Departamento de Informática da Faculdade de

Ciências da Universidade de Lisboa

Robôs Móveis 2022/2023

Final Project
Thymio Robot Shape Finder

Group 9

Student	Number
José Siopa	fc60716
Martim Costa	fc60504

May 22, 2023

1 Introduction

The Thymio robot is an educational platform designed to introduce students to the field of robotics and programming. It provides a user-friendly and adaptable platform for education and experimentation. As part of our class's curriculum, we started working on a project that aimed to increase the Thymio robot's functionalities by giving it the ability to navigate autonomously in a controlled environment and recognize a certain shape specified by the user.

This report's objective is to outline the development and results of the project. The project developed and put into practice a solution using the knowledge we learned in class.

2 Objectives

- Create a navigation system for the Thymio robot so that it may travel on its own inside a predetermined environment while avoiding obstacles and adhering to a predetermined course.
- Implement shape identification: Using image processing techniques to recognize and distinguish between various shapes, the Thymio robot should be able to identify a particular shape that the user has specified.
- Integrate classroom knowledge: The project's goal was to use the theoretical ideas and practical strategies we acquired in class, such as programming, using sensors, and implementing algorithms, to achieve the project's goals.

3 Approach

The project was coded using a Python-based methodology that made use of the behavioral tree concept. The Thymio robot's navigation and shape identification tasks could be carried out in a well-organized manner thanks to the behavioral tree that was developed utilizing sequential and selector nodes.

Sequential nodes were used to define a sequence of actions to be executed in a specific order. The Thymio robot was able to carry out a few navigational tasks thanks to these nodes, including moving ahead, detecting obstacles, and making decisions based on sensor readings.

Selector nodes allowed the Thymio robot to choose between different actions based on certain conditions.

We attached a camera to the top of the Thymio robot to detect shapes whenever it reached an obstacle.

4 Implementation

The OpenCV library for image processing and the Thymio Direct library for controlling the Thymio robot were used in the project's development. We were able to use these libraries to combine the required parts and develop a whole set of behaviors for the Thymio robot, enabling it to move around and recognize shapes in the controlled environment, with a camera attached to it.

The behaviours implemented were:

- **Obstacle Detected**: This behavior is triggered when the Thymio robot detects an obstacle in its path. It starts the obstacle handling selector, which chooses the best course of action depending on sensor readings and the robot's current condition.
- **Move Forward**: The Thymio robot is told to advance in a straight line at a certain constant speed. It is incorporated into the move sequence to provide easy environment navigation.
- **Align**: The Thymio robot is aligned with an obstacle using the align behavior. It uses the robot's proximity sensors to align itself with the object.
- **Backoff**: The Thymio robot's backoff behavior is activated after aligning itself with the obstacle, to give the camera a good frame of the paper.
- **Rotate**: The Thymio robot is told to rotate in place using the rotate behavior, for a certain time. It has a higher chance to turn right, then left (70% for right).
- **Has Detected Correct Shape**: When the right shape is found, this behavior turns on the Thymio robot's flashlights, acting as a visible signal. It offers visual proof that the shape was correctly identified.
- **Idle**: The Thymio robot's default state when it is not actively performing any particular action is represented by its idle behavior. It enables the robot to remain still until other behaviors or circumstances cause it to move.

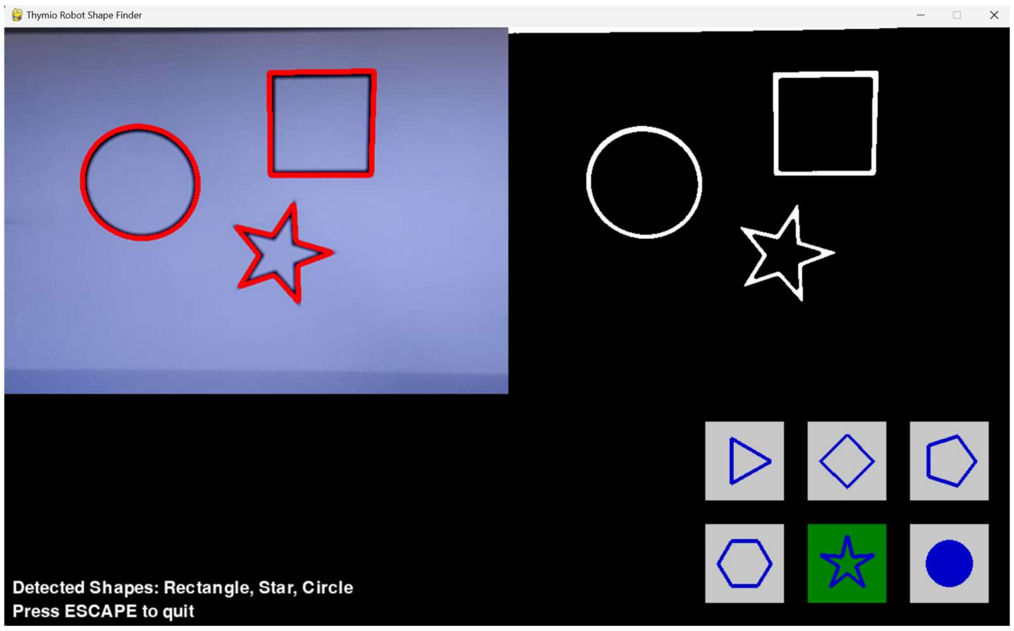
Using the behavioral tree technique, we created several sequences and selectors that implemented these behaviors:

- **Shape Detect Sequence**: This sequence uses the behaviors "Has Detected Correct Shape," "Flash Lights," and "Idle." It is in charge of recognizing the user-selected shape and providing the appropriate visual feedback via the robot's flashing lights.
- **Align Sequence**: The "Align," "Backoff," and obstacle handling selector actions are combined in the "Align" sequence. It makes sure that the Thymio robot is correctly oriented in the direction of the object, and uses the backoff behaviour to better align the camera with the possible obstacle.
- **Obstacle Handling Selector**: This selector determines the appropriate course of action when an obstacle is detected. It selects between the shape detect behavior and the rotate behavior based on the sensor readings and the current robot state.
- **Move Sequence**: This move sequence makes use of the behaviors "Obstacle Detected" and "Move Forward." It enables the Thymio robot to move through its surroundings on its own and deal with any challenges it encounters.

- **Top Selector**: The Thymio robot's top selector controls how decisions are made in general. It makes use of the move sequence and the align sequence to choose the best course of action.

For the shape detection we implemented the following transformations to the frames:

- **Black and white conversion**: The acquired image was initially changed from color to black and white. This conversion reduced the amount of information that needed to be processed, simplifying the subsequent analysis.
- **Blur Effect**: A blur effect was applied to the black and white image to lessen the appearance of shadows and to smooth out any irregularities in shape detection.
- **Thresholding**: The next step involved thresholding the blurred image to convert it into a binary image. The thresholding procedure divided the grayscale image into the categories of black and white.
- **Color Inversion**: The colors in the thresholded image were inverted to make them easier to utilize and process later. The shapes were now able to be used as black contours on a white background thanks to this.
- **Contour Extraction**: The OpenCV function *findContours* was used to extract the contours from the binary image. The shapes that were present in the scene were identified by identifying and extracting the contours from the binary image.
- **Filtering by Area**: The extracted contours were filtered based on their area. Shapes that were either unusually huge or little in their areas were eliminated since they weren't likely to be the specific shapes we were looking for. This filtering procedure helped to reduce noise and concentrate on the possible shapes that we were looking for.
- **Shape Identification**: The shape of the discovered object was identified by calculating the number of edges that persisted after the filtering phase. For example, if a shape had four edges, it was categorized as a rectangle, five indicated a pentagon, and so on.



Detected Shapes: Rectangle, Star, Circle
Press ESCAPE to quit

Figure 1 - UI of the application

5 Results

For the results, we found that depending on the obstacles placement the robot could find itself stuck for a while before being able to break out from the situation.

The camera detected many shapes in the environment, so it was a possibility for it to falsely detect the selected shape sometimes.

The communication between the Thymio and the python code had a slight delay so we couldn't implement precise behaviors, like avoiding a fall since the delay made it so it couldn't react in time.

Overall, the robot was able to navigate in fairly simple contained environments and detect simple shapes if the frame didn't have a high number of other details in it.

6 Final Comments

We found this project to help us develop our expertise in planning and coding the behavior of a robot, as well of the nuance that are real time sensor outputs and response times, that limited the way of doing certain behaviors or in some cases made it impossible to create.

In the beginning, we also wanted to implement a mapping behavior, but due to time constraints and, when early testing, we found that the odometry implementation of the robot was unreliable and decided to focus on the basic idea of the project.

7 Bibliography

1. Mobsya Association, "Programming with Python", https://www.thymio.org/products/programming-with-thymio-suite/programming-with-python/
2. EPFL – Mobots, "thymiodirect 0.1.2", https://github.com/epfl-mobots/thymio-python