# homework3

April 2025

Hi, everyone. This is the third homework of our computer vision class. This assignment includes two parts: short answer and programming problems. Here are the **requirements**:

- You are required to use Python for all the programming tasks.
- We recommend you typeset your report using LATEX and submit the PDF. You are asked to submit a single zip file on WebLearning. The zip file should be consisted of the report and your code.
- Referring to the public code on the internet is allowed, but copying is absolutely forbidden.
- Please finish the programming and report independently.
- We provide the code framework for you (see in src.zip attached). Zip all your files and name it as: name_id_hw_3.zip,
  eg: zhangsan_2020123456_hw_3.zip, and the structure of directory should be like this:

```
zhangsan_2020123456_hw_3
├── report.pdf:  your report file
└── src:  put your code
    ├── README.md:  more details for code implementations
    ├── problem2_analyzing:  solution of Road Analyzing problem
    └── etc...
```

## 1 3D Location Transformation

Attached is an image car.png recorded with a camera mounted on a car. The focal length of the camera is 721.5, and the principal point is (609.6, 172.9). We know that the camera was attached to the car at a distance of 1.7 meters above ground.

a) Write the internal camera parameter matrix K.

b) Write the equation of the ground plane in camera's coordinate system. You can assume that the camera's image plane is orthogonal to the ground.

Figure 1: Demo Car

c) How would you compute the 3D location of a 2D point (x,y) in the image by assuming that the point lies on the ground? You can assume that the camera's image plane is orthogonal to the ground. No need to write code, write a mathematical explanation.

## 2 Road Analyzing

In this exercise you are given stereo pairs of images from the autonomous driving dataset KITTI. The images have been recorded with a car driving on the road. Your goal is to create a simple system that analyzes the road ahead of the driving car. Include your code to your solution document.

There are two parallel cameras on the car and the images are saved in the folder data/left and data/right. And for each image, there is a file with extension *left_disparity.png* in the data/detections that contains the estimated disparity. The camera calibrations (focal length, principle point and baseline) are saved in data/calib. Note that the coordinate origin in image plane is the top left pixel, so the coordinate of principle point is relative to it.

Under the folder data/detections, there are also the detection results for each left image. The results are saved in Matlab (.MAT) format. Use the SCIPY.IO.LOADMAT function to load this data. each value in the key *dets* represents a rectangle (called bounding box) around what the detector thinks it's an object. The bounding box is represented with two corner points, the top left corner $(x_{left}, y_{top})$ and the bottom right corner $(x_{right}, y_{right})$. The value of the key *dets* has the following information: $[x_{left}, y_{top}, x_{right}, y_{bottom}, id, score]$. Here SCORE is the confidence of the detection, i.e., it reflects how much a detector believes there is an object in that location. The higher the better. The variable ID reflects the viewpoint of the detection. You can ignore the SCORE and ID for this assignment.

a) For each image compute depth and visualize it.

b) Visualize the bounding boxes in the images. Then, for each bounding box, calculate the 3D coordinate of the diagonal intersection. For convenience, use the camera as the origin of the 3D world coordinate system.

# 3 Self-driving Detection (Bonus)

This is a bonus problem. You will get 1 extra point in your final grade if you successfully solve this problem.

In this exercise, you are given three images (data/image_*.png) from an autonomous driving simulator and their measurements (data/measurements_*.json) that record the spatial information of all the objects at those moments. For example, there are many different measurements in the key *playerMeasurements*: the *transform* represents the location coordinates and rotation pose in the world coordinate system; the *boundingBox* is the bounding boxes of the vehicle. And *nonPlayerAgents* are many other objects in the simulator, such as other vehicle, pedestrians, traffic lights and etc. Note that, the transform information of bounding boxes are relative to the agents and that of agents are relative to the world. For more details, please read the https://carla.readthedocs.io/en/0.8.4/measurements/#player-measurements if necessary.

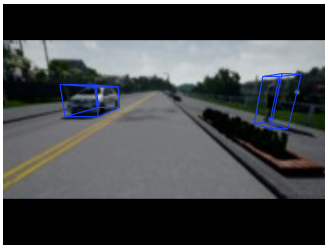

Figure 2: The example of drawing the bounding boxes on the image.

You are asked to draw the bounding boxes of the non-player-agents in the corresponding image, like Figure 2. The image shape is (800,600) and the principle point is just the centre point of the image. The FOV of the camera is 100. You only need to draw the bounding boxes for two types of agents: vehicle and pedestrians within 50 metres. Note that, take care of the names and directions of the axes in Figure 3 because the x-axis is the direction in which the car is moving.

If you need some help, please refer to the code in https://github.com/carla-simulator/carla/blob/master/PythonAPI/examples/client_bounding_boxes.py.

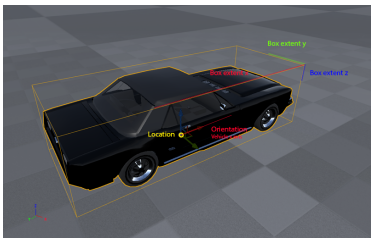


Figure 3: The schematic of the coordinate system of Carla.