Schrauben und dazugehörigen Muttern bei der Bewegung des Roboters sehr leicht lösen. Dies kann dazu führen, dass der Roboter in seine „größeren“ Einzelteile zerlegt werden muss, um die losen Schrauben wieder zu installieren. Abschließend wurden zu Beginn des Projektes falsche Batterien geliefert, welche für den Antrieb der Roboter nicht verwendet werden konnten. Es wurden darauf hin, geeignete Akkus mit entsprechenden Ladegeräten bestellt. Der Roboter und die Akkus sind in den Abbildungen 1 bis 4 ersichtlich. Diese zeigen bereits die befestigten ArUco-Marker auf den Robotern.

Abbildung 1: Roboter mit ArUco-Marker

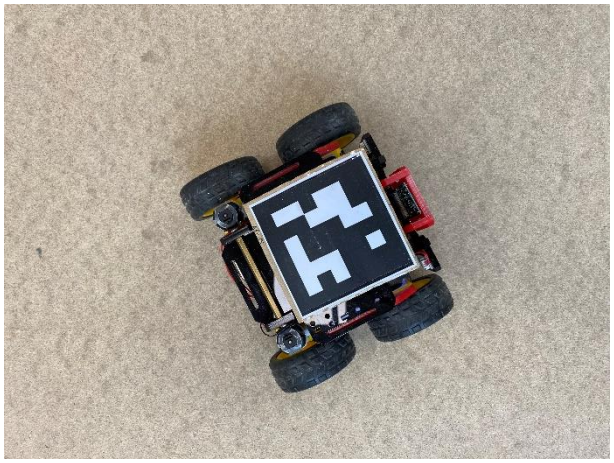

Abbildung 2: Roboter mit ArUco-Marker gesehen von oben

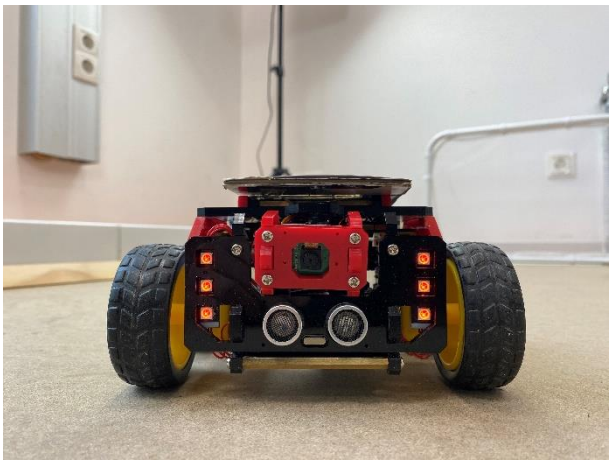

Abbildung 3: Roboter gesehen von Vorne

Abbildung 4: Geeignete Akkus der Roboter mit entsprechenden Ladegeräten

Spielfeld

Das Spielfeld der Roboter besteht aus einer schweren 143 cm breiten und 275 cm langen Holzplatte. Auf dieser Holzplatte wurden die bestellten Bandenteile des Spielfeldes mittels Vorbohrungen und Quelldübeln (Abbildung 9) befestigt, so dass sich die Banden nicht mehr bewegen lassen. Die 10 cm hohen Bandenelemente spannen, nach deren Befestigung, ein 140 cm breites und 250 cm langes Spielfeld auf, in welchem sich die Roboter bewegen können. Die Tore sind in diese Bandenteile integriert und umfassen eine Tor-Länge von 58 cm (Abbildung 5). In einer Ecke wird dann der sogenannte „Nullpunkt-Turm“ aufgestellt, welcher einen ArUco-Marker auf einer höheren Kartonbox bezeichnet. Dieser ArUco-Marker wird von der Overhead-Kamera als Ursprungspunkt des über das Spielfeld gespannte Koordinatensystem erkannt (Abbildung 8).

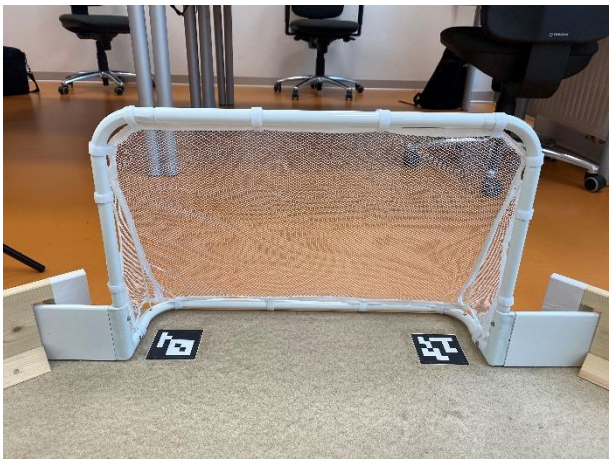

Abbildung 5: Tor des Spielfelds mit aufgeklebten ArUco-Markern

Abbildung 6: Zusätzliche Banden Elemente, um Deadlocks in Spielfeldecken zu verhindern

Abbildung 7: Zusätzliche Holzleisten entlang der Bande, um das Abrollen des Balles zu verbessern

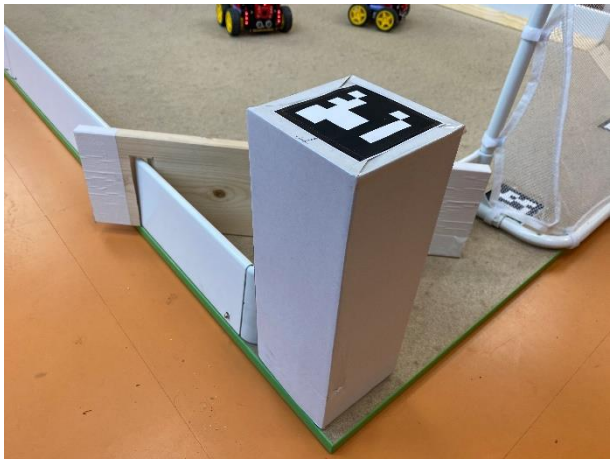

Abbildung 8: „Nullpunkt-Turm“

Abbildung 9: Verwendete Quelldübel

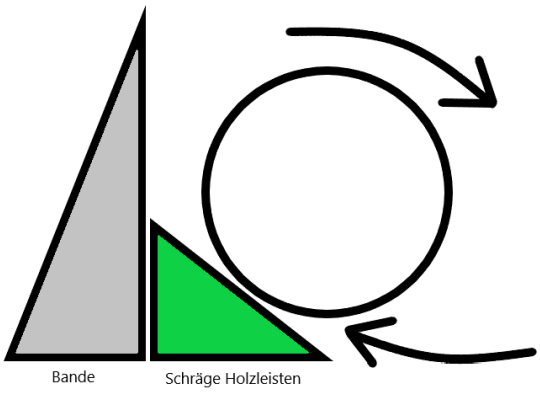

Abbildung 10: Funktionsweise der Holzleisten entlang der Spielfeldbanden

Zusätzlich für die Ecken des Spielfeldes eigens Holzbanden gefertigte, welche verhindern sollen, dass der Ball in der Ecke stecken bleibt und sich die Roboter in einem Deadlock stecken bleiben. Abschließend wurden entlang der Banden dreieckig-geschnittene Holzleisten installiert, welche das

Abrollen des Balles an der Bande ermöglichen sollen (Abbildung 7). Damit sollen ebenfalls Deadlocks an der Bande verhindert werden (Abbildung 10).

Spielball

Als „Spielgerät“ wurde zuerst der mitgelieferte Ball des Spielfeldes (Abbildung 11) getestet. Dieser hatte auf Grund seines geringen Gewichts und Form (man beachte die Klebkannte in der Mitte des Balles) ein sehr gutes Rollverhalten. Allerdings war die Farbgebung des Balles für die Erstellung einer geeigneten HSV-Maske ungeeignet, da der Ball drei unterschiedliche Farben aufweist. Daraufhin wurde ein regulärer Tennisball in Betracht gezogen, um denselben Objekterkennungsvorgang (siehe Abschnitt Bilderkennung) an einem Ball in einer Farbe zu testen. Der Tennisball konnte von der Overhead-Kamera sehr gut erkannt werden. Allerdings ist dieser Ball zu schwer, um von den Robotern weit gespielt zu werden. Deshalb wurden die drei Bälle in Abbildung 13 bestellt, welche das gleiche Rollverhalten wie der mitgelieferte Ball aufweisen und von der Kamera sehr gut erkannt werden können. Das sich der blaue Ball farblich am meisten von der Farbgebung der Roboter unterscheidet, wurde für das Spiel dieser Ball verwendet.

Abbildung 11: Der mitgelieferte Ball des Spielfeldes erwies sich auf Grund seiner Farbgebung als ungeeignet

Abbildung 12: Tennisball diente zum ersten Test der Objekterkennung eines Balles in einer einheitlichen Farbe

Abbildung 13: Die finale Wahl: Drei Schaumstoffbälle in den Farben Rot, Blau und Gelb

Overhead-Kamera & Beleuchtung

Als Overhead-Kamera diente zu Beginn des Projekts die in Abbildung 14 dargestellte Webcam. Um mit dieser Kamera das gesamte Spielfeld zu erkennen, musste das Stativ auf seine Maximalhöhe ausgefahren werden.

Abbildung 14: Initialgelieferte Overhead-Kamera

Abbildung 15: Logitech Webcam auf dem Stativ, befestigt mittels Kabelbindern

Abbildung 16: Stativ für Overhead-Kamera, positioniert auf Höhe der Mittellinie des Spielfelds

Abbildung 17: "Flutlicht" - Fotostudioleuchten

Abbildung 18: Aufbau des Stativs für die Befestigung der Overhead-Kamera

Allerdings konnten ArUCo-Marker, welche auf den Robotern befestigt wurden, in den Ecken des Spielfeldes von dieser Webcam nicht erkannt werden. Aus diesem Grund wurde über das Department eine Logitech C930e Kamera (siehe Abbildung 15) bestellt, welche nicht nur dieses Problem löste, sondern auch das Spielfeld auf einer viel kleineren Stativhöhe überblickt. Das Stativ für die Overhead-Kamera wird ca. auf Höhe der Mittellinie des Spielfeldes platziert und auf eine Höhe von gut zwei Metern ausgefahren. Zur besseren Beleuchtung des Spielfelds dienen zwei Fotostudioleuchten, welche in zwei gegenüberliegenden Ecken des Spielfeldes aufgestellt werden.

Router & Datenbank Raspberry Pi

Abbildung 19: Router für lokales Netzwerk, in welchem die Roboter agieren

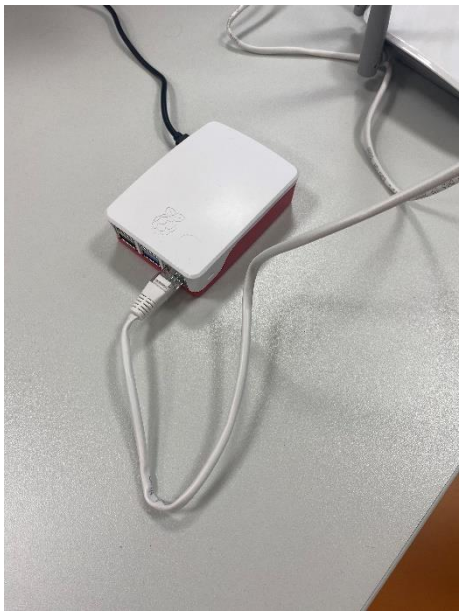

Abbildung 20: Raspberry Pi als Host für die Datenbank der übertragenen Spieldaten

Um einen Datenaustausch zwischen dem stationären PC / Raspberry Pi zu ermöglichen wird ein eigenes Netzwerk benötigt. Dieses wurde mittels einem TP Link Archer XXX WLAN-Router erstellt. Der **WLAN-Name lautet *Fußball_5GHZ*** und das **Passwort *123456789***. Wir entschieden uns bewusst ein reines 5 GHZ WLAN zu verwenden, da dieses Band performanter und störungsfreier als das herkömmliche 2,4GHz Band ist. Da es bei den statischen IPs im Netzwerk mit den Raspberry PIs Probleme gab, verwenden alle Teilnehmer des Netzwerks eine dynamische IP, welche per DHCP vergeben wird. Die per DHCP vergebene IPs können im Webinterface unter http://192.168.0.1/ im Untermenü DHCP/DHCP Client List eingesehen werden.

Für das Logging wurde eine mariaDB Instanz auf einem Raspberry PI installiert. Detaillierter Beschreibung dazu findet man im **Fehler! Verweisquelle konnte nicht gefunden werden.** Kapitel.

Technische Umsetzung

Softwarearchitektur – „Big Picture“

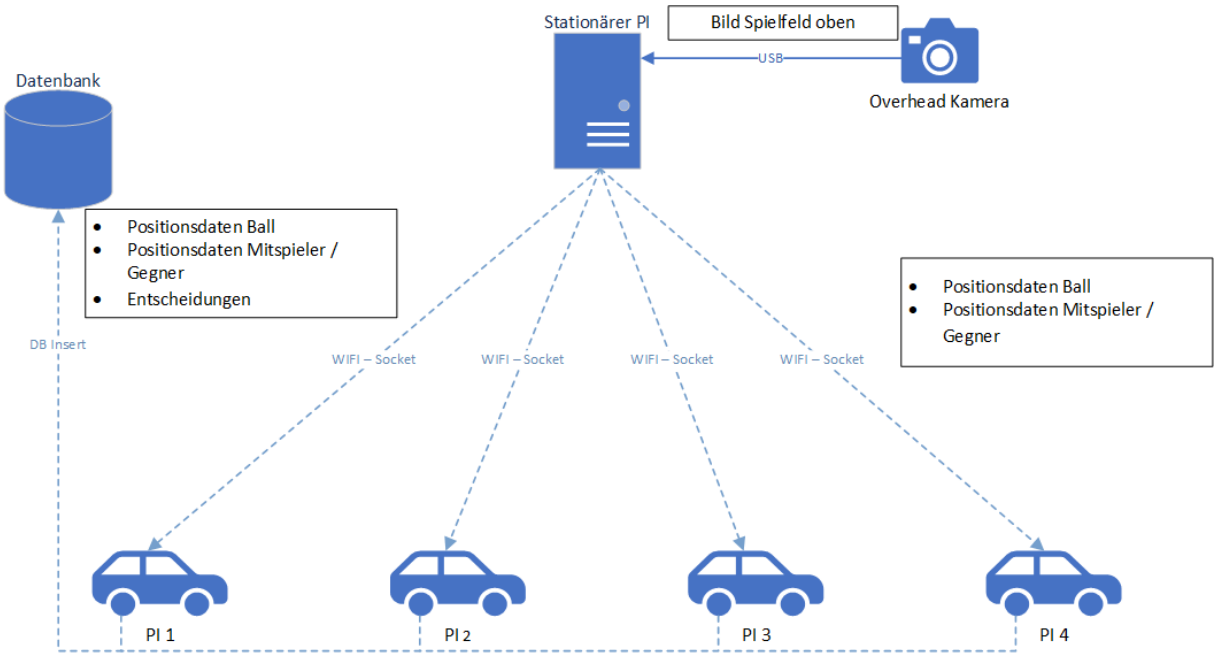

Abbildung 21: Softwarearchitektur als „Big Picture“ des Projektes

Grundlegend besteht die Applikation aus 3 Komponenten:

- Stationärer PC (oder Raspberry PI) mit verbundener Overhead Kamera
- Raspberry PI am AdeeptAWR Roboter
- Raspberry PI mit SQL-Datenbank für Logging

Die an den stationären PC angebundene Kamera liefert die Positionsdaten der 4 Roboter sowie die Position des Balls. Diese Daten werden im JSON Format mittels TCP Socket an die Roboter alle 0,2 Sekunden gesendet. Die Roboter entscheiden anhand dieser Daten ihre nächste Aktion um möglichst ein Tor zu schießen bzw. vermeiden ein Tor zu bekommen. Alle 0,5 Sekunden werden alle Positionsdaten und die aktuelle Entscheidung des Roboters in eine Tabelle in der Datenbank auf einem eigenen Raspberry PI geloggt.

Objektstruktur

Player

Die Player Klasse repräsentiert die Spieler selbst. Diese Klasse hat die folgenden drei Hauptaufgabenbereiche:

- Verbindung zum Refereeserver
- Verbindung und Logging zur Datenbank
- Bewegung der Roboter

Die Verbindung zum Refereeserver und die Verbindung und das Logging zur Datenbank wird im Kapitel Kommunikation näher beschrieben. Die Bewegung der Roboter erfolgt nach dem folgenden Prinzip:

1. Angabe der gewünschten Position
2. Ermittlung der Koordinaten und der Rotation des Roboters
3. Ermittlung von blockenden Objekten
4. Ermittlung der Drehrichtung, des Drehwinkels und Bewegungsrichtung des Roboters
5. Drehen des Roboters in Drehrichtung, bis Roboter ungefähr zum Ziel zeigt
6. Bewegen des Roboters in Bewegungsrichtung, bis Roboter am Ziel angelangt ist

Die Angabe der gewünschten Position erfolgt über die Parameter der dazugehörigen Funktion. In der Regel sind dies entweder die Koordinaten des Balles, der Mittellinie, es eigenen Tores, der Startpunkte oder eines Ausweichpunktes.

Die Ermittlung der Koordinaten und der Rotation des Roboters erfolgt durch die Daten, die vom Referee erhalten werden. Der Roboter findet über seine eigene ID die ihm zugeordneten Positions- und Rotationsdaten und verwendet diese in den nächsten Schritten.

Bei der Ermittlung der blockenden Objekte wird der direkte Pfad zwischen dem Roboter und seinem Ziel errechnet und alle Objekte, die sich auf diesem Pfad befinden, werden als blockierendes Objekt registriert. Wenn mindestens ein Objekt als blockierend registriert wird, errechnet sich der Spieler einen Punkt wie er dem Objekt ausweichen kann und nimmt diesen Punkt als Zwischenziel.

Bei der Ermittlung der Drehrichtung, des Drehwinkels und der Bewegungsrichtung, errechnet sich der Spieler zuerst den Winkel zwischen seiner Vorderseite und seinem Ziel. Anhand dessen, errechnet sich der Spieler jene Richtung, in die er sich drehen muss, um mit seiner Vorderseite nach dem Drehen zum Ziel zu schauen. Danach errechnet er sich den Winkel und die Richtung, die er benötigt, um mit seiner Rückseite zum Ball zu zeigen. Der tatsächliche Drehwinkel ist der Minimalwinkel der beiden errechneten Winkel. Die Dreh- und Bewegungsrichtungen, sind die zum Winkel dazugehörigen Richtungen.

Bei Schritt fünf dreht sich der Spieler solange in die Drehrichtung zum Ziel, bis der Drehwinkel zum Ziel einen gewissen Epsilon-Wert unterschreitet (zurzeit 20°).

Bei Schritt sechs bewegt sich der Spieler solange in die Bewegungsrichtung, bis das Ziel erreich wird, oder der Drehwinkel größer wird als der Epsilon-Winkel. Sollte der Drehwinkel größer werden als der Epsilon-Winkel geht der Spieler wieder zu Schritt fünf.

Referee / Schiedsrichter

Damit ein erfolgreiches und faires Spiel stattfinden kann, muss es ein Regelwerk geben. Dies wird durch eine neutral-beobachtende Instanz überprüft und dessen Einhaltung wird durch diese sichergestellt. Für diesen Zweck wurde ein Schiedsrichter implementiert. Dieser ist verantwortlich für das Starten, Stoppen und Pausieren des Spiels und der Spielzeit, der Torerkennung, der Ballerkennung und ob die Spieler bereit für den Anstoß sind. Zusätzlich hält der Schiedsrichter den aktuellen Punktestand fest und ändert diesen, sobald ein Tor fällt.

Zur Messung der Spielzeit wurde eine Gametimer Klasse entwickelt. Diese benutzt die time Bibliothek, um die Zeit zu messen. Sie verfügt über einfach Methoden wie Start, Stopp, fortsetzen (resume) und zurücksetzen (reset). Der Schiedsrichter hat dabei die alleinige Kontrolle über die Spielzeit. Wenn ein Tor fällt, wird die Zeit pausiert und der GameStatus auf „INIT“ gesetzt. Dies lässt die Roboter/Spieler auf die Initialpositionen zurückkehren. Das Spiel wird fortgesetzt, die Spielzeit wieder gestartet (resume) und der GameStatus auf „START“ gesetzt, wenn der Schiedsrichter den Ball im Zentrum des Feldes erkennt.

Bei einer Unterbrechung (z.B. Tor) ist ein manuelles Eingreifen der Beobachter notwendig, um den Ball wieder zentral im Spielfeld zu platzieren, um damit die Ausgangssituation für die nächste Runde zu schaffen. Außerdem markiert der Schiedsrichter die Torlinien und die unterschiedlichen Mitspieler der Teams für den Beobachter (Grün & Rot). Nach Ablauf der Spielzeit werden das Spiel gestoppt und der GameStatus auf „STOP“ gesetzt. Der GameStatus wird nur auf „PAUSE“ gesetzte, wenn im GUI der „Pause“ Button gedrückt wird. Dabei wird der Gametimer pausiert und die Roboter brechen ihre derzeitigen Aktionen ab. Im normalen Spielbetrieb ist es normalerweise nicht notwendig das Spiel manuell zu pausieren.

Nach einem Tor, oder beim initialen Start des Spiels überprüft der Schiedsrichter, ob alle Spieler verbunden sind. Sollte ein oder mehrere Spieler keine Verbindung zum Server haben, wird das Spiel nicht gestartet. Das Spiel startet erst wenn alle Roboter auf den Initialpositionen sind und eine Verbindung zum Server haben.

Die Torpfosten bzw. die Torpositionen werden mit Aruco-Marker deklariert. Der Ball, da dieser einfarbig ist, wird mit einer HSV-Maske verfolgt. Die genaue Beschreibung wird im Kapitel „Bilderkennung“ erklärt.

Bilderkennung

ArUco-Marker und Koordinatensystem

In diesem Kapitel wird auf die Verwendung von ArUco-Marker und der Implementierung eines eigenen Koordinatensystems eingegangen.

ArUco-Marker

Die ArUco-Marker werden im Projekt verwendet um die Position, die Rotation und die Identität der Roboter anhand des Kamerabildes festzustellen. Die ArUco-Marker wurden an der Oberseite der Roboter angebracht (siehe Abbildung 2) und die ID der Marker und der Roboter wurde am Rand des ArUco-Markers festgehalten (z.B.: A2=P1 -> ArUco-Marker ID 2 = Player ID 1). Das Detektieren der Marker erfolgt mit dem Packet *cv2.aruco,* welches von OpenCV stammt. Hierbei ist zu erwähnen, dass jeder einzelne Frame der Videokamera durchsucht wird und dann die Marker in diesem Frame markiert werden (siehe aruco_tracker.py: ArucoTracker.find_markers).

Koordinatensystem

Um zu verhindern, dass das Feld und die Kamera immer exakt gleich zueinanderstehen müssen, wurde ein eigenes Koordinatensystem implementiert, das sich vom Koordinatensystem der Kamera unterscheidet. Bei diesem selbst implementierten Koordinatensystem ist der Ursprung stehts der sogenannte Nullpunkt-Turm (siehe Abbildung 8). Alle Koordinaten der ArUco-Marker, die über die Kamera erfasst werden, werden in das neue Koordinatensystem übersetzt und erst dann in des GameData Objekt gespeichert (siehe Game Data Object).

HSV

Ein Raspberry PI bietet wenig Leistung im Vergleich zu herkömmlichen Computer Systemen. Um die vorhandene Leistung so effizient wie möglich zu nutzen, wurde die Ball-Verfolgung mit einem HSV-Filter implementiert. Dies ist bei dem Spielball besonders gut möglich, da dieser einfarbig ist. Damit ein HSV-Filter erstellt werden kann, muss der Farbort des Farbtons im HSV-Farbraum definiert werden

(siehe Abbildung 22). Dieser Farbort besteht aus einer oberen und einer unteren Grenze, welche einen bestimmten Bereich im Farbraum eingrenzen.

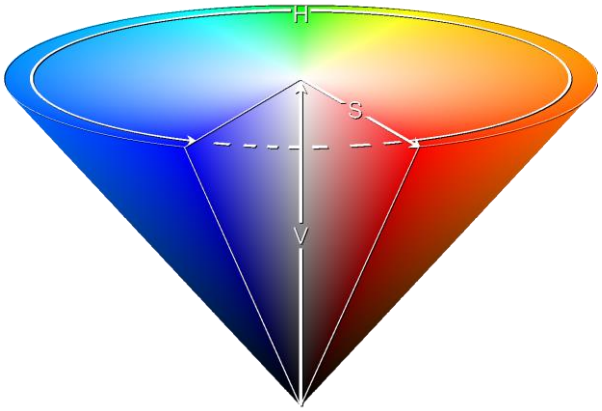

Abbildung 22: HSV-Filterraum

Diese Grenzen bestehen aus drei unterschiedlichen Parametern. Dem Farbwert H, der Sättigung S und der Dunkelstufe V. Um diese Grenzwerte zu ermitteln wurde ein Script verwendet, welches ein Foto in den HSV-Farbraum konvertiert. Außerdem erzeugt dieses Script drei Regler (H, S & V) um bei jedem der drei Werte eine obere und untere Grenze zu ermitteln. Diese HSV-Grenzwerte erzeugen einen Filter, um den Roten Ball zu verfolgen.

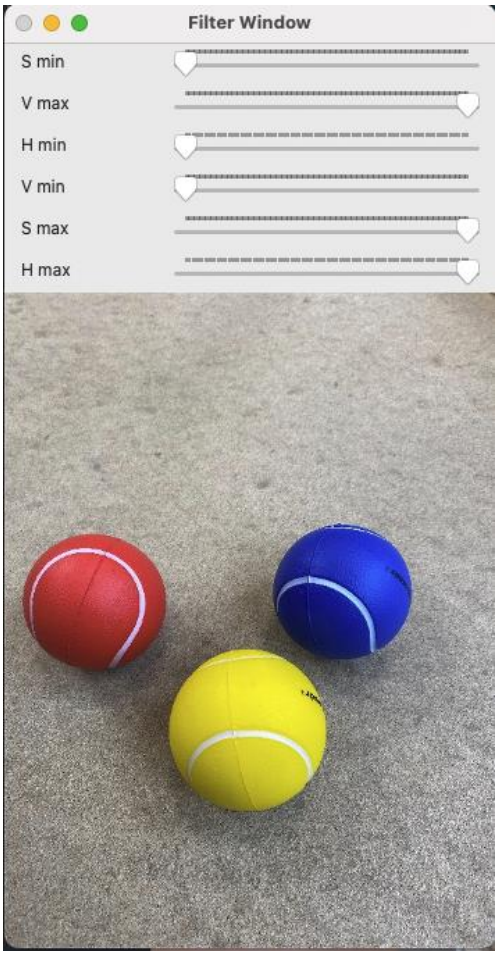

Abbildung 23: HSV-Regler in Default-Einstellungen

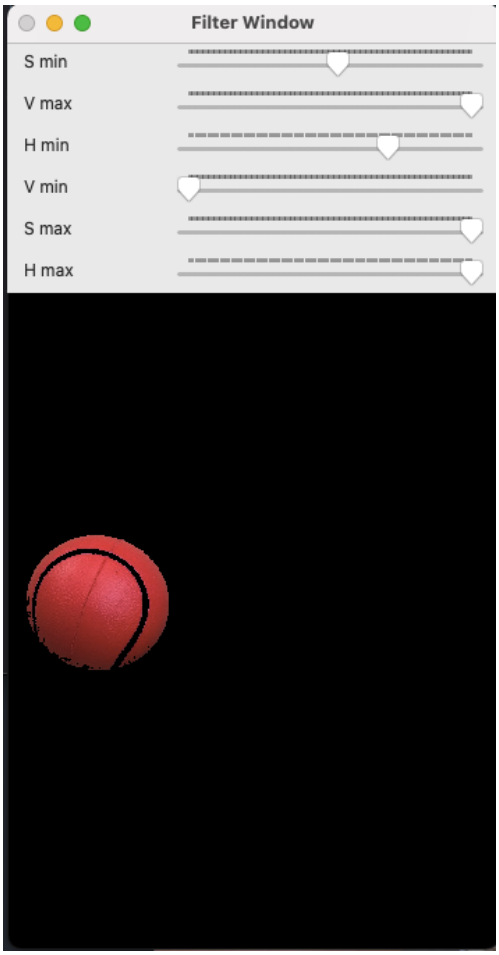

Abbildung 24: HSV-Einstellungen, um roten Ball zu erkennen

Bei der Implementierung wird nun jeder eingehende Frame in den HSV-Farbraum konvertiert. Anschließend wird mit dem erzeugten Filter nach Objekten im Frame gesucht. Wichtig dabei ist nicht jeden gefundenen Pixel, in der richtigen Farbe, als „Ball“ zu klassifizieren. Daher wird überprüft, ob das gefundene Objekt eine bestimmte Mindestgröße hat. Diese Variiert je nach Abstand von der Kamera zum verfolgenden Objekt. Wird ein nun ein valides Objekt gefunden, werden die Koordinaten des größten Objektes im Frame zurückgegeben. Während des Spieles ist es wichtig, nicht mit einem ähnlich farbigen Gegenstand in das Bild der Kamera zu gelangen, da jedes gefundene Objekt sofort für den Ball gehalten wird. Die Koordinaten werden an die Roboter übergeben, welche dann auf dieses Objekt zufahren.

Kommunikation

Für den Austausch der Positions- und Spielstanddaten wurde ein eigens entwickeltes TCP/IP Socket Framework entwickelt. Die Clients sowie der Server ist ausfallsicher sowie designt. Das bedeutet dass

der Server sowie die Clients beendet werden können, ohne dass dies eine Auswirkung auf das Gesamtsystem hat. Der Server akzeptiert außerdem mehrere Clients (in unserem Fall an die 4 Roboterclients) und leitet die Positions- sowie Spielstanddaten an alle verbunden Clients weiter. Der Server warte auf dem Port 65432 auf eingehende Verbindungen zum Registrieren der Clients.

Da TCP Sockets keine Nachrichten basiertes Protokoll ist und lediglich Bytes empfängt, mussten wir ein System implementieren um Nachricht für Nachricht zu Senden/Empfangen. Vor dem Senden des serialisierten JSON Objekts, wird die Länge des JSON Objects in ein Preamble Flag (>L + Länge) in der Nachricht gesetzt. Diese Länge der Nachricht belegen immer die ersten 4 Stellen der Nachricht. Der Client liest immer zuerst die ersten Stellen der Nachricht aus, und empfängt dann exakt so viele Stellen des TCP Socket Streams wie lang die Nachricht ist.

Game Data Object

Das GameData Object ist die Datenstruktur, welche per Socket zwischen dem Server und den Clients ausgetauscht bzw. als Broadcast versendet wird. Es stellt ein Python Object dar mit folgenden Daten:

- Ballposition
- Spielerpositionen und Ausrichtung
- Initale Positionen
- Center Position
- Tor Positionen
- Spielstände
- Spielstatus
- Zeit

Die Klasse, welche den Server Socket erstellt und Nachrichten sendet lautet *multiClientServer.py*.

Die Klasse, welchen den Client Socket erstellt und Nachrichten empfängt lautet *clientSocket.py*.

Datenbank & Logging

Für das Logging der Aktionen und den aktuellen Positionsdaten, entschied das Team, eine SQL-Datenbank zu verwenden. Dafür wurde auf einem Raspberry PI eine MariaDB Instanz installiert und eine Datenbank sowie Tabelle angelegt. Die Setup Befehle für die Roboter PIs und Logging PI sehen wie folgt aus:

```
Setup on raspberry pi logging server

1. sudo apt-get install mariadb-server
```

```
2. in file /etc/mysql/mariadb.conf.d/50-server.cnf
   modify bind-address = 127.0.0.1 to bind-address = 0.0.0.0
   in order to activate access from remote clients

Useful aliases:
   - db_start
   - db_stop
   - db_restart
   - db_status


Setup on rasperry pi logging clients

1. sudo apt-get install -y libmariadb-dev
2. sudo pip3 install mariadb
```

Code Snippet 1 - Setup DB Client/Server

Im folgenden SQL Script werden Datenbank, Tabelle, User und die Berechtigungen für einen Remotezugriff erstellt:

```
CREATE DATABASE football;

CREATE USER 'server'@'localhost' IDENTIFIED BY 'password';
CREATE USER 'manfred'@'localhost' IDENTIFIED BY 'password';
CREATE USER 'fattuesday'@'localhost' IDENTIFIED BY 'password';
CREATE USER 'bob'@'localhost' IDENTIFIED BY 'password';
CREATE USER 'timo'@'localhost' IDENTIFIED BY 'password';

GRANT ALL ON football.* to 'server'@'192.168.%' IDENTIFIED BY 'password' WITH
GRANT OPTION;
GRANT ALL ON football.* to 'manfred'@'192.168.%' IDENTIFIED BY 'password' WITH
GRANT OPTION;
GRANT ALL ON football.* to 'fattuesday'@'192.168.%' IDENTIFIED BY 'password' WITH
GRANT OPTION;
GRANT ALL ON football.* to 'bob'@'192.168.%' IDENTIFIED BY 'password' WITH GRANT
OPTION;
GRANT ALL ON football.* to 'timo'@'192.168.%' IDENTIFIED BY 'password' WITH GRANT
OPTION;



CREATE TABLE football.game_data (
  id int(11) NOT NULL AUTO_INCREMENT,
  time float NOT NULL,
  player_id int(3) NOT NULL,
  mate_id int(3) NOT NULL,
  team_id int(3) NOT NULL,
  decision varchar(255),
  decision_count int(10),
  goal_team_1 int(3) NOT NULL,
  goal_team_2 int(3) NOT NULL,
  goal_1_post_1_x float NOT NULL,
  goal_1_post_1_y float NOT NULL,
  goal_1_post_2_x float NOT NULL,
  goal_1_post_2_y float NOT NULL,
  goal_2_post_1_x float NOT NULL,
  goal_2_post_1_y float NOT NULL,
  goal_2_post_2_x float NOT NULL,
  goal_2_post_2_y float NOT NULL,
```

```
  player_1_x float NOT NULL,
  player_1_y float NOT NULL,
  player_1_alpha float NOT NULL,
  player_2_x float NOT NULL,
  player_2_y float NOT NULL,
  player_2_alpha float NOT NULL,
  player_3_x float NOT NULL,
  player_3_y float NOT NULL,
  player_3_alpha float NOT NULL,
  player_4_x float NOT NULL,
  player_4_y float NOT NULL,
  player_4_alpha float NOT NULL,
  ball_x float NOT NULL,
  ball_y float NOT NULL,
  time_stamp timestamp DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (id)
)
ENGINE = INNODB,
COMMENT = 'game_data table which stores all relevant data which led to players
decision';
```

Code Snippet 2 - Datenbank, Tabellen, User und Berechtigung Erstellung

Um die Daten der erzeugten Logs zu exportieren, kann mit jedem gängigen MySQL-Programm zugegriffen werden. Wir verwendeten dafür das frei erhältliche Programm *dbForge Studio for MYSQL*.

Das Schreiben der Daten von den Robotern passiert in einem eigenen Thread welcher sich neben dem regelmäßigen Schreiben der Daten, ständig überprüft, ob die Verbindung zur Datenbank noch aufrecht ist und im Falle eines Verbindungsverlust sich neu mit der Datenbank verbindet. Das komplette Datenbank Log Handling findet in der Methode *log_to_database* in der *Player* Klasse statt.

Vordefinierter Entscheidungsbaum

Die allgemeinen Fähigkeiten der Roboter beinhalten die folgenden Bewegungen:

- Gerade ausfahren
- Rückwärtsfahren
- Drehung nach links
- Drehung nach rechts

Es wurde auf die Integration einer Schussvorrichtung an der Frontseite des Roboters verzichtet, da sonst der Ultraschall-Sensor verdeckt werden würde. Um jedoch, die Schussmöglichkeiten der Roboter zu erhöhen, suchen die Roboter bevor einer Spielaktion den geringsten Rotationswinkel, um sich in einer Linie zu dem Ball zu positionieren. Es spielt dabei keine Rolle, ob die Front- oder Rückseite des Roboters direkt zum Ball steht da die Roboter direkt auf den Ball zufahren müssen, um ihn zu spielen.

Da die Spieleigenschaften der Roboter in einem Fußballspiel unbekannt waren und Trainingsdaten aus durchgeführten Spielen nicht zur Verfügung standen, fiel die Entscheidung des Projektteams auf die Erstellung eines vordefinierten Entscheidungsbaum, nach welchem die Roboter voranalysierte Spielsituationen interpretieren und entsprechende Aktionen durchführen sollen. Während der Analyse von möglichen Spielsituationen wurden die folgenden Spielaktionen definiert, welche:

- **Guard**: Der Roboter fährt in das eigene Tor, um es abzudecken und ggf. ein Tor der gegnerischen Mannschaft zu verhindern.
- **Shoot**: Der Roboter fährt zu einer Position, welche es ihm ermöglicht so auf den Ball zu fahren, dass der Ball in direkter Linie zum Tor geschossen werden kann.
- **Intercept**: Der Roboter fährt direkt auf den Ball zu, um den gegnerischen Ballbesitz zu unterbrechen.
- **Recover**: Sollte der Ball hinter beiden Spielern eines Teams zu liegen kommen, fährt jener Roboter, der am nächsten zum Ball steht zum Ball und versucht diesen vom eigenen Tor zu entfernen und versucht hinter den Ball zu kommen.
- **Center**: Während der andere Teamspieler eine Aktion durchführt, bewegt sich der andere Roboter auf die Mittellinie, um sich strategisch für darauffolgende Entscheidungen zu positionieren.

Die entsprechenden Spielsituationen sind für das Spiel in der eigenen und gegnerischen Hälfte in Abbildung 22 und 23 dargestellt. Die einzelnen Situationen sind mit Großbuchstaben gekennzeichnet, damit deren durchzuführenden Aktionen im Entscheidungsbaum (siehe Abbildung 24) ersichtlich sind. Die Zahlen 1 und 2 stehen dabei für die beiden Spieler eines Teams und deren Aktionen sind rechts neben dem Bild dokumentiert. Diese Spielsituationsanalyse gilt für beide Teams.

Der Entscheidungsbaum ist darauf ausgelegt, dass sich die Spieler eines Teams je nach ihrer Position entsprechend ergänzen und zwischen einer „aktiven“ (Recover, Intercept, Shoot) und einer „passiven“ (Center, Guard) Aktion wechseln. Die wichtigsten Entscheidungen, welche die Spieler evaluieren müssen, sind dabei:

- In welcher Hälfte befindet sich der Ball?
- Bin ich jener Spieler, welcher am nächsten am Ball ist?
- Ist der Ball hinter mir?
- Ist der Ball hinter meinem Teamkameraden?

Der Spieler, welcher am nächsten zum Ball steht, soll sich dann für eine aktive Aktion entscheiden, während der andere Spieler zu einer passiven wechselt, um das eigene Tor abzusichern. Um

zusätzliche Zielstrebigkeit bei der Aktion Shoot zu verankern, verzichtet ein Spieler ab einer gewissen Entfernung zum gegnerischen Tor (sehr nahe am gegnerischen Tor) auf die entsprechende Ausrichtung vor dem Schuss und schießt den Ball direkt auf das Tor.

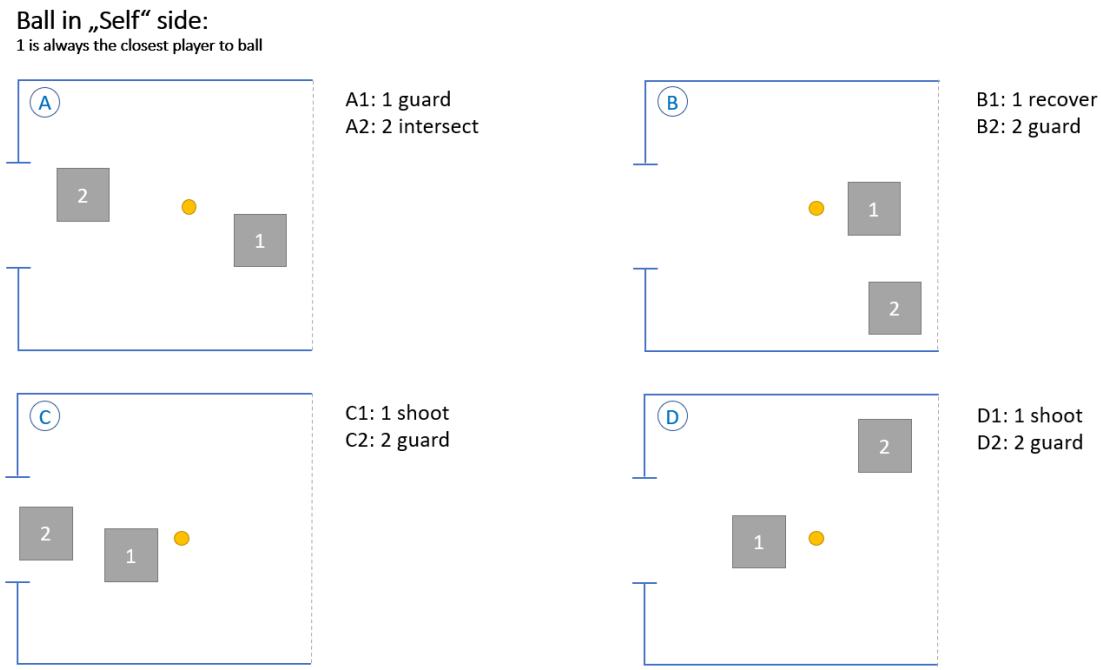

Abbildung 25: Spielsituationen in eigener Hälfte

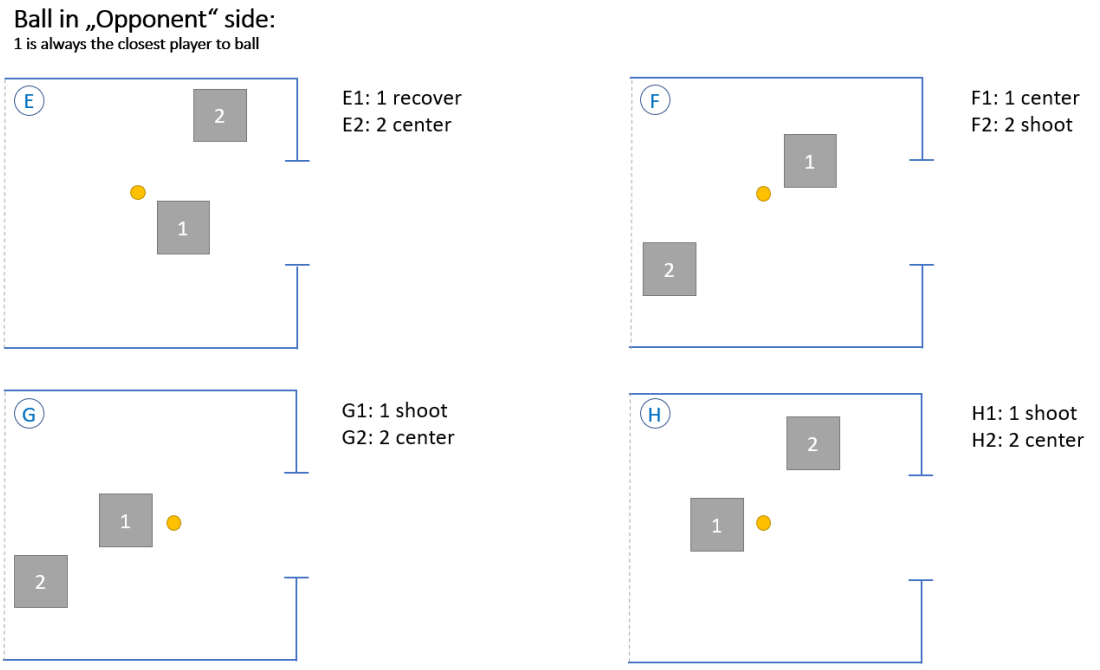

Abbildung 26:Spielsituationen in gegnerischer Hälfte

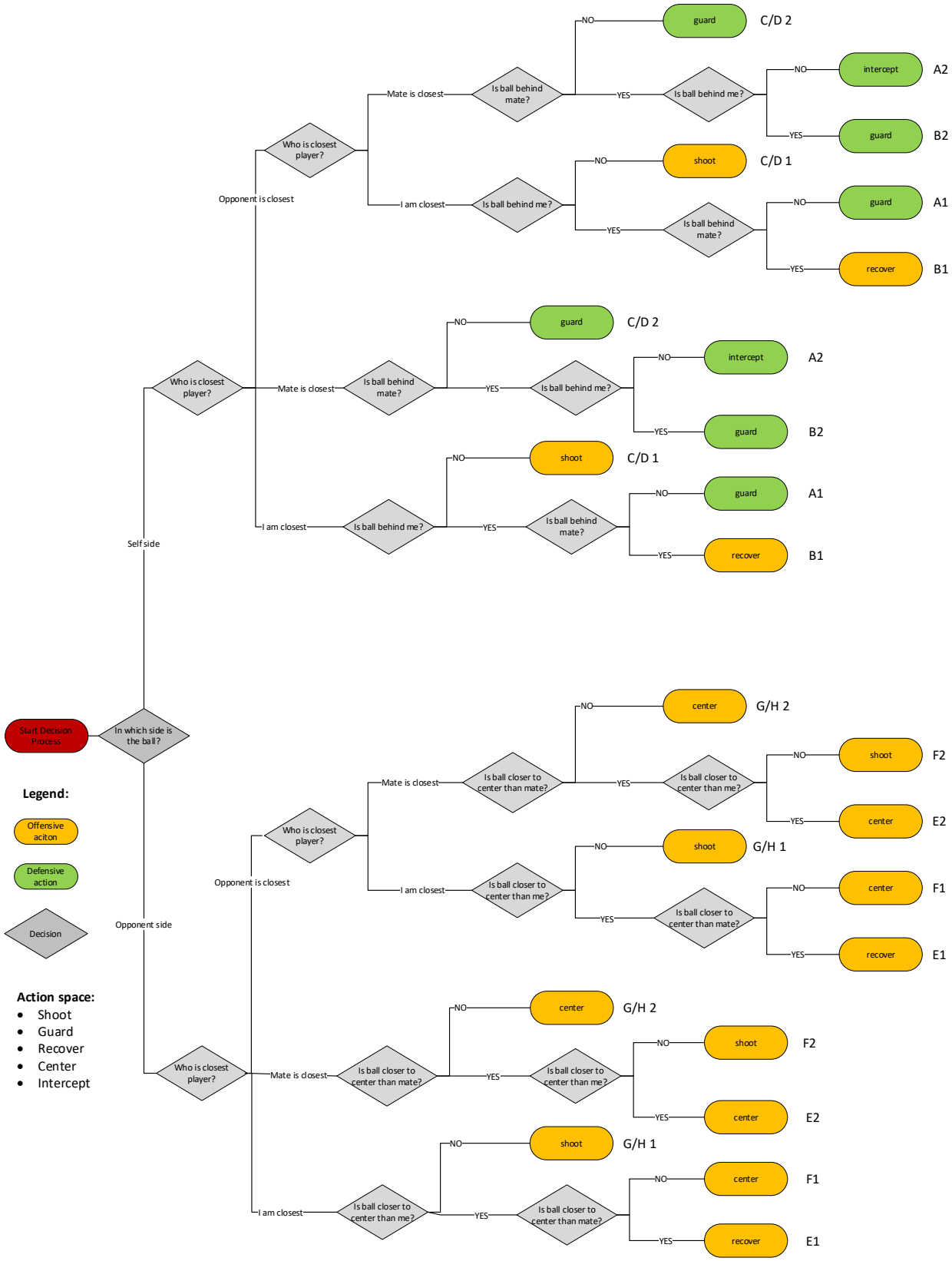

Abbildung 27:Vordefinierter Entscheidungsbaum, basierend auf den identifizierten Spielsituationen

Data Science

Data Labeling für Supervised Learning

Gemäß dem zuvor dargestellten vordefinierten Datensatz wurden mehrere Test-Matches durchgeführt, dessen Spieldaten in der Datenbank aufgezeichnet wurden. Der finale Trainingsdatensatz beinhaltet 10.246 Dateneinträge, welche gemäß entsprechenden Regeln gelabelt wurden, um einen Supervised Learning Ansatz einsetzen zu können. Ganz allgemein ist der Aktionserfolg die wichtigste Größe, um eine Entscheidung zu bewerten. Der Aktionserfolg ist maßgeblich damit verbunden, wie nahe man den Ball in die Richtung des gegnerischen Tors spielen konnte. Dies gilt allerdings nur für die aktiven Aktionen *Shoot*, *Intercept* und *Recover*. Die passiven Aktionen *Center* und *Guard* wurden mit repräsentativen Konstanten für Erfolg und Misserfolg bewertet, da diese nicht so stark zu einem Torerfolg beitragen. Da es bei einer Entscheidung allerdings auch zu keiner Verbesserung der Ballposition, aus Sicht des jeweiligen Teams, kommen kann, muss dieser Erfolg auch negative bewertet werden können. Ebenfalls ist ein erzieltes Tor entsprechend höher zu honorieren beziehungsweise ein erhaltenes Tor zu bestrafen. Die folgenden Regeln fassen den Labeling-Prozess gemäß den oben beschriebenen Kriterien zusammen:

- A: Erzieltes Tor (+5), erhaltenes Tor (-5), kein Tor (± 0)
- B: Ball näher zu gegnerischem Tor
    - SHOOT, INTERCEPT & RECOVER. [-1, 1]
    - GUARD, CENTER: (-0.25; 0.25)
- Gesamtscore: A + B

Regression mit XGBoost

Um den Score für jede Aktion anhand der aktuellen Positionsdaten und Spielstände zu bewerten wurde ein Model mit den gesammelten und gelabelten Daten trainiert. Dafür wurde das XGBoost Framework verwendet. Die folgenden Schritte stellen den Vorgang der Modellerstellung dar:

1. Einlesen der Daten aus einem csv File.
2. Konvertieren der kategorischen Features in numerisches Encoding mittels SKLearn LabelEncoder
    a. Decision, player_id, mate_id, team_id
3. Separieren von Score Labels vom Rest der Features
4. Normieren der Scores mittels Sklearn Min Max Scaler (0 – 1)
    a. 0 = Schlechter Score

    b. 1 = Guter Score
5. Test und Trainingdaten Split mit 80% Trainings- und 20% Testdaten
6. Erstellen des XGBoost Modells mit Ziel Regression und RMSE als Fehlermesseinheit
7. Model Fit mit Trainingsdaten

Durch manuelles Optimieren der XGBoost Hyperparameter erreichten wir einen RMSE Score von 0,049115 welches den durchschnittlichen Fehler des vorhergesagten Scores darstellt.

Random Forest Regression

Um einen weiteren Data Science Algorithmus an den Trainingsdaten zu trainieren fiel die Wahl auf die Regression mittels Random Forest. Die ein entsprechendes Label (Score) sollte für die einzelnen Entscheidungen damit vorhergesagt werden

1. Einlesen der Daten aus einem csv File.
2. Konvertieren der kategorischen Features in numerisches Encoding mittels SKLearn LabelEncoder
    a. Decision, player_id, mate_id, team_id
3. Separieren von Score Labels vom Rest der Features
4. Normieren der Scores mittels Sklearn Min Max Scaler (0 – 1)
    a. 0 = Schlechter Score
    b. 1 = Guter Score
5. Test und Trainingdaten Split mit 80% Trainings- und 20% Testdaten
6. Erstellen des Random Forest Modells (SKLearn Framework) mit bis zu 5 Ästen
7. Model Fit mit Trainingsdaten

Mit dieser Methode konnte ein RMSE Score von 0,053947 erzielt werden.

Lasso Regression

Neben der Random Forest Regression wurde auch mittels der LASSO (least absolute shrinkage and selection operator) Regressionsmethode versucht, um einen Spielzug vorherzusagen. Um ein LASSO Vorhersagemodel zu erstellen sind folgende Schritte notwendig:

1. Einlesen der Daten aus einem csv File.
2. Konvertieren der kategorischen Features in numerisches Encoding
3. Separieren von Score Labels vom Rest der Features

4. Erstellen eines Hyperparameter Suchraumes
5. Normieren der Scores mittels Sklearn Min Max Scaler (0 – 1)
    a. 0 = Schlechter Score
    b. 1 = Guter Score
6. Test und Trainingsdaten Split mit 80% Trainings- und 20% Testdaten
7. Erstellen eines Lasso Modells mit SKLearn
8. Model Fit mit Trainingsdaten mit allen möglichen Kombinationen im Suchraum
9. Validierung der Ergebnisse mittels K-Fold-Crossvalidation.

LASSO Regression führte zu einem RMSE Score von 0,1306.

Empfehlungen

- Ballführung & Action Space der Roboter eingeschränkt
- Taktikimplementierung eingeschränkt
- „Indirekte“ Kommunikation innerhalb des Teams
- Deadlocks nicht immer verhinderbar
- Use Case für Reinforcement Learning

Quellen

Adeept Roboter: https://www.adeept.com/awr_p0122.html

HSV-Filterraum Abbildung 22:

https://de.wikipedia.org/wiki/HSV-Farbraum#/media/Datei:HSV_cone.png