Exam Project : Robot Tag

Frederik Bechmann Faarup (frfa@itu.dk), German Alexander Garcia Angus (gega@itu.dk), Mads Meinert Andersen (mmea@itu.dk)

1 Introduction

This project revolves around a pragmatic challenge—a game of robot tag featuring Thymio II robots. The task at hand involves creating two roles for our robotic participants: a seeker, responsible for tagging others, and an avoider, tasked with evading the seeker.

The game unfolds in a rectangular arena with a white floor, a gray safe zone, and a black perimeter. The seeker begins in the center, emitting red from its LEDs, while avoiders, stationed in the corners, emit blue light. Communication between robots relies on IR signals.

Our chosen strategy involves a blend of methodologies. For the seeker, we employ a behavior-based approach, integrating image processing to enhance navigation and tagging. The avoider utilizes Q-learning, a reinforcement learning algorithm, to navigate around the arena strategically.

This report details how we designed our controllers, explored various paths through both physical testing and simulation, and describes how we adapted based on the outcomes.

2 Seeker

2.1 Initial Plan

Initially, our plan for the seeker controller involved applying Q-learning to optimize the tagging of avoiders. The concept was to integrate camera input with wheel output, allowing the robot to move randomly until an avoider was detected in its frame, at which point it would navigate towards the avoider. The states and actions for this approach were outlined as follows:

States/Actions	Foward	Backwards	Right	Left
Nothing in sight	0	0	0	0
Blue left	0	0	0	0
Blue right	0	0	0	0
Blue middle	0	0	0	0

TABLE 1: States and actions for initial seeker plan, where ”left”, ”right” and ”middle” would be divided according to indices of pixels from the camera.

Rewards were assigned when the robot approached within a certain distance. To refine the system's accuracy, we conducted tests, primarily focusing on determining the distance at which the IR communication sensor could effectively tag an avoider. The test protocol was as follows:

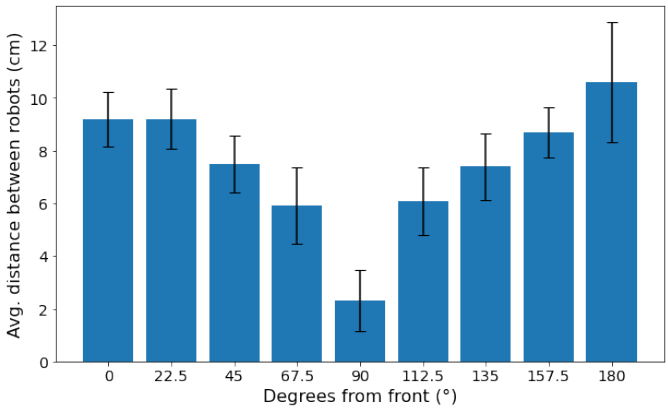

Fig. 1: Average distances from the seeker robot to the avoider robot before the latter is tagged. The avoider is positioned with different angles relative to the seeker, such as to see how its color signal affects the avoider. We took ten measures at each degree point up to 180 degrees and computed the average. The standard deviations are included, too.

1) Borrow an avoider from another group.
2) Constantly emit ”1” using our seeker controller.
3) Place the seeker at various angles around the avoider robot (starting from 0 degrees and increasing by 22.5 degrees for each position, up to 180 degrees)[1].
4) Start the seeker approximately 20 centimeters from the avoider, gradually moving it closer until the avoider turned purple.
5) Record the measurement from the edge of the seeker to the edge of the avoider with a measuring tape.
6) Repeat steps 4 and 5 ten times for each position.

In Figure 1, the results of our tests are depicted. The x-axis represents the degrees of the avoider relative to the seeker, while the y-axis is the average distance of ten measures before the avoider was tagged at that angle.

The graph illustrates that the distance required to tag an avoider significantly varies based on the tagging angle, and particularly, that the avoider needs to be very close before being tagged when it is on the side (90 degrees angle) of the seeker.

1. For clarity, it's important to mention that the reverse orientation (minus degrees from the starting point) is interpreted to be the same.

The idea behind the tests was to establish a threshold where we would be nearly guaranteed to have tagged an avoider. However, due to limitations in our hardware, pinpointing a specific distance to an object posed challenges. These challenges involved determining the precise distance to an object in front of our robot. The Thymio was equipped with a Lidar sensor on top; however, in previous assignments involving the use and testing of the Lidar sensor, once the object in front of the robot came within a certain threshold (10 centimeters in our case), it could no longer accurately detect the distance. Additionally, the only other way to detect objects in front was with the IR Proximity Sensors on the Thymio. However, testing those in previous projects revealed that once the distance exceeds 5 centimeters, they no longer detect anything.

Consequently, we chose an alternative approach by assessing the amount of blue in the camera frame when an avoider was closer than our threshold. To determine the quantity of blue pixels in the frame under these conditions, we developed a test protocol.

During the protocol discussion, a couple of conclusions emerged. Firstly, this approach appeared more complex than initially intended. Secondly, the group was concurrently engaged in various activities—some were testing physical boundaries, while others were setting up simulations.

At the beginning of the project, we agreed that prioritizing the training of our Q-learning algorithm within a simulation was a great way to enhance efficiency, with the goal of achieving a faster development cycle and execution.

However, simulating this Q-learning posed challenges. It proved to be quite difficult as it involved incorporating multiple robots within the simulation and creating a ”fake” camera that could detect these additional robots. Therefore we decided on a different approach for our seeker controller.

2.2 Final Plan

Instead of the Q-learning approach, we made a strategic shift towards behavior-based robotics for our seeker controller. This alternative method centered on a more straightforward and reactive approach to navigate the robot effectively.

Our new strategy involved capturing images using the camera, with each image divided into three distinct parts: left, middle, and right. Applying image processing techniques on each segment, we checked the amount of blue in each section. If the left segment had the most blue, the robot would turn left; if the right segment had the most, it would turn right. Conversely, if the middle had the highest concentration of blue, the robot would proceed straight ahead. If there were no blue in all segments, the robot would initiate a spinning motion until blue was detected again.

This behavior-based approach aimed to simplify the decision-making process, providing the seeker with an adaptive and responsive mechanism for movement.

For image processing, we utilized the OpenCV Python library to analyze the presence of a specific color range in the images. We applied a color filter defined by the lower and upper color bounds. The lower color is set to:

[90, 50, 50]

and the upper color is set to:

[128, 238, 255]

We proceeded to create a binary mask using OpenCV's **‘inRange‘** method, isolating pixels within the specified color range and designating them as white, while pixels outside the range were set to black. Subsequently, we employed the **‘countNonZero‘** function to calculate the count of non-black pixels in the mask.

By applying this process to each third of the image, we were able to compare which part had the most color within our specified range and act accordingly.

One of the challenges we encountered using this approach was determining the correct lower and upper bounds. Initially, we adopted a method where we took a picture of a blue avoider with the camera, used a color picker tool, and added 20% to the values for our upper bound while subtracting 20% from the values for our lower bound. However, during testing, we observed that the camera picked up a considerable amount of color even when not facing an avoider.

To address this, we experimented with adjustments, trying increments of 5%. With a 5% adjustment, our seeker struggled to identify any avoiders when positioned further than 30 centimeters away. At 10%, the performance improved, but it still couldn't reliably detect an avoider on the opposite side of the arena. However, a 15% adjustment seemed to strike a balance, enabling the seeker to spot avoiders on the opposite side of the arena. Despite these adjustments, we continued to capture colors outside the boundaries when not facing avoiders, with the frequency decreasing as we lowered our percentage.

We also experimented with Chat-GPT 4 by uploading images taken with our seeker camera and asking Chat-GPT to analyze them and identify upper and lower bounds for image recognition. However, the results weren't very useful, as it consistently detected color within the boundaries even when not facing an avoider.

Another issue we faced was related to image processing delay. Through tests, we observed that the time elapsed from capturing the photo to executing actions based on the processed image could place us in a different state than anticipated.

For instance, if the instruction was to spin left and an avoider entered the frame, the processing would position the avoider on the left side of the frame, prompting the robot to turn left.

However, by the time the instruction reached our robot, the seeker would have turned so much left, that the avoider would would be in the right frame of the seeker.

In the subsequent iteration of behavior, the seeker would capture a new image, having turned quickly enough for the avoider to be on the right side of the frame, leading the seeker to turn right, meaning we would be stuck in a loop where the seeker would just turn from left to right.

During the initial tests of this issue, we experimented with a stationary avoider. However, as we transitioned to testing with moving avoiders, this problem became even more noticeable.

While speed is typically advantageous in playing tag, it becomes less beneficial if the seeker is unable to accurately locate avoiders. Consequently, our solution was to reduce the speed of our seeker. This adjustment allowed the

avoiders more time in each section of our image, providing an extended period to react to potential variations or discrepancies.

Our journey in creating an effective robot tag seeker involved strategic shifts and adaptive solutions. Initially centered on Q-learning, we transitioned to behavior-based robotics for simplicity and responsiveness. Utilizing image processing, we segmented frames to determine blue prevalence, adjusting seeker speed for balance. Despite achieving a functional seeker capable of finding and tagging avoiders, we recognize that the persistent hardware limitations, particularly in the camera quality, remain a significant hurdle. Specially the camera's tendency to detect blue even in the absence of avoiders.

3 AVOIDER

When building the avoider behavior, we decided to attempt Q-learning. We wanted to use a very simple approach, as we quickly realised that a more advanced approach to states, actions and rewards came with assumptions that did not always hold. This comes from the nature of the formula for Q-learning, where the Q-table after a given state *s* and a given action *a* is updated at each iteration as such:

$$Q(s,a) = (1-\alpha)\cdot Q(s,a) + \alpha\cdot(R+\gamma\cdot max(Q(s',a'))) \quad (1)$$

Where α and γ are hyperparameters. *R* is the intermediate reward of the action, and *Q(s',a')* represents the Q-value for the next state. In other words, in order to update a Q-value in the simulation after an action, we had to know which state preceded the action. This quickly became infeasible due to the non-deterministic nature of the robot movements.

With that in mind, we implemented a simple Q-learning algorithm with the goal of staying within the borders of the arena by using the two ground sensors beneath the robot. Table 2 shows the initialised Q-table. The states represent the two sensor values, set to binary by a threshold in *light reflected*, which we found after practical testing. A sensor was above the black border line if it reflected less than 100 light. The driving speed of the robot was in simulation set to 200 for driving forward and backwards. For right and left, the robot simply had its wheels rotate with the driving speed in opposite directions. As it can be seen in the table, we initialised the value of *Q(Both black, Forward)* to *-20* as we realised that this behavior was unwanted.

States/Actions	Foward	Backwards	Right	Left
Both white	0	0	0	0
Left black	0	0	0	0
Right black	0	0	0	0
Both black	-20	0	0	0

TABLE 2: Initialised Q-table of the avoider robot

In the simulation we experimented with different variables, such as the size of the arena, the timestep of the robot, the wheel speeds, α & γ of Q-learning and number of simulated iterations. We noticed that since the robot would initially not know what to do, it would occasionally drive out of the bounds of the arena, where it would be stuck forever. To avoid this, we reset the robot position to the middle of the arena and updated the Q-table. For reproducability, all the parameters and values of the final Q-learning simulation used are listed in Table 3.

Parameter	**Value**
Arena size	40x40 cm
Robot speed	200 rpm
α (learning rate)	0.1
γ	0.9
ϵ (random action during learning)	0.2
Iterations	5000
Simulated ground sensors degrees from robot direction	5°
Simulated ground sensors threholds distance	1 cm
Reward, both white	1
Reward, left black	10
Reward, right black	10
Reward, both black	-100

TABLE 3: Parameters of Q-learning in simulation

To simulate the ground sensors, we used the direction of the robot in each simulated step to which we added and subtracted $5°$ (or $\pi/180 \cdot 5 =\sim 0.09$ in radians) for the left and right sensor respectively. This created two straight lines pointing slightly left and right in front of the robot. We calculated the intersection point of the two lines with the arena borders, and then used a threshold distance of *1 cm* to determine if either of the ground sensors was on top of the arena border.

Table 3 includes the final rewards after testing. They were chosen because we wanted to penalize driving outside the borders of the arena and reward line-following, as the latter meant being at the borders of the arena and therefore in theory making the avoider hard to tag.

We also experimented with adding additional actions, such as "big forward" and "big backwards", with double the speed, to allow more complicated behavior. We theorized that this would allow the robot to move quicker away from the black line to avoid penalty, but it did not prove successful.

We used a neat trick to solve the problem from Equation 1 that we needed to know the preceding state after an action in each iteration of the simulation. Whenever an action *a'* was chosen (either based on the exploitation of the Q-table or by exploration), we spawned a copy of our robot, $robot_{theta}$, with identical position and direction. Then, $robot_{theta}$ took the action *a'*, after which we computed the new state from the sensor lines. Finally, we updated the Q-table with the new rewards and made the original robot take the action.

In the end after running the final simulation, we implemented the resulting Q-table as behavior in the physical robot (with the randomness omitted) and tested it in the arena ten times before the final tag-play. We experienced that the robot quick found the line at the border of the arena, where it had a couple of behaviors. Either the robot followed the line, if it was pointing sideways to the line, or it would be stuck driving backwards and forwards, if it was facing the line. The latter can be explained because the final Q-table had a high reward when driving forward, when both sensors were white, or backwards, when both sensors were black. This resulted in the robot driving backwards and forwards at the line.

4 Results

On the 13th of December we had a competition, where we would compete against the other groups, participating as both seeker and avoider.

4.1 Seeker

Our seeker did unfortunately not manage to catch all 4 avoiders within the time limit of three minutes. The most notable observations made were; Firstly, the speed of the seeker, which we intentionally lowered strategically for better image processing did not prove to be beneficial, as the avoiders manage to speed away from us. Secondly, the camera tended to pick up colors in the "wrong" sections on the frames, prompting the seeker to turn right or left when an avoider was placed in the middle - we suspect some deviation in light difference from the testing room to the competition room. All testing regarding the seeker was performed in room 5a60, while the competition was held in 5a12-14.

At the beginning of the round, we observed that our seeker successfully identified and "hunted" an avoider. However, as the avoider sped away, we only managed to get within a reasonable tagging distance when facing the side of the avoider. As Figure 1 shows, when taking a 90ish degree angle to catch the avoider, it requires the seeker to be a lot closer than other angles.

Unfortunately, within the first few seconds of the round, two of our opposing avoiders exceeded the track limits (still displaying blue with their LEDs). This might explain why our robot picked up blue from unwanted places, as the reflected light from disqualified avoiders could have interfered with our camera.

Overall, we were disappointed with the performance of our seeker. We observed the wanted behavior in glimpses, but not consistently enough to challenge for the victory.

4.2 Avoider

Our avoider managed to not be caught by an enemy seeker in 1 of 4 rounds. We observed that our Q-learning worked as intended, as we did not exceed the track limits in any of the rounds. However, we still observed some limitations with our avoider. Firstly, we did not utilize the safe zone. This is not surprising, as our reinforcement learning algorithm did not include support for that specific state. Secondly, we were not actively avoiding the seeker. In one of the rounds, an enemy seeker was faced to directly target our avoider from the get go. As we don't actively run away from avoiders, this caused us to be caught within the first seconds of the round. Again this behavior is not surprising, as it is yet to be incorporated into our avoider.

Overall, we are happy with the performance of our avoider, because the Q-learning algorithm worked as expected, based on the results of our simulations and testing, but we still recognize the shortcomings of our avoider.

5 Discussion

Throughout the course, we've experienced technical issues and hardware shortcomings, from *LEGO MINDSTORMS EV3* color sensors, to Lidar sensors, and, in this final project, a camera. It is hard to pinpoint the exact issue with the camera, as there are a lot of factors involved, but we believe one of the main reasons for the poor competition performance was due to a difference in lighting and setup from the testing location to the competition location.

After the competition, we realized that the combination of potential angles for the robots to tag each other were almost endless. We observed that when a seeker and an avoider drove next to each other (side by side), it resulted in an "accidental" tag. This could potentially increase our tactical options, however, it would be difficult to change the behavior with our current knowledge of the hardware, as the input devices didn't allow us to record this situation.

Additionally, a consideration emerged regarding the implementation of a new behavior for our seeker, where the seeker would initiate forward movement after completing its rotational behavior, particularly in the event of failing to detect any blue spot indicative of the avoider's presence. This cycle, consisting of alternating between moving forward and rotating would be persistently executed, allowing the seeker to adaptively react to any visual cues of the targeted image in its environment and with the intention of gradually decrease the distance between the seeker and the avoider.

Although we were more satisfied with the performance of the avoider, we also realize that it had shortcomings and that developing the "perfect" avoider would be far more complex. The incorporation of actively searching for the safe zone, while simultaneously avoiding the seeker, would require far more states and a completely new rewards system, given that we kept a reinforcement learning approach.

One of the benefits with our current avoider controller, was the ease of testing as we were independent of other groups. Incorporating seeker-dependant functionality in the avoider, would require physical testing with seekers, to experiment with color bounds, distance bounds, etc. Although physical testing sounds straightforward, we experienced it to be very time consuming, due to the fact that we had to coordinate with other groups.

Of course, these physical constraints could be solved with simulation, but this brings forth the difficulty of imitating the exact conditions of the physical world in simulation. Even when simulating our simple Q-learning algorithm, we had to imitate the sensors with geometric lines. As the complexity of the model increases, the spaces that errors can occur increases, too.

6 Conclusion

We have successfully managed to implement a reinforcement learning algorithm to control our avoider, by utilizing a simulation environment for training and optimization. Possible solutions for the shortcomings, such as searching for the safe zone and actively avoiding seekers, were brainstormed; adding new states and reward functions to our Q-learning table, were deemed the optimal addition. However, adding complexity to the model means increasing the room for error.

We have experienced difficulties with our seeker, especially when processing images with the provided hardware. We tried to tackle these challenges by testing the color

bounds used in the image processing script, as well as altering the speed of the robot to maximize the time an avoider was within our frame.

In conclusion, while acknowledging the current limitations, we are optimistic about the potential advancements by continuing to utilize experimental methods, behavior-based robotics, sensors, simulations, and reinforcement learning.