Follow that Thymio

1st Alessandro De Grandi
Master Student of Informatics (First year)
USI
Lugano, Switzerland
degraa@usi.ch

2nd Filippo Casari
Master Student of Informatics (First year)
USI
Lugano, Switzerland
casarfi@usi.ch

***Abstract*—The focus of this project is to build a system that allows a RoboMaster to follow a Thymio in simulated environments created in CoppeliaSim, using ROS controllers and machine learning models.**

I. Introduction

This project is divided into 5 main tasks:

- Building a controller to move Thymio.
- Collecting a Dataset of images of Thymio from the RoboMasters camera.
- Building and training a machine learning model that can be used to detect Thymio's position relative to the observer.
- Building a controller that allows RoboMaster to follow Thymio.
- Testing the controllers and the model in simulated environments.

Each task was accomplished in its simplest form initially, then refined overtime, with an iterative process of tests, failures and improvements described in the next sections, aimed at achieving the final goal of obtaining a controller that allows the RoboMaster to recognize the position of Thymio relative to itself, and act accordingly to follow its movements.

II. Moving Thymio

To simplify the subsequent tasks, it was convenient to make Thymio move in a predictable way. To do that we implemented a simple controller (Fig. 1) that allows Thymio to follow a black line on the ground. Assuming that Thymio starts on the line, we use the 2 ground sensors, left and right. If the left sensor is not detecting the line anymore, Thymio will steer right, if the right sensor is not detecting the line, it will steer left, otherwise it will keep going forward. In its simplicity, this approach works well.

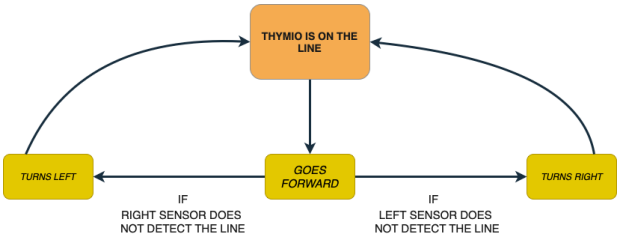


Fig. 1. Thymio controller logic

III. Collecting the dataset

The initial approach of data collection was to gather two classes of pictures separately: left and right, each in an automated way using appropriately programmed controllers, eg: place a rotating thymio on the left/right and a Robomaster going backward and forward. This way we could rapidly gather a small initial dataset (1000 images) and start training and testing the models. For further improvements we had to increase the size of the dataset by using similar techniques, eg: placing Thymio on a circular line path with Robomaster rotating in the middle (Fig. 3). As the models got better we could also include into the dataset pictures manually gathered during tests, especially in the specific instances where a model was failing.

A. Preprocessing

The RoboMaster camera acquires images with a resolution of 360 by 640 pixels. These images are then cropped, maintaining only the bottom half portion, and rescaled by 50%, then a normalization is applied, using mean and standard deviation of the entire dataset.

B. Augmentation

We incrementally gathered 10272 images of Thymio(left and right). By flipping the images horizontally, and inverting the labels, the size of the dataset doubles. Also, because we changed our classification approach to a Thymio or no-Thymio classification, the images are split in two halves vertically, doubling again the dataset size, another automatic re-labelling was necessary.

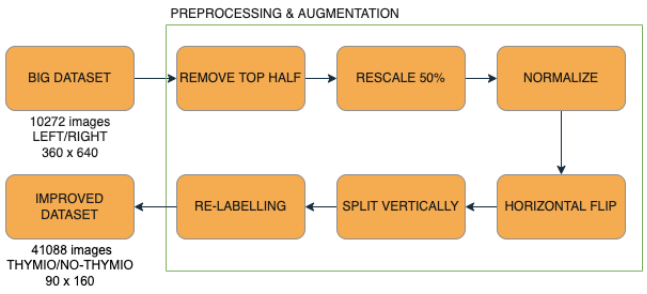


Fig. 2. Preprocessing & Augmentation

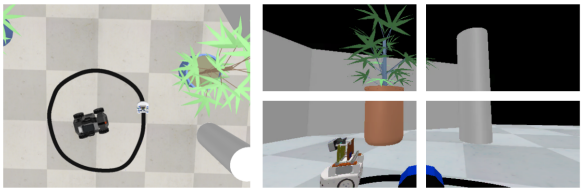


Fig. 3. Automatic dataset collection

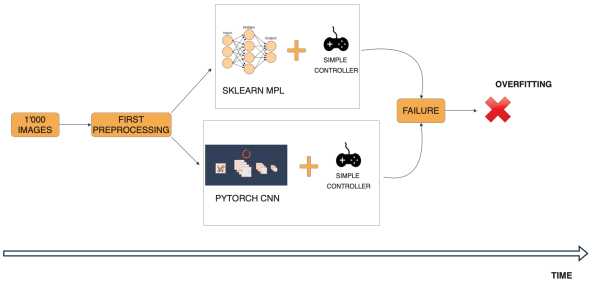


Fig. 4. Failures

C. Final Dataset

The final dataset consists of 41088 images with a resolution of 90 by 160.

IV. The Models

A. CNN from Pytorch

We built a custom CNN model using pyTorch, initially with the task of classifying if Thymio is either on the left or right side of the image taken from Robomaster camera:

Even though the model had great accuracy on training, validation and test set. It would not perform as expected when used in a simulation. For example: In new and unseen scenes it would randomly go towards walls or other objects, and in scenes that we used to build the dataset, the Robomaster seemed to follow Thymio, only to then surpass it and finish the path on its own! It was clearly overfitting the scenes without generalizing.

B. MPL from Sklearn

Even though the CNN seemed to be the best choice to us for this kind of project, we also considered to use a different, lighter, model. Indeed, we implemented an sklearn Multi-layer Perceptron classifier that takes as input the same dataset used by the CNN. However, the input needed to be reshaped for the feed forward network. Consequently, we cut the top half of the images, we re-scaled them to one third (for better performance) and, finally we flattened the pixels of every image. For the MLP classifier we used the following hyperparameters:

- 100 hidden layers
- Relu activation function
- tolerance = 10e-3
- early stopping: True

After training the model, we tested it within the simulation and, the RoboMaster followed the Thymio. Nevertheless, after changing the scene the model did not recognize the Thymio properly. In fact, we suspected that also this sklearn model tended to overfit the scene like the CNN (Fig. 4).
After several experiments, we concluded that the CNN had to be the best choice because of the spatial information within the image, that is otherwise lost by linearizing the input for the MLP. So we moved forward with the CNN approach and abandoned the sklearn model.

C. Improved approach

To solve this problem we had to rethink our approach. By splitting the image in half vertically and making two separate predictions we could use the same dataset and effectively double its size, keep a similar but smaller model (input smaller images), and just change the task to classify if Thymio is in the image or not. This also allows for greater flexibility in the controller part as described in the next section.

D. Final model

- Convolutional Neural Network (CNN)
- Relu activation function
- Binary Cross-Entropy(BCE) loss
- Learning rate = 0.001
- Momentum = 0.9
- Early stopping at 98% validation accuracy
- Classify Thymio or no-Thymio

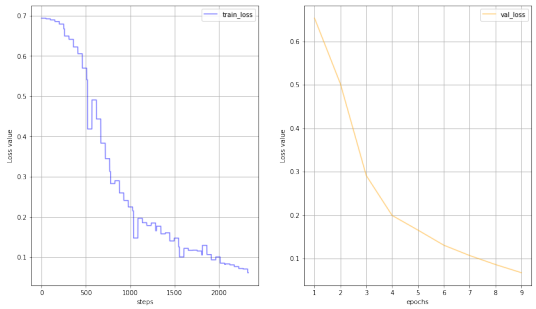


Fig. 5. Training & Validation losses

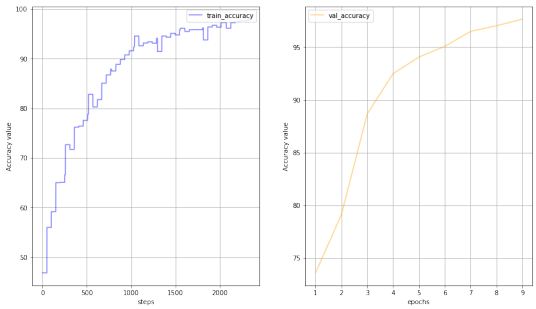


Fig. 6. Training & Validation accuracies

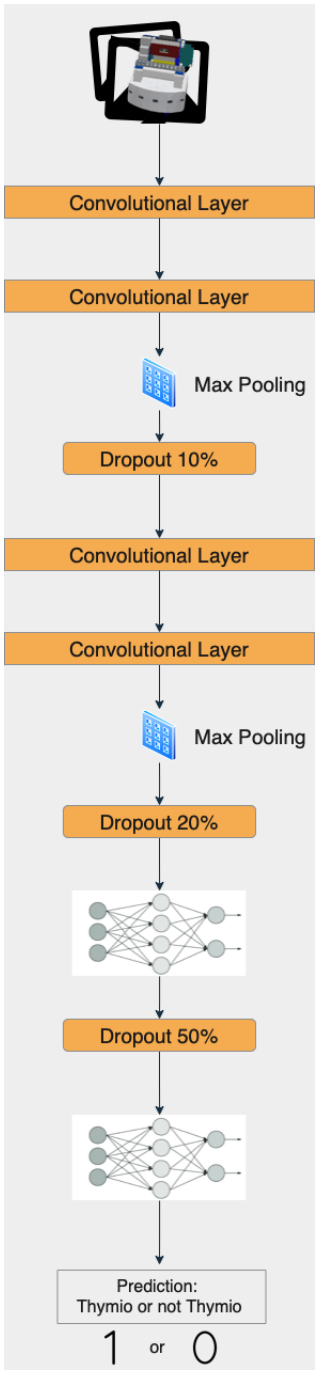


Fig. 7. Pytorch Model

V. The Robomaster controller

Initially we wrote a simple controller that would keep the Robomaster going forward and try to use the trained classifier predictions, left or right, to steer. As described in the previous section the results were not satisfying.

A. Improved approach

By simplifying the classifier component to essentially a Thymio or not Thymio classifier, we also obtained more information that could be used to implement a more sophisticated state machine illustrated in Fig. 9 . By making two separate predictions at every timestep (Fig. 8).

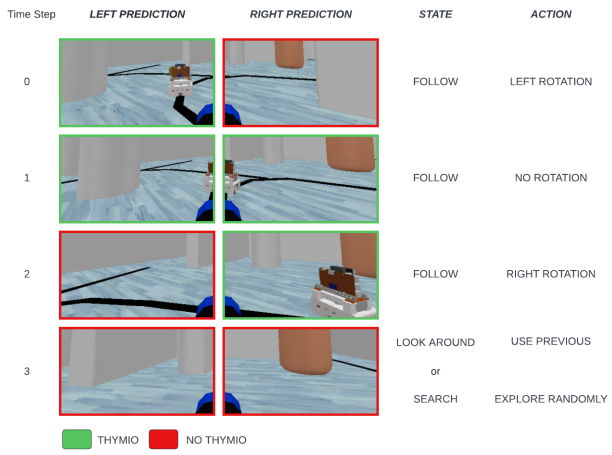


Fig. 8. Improved approach

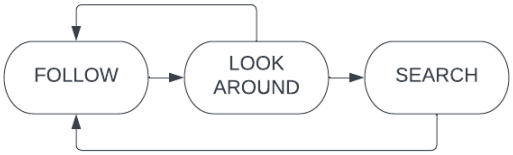


Fig. 9. Robomaster controller logic

B. FOLLOW

The rotation of the RoboMaster in the FOLLOW state is determined by the two predictions made by the model. The forward motion is controlled using 3 forward looking sensors (picture) that allow to disable RoboMasters forward motion when an obstacle is detected in front of it, the addition of the sensors to the Robomaster model is also very useful to allow Robomaster to move faster than Thymio when it needs to catch up , and slow down or stop when it gets too close. Meanwhile both its actions and predictions are buffered in two fixed size double-ended queues, that save the most recent predictions and actions(left/right rotation), discarding the oldest ones at every time step, this allows for some uncertainty and misclassifications of the model, and also in same cases to proportionally adjust speed, for example when Thymio exits the field of view, as the predictions buffer gets filled with no-Thymio predictions, the Robomaster will slow down proportionally, until the queue is full of no-Thymio, at that point the state will change to LOOK AROUND.

C. LOOK AROUND

In the LOOK AROUND state Robomaster will attempt to recover sight of Thymio, by consuming the actions buffer

and rotating towards the directions where Thymio was last seen, until the buffer is empty, then the state will change to SEARCH.

D. SEARCH

In the SEARCH state RoboMaster will simply move forward until an obstacle is detected by its sensors (Fig. 10) and then rotate until it is safe again to move forward, this way it will randomly explore the room, and meanwhile looking around for Thymio, until there are enough predictions of Thymio in the queue, then it will move back to the FOLLOW state.

Fig. 10. Robomaster sensors configuration

VI. Final tests & results

A. Successes

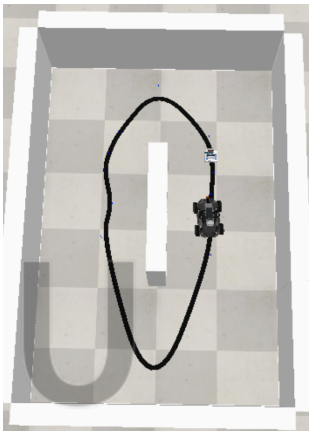
We have tested the system in multiple seen and unseen scenes, and it works quite well, the Robomaster is able to follow Thymio along its path, and is able to recover sight of it when it moves out of the field of vision, for example during narrow curves, it is indeed not perfect and might sometimes unexpectedly fail, especially in untested scenes where objects might be confused for a Thymio.

B. Needs further improvements

However, the searching algorithm is too naive and would need a lot of improvements to allow RoboMaster to find Thymio in scenarios where it has completely no sight of it. In those cases it will mostly explore randomly, avoiding walls, but without being able to find Thymio, or when it finds it, it might approach from behind and successfully be able to follow it, but otherwise the two might collide or other unexpected or untested behaviours might happen, this is a matter for an entire other project, but nonetheless it is an area of improvement that the system allows, so it was worth including it in the project.

VII. Conclusion

In conclusion, we overall consider the project successful in its initial goal of bulding a system that allows Robomaster to follow Thymio (Fig VII).



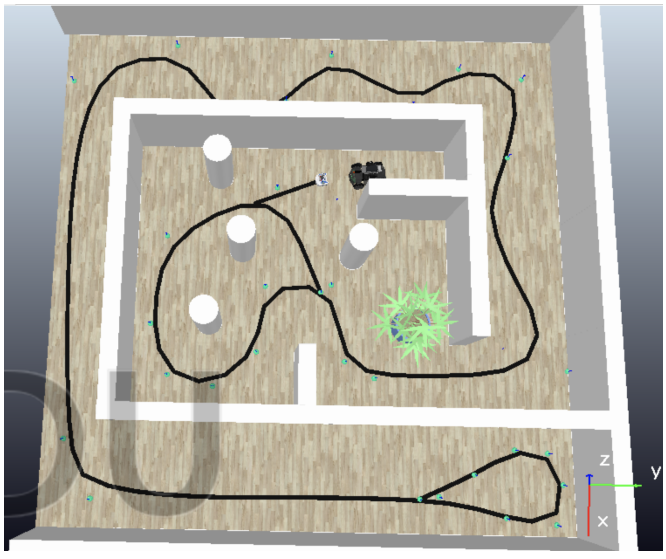


Fig. 11. Successfull tests where RoboMaster is able to follow Thymio along all its path