Autonomous Vehicle Driving

Final Project – A.Y. 2021/22

Alessio Pepe	0622701463	a.pepe108@studenti.unisa.it
Grazia D'Amore	0622701462	g.damore15@studenti.unisa.it
Giuseppina Di Paolo	0622701510	g.dipaolo5@studenti.unisa.it
Teresa Tortorella	0622701507	t.tortorella3@studenti.unisa.it

1 Executive Summary

It is required to implement a behavioral planner which allows to manage collisions with other vehicles or pedestrians, and it manages the presence of traffic lights. The provided solution includes two RGB cameras, a depth camera, and a semantic segmentation camera. The RGB camera data are used to detect people and vehicles. We don't use tracking algorithm, but we use "perfect" orientation and speed data from CARLA. YOLO detector is used to detect people and vehicles, meanwhile depth camera and semantic segmentation are jointly used to compute the average depth of the detected point.

1.1 ODD

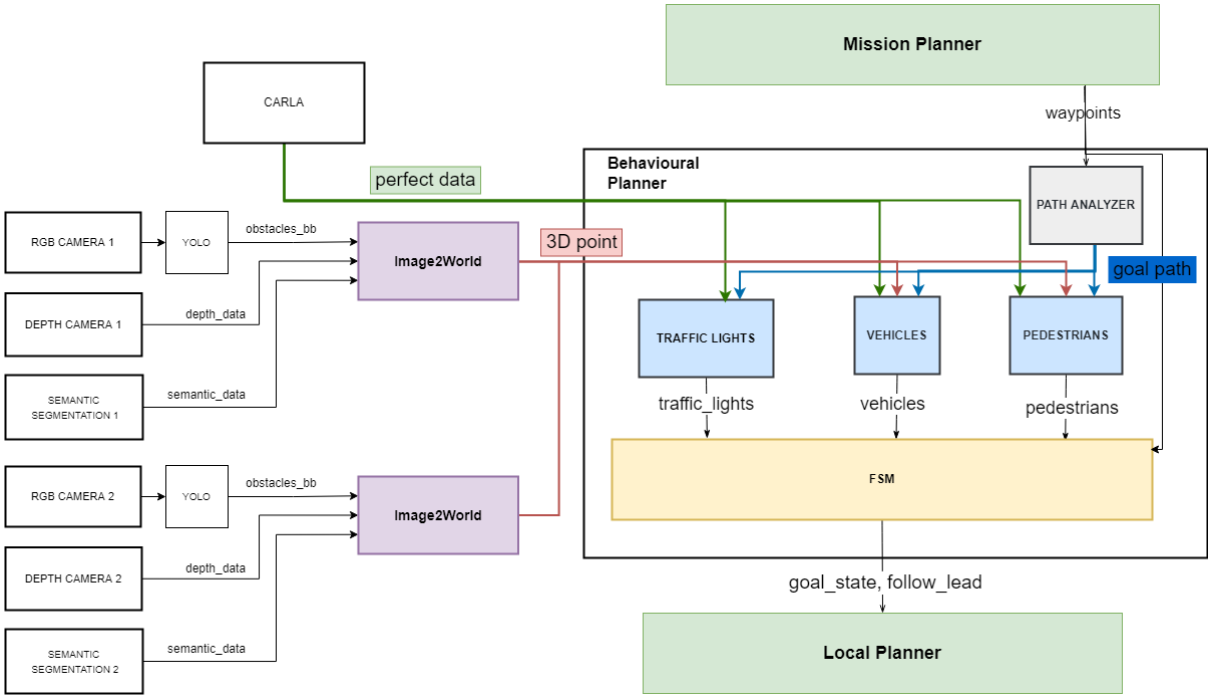
Our behavioral planner is positioned in the following ODD:

- Lateral control is provided, but it only takes care of following the right trajectory.
- The longitudinal control, which takes care of maintaining speed.
- OEDR: the vehicle is capable of handling dangerous situations, such as pedestrians crossing the road.
- The vehicle is not able to autonomously manage dangerous situations that require a response to the moral experiment. For example, the vehicle is unable to choose whether the lesser evil is crashing into the car in front of it or having several pedestrians injure themselves on it (pedestrian suicide in Carla).
- The vehicle is limited to a situation where the map waypoints are known. Furthermore, data on the position and status of traffic lights must also be known.
- The vehicle can detect people and other vehicles from the cameras but the information about speed and orientation are known.
- It is assumed that depth data and semantic segmentation are "perfect".
- It is assumed that in most cases the other agents comply with the rules of the road, for example that the vehicles all walk in the right direction and in the right lane, and that they do not crash into other vehicles (although this does not always occur in simulations).

2 Architecture

2.1 Behavioral Planner

The architecture of the behavioral planner is located between the Mission Planner and the Local Planner. You can see them in the following figure.



We can see that we receive the path waypoints from the Carla Mission Planner, and the perfect data from Carla as if we were on a smart road where traffic lights are known. The position of vehicles and pedestrians is not known, but we can assume that we know their speed and orientation without using a tracking algorithm.

- The module ***Path Analyzer*** creates the path considering the waypoints based on the lookahead, directly proportional to the speed.
- The module ***Traffic Lights*** take as input the calculated route and the Carla's data and check if traffic light is present in the route and its state.
- The modules ***Pedestrians*** and ***Vehicles*** takes as input the calculated route, the position of the vehicle computed from the camera's image, and "perfect" data for speed and orientation. These modules check if some of these objects are present in the route and if they are relevant for the specific scene. As regards vehicles, for example, only those present in the same lane as the car are considered lead vehicles. Meanwhile detected pedestrians are considered obstacles either they are on the road either they are on the sidewalk and will arrive on the road in front of the car in at most double the time it will take the car to get to that point.
- The waypoints and the information provided by these other modules are taken as input by the ***FSM***, which calculates the outputs to be provided to the local planner, which is the desired state and the eventual vehicle to follow by adapting the trajectory.

2.2 Local Planner

The module ***Local Planner*** has been modified to implement a Stanley controller tuned for handling the lateral control.

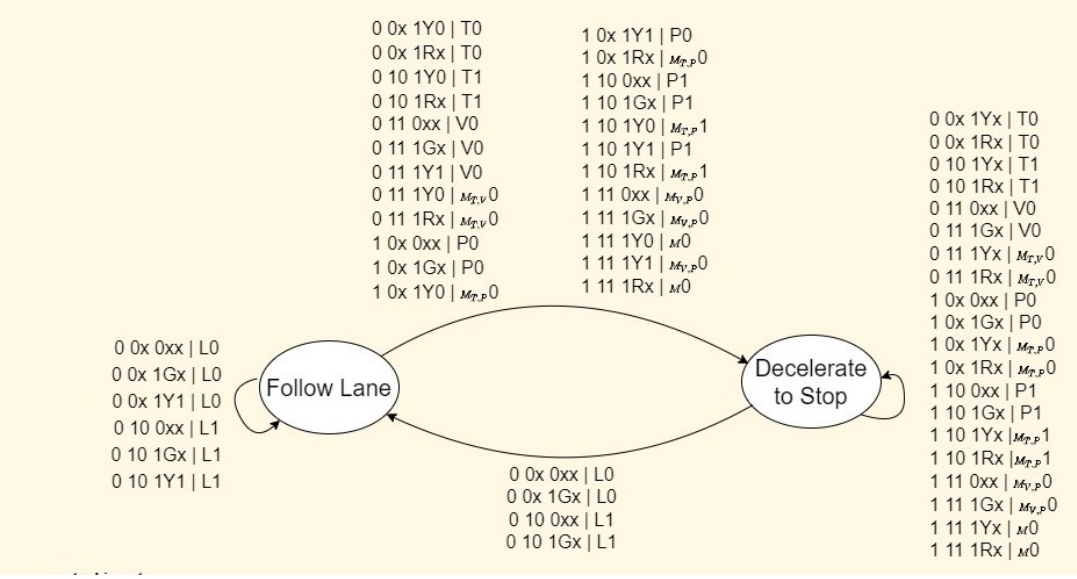
2.3 FSM

Two states are defined in the FSM, with the following meaning:

- **Follow Lane**: In this state the car follows the trajectory in the absence of events that involve having to stop in front of a pedestrian or a traffic light. As for the vehicles, being managed at a lower level (in the local planner) it is only indicated whether or not to follow the lead vehicle and the status of the latter.
- **Decelerate to Stop**: In this state, a pedestrian has detected who is or will be on the road for which it is necessary to stop, or a traffic light that has been red or yellow (but without the car having yet engaged the intersection), or a car is stopped in front of us.

The transitions cover all combinations of the following possibilities, giving priority to the closest obstacle:

- **Traffic light**: The favorable occasions to pass the traffic light are those in which it is green or yellow but only if the vehicle has already engaged the intersection. The unfavorable occasions are all the others, which is with a yellow traffic light and without having occupied the intersection or with a red light.
- **Vehicle**: If there is a vehicle in the path of the car, it must adapt its speed to maintain the gap. If the vehicle is stopped we need to stop the car.
- **Pedestrian**: If a pedestrian is in the middle of the road in the lane of the car or if it is expected that it will arrive on the trajectory in at most twice the time it will take the car to arrive at the same point, the car must stop before the pedestrian and until this will not leave the lane.



control input

$$\text{pedestrian}_{\text{found}} | \text{lead\_vehicle}_{\text{found}} | \text{lead\_vehicle}_{\text{speed}} <= \text{speed}_{TH} | \text{trafficlight}_{\text{found}} | \text{trafficlight}_{\text{state}} | \frac{\text{distance}_{\text{to trafficlight}}}{\text{car}_{\text{velocity}}} < \text{time}_{\text{yellow}}$$

Is 1 if the car can pass the trafficlights before it turns red. This because it's unsafe stop brake too late.

control output

$$\text{goal}_{\text{index}} | \text{lead vehicle}$$

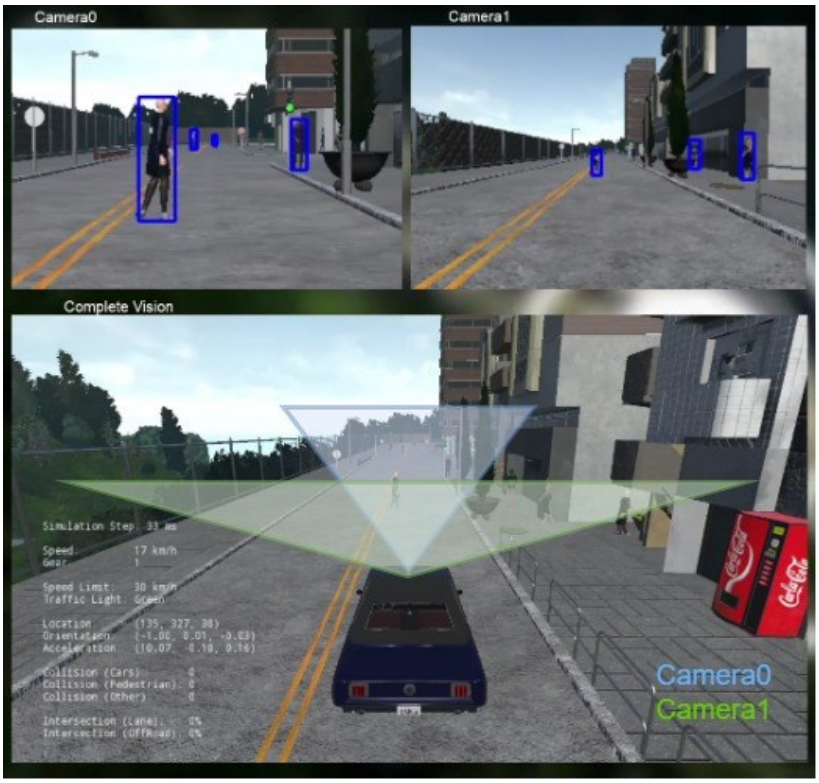
P: pedestrian
V: vehicle
T: trafficlight
L: lookahead

$M_{T,P} = \min(T, P)$
$M_{T,V} = \min(T, V)$
$M_{V,P} = \min(V, P)$
$M = \min(T, V, P)$

Final project note: We did not edit the FSM to manage the new situation who comes from the real data, for example the ghost situation.

2.4 Camera Data

To detect the vehicle and the pedestrians in front of us, we decided to use two different cameras, placed as you can see in the following image.

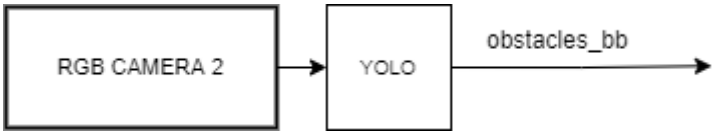


We need the *Camera0* to see the object far from the vehicle. Otherwise, the dimension of that in pixel is not enough to be detected by the net.

The *Camera1* is needed instead to detect closest object and people close on the Sidewalk.

2.4.1 Detection

This dimensionality problem comes by the camera sensor dimension. In fact, the two cameras have a resolution of 300px by 200px. We need this low resolution to guarantee the detector a frame rate faster or equal to 30fps for both the cameras.



The detector we choose and used is Yolo v5 small. This because multiple reason:

- Considering the benchmark and pretending to have a decent hardware, we can guarantee the computation in less than 1/30s
- The performance of this detector is good for the problem we are resolving.

- Compared to other pretrained net, the implementation is easy.

We also considered this benchmark, but we used a lower resolution for the images to make our simulation faster, reducing the performances.

Model	size (pixels)	mAPval 0.5:0.95	mAPval 0.5	Speed V100 b32 (ms)	params (M)	FLOPs @640 (B)
YOLOv5n	640	28.0	45.7	0.6	1.9	4.5
YOLOv5s	640	37.4	56.8	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	4.8	86.7	205.7

Figure 1 - Source: https://github.com/ultralytics/yolov5

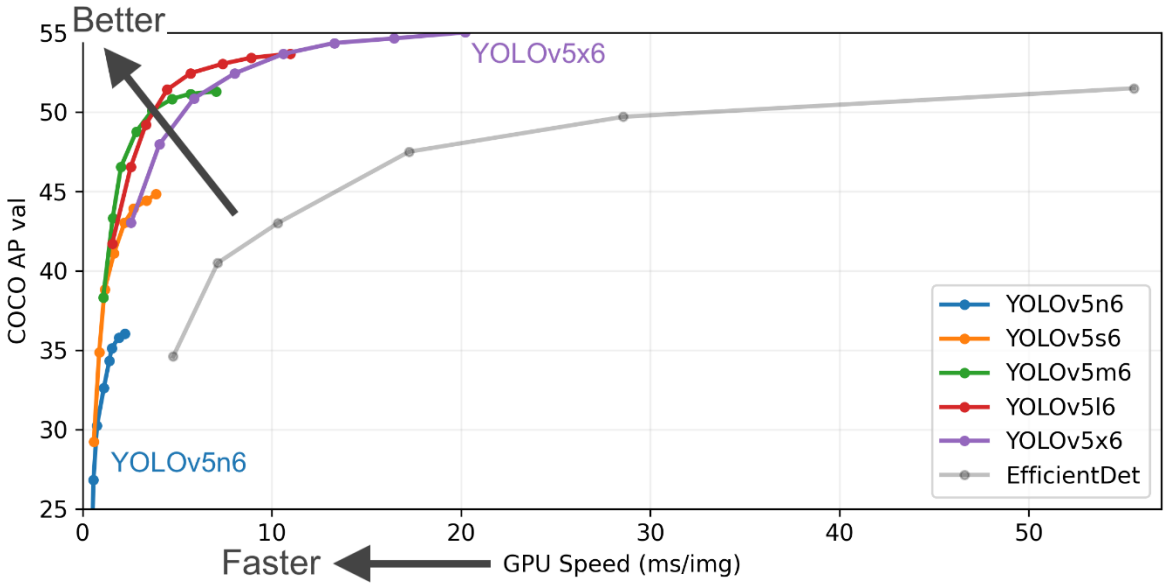


Figure 2 - Source: https://github.com/ultralytics/yolov5

2.4.1.1 Implementation specifications

As we can see, the dimensions of the camera sensors can be bigger than that we selected, respecting the frame rate specification.

However, this dimension is needed also by us, because the simulation on our hardware is much slower than the times reported in the paper.

We also use a low-quality world for the simulation. This quality will affect the performance of all system. Unfortunately, the simulation with an increased quality is too slower to be executed in the times.

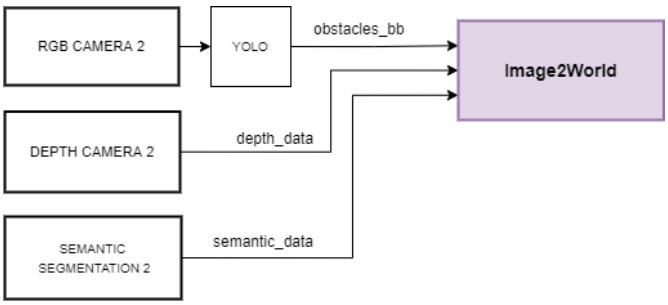
For this reason, in the test provided by us, some error that happen can be avoided because they depend on detection errors, who will not happen with a better quality of the world.

2.4.2 Map detected object in world

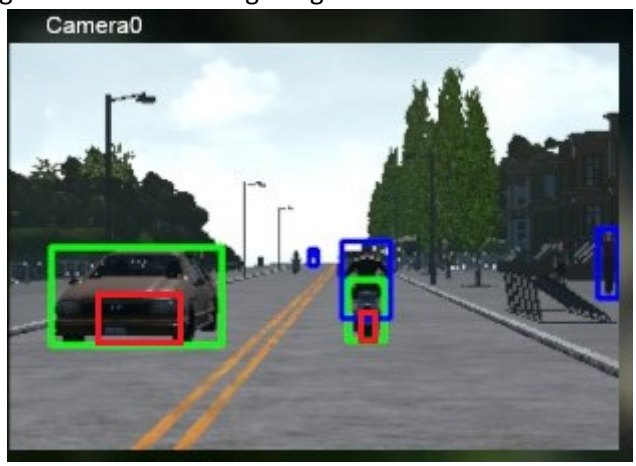
For each camera, we used a perfect Semantic Segmentation and a Perfect depth camera. In a real situation, at least the semantic segmentation needs to be computed with some algorithms. For that reason, the performance will surely decrease.

To map the object from the image to the real world, we made the following module. The module ***Image2World*** projects the detected point from camera to 3D world starting from image bounding boxes and depth of the object. The computed bounding box and its relative depth are used to project the point in the vehicle frame through the extrinsic and intrinsic matrices.

The information to give at this module is computed as follows:



1. YOLOv5 is applied on the RGB camera image to detect the objects and get the bounding boxes of the image
2. The depth is calculated as the depth mean in the point who semantic segmentation contains the chosen object. In order to avoid a mismatching with different objects of the same category (i.e. vehicles or pedestrian very close to each other) we have considered only the low-central part of the bounding box as the following image show



Green	YOLO vehicle bounding box
Blue	YOLO pedestrian bounding box
Red	cut bounding box

3. To avoid too much false positive detection, if in the boxes are not present any pixel with the required class, we do not consider that object.

To improve this process, we can surely use a Detection and Mask net.

3 Analysis

3.1 Quantitative Analysis

This section shows the tests carried out and their descriptions. In the repository, however, you can also see the execution of these. In the following test cases, pedestrians on the sidewalk keeping a direction that doesn't cross our vehicle direction are not considered as cases of interest even though they are managed avoiding useless brake.

3.1.1 Test 1

Code	Traffic Lights	pedestrians	vehicles	situation passed
116-140-450-1-1-1	1 yellow	4	0	5/5

In this test case we have:

- A pedestrian crossing the way, so the car stops before the pedestrian and restart when he's passed.
- A yellow traffic light, so the car decelerates to stop before the traffic light that turns red.
- A pedestrian on the lane, so the car decelerates while the pedestrian leaves the street.
- Two pedestrians that cross the street, so the car stops until they leave the street.

3.1.2 Test 2

Code	Traffic Lights	pedestrians	vehicles	situation passed
116-140-300-150-1-1	1 red	3	1 motorcycle	4/5

In this test case we have:

- A motorcycle in front of our vehicle, so the car follows it
  The motorcycle stops at a red traffic light, so the car stops behind it until it restarts at the green light
- The motorcycle stops to let a pedestrian cross, so the car stops behind it until it restarts
- The motorcycle stops behind two cars stopped at a red traffic light, so the car stops
- The motorcycle restarts but the car stay stopped to let two pedestrians cross
- A pedestrian suicide on the car
- A detection error is verified, so the car doesn't see the motorcycle and collides with it

3.1.3 Test 3

Code	Traffic Lights	pedestrians	vehicles	situation passed
89-119-150-150-3-1	1 yellow	1	1 motorcycle	3/3

In this test case we have:

- A motorcycle in front of our vehicle, so the car follows it
- The motorcycle stops behind some cars stopped at a red traffic light, so the car stops
- A pedestrian crosses the street while we are already stopped
- The motorcycle restarts when the traffic light turns green, so the car follows it
- The traffic light turns yellow, so the car decelerates to stop

3.1.4 Test 4

Code	Traffic Lights	pedestrians	vehicles	situation passed
116-140-1-450-1-1	1 yellow	0	1 car	2/2

In this test case we have:

- A yellow traffic light, so the car decelerates to stop until it turns green
- A car is stopped at a red traffic light behind other cars, so our vehicle stops until they restart

3.1.5 Test 5

Code	Traffic Lights	pedestrians	vehicles	situation passed
130-12-100-100-0-0	2 red	1	1 car	4/4

In this test case we have:

- A car in front of our vehicle, so we follow it
- A pedestrian crosses the street, so we stop to let him pass but he suicides on our vehicle
- Our vehicle restarts and then it stops behind a car stopped at a red traffic light until it turns green
- A red traffic light is present, so the vehicle stops

3.1.6 Test 6

Code	Traffic Lights	pedestrians	vehicles	situation passed
123-46-150-150-3-1	4 red	0	1 bike 1 car	6/6

In this test case we have:

- A bike in front of our vehicle, so we follow it
- The bike stops at 2 red traffic light, so we stop behind it
- A car in front of our vehicle, so we follow it
- The car stops at two red traffic light, so we stop behind it

3.1.7 Test 7

Code	Traffic Lights	pedestrians	vehicles	situation passed
26-15-1-1-1-1	1 green 1 yellow	0	0	2/2

In this test case we have:

- A green traffic light is present, so the car continues on its path
- A traffic light turns yellow when the car has already joined the intersection, so the car continues on its path

3.1.8 Test 8

Code	Traffic Lights	pedestrians	vehicles	situation passed
125-28-1-1-1-1	1 red	0	0	1/1

In this test case we have:

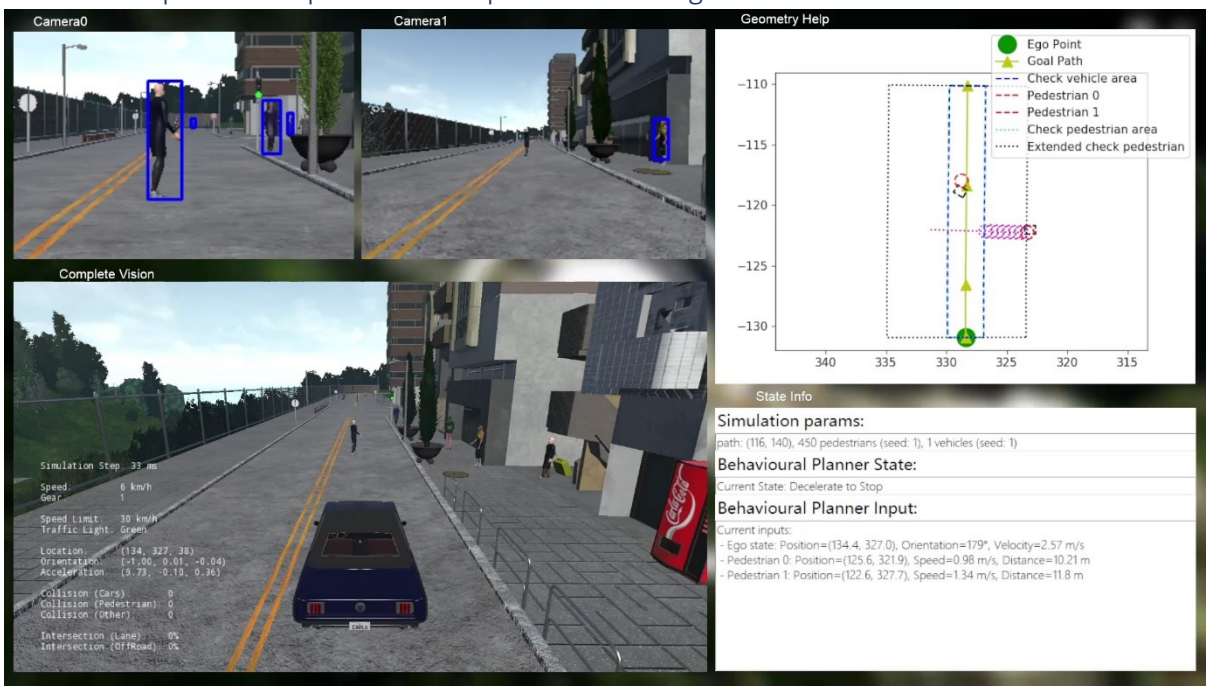
- A red traffic light, so the car stops until it turns green and the restarts

3.2 Qualitative analysis

3.2.1 Yellow traffic light

We can see in the information in the bottom left text the traffic light is yellow. But the state is “Follow Lane” because the car has already engaged the intersection.

3.2.2 Real pedestrian position and “perfect” tracking



In the left plot we can see the real position of the pedestrian in black and in red the predicted bounding box. The pedestrian 0 is the closest pedestrian, the one who is detected on the sidewalk by the

camera1 and predicted to be in the middle of the road with the “perfect data”. The pedestrian 1 is the one detected by the camera1, already in the middle of the road.

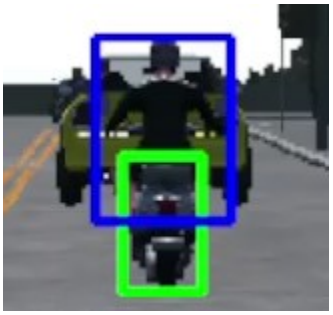
3.2.3 Real vehicles position and “perfect” tracking

As we can see in the image, the position obtained from the image of the car is plotted in red, the real position is plotted in black. The information about speed and orientation, used for the follow leader state, are obtained from the perfect data.

3.2.4 People and Vehicle detection issues

3.2.4.1 Riders and motorcycle

YOLO detects people on motorcycle. We don’t want to consider the rider as a pedestrian to avoid useless stop, so we filter YOLO detection with semantic segmentation data to consider the rider and the motorcycle as one vehicle. In fact, CARLA semantic segmentation data classifies the motorcycle as vehicle and the rider as Nothing.



3.2.4.2 False positive detection

In some cases, YOLO detects vehicle or pedestrian where they are not present. So, we filter YOLO bounding boxes with “perfect” semantic segmentation data not considering the detected object if there is no matching for its category (vehicle or pedestrian) in semantic segmentation.

3.2.4.3 False negative detection

In some cases, YOLO can’t detect people or vehicle (i.e. occlusions, pedestrian in front of a car, etc.). This type of error is due to the choice of small version of YOLO that is a right tradeoff between accuracy and performances.

3.2.4.4 Occlusions

In some cases, YOLO can’t detect people or vehicle because of occlusions (i.e. people behind street lamp/tree or other street objects, vehicle behind pedestrians or other street objects). The cases in which are involved both pedestrians and vehicles are managed because the implemented FSM considers only the closer object (so the occluded object is not considered). The other cases can be a problem if the occlusion is persistent for several frames.

3.2.4.5 Ghost frame

The provided solution doesn’t use a tracking algorithm, so in some cases people or vehicle detection is missed. This is not a problem if detection is missed for few frames but it causes instability on our FSM that changes state 2 times (to another state and back) making useless brake and throttle.

3.3 Comparison with just “perfect” data

As we can see from the previous test, the global performance is decreased in terms of useless brake and acceleration. This because the false negative detection, that we have not managed because we do not have implemented a tracking algorithm.

In our test no critical issues was happened, but the missing detection of a pedestrian or vehicle for many frames can be create a big issue. For this reason, a future improvement on this work can be the use of a lidar or a radar to manage this detection error.

3.4 Video Demo

You can find the video in the following 3 link.

- Traffic lights: https://github.com/pepealessio/CARLA-behavioural-planner/blob/with-detection/doc/video/completo_semafori-converted.mp4
- Vehicles: https://github.com/pepealessio/CARLA-behavioural-planner/blob/with-detection/doc/video/completo_veicoli-converted.mp4
- Pedestrian: https://github.com/pepealessio/CARLA-behavioural-planner/blob/with-detection/doc/video/completo_pedoni-converted.mp4

You can also find the execution of all tests at these reports:

- Perfect data: https://github.com/pepealessio/CARLA-behavioural-planner/blob/perfect-data/README.md#scenarios-of-interest
- With detection: https://github.com/pepealessio/CARLA-behavioural-planner/tree/with-detection#scenarios-of-interest

4 Conclusions

The provided solution implemented a behavioral planner that manage collisions with other vehicles or pedestrians and the presence of traffic lights. The first solution used “perfect” data from CARLA simulator, meanwhile this solution employs two RGB cameras and a YOLO detector to detect people and vehicle. Both the solutions have been tested, the results of the first solution are reported in the delivered midterm report. The ultimate solution that uses jointly real and perfect data is described in this report, most of the reported tests have been completed successfully.