Robotics Project 2020/2021

1st Stefano Carlo Lambertenghi
Master Student of AI (First year)
USI
Lugano, Switzerland
lambest@usi.ch

2nd Felix Boelter
Master Student of AI (First year)
USI
Lugano, Switzerland
boeltf@usi.ch

***Abstract*—Robot maneuvering in road-like environments. A controller implementation of lane following, road sign detecting, obstacle avoiding robots.**

I. Introduction

This project is divided in 6 steps, each of them will be subsequent to the latter, with the objective of being able to have a working prototype even if a section is not fully implemented. Nonetheless we believe that each function is implemented successfully as we have intended it to be executed. The main idea is the creation of a road-like environment where the actors have the objective of successfully maneuvering without crashing and by following the world rules, such as speed limits and precedence at cross roads.
The following fundamental steps will be explained in detail as sections of this paper:

- World creation
- Lane identification
- lane following and error correction
- Cross road detection and obstacle avoidance
- Road sign detection
- HUD display
- Full logical map
- results

II. World creation

The world will consist of a textured floor where lanes of 2 colors will follow a rounded rectangular shape with the presence of intersections between 2 different coloured lanes. Moreover, 3 possible road signs will be placed where a certain action should be performed by our actors.

A. Componible tiles

To create the floor texture, displaying the blue and red lanes, 5 '.png' files have been created. One will display a crossing between lanes of the 2 possible colors, meanwhile there will be a straight and a round section for each of the 2 colors. The goal is to create a dependable group of objects that if differently placed can easily create multiple world configurations in Gazebo simulations. Each road "tile" will be of size 2x2 meters and it's center will be exactly 2 meters apart from any adjacent block.

B. Roadsigns

3 Road signs have been created: Stop, Slow down, No speed limit. For each of them we expect the actors to perform the actions they correspond to. To place them in the world, the chosen images will be used as textures of 0.4x0.4 meter planes, that will be placed close to where their logic should be followed. To facilitate the detection step, all signs will be at the right side of a lane, at a distance of 0.5 meters and their center will be 0.4 high from the ground plane. This will allow for the analysis of only a small portion of the robot's camera feed.

C. Layout

For each of the 4 cross roads, one incoming lane will be accompanied by a stop sign and the other with a slow down sign, so that if 2 robots are on a colliding trajectory, one will have priority of movement over the other. For each of the 2 colored lanes there will be 2 crossings with a stop sign and 2 with a slow down sign. Speed up signs will be placed at the start of long straight sections, and slow down signs will be placed before turns. The signs are placed with the assumption that both robots will follow the path in a clockwise fashion.

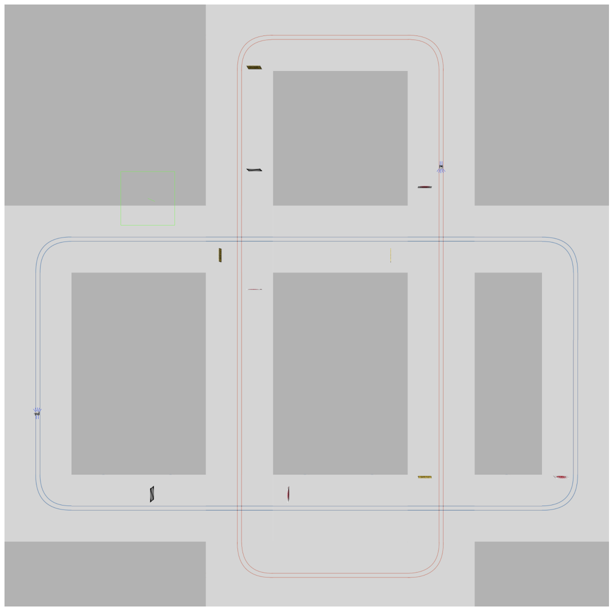


Fig. 1. Lanes layout

III. Lane identification

To keep the Thymios inside the lanes in our world, we will have to first isolate red from blue lanes, so that a ”red” robot will ignore blue lanes and vice-versa. Then, we will need to follow some logical process to identify if a robot is centered or angled towards one of the lanes sides, assuming that the robot is not outside of the lane area.

A. Image preprocessing

To isolate the red or blue lanes from the other colour, a mask is used. For each color a lower and upper bound R G B threshold is manually identified beforehand and applied to the incoming camera feed at runtime. The resulting product will be a black image with only the red or blue portions of the image remaining. As we can assume, as a given, that any red or blue road signs will be in the top half of the feed, we will not have to worry about any interference at later steps.

B. Lanes centering detection

To identify if a lane is centered or shifted in one direction from the robot's point of view, we will decide 2 horizontal pixel lines and decide if the lane at that viewing height is straightly aligned to our robot. A sweep of all pixels at the 2 given heights will be performed, and the x coordinate of any point which is not black will be saved in a vector. The first and the last of those x coordinates will then be used. It will be necessary to decide based on 5 possible conditions:

- Both higher and lower y lines contain at least 2 points:
  A mean of the top and bottom extreme points x coordinates is calculated and if it is closer than a threshold t to the center of the image for both, the robot is centered.
  If only the top line is shifted, a turn will be expected at a later step.
  Finally, if the robot is shifted towards the opposite way than the mean of the lines for both top and bottom observed heights, the robot is leaning in the opposite direction than the lines' mean.
- Top line contains one point:
  If the top line contains only one point it means that the robot is inside of a turn or greatly shifted in a straight area.
  If the bottom line's mean is outside of t, then the robot will need to turn toward the direction of the point's mean.
- Top line contains no points and bottom line contains 2 points. The robot has found the end of the lane(not tested) or it finds itself in a sharp turn.
- Top line contains no points and bottom line contains 1 point:
  The robot has to turn in the direction of the point compared to the center of the image.
- Top line contains at least one point and bottom line doesn't:
  The robot is outside of the lanes

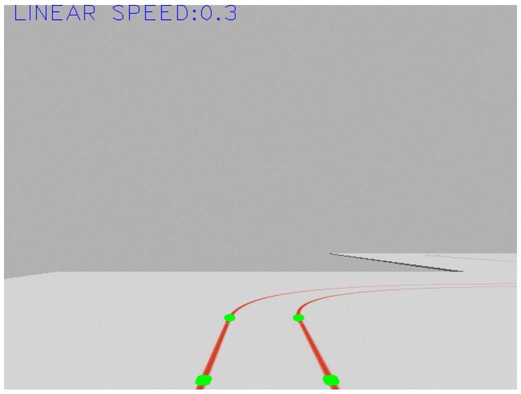


Fig. 2. Lanes with detected points on high and low lines

IV. Lane following and error correction

A. lane following

Even if we are able to classify the robot's condition with multiple points, the easiest and most efficient way to guide the robot is found by only deciding the steering angle by observing mostly the bottom line which is chosen at a very low height(close to the bottom of the image).
The following logical diagram shows the decision making process of or controller:
If one of the lines has only one point which are colored, the other will be set as a number outside the threshold
If one of the lines has no points which are colored, the direction will not be calculated

Algorithm 1: Direction detection Algorithm

```
Result: direction
dir ← straight;
if mean of top line is inside of threshold $t$ then
    dir ← straight
else
    if bottom line contains more than one point then
        if bottom line mean is inside of threshold then
            dir ← straight
        else
            if bottom line mean is left of treshold then
                dir ← left
            else
                dir ← right
            end
        end
    else
        the direction is not updated
    end
end
```

B. lane following

Once the direction is retrieved, the angular velocity of the robot will be between 0 and 0.3 if it is misaligned and 0 if it is detected as straight.
The velocity is calculated by dividing the distance in pixels to the center detected on the lower y line by 1650, creating

a system that will adapt the turning radius in proportion to how much the robot is misaligned.
The choice of positive or negative angle is based on the direction given by the detector algorithm.

C. error correction

To return inside a lane, if the system fails, when no lane points are detected the mean will be set to 2000 or 0, creating a condition where the robots will believe to be outside of the threshold, and will attempt to turn around until it detects a lane again.
The maximum of turning angle of 0.3 is used to avoid a unlimited turning angle when the system is in an unexpected position.
This logic is only a fail safe as the robot will not change direction when less than 2 points are detected on the bottom line, a condition that can be reached only if the turning angle was left or right before loosing position, and if the turning angle is not zero, the robot will still follow a circle until the lanes are detected again.
The only situation that could warrant for such a correction mechanism is a failed turn, as the robot would continue straight uncontrollably.

V. Cross roads

Using the same lane detection method as the previous steps, if the bottom points are not detected, 2 conditions could be the cause:

- Robot has lost its lane
- Robot has detected a crossroad

If a crossroad is detected, the robot should be able to avoid other robots that might be passing using the crossing lane.
To detect the presence of another robot, we will use the 5 available front sensors, and in case an obstacle is found, the robot will not move until his path is free again.
As the Mighty Thymio contains some blue components, the blue-following robot might find himself confused as whereas he is straight or on a lane exiting course.
To avoid any lanes misclassification, the lower detection line has been lowered to the functional minimum and also the robot will wait a certain number of steps before proceeding again on his path.
This logic will then be improved when road signs are implemented.
the following algorithm explains the cross road passing logic:

Algorithm 2: Crossroad object avoidance algorithm

```
Result: Safely pass the crossroad
as soon as a crossroad is detected while Crossroad is
  not passed do
    retrieve sensor measurements
    if sensors don't detect obstacle then
        Move forward one step;
    else
        while Obstacle is detected do
            Wait still
            retrieve sensor measurements
        end
        Wait 200 iters
    end
end
```

VI. Road sign detection

To facilitate detection, the following precautions are met:

- Road signs will only be at the same height
- Road signs will only be to the right of Thymio

A. Image pre-processing

Assuming that the road signs will only be on the top right of the image will allow for the use of smaller inputs to our detection model.
We will then be able to crop the input feed to a small square portion and as a result we will also reduce the times that more than one sign is seen by the detector.
As all signs are bordered by a black background, a big risk of misclassification is expected as the model might learn to classify a black square as a road sign.
The use of a mask will solve this issue by removing every pixel of the gray background and replacing it with black.
This will result in the model only seeing the road sign outline or a black background.

B. Training data

The training data was taken from the Gazebo world, where we captured a masked image every 2 frames, an example of such an image is seen hereafter,

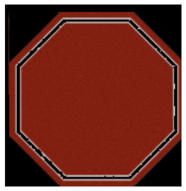


Fig. 3. Training Image of Stop Sign

In total we used 771 images, where 231 of them were used in training and 155 of them used in validation. We used a supervised learning approach which tries to classify the images by learning how the images correlate to the labels that were given to them. The labels were created as 0 (Slow Down),

1 (Stop), 2 (No Limit) and 3 (No Sign). The data was then prepared by transforming all the image data to tensors, which is a multi dimensional array used in Deep Learning, this was then then used to create the training and validation split.
The amount of images we created for each class are,

- Slow Down: 168,
- Stop: 124,
- No Limit: 113,
- No Sign: 366

C. Model

The model which has been used to classify the images is the DenseNet [1].

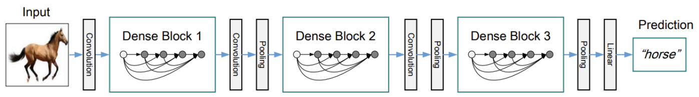


Fig. 4. Taken from [1, p. 3], which shows a DenseNet with three dense blocks.

The network which was used, named 'DenseNet-121', can be seen in the following table,

TABLE I
DENSENET-121 ARCHITECTURE [1, P. 4]. EACH "CONV" LAYER SHOWN IN THE TABLE CORRESPONDS TO THE SEQUENCE BN-RELU-CONV

Layers	Output Size	DenseNet-121
Convolution	112×112	7×7 conv, stride 2
Pooling	56×56	3×3 max pool, stride 2
Dense Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56×56	1×1 conv
	28×28	2×2 average pool, stride 2
Dense Block (2)	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28×28	1×1 conv
	14×14	2×2 average pool, stride 2
Dense Block (3)	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$
Transition Layer (3)	14×14	1×1 conv
	7×7	2×2 average pool, stride 2
Dense Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$
Classification Layer	1×1	7×7 global average pooling
		4D fully-connected, softmax

D. Training

We trained the model using PyTorch with the hyper-parameters,

- Initial Learning Rate: 0.01,
- Batch Size: 2,
- Optimizer: Adam,
- Learning rate reducer on plateau: 0.5 with a patience of 11.

For the loss function we used the 'CrossEntropyLoss' from PyTorch and we reduced the learning rate if the validation loss didn't achieve a better result in 11 epochs. Furthermore, we saved the model every epoch as well as our best model with the lowest validation loss.

E. Results

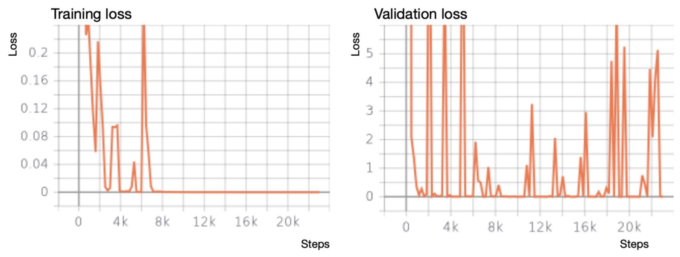


Fig. 5. Left: Training Loss, Right: Validation Loss

We trained the model for 100 epochs and from Fig. 5 we can see that both the training and validation loss converged to 0. The best model was obtained at the end of the 83^{rd} epoch, with a validation loss value of 4.0926×10^{-6}. To test the model, we loaded the model and predicted the images with the image outputs, which are obtained through the camera of the Thymio, as can be seen in Fig. 7.

VII. IMAGE OUTPUT

To show the full controller's choices, a HUD system has been added to the input stream before publishing it to ROS as a topic.
6 elements are over impressed to the original input:

- Linear speed of robot
- Crossroad detection state
  This component will output "Found crossroad" if the controller detects a crossroad

- Collision detection state
  This component will output "Obstacle detected" if the proximity sensor detect an obstacle

- Roadsign classifier input overlay:
  The area where a roadsign might be present is overlayed by the masked image that is passed to the classifier, to visualize what input it will use for detection

- Lane detection points
  Circles will be printed in the area where a lane is detected
- Roadsign match visualization
  If a roadsign is found a green square over the checked area will appear
- Logic visualizer
  The system will output what behaviour it will follow after a roadsign input

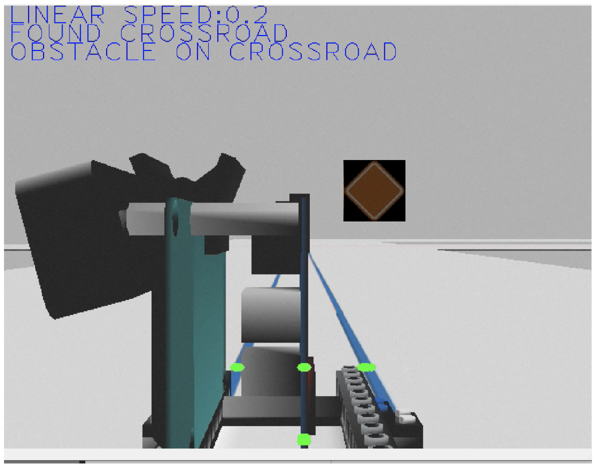


Fig. 6. Example of Speed, Crossroad Detect, Obstacle detect

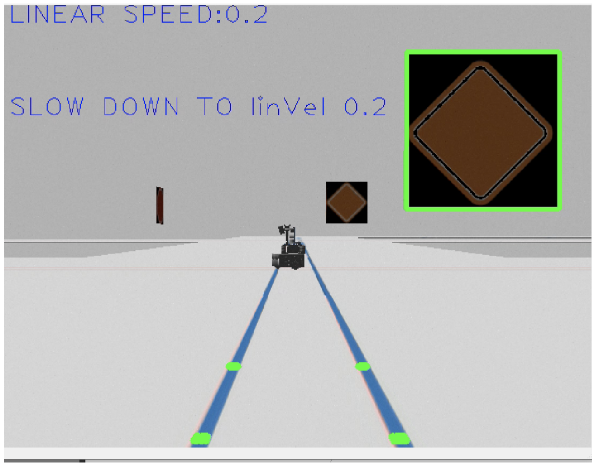


Fig. 7. Example of Speed, Roadsign Logic, Roadsign detect

VIII. Full logical map

A logical map of the robot's behaviour is represented hereafter

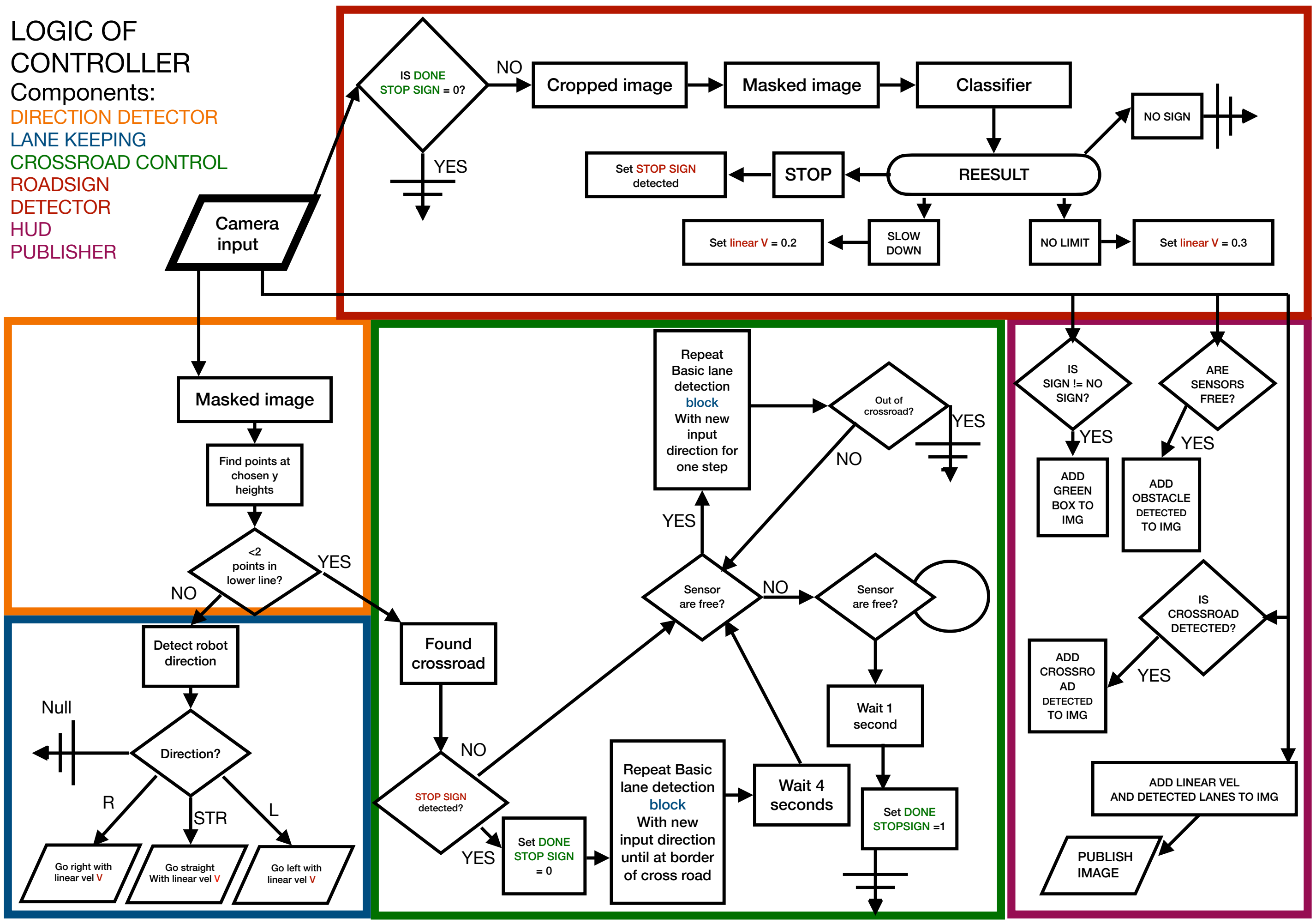


Fig. 8. Full logical Map

IX. Conclusion

In conclusion, we have created a road-like environment, where a robot uses the camera to stay inside of a lane and can turn left or right depending on where the road is moving towards. Moreover, we created a classifier that can detect road signs and can classify them accordingly, where the classification results in a behaviour that the robot follows. A few temporal sequences of behaviour will follow, demonstrating how robots comply with the given logic.

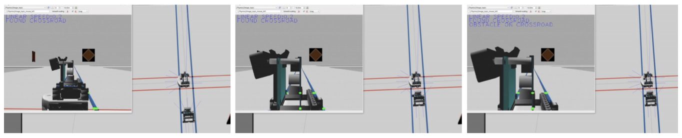


Fig. 9. Object Avoidance Sequence

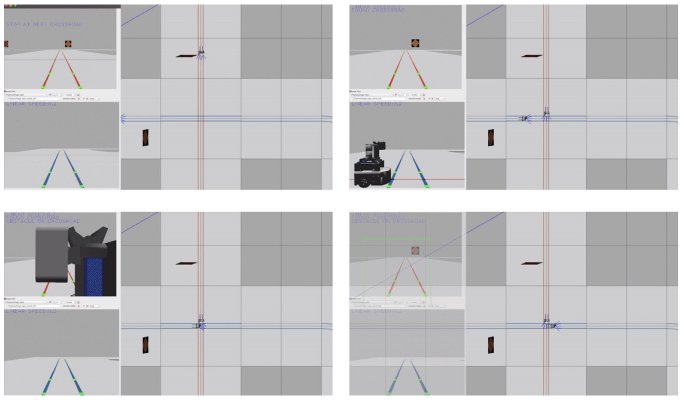
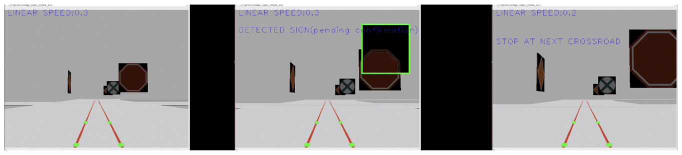


Fig. 10. Stop sign Logic Sequence



LINEAR SPEED:0.3

LINEAR SPEED:0.3

DETECTED SIGN(pending confirmation)

LINEAR SPEED:0.2

STOP AT NEXT CROSSROAD

Fig. 11. Road sign detection Sequence

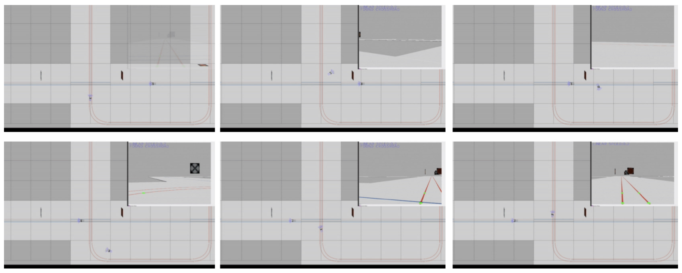


Fig. 12. Recovery Sequence

References

[1] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708.