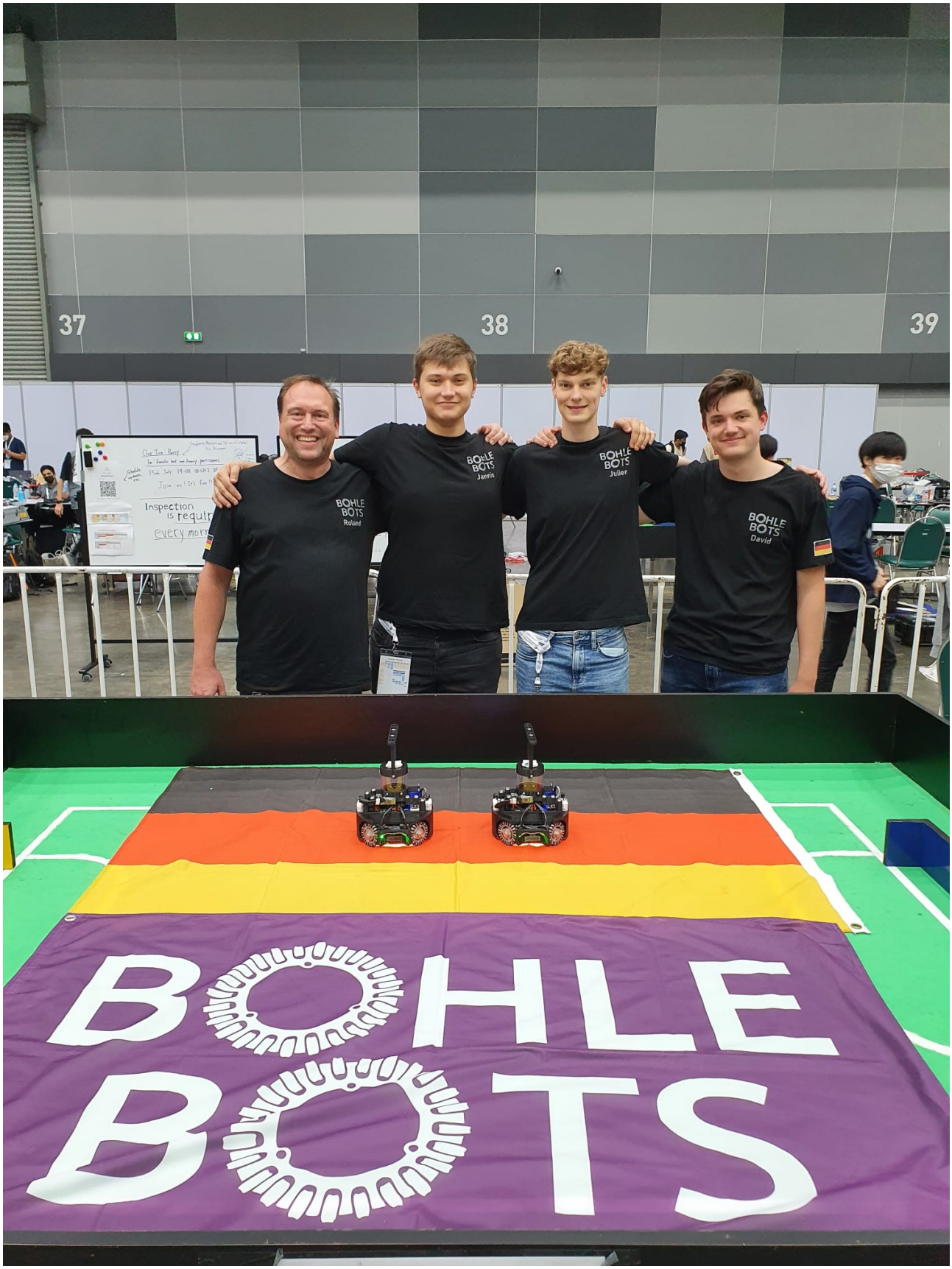
Bohlebots Quadro 2022



Inhaltsverzeichnis

1. Introduction

This file is a summary of the presentation the Team Bohlebots Quadro gave at the symposium of the Robocup 2022 as asked by the organizer of the soccer competition, after winning the world cup in the 2v2 Open League.

So this article focuses mainly on what we think are the biggest innovations we did this year. Due to the time limit our presentation had on the symposium, this file isn't as detailed as it could be.

For further information, we recommend our poster and website as well as our Github for all files and documentations as our robot is completely open source.

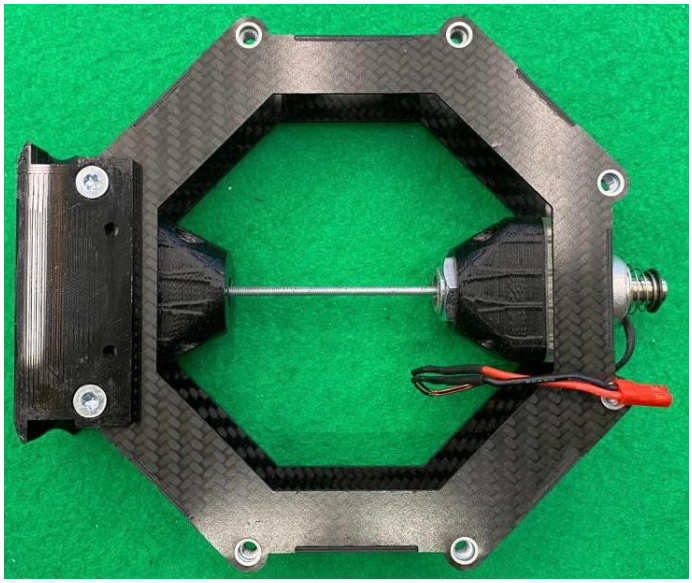
2. Hardware

2.1. Wheels

A lot of teams have problems with wheels falling off their motors. That is why we decided to design and build our wheels ourselves. We decided not to arrange the small wheels in a straight line, but slightly offset. We also have small "X-wheels" instead of "O-wheels", to have better grip on the surface. Also we do not fix the wheels to the engine with only one bolt, but use a clamping cone, which is also used for model airplanes to hold the wheel to the engine with several points. This prevents the wheel from falling off.

2.2. Kicker

We did not have enough space in the front of our robot for the kicker, because of our “very long“ motors. To not restructure our whole robot or take new motors, we simply put the kicker in the back. Hence we need to kick trough the whole robot.



2.3. Ball control

Also we have decided not to use a dribbler, because it is very complex and it would have taken too much time to build a really good dribbler. But now we use a silicone tape to stop and better control the ball. Because of its high friction the silicon prevents the ball from spinning, while the robot drives with it therefore negating all its momentum that would push it outside the robot. This technique allows us to perform a planetary movement around the ball with which we can make narrow curves without ever losing the ball.

3. Vision

3.1. Mirror

We as a team thought a lot about the right mirror shape, because all had their pros ans cons.

A hyperbolic one wouldn‘t distort the image, a spherical one is great for seeing objects near the robot, but only a cone can see everything properly in a long distance. So what we did in the end was trying to mix those shapes to get the best out of all. So at the end we decided to stick to a cone-like mirror with an spherical top, as you can see on our poster. With this we had no problem seeing the ball, no matter its distance to the robot. The image nevertheless was still distorted, but this turned out not to bother us at all as you will read later on.

3.2. AI

When approaching how to see the ball we wanted to create a very stable system. We wanted to be resistant against deviations in the light conditions, ball colors and other irritations such as hands inside the field.

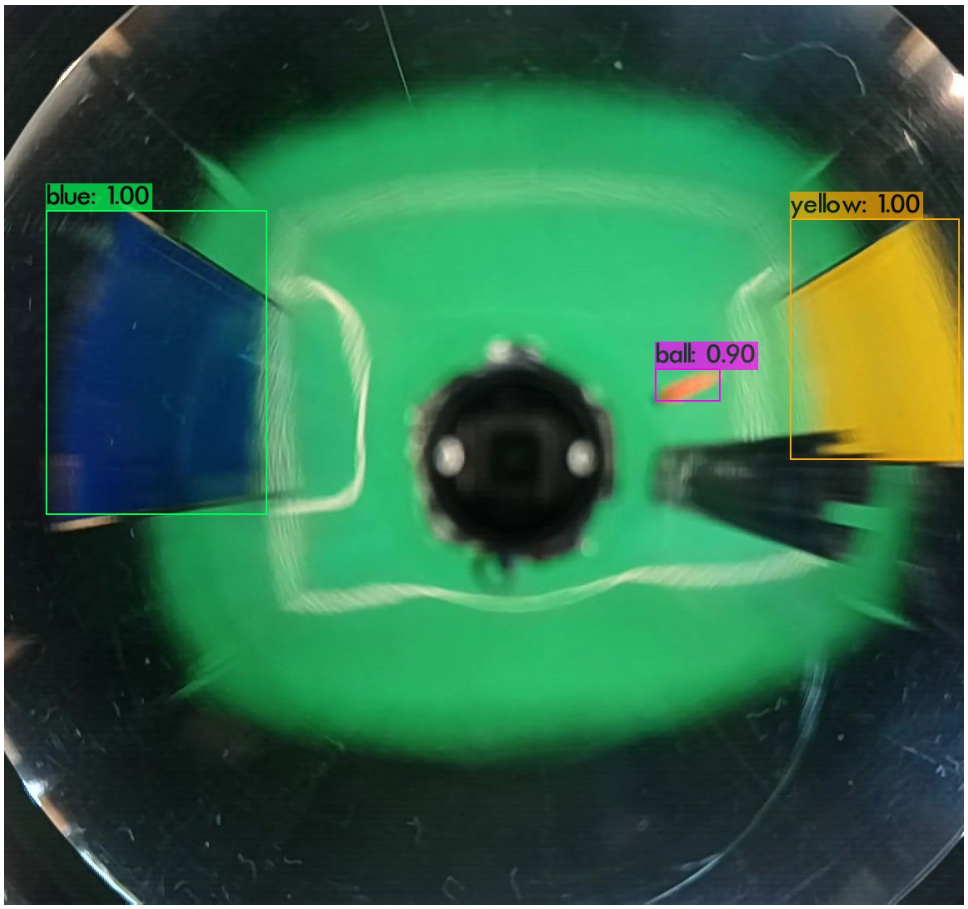
In order to reach those expectations we decided to take a risk and not to go for a simple color blob detection, but to train an object detection AI. For that we chose to use Yolov4. We trained our model with darknet and it worked quite well, but the main challenge is to have the AI work on a small device while being really fast.

We switched to Yolov4-tiny and the results were much better. After a long journey of formatting the weights to a device friendly format all was set up. Except that we couldn't detect the ball if it was too small. To fix this we tried two things, first to use custom anchors and second to add an additional layer in front of the neural network. At the end we stuck to the anchors, because it produced better results after all. To choose the right anchors, we actually asked Luxonis, the company which made our camera, which ones to choose in order to get a proper detection.

At the end we made a training video ourselves from which we extracted 10.000 images and annotated them by hand. After that we multiplied our data by a factor of 5 by using different filters, such as color, saturation, etc. to use the same annotations on multiple images.

After our first AI we could of course let the previous AI annotate our dataset and just correct the mistakes, which made the whole data preparing process much shorter.

After that the results were promising:



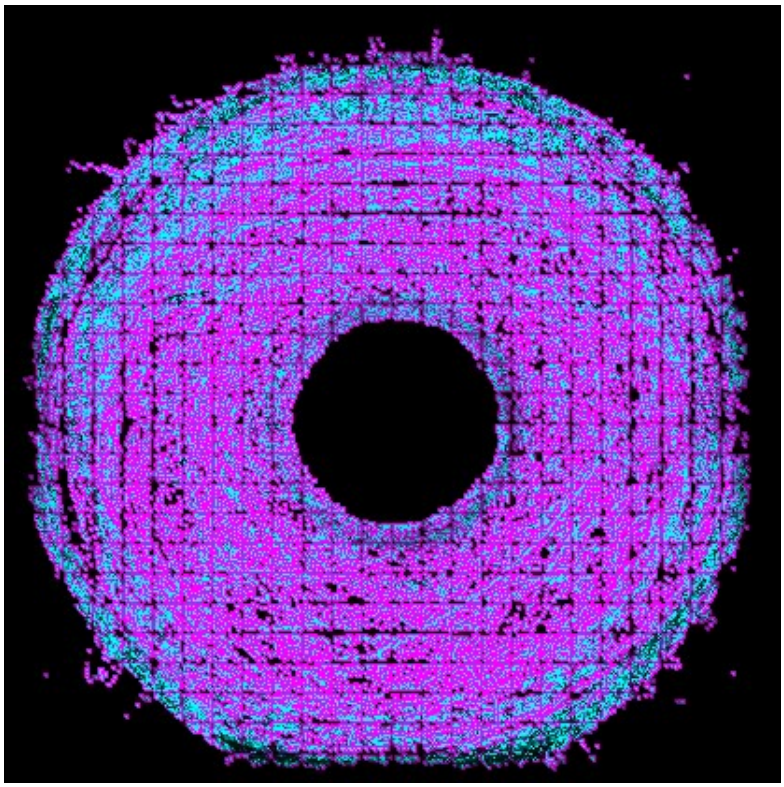
3.3. Devices

Initially we planned to use a google coral to run our model but we had lots of issues with it, mainly because it is optimized for integer operations and our model was based on floats. You can imagine that the formatting wasn't fun after all.

But luckily we found the Oak One Camera with a builtin Intel Myriad X TPU. It was perfect for our demands, because it instantly processed the images inside the camera and just sends out the coordinates of the object via USB. So we just had to format our model to work on the oak and load it in at the beginning of our program. To load the model on the camera and to receive those coordinates and to polish them we simply used a Raspberry Pi CM4.

3.4. AI Kpi

After everything was finished we wanted to check whether the AI was working as expected on every possible ball position. For this we created a program that saved the accuracy of the AI in relation to the position of the ball in the field and visualize them, as you can see here:



The black areas are the ones where the ball never shows up, which you can see is insight the robot and outside the mirror.

The blue and pink pixels give us the important information, as the brightness of the pixels is proportional to the accuracy of the AI.

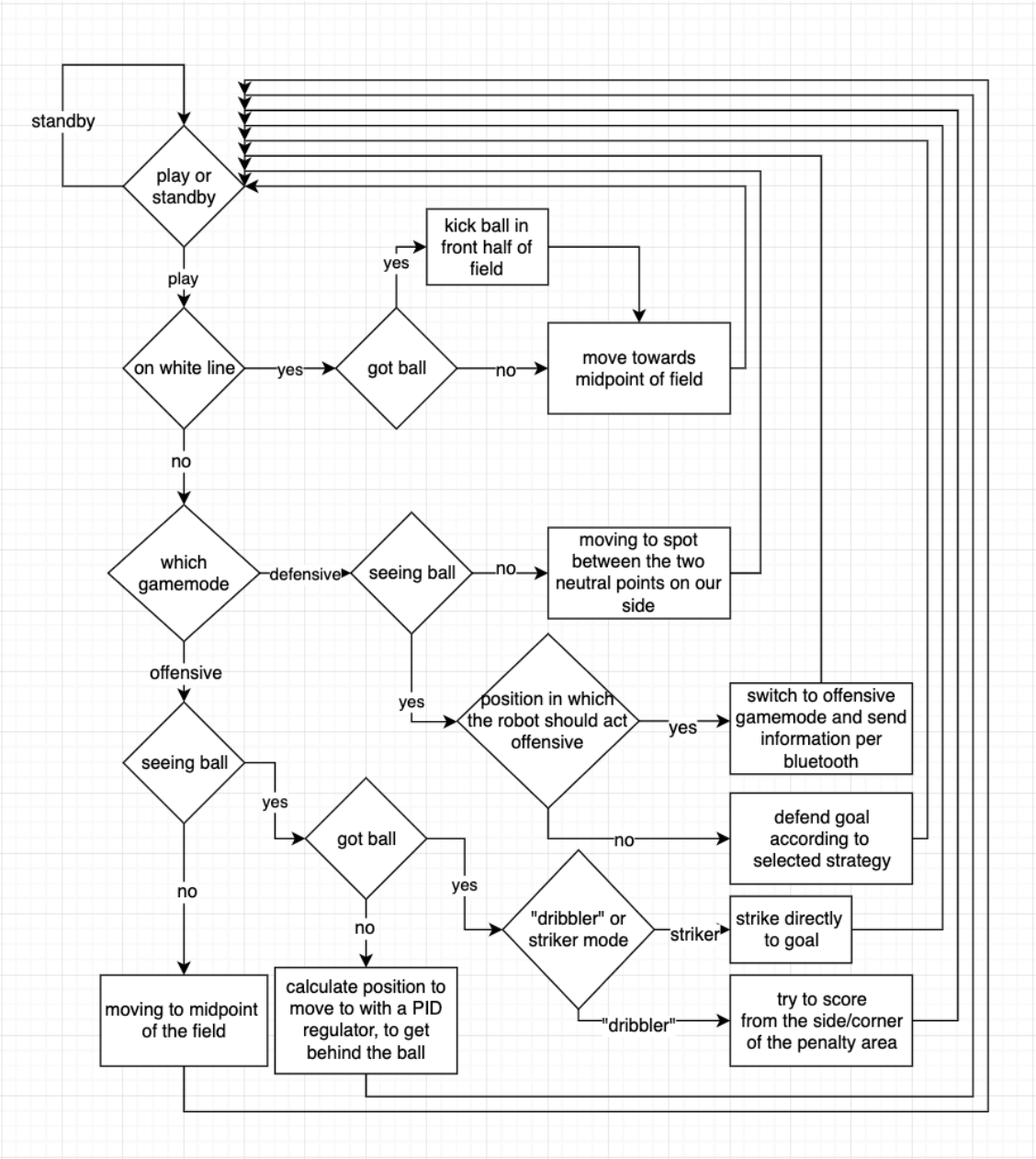
Blue pixels stand for the places where the ball actually was during the testing, while pink are positions we assumed according to nearby positions in an addition to our data to make the conclusions you get out of this KPI more intuitively readable.

4. Main code logic

4.1. Basic idea

Our code to decide which action should be performed based on the sensors data is written in Arduino and serves two main goals.
First to keep our gameplay as safe in terms of not going out of bounds or leaving our own goal uncovered and second to be as simple as possible for the complexity of gameplay that we want to create.

Our fundamental code logic looks like this:



Hidden in this diagram are our three greatest software achievements (4.2, 4.3, 4.4) after our AI for object detection.

4.2. Gamemodes

First of all our four game strategies.
The most basic is the goalie that tries to block the way between the ball and the center of our goal to prevent the opponent from scoring a goal.
A bit more agile but still defensive is our defender that tries to get the ball out of our half of the field into the opponent's half by pushing or kicking it.
Our first offensive mode is the striker that gets behind the ball to then move forward and kick or push the ball into the goal to score by relying on speed and a straight line of movement.
Last but not least is our second offensive gamemode which is the most advanced. The “dribbler“ gamemode tries to get behind the ball, move along the side of the field with the ball turned away from the opponents goal, so that they ideally cannot see the ball until the robot is in the corner pocket of the field. There it performs an orbital movement around the ball and goal to then score.

4.3. Positioning

All of these game stategies and movements rely on something very important: knowing your position in the field.
Thanks to the extremely solid detection of the goals by the AI we can use them as fixpoints for the positioning.
We can calculate the direction and relative distance to the goals, the ball and the midpoint of the field by using the angle between the goals in the mirror and the ratio of radius to maximum possible radius of each object we want to know the position to.
Therefore, we can for example evaluate the maximum speed we can move into a specific direction from the current position in the field to accelerate towards the center of the field and decelerate just enough not to slide over the line.

4.4. Movement on the field

Because we have this not very precise but robust positioning, we were able to improve our movement from “move into this direction with this speed“ to “move to this spot as fast as possible“ combined with a system similar to a PID regulator.
This gave us the opportunity to keep our precision in the movement while nearly doubling our speed.

5. Conclusion

Because of the big rule changes this year we saw a lot of interesting robots in this competition. And most of them were very good on paper but they weren't able to show this on the field. Which is why we think what really distinguished us from the rest was that we tried to keep things simple wherever we could and change only one very import thing at a time. And if we made a change we did it carefully and thoughtfully.

This approach allowed us to include new systems as shown above, while always having a solid and not over-engineered base which we could depend on.

This combination of a solid and stable system with innovative technology where it is important is what made us win after all.