BLG 435E - Artificial Intelligence
Final Project / Duckietown

İhsan Soydemir
150180702
soydemir18@itu.edu.tr

Hasan Hüseyin Kaçmaz
150170705
kacmazh17@itu.edu.tr

Enes Furkan Örnek
150160137
orneke16@itu.edu.tr

Talha Çomak
150160726
comak16@itu.edu.tr

***Abstract*—In this project, we tried to run duckie in *gym-duckietown* [1] environment. In this environment, there are some obstacles, traffic signs, and roads. Our purpose is to move duckie on the road without crashing into obstacles and breaking the traffic rules. The task of the duckie is to travel as a taxi. Duckie starts at a starting point and tries to pick up the passenger and continue its way. Duckie (as a robotaxi) will drop the picked-up passenger off on the route. To able to that, we trained a model with SVM (and a Pyramid method) to identify traffic signs. We also used line detection techniques such as Hough Lines to prevent the vehicle go off from the road. Using all these methods, we successfully moved duckie in different maps. We demonstrated these abilities by picking up & dropping off the passenger in the following video:**

CLICK FOR THE VIDEO LINK

***Index Terms*—duckietown, artificial intelligence, agent, SIFT, image processing, path planning**

I. Introduction

Vehicle technology has started to shift in different directions in recent years. Considering new industrial developments, it can be seen that investments and research on autonomous vehicles are increasing day by day. With today's technology producing fully autonomous vehicle is a hard topic, however; to create new approaches and test them simulations are important tools. Autonomous vehicles also have advantages such as providing traffic safety by minimizing human error. *Gym-duckietown* is an environment to simulate autonomous vehicle movements. In this environment, we tried to solve a scenario which can be experienced in real life by autonomous taxi companies. In this scenario, our duckie vehicle takes a passenger to him/her destination points. During the travel, duckie tries to obey the traffic rules and not crashing the obstacles.

II. Methods

To move the vehicle autonomously three different nodes are working simultaneously. The movements of duckie are decided according to the various inputs such as lane position, detected traffic signs, shortest path, etc. In order to follow middle of the lines, we are using a PD controller.

Initially, the map is sliced into a grid, and a map is assigned to a 2D array. After these slicing operations, the duckie starts to move over the lines and looking for a stop sign for picking up the passenger. After the first detection of the stop sign, duckie stops for 10 seconds, to allow the passenger to get in.

After retrieving the destination point from the passenger, the duckie finds the shortest path to the destination point using $A*$ algorithm. It continues to move again while following the middle of the road lines. If the duckie encounters a sign during traveling, and if the sign affects the path, the new path is determined again according to a star. These processes are repeats until the duckie arrives at destination point. Once it arrives at the destination it again waits for 10 seconds to drop-off the passenger. Finally, it looks for another passenger by moving around the map. In Figure 1 below, the general flow of the duckie is described in a diagram:

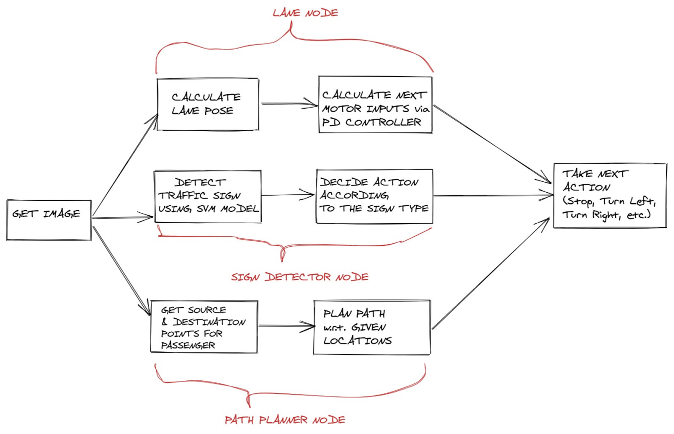
LANE NODE
CALCULATE LANE POSE
CALCULATE NEXT MOTOR INPUTS via PD CONTROLLER
GET IMAGE
DETECT TRAFFIC SIGN USING SVM MODEL
DECIDE ACTION ACCORDING TO THE SIGN TYPE
TAKE NEXT ACTION (Stop, Turn Left, Turn Right, etc.)
SIGN DETECTOR NODE
GET SOURCE & DESTINATION POINTS FOR PASSENGER
PLAN PATH w.r.t. GIVEN LOCATIONS
PATH PLANNER NODE

Fig. 1. The general structure of algorithm

In the following sub-sections, 3 nodes that are previously mentioned will be explained. We will start by the path planner.

A. Path Planner

We created the given map as a 2D array from the top-view of the environment as can be seen from Figure 2. In this map, there will be a passenger which will give the coordinate points of current and destination location. As a given point we solve the problem with $A*$ algorithm to find the minimum costly path. The advantage of this algorithm is possible to implement, change or add other distances to the distances used as a criterion [2]. This algorithm updates itself during travel according to traffic signs the vehicle encountered.

Euclidean Distance function is used as a heuristic function because it fits this problem and easy to implement and gave accurate results. As can be seen from Figure 3, at the beginning

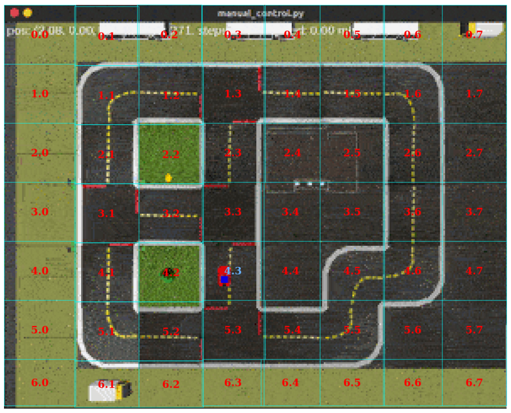

Fig. 2. Example of a map assigned to array

of the scenario created, in the next taxi station, taxi takes a passenger and generates a new path to the destination point and gave these way-points to the taxi's autopilot. Thanks to $A*$ algorithm's dynamic structure, if the current path is not available due to traffic signs or others, $A*$ algorithm re-run for generating a new path from current location to the destination point.

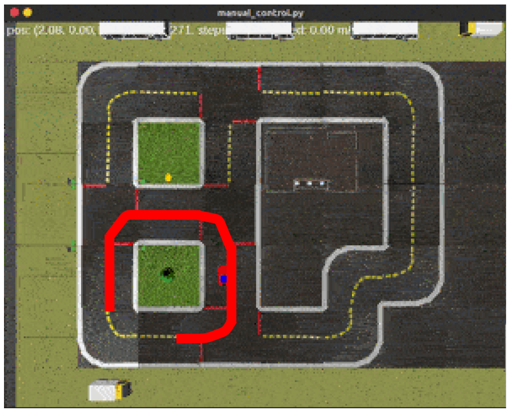

Fig. 3. Example of generated path

Also, we added the feature that the vehicle waits for 10 seconds when it arrives at passenger points and destination points to make the example more realistic. Finally, when the vehicle encountered a stop sign on the destination point it sets down the passenger.

B. Sign Detector Node

For self-driving agents, visual perception is one of the most critical aspects. Perception refers to the ability of an autonomous system to collect information and extract knowledge from the environment. In this project, we used image processing to perform lane and traffic sign detection. First, to detect lanes & traffic signs, we have tried to implement various algorithms including both traditional and novel methods.

Traditional methods are implemented using the *OpenCV* library. We started by preprocessing the given image input, then applied the *Hough Line* transformation. These operations successfully yielded lanes. For the traffic sign detection, we first trained an *SVM* classifier. we have created a dataset using online resources. We also added cropped sign images from the *DuckieTown* environment to the dataset.

Since *SVM* classifier had some speed & accuracy issues, we also tried another method that simply slides a window over the image. It first creates a prototype and creates a pyramid by downsampling the provided image. Detections are decided by calculating the mean squared error. Thanks to this fast method, we picked and dropped passengers from the appropriate location. To identify encountered signs and acts according to them we create a scale-invariant feature transform *SIFT* model which classifies the traffic signs. We collected sign data from the internet and downloaded *gym-duckietown* repository. After collecting data we made some prepossess to make images useful. In that case, we utilized *OpenCV* functions like edge, shape detection, etc.. Finally, we used SIFT classifier and choose the largest contour.

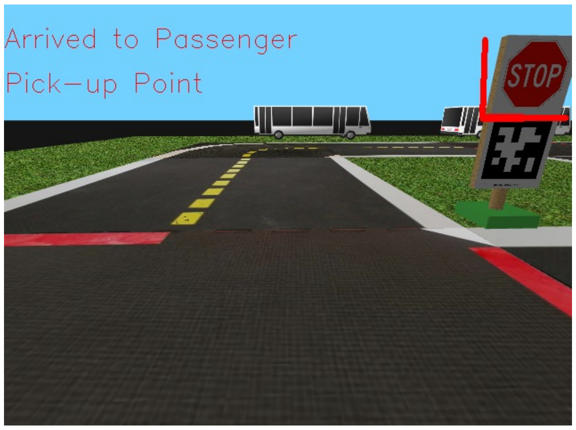

Fig. 4. Example of sign detector.

C. Lane Detector Node

PID, proportional-integral-derivative controller control loop method, is a feedback controller method widely used in industrial control systems [3]. In this project, the vehicle tries to go from the middle of the lines with the pd controller created using only P and D operations to provide a proper system control. Lane position input comes to the pd controller and provides calculation of next position. Thanks to this controller, the robot never goes out of the map, or grass parts and it successfully follows lanes.

D. States

Wait: Waiting for passenger

Go to the passenger: If a passenger exist, vehicle will go to the passenger and get him/her.

Drop the passenger off: When vehicle sees the stop sign, it will wait 10 seconds and drop off the passenger.

Get the passenger: When the vehicle goes to pick up the passenger, it waits 10 seconds.

E. Constraints

Vehicle moves according to traffic signs. Each sign and traffic sign is a constraints.

No turn left sign: If the vehicle sees this sign it won't turn left.

No turn right sign: If the vehicle sees this sign it won't turn right.

Stop sign: If vehicle sees this sign it will not move for ~10 seconds. This sign is the only place where vehicle can drop or pick up the passenger.

F. Actions

Turn right: Vehicle can turn right if there is no constraint.

Turn left: Vehicle can turn left if there is no constraint.

Go forward: Vehicle can go forward if there is no constraint.

Stop: Vehicle can stop when there is a stop signal. It can not stop anywhere else.

III. Work Assignment

Tasks:
0 - Sign detection with image processing
1 - Extracting the map
2 - Path determining with A star (A*)
3 - State diagram/table
4 - Controlling duckie with PID

In this project, as a group of 4, all team members took an active part in the study. There were several sub-topics in this project because the *gym-duckietown* environment which we selected was a complicated environment. There were several problems like image processing, autonomous driving & smart systems, controlling, etc. Nowadays, autonomous driving is one the most challenging and popular topics. During the project, we focused on these topics. The tasks & assignee can be seen in the table below:

TASKS	**ASSIGNED TO**
Image Processing	İhsan SOYDEMİR
Map Extraction and A*	Hasan Hüseyin KAÇMAZ
State diagram / table and general flow	Enes Furkan ÖRNEK
Controlling duckie with PID	Talha ÇOMAK

First of all Ihsan (Ihsan SOYDEMİR) worked on *Image Processing* and detection of lanes and traffic signs. He used the *OpenCV* library for this problem and also used traditional machine learning algorithms. Hüseyin (Hasan Hüseyin KAÇMAZ) worked on path planning and map extraction. He arranged given maps and edited traffic signs and others. He also worked on the *A** algorithm for generating the shortest paths dynamically. Enes (Enes Furkan ÖRNEK) worked on state diagrams, smart systems, and general flows of the algorithm. He also worked on the development environment. He analyzed image processing results and path way-points. Talha (Talha ÇOMAK) worked on controlling the duckie, aka Taxi. He used a *PD* controller for movements. He analysed and optimized the parameters of the *PD* values.

Everyone in our project group worked keenly and we successfully finished all of our tasks mentioned in the project proposal.

IV. Acknowledgment

The autonomous vehicle technology is getting more and more attract every day. Although with current technology it is not possible to make fully reliable vehicles which moves around the towns, we are getting closer with every passing day to days we made them true. In this case simulators like we did in this projects are takes important rolls to implement new approaches and experience them. We tried to simulate a real world scenario of autonomous taxi which takes passenger from station and get he/she to destination points. Also, during the travel we observed behaviour of the vehicle when it encountered with traffic signs.

References

[1] C. Boisvert et al., "Duckietown Environments for OpenAI Gym", (2018), GitHub repository, https://github.com/duckietown/gym-duckietown.

[2] F. Ducho˘n, A. Babinec, M. Kajan, P. Be˘no, M. Florek, T. Fico, and L. Juri˘sica,"Path planning with modified a star algorithm for a mobile robot", Procedia Engi-neering, vol. 96, pp. 59–69, 2014.

[3] Ni contributors, "Pid theory explained", 2020, [Online;accessed17-March-2020].[On-line].Available: https://www.ni.com/en-tr/innovations/white-papers/06/pid-theory-explained.html