Global and Local Motion Planning for Self-driving Car using Carla Simulator

Siyi Dai, Batu Kaan Ozen, Yanni Zhang, Yiming Wei
TAS Final Project
Supervisor: Regine Hartwig

Chair of Automatic Control Engineering
Technical University of Munich

Contents

Overview - Motivation & Project Description

Autonomous self-driving car application

- Computer Vision
- Global Planner
- Local Planner

Carla Simulator with Coursera instruction. [*Coursera Motion Planning for Self-Driving Cars* n.d.]

Demonstration for our Project

Overview - Task Distribution

Task 1: Object Detection using semantic segmentation and depth map for static object (Batu Kaan Ozen)

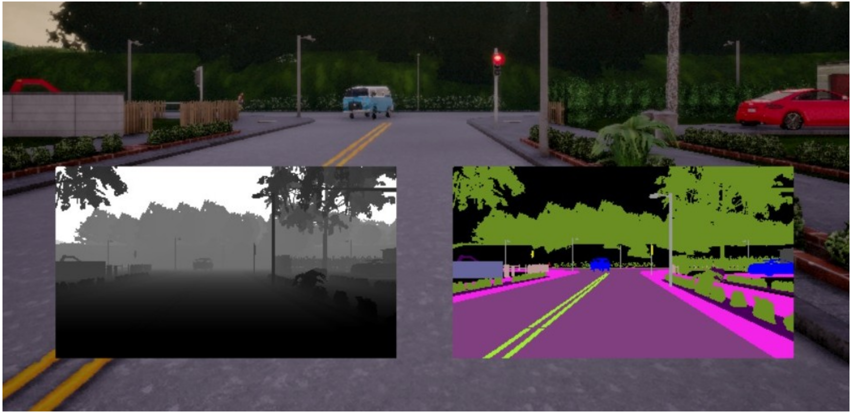
Task 2: Global Planner using K-Nearest-Neighbour and A* algorithm (Yanni Zhang)

Task 3: Local Planner include velocity planner, behavioural planner, path optimizer, and collision checker (Siyi Dai)

Task 4: Controller with Stanley Method as lateral controller and PID Controller as longitudinal controller (Yiming Wei)

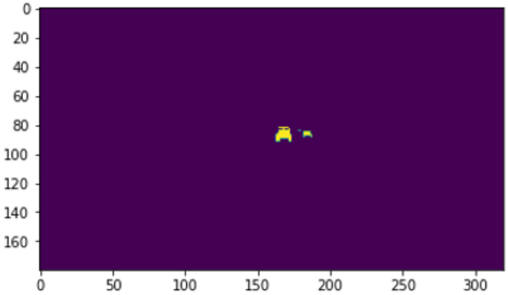
Object Detection

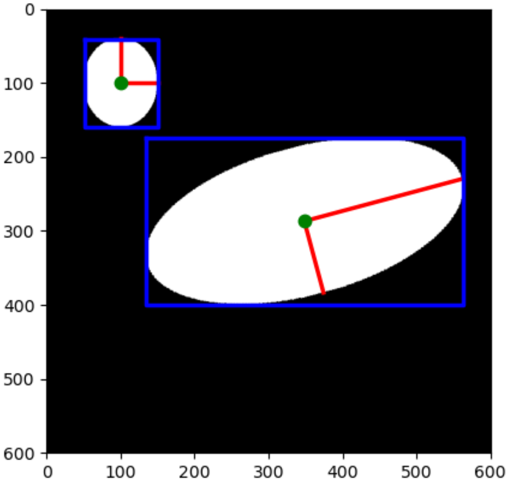
Carla gives us depth map and semantic segmented image.



Object Detection

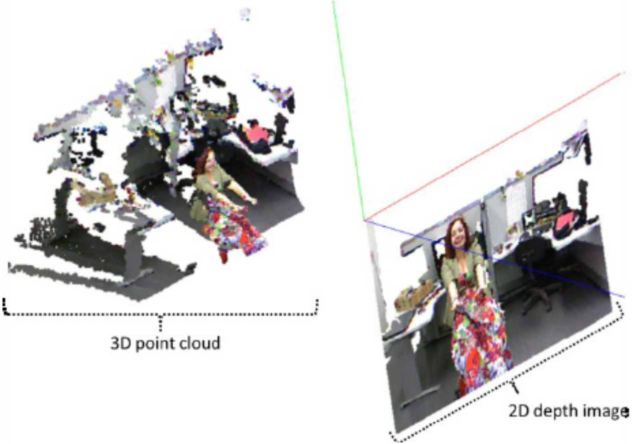
Apply threshold to detect car and use region props algorithm to find center of object.





Object Detection

Convert center position of car using inverse intrinsic matrix to 3d point. It gives you location of obstacle.

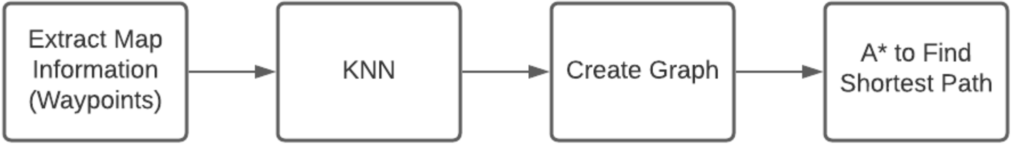


Object Detection

Pixel coordinate to 3D coordinate conversation using depth and focal length information.

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = z \underbrace{\begin{bmatrix} 1/f_x & 0 & 0 & 0 \\ 0 & 1/f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{inverse\ with\ c_x, c_y, S=0} \begin{bmatrix} u \\ v \\ 1 \\ 1/z \end{bmatrix}$$

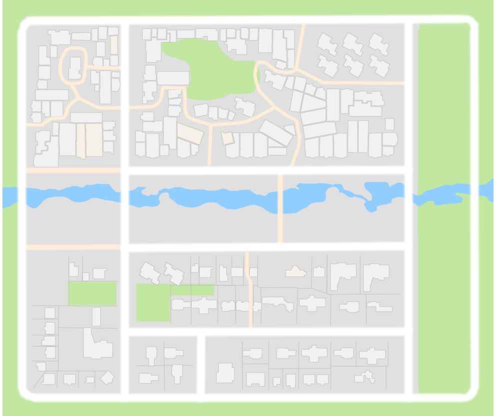
Global Planner - Flow Chart

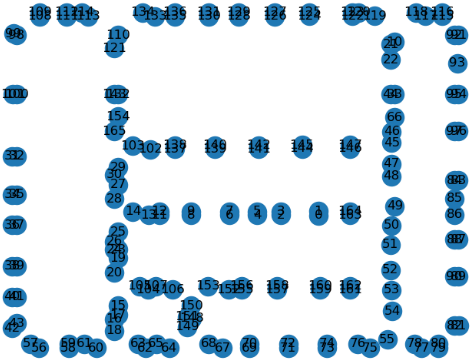


Graph: $G = \text{networkx.Graph()} \rightarrow \text{G.add\_node()} \rightarrow \text{G.add\_edge()}$

[*KNN sklearn - Nearest Neighbors* n.d.]

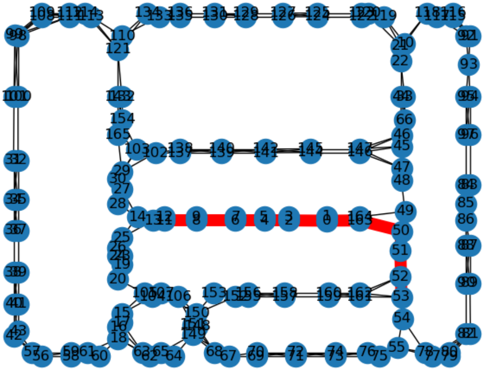
Global Planner - Map & Waypoints





Map $\leftrightarrow$ Nodes

Global Planner

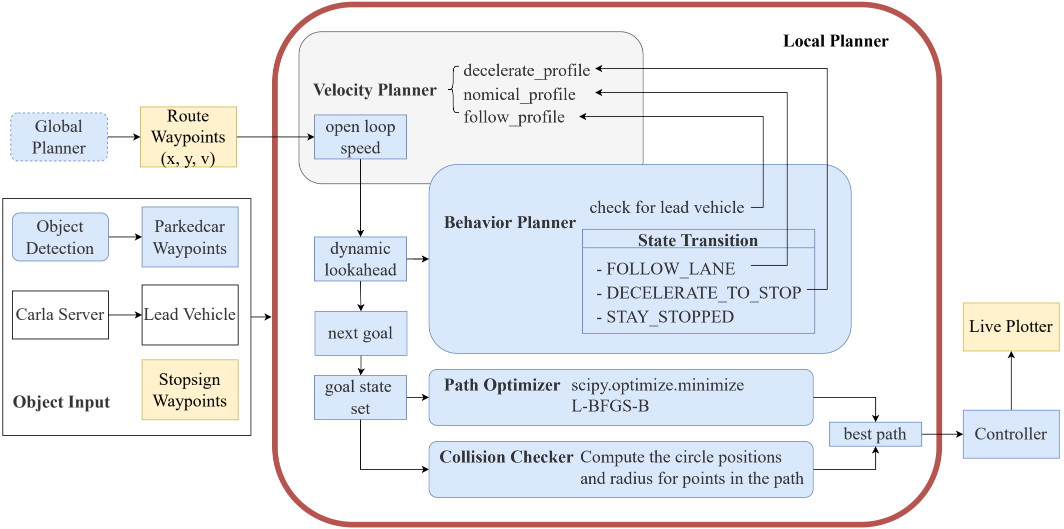


Graph Created Based on Map Information and KNN & A* Path (Red)

Starting Node 13 $\rightarrow$ Goal Node 53

$K = 5$ (Connected to 4 Points Except Itself)

Local Planner



Velocity Planner & Behavioural Planner

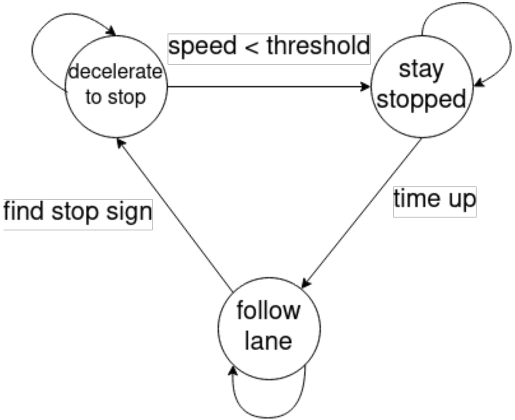
Behavioural Planner:

- Follow lead vehicle
- State transition
  - Follow lane
  - Decelerate to stop
  - Stay stopped (count 10)

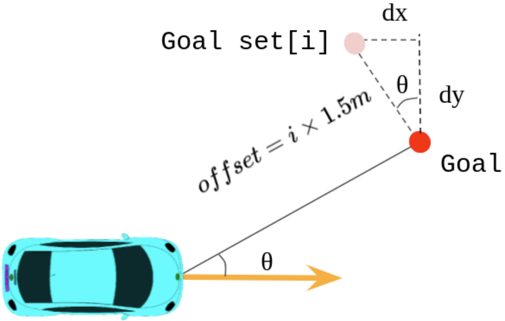
↓

Velocity Planner:

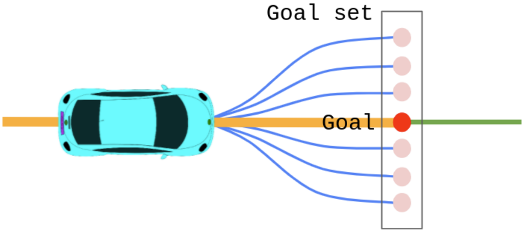
- Follow profile
- Nominal profile
- Decelerate profile



Goal set & Path Optimizer



(a) Compute goal set based on the goal state



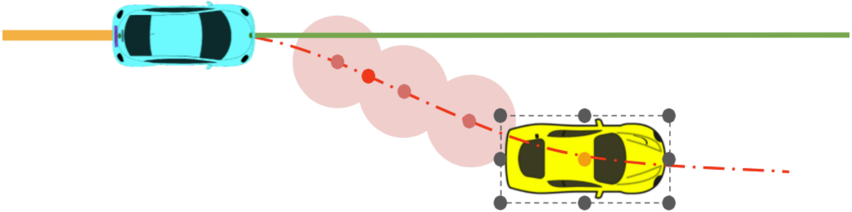
(b) Optimize the planned paths based on the goal set

`scipy.optimize.minimize(method='L-BFGS-B')` ⇒ Spiral parameters

`scipy.integrate.cumtrapz` ⇒ Path waypoints

[*Scipy Minimize-L-BFGS-B Documentation* n.d.] [*Scipy Integrate-cumtrapz Documentation* n.d.]

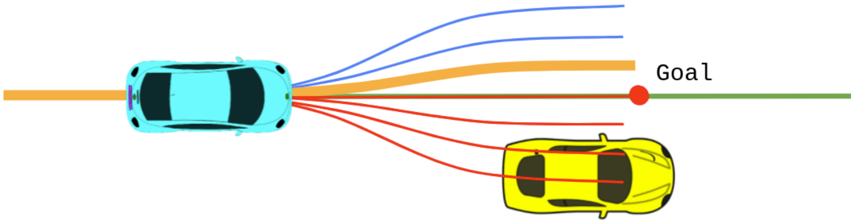
Collision Checker



Circle method indicating an approximate collision border for the vehicle

arc length distance for each circle = `[-1, 1, 3]`
circle radius = `1.5 m`

Best Path Selection



Select the best path based on selection score

Score = "distance from centerline" score + "proximity to other colliding paths" score

Best path ⇒ Path with minimum score

Summary

Achievements:

- Object detection for parked car waypoints calculation
- Global Planner for Carla default maps
- Local Planner for driving, stop, and obstacle avoidance
- Controller with PID Controller and Stanley Method

Challenges:

- Access to the Coursera map
- Learning using Carla from zero
- Graph processing power restriction

Further Research:

- Apply object detection to dynamic objects
- Apply global planner to coursera map

References

Controller designing. https://skill-lync.com/student-projects/designing-a-controller-for-controlling-lateral-and-longitudinal-movement-of-self-driving-car-using-python-and-test-it-by-using-carla-simulator. Accessed: 2022-02-08.

Coursera Motion Planning for Self-Driving Cars. https://www.coursera.org/learn/motion-planning-self-driving-cars/home/welcome. Accessed: 2022-02-08.

KNN sklearn - Nearest Neighbors. https://scikit-learn.org/stable/modules/neighbors.html. Accessed: 2022-02-08.

Scipy Integrate-cumtrapz Documentation. https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.integrate.cumtrapz.html. Accessed: 2022-02-08.

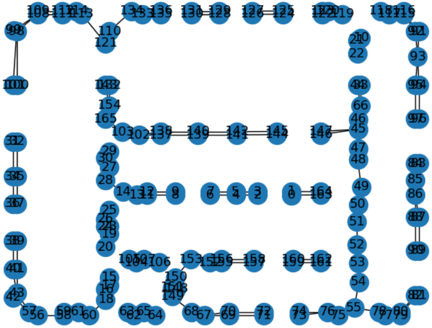
Scipy Minimize-L-BFGS-B Documentation. https://docs.scipy.org/doc/scipy-1.8.0/html-scipyorg/reference/optimize.minimize-lbfgsb.html. Accessed: 2022-02-08.

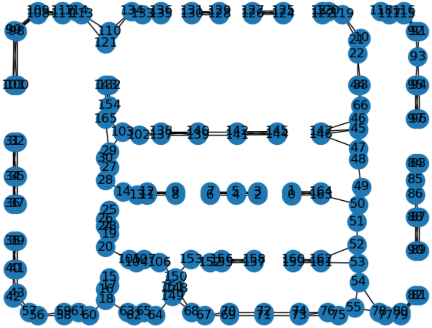
Appendix - Global Planner - Code for Map Extraction

```
# We load the default settings to the client.
scene = client.load_settings(CarlaSettings())
try:
    image = mpimg.imread('carla/planner/%s.png' % scene.map_name)
    carla_map = CarlaMap(scene.map_name, 0.1653, 50)
```

CarlaMap is a class defined by Carla which can extract map information

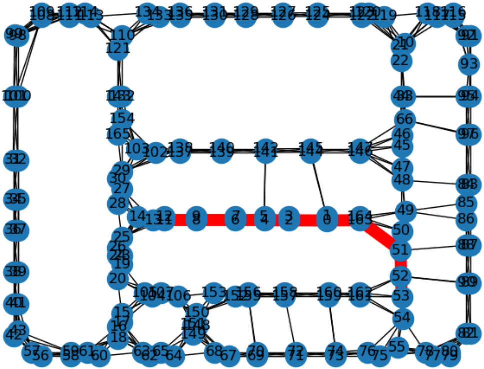
Appendix - Global Planner - K Too Small





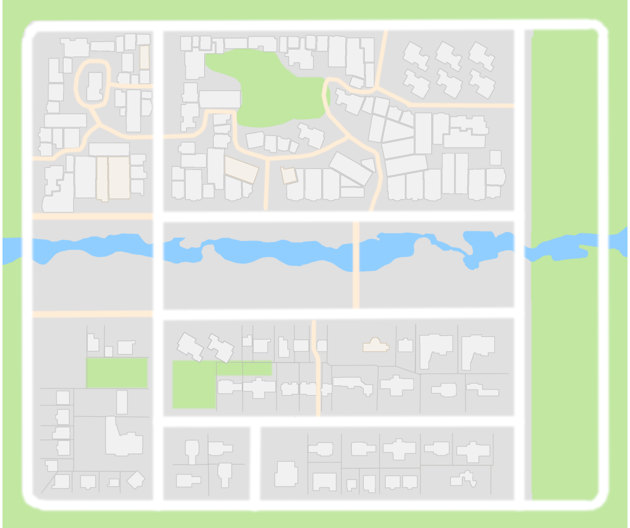
Graph Created Based on Map Information and KNN & A* Path (Red)
K = 3 (Connected to 2 Points Except Itself) & K = 4 (Connected to 3 Points Except Itself)

Appendix - Global Planner - K Too Large



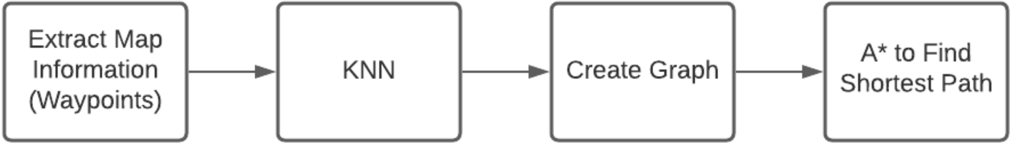
Graph Created Based on Map Information and KNN & A* Path (Red)
K = 7 (Connected to 6 Points Except Itself)

Appendix - Global Planner - Map



Map (Provided by Carla)

Appendix - Global Planner - Python Package



KNN: NearestNeighbors from sklearn.neighbors
A*: astar_path from networkx
Graph: $G = \text{networkx.Graph()} \rightarrow \text{G.add\_node()} \rightarrow \text{G.add\_edge()}$

Appendix - Controller

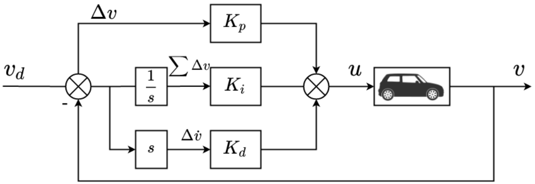
Longitudinal Controller: Speed Control $\rightarrow$ throttle

PID Controller:

- Error of current speed and desired speed
- Integral of error
- Changing rate of error

$$u = K_p \Delta v + K_i \sum \Delta v + K_d \Delta \dot{v}$$

with $\Delta v = v_d - v$



Appendix - Controller

Lateral Controller: Orientation Control → steer

Stanley Method:

- Heading error
- Cross-track error

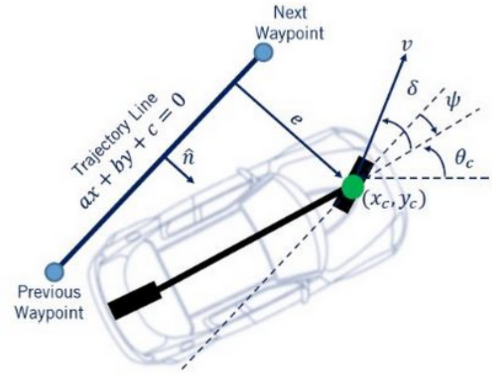
Cross track steering: $\arctan(\frac{k_p e}{v})$

Heading error:

$$\psi = \arctan(\frac{-a}{b}) - \theta_c$$

Steering input:

$$\delta(t) = \psi(t) + \arctan(\frac{k_p e(t)}{k_s + v(t)})$$



`Designing a Controller for controlling lateral and longitudinal movement` [*Controller designing* n.d.]

Appendix - Perception - Focal length calculation

$$f = \frac{(H \times \mathrm{WD})}{\mathrm{FOV}}$$

Appendix - Perception- Focal length