CVL, ETH Zurich — Deep Learning for Autonomous Driving

Project 1: Understanding Multimodal Driving Data

28/02/20 - 20/03/20

Overview: Successfully completing all projects is compulsory for attending the exam. The projects can be completed in groups of 2. This project will affect your course grade by 10%. For further questions contact: Ozan Unal (ouenal@ethz.ch) or Dengxin Dai (dai@vision.ee.ethz.ch).

Submission: For each problem statement, the report should contain a concise description of your solution. You should also include the resulting figure and indicate the code files for the task. Submissions must be made in a ZIP file to Ozan Unal (ouenal@ethz.ch) by 20 March 2020. The ZIP file should include a report (PDF) for your answers and your Python code. The code will be checked.

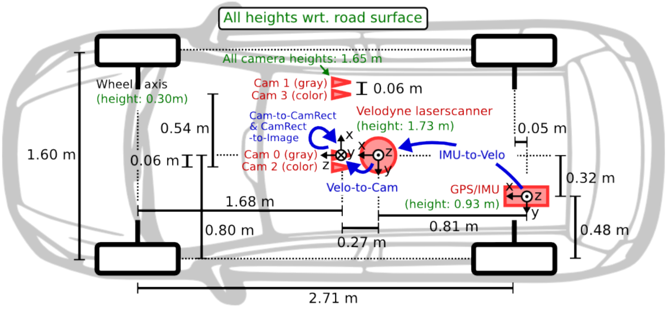


Figure 1: Camera setup.

Materials: In the gitlab repository[1] available to you are two autonomous driving scenes. To save you time, the scenes are given as a python dictionary and can be accessed via the provided **load_data.py**. Make sure to install **pickle**.

```python
from load_data import load_data

data_path = os.path.join('your/data/dir','data_name.p')
data = load_data(data_path)
```

The data dictionaries provided contain the following keys:

- **velodyne** is the point cloud of the scene. The LIDAR scanner used is the Velodyne HDL-64E that spins at 10 frames per second, capturing approximately 100k points per frame. This sensor has 64 channels. More information about the sensor can be found in the associated data sheet.

  The point cloud is given as a (**num_points** x **4**) numpy.array object. The first three dimensions of the array contain the x, y and z coordinates stored in metric (m) using the velodyne coordinate system. The fourth dimension is the reflectance intensity value which is between 0 and 1.

- **image_2** is the RGB image received from left RGB camera (Cam 2 in Fig.1). A capture by the camera is triggered when the velodyne is looking exactly forward.

[1] https://gitlab.ethz.ch/ouenal/deep-learning-for-autonomous-driving

- **P_rect_X0** gives the intrinsic projection matrices to Cam X after rectification, given as a (**3**x**4**) numpy.array.
  The notation is given as P_rect_destinationFrame-originFrame.
- **K_camX** provide the pinhole camera intrinsics from Cam 0 to Cam X and are given as (**3**x**3**) numpy.array objects.

Only for **segmentation_data.p** the following *additional* keys can be found:

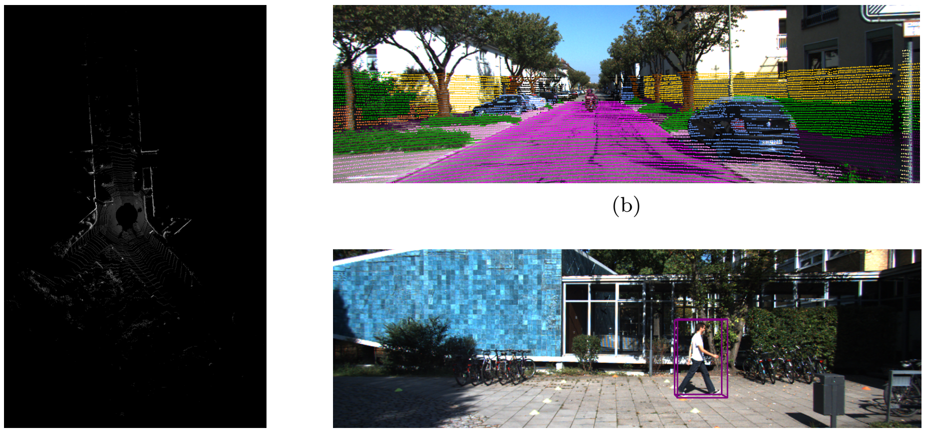
- **T_camX_velo** are the homogeneous velodyne to rectified camera coordinate transformations and are provided as (**4**x**4**) numpy.array objects.
  The notation is given as T_destinationFrame_originFrame.
- **sem_label** is a (**num_points**,) numpy.array object that gives the semantic label of each point within the scene.
- **color_map** is a dictionary which maps numeric semantic labels to a BGR color for visualization.
  Example: 10: [245, 150, 100] # car, blue-ish
- **labels** is a dictionary which maps the numeric semantic labels to a string class.
  Example: 10: "car"

Only for **detection_data.p** the following *additional* key can be found:

- **objects** contains a list of lists. Each sublist has 8 columns of the following:
  1 **type** describes the type of object: 'Car', 'Van', 'Truck', 'Pedestrian', 'Person_sitting', 'Cyclist', 'Tram', 'Misc' or 'DontCare'
  3 **dimensions** 3D object dimensions: height, width, length (in meters)
  3 **location** 3D object location x,y,z in Cam 0 coordinates (in meters)
  1 **rotation_y** rotation ry around Y-axis in Cam 0 coordinates $[-\pi : \pi]$
  Example: [Car, 1.65, 1.67, 3.64, -0.65, 1.71, 46.70, -1.59]

The coordinate systems are defined the following way, where directions are informally given from the drivers view, when looking forward onto the road:

- **Velodyne:** x: forward, y: left, z: up
- **Camera:** x: right, y: down, z: forward



(a) (b) (c)

Figure 2: Example results for problems 1-2. (a) BEV image, (b) Point cloud projected onto image and colored according to semantic labels, (c) 3D bounding box projected onto the image.

Problem 1. Bird's Eye View (1.5 point(s))

Fun fact: The bird's eye view (BEV) is an elevated view of a scene from directly above. Although BEV brings information loss during projection and discretization, crucially it preserves the metric space. Remember, objects we face in autonomous driving scenes such as cars or pedestrians do not fly (at least not yet)! This allows, detection models to explore priors about the size and shape of the object categories in the BEV view *without* obscurances.

Display the BEV image of the given scene in **segmentation_data.p** with pixel intensities corresponding to the points' respective reflectance values. A reference solution can be seen in Fig.2a where the projected point cloud is discretized into a 2D grid with resolution of 0.2m, 0.2m and 0.3m in x, y and z coordinates respectively.
Tip: Having problems visualizing your data? Maybe today is the day you learn[2] about cv2!

Problem 2. Projection onto Image Plane (2.5+1=3.5 point(s))

a. Semantic Segmentation: Displaying Semantic Labels

Project the point cloud of **segmentation_data.p** onto the image of Cam 2. Color each point projected according to their respective label. The color map for each label is provided with the data. A reference solution can be seen in Fig.2b.
Hint: Make sure to filter your point cloud! The provided velodyne scans are 360°. Projection equations will give you a result even for points that are behind the camera which will get projected as if they were in front, but vertically mirrored.

b. Object Detection: Drawing 3D Bounding Boxes

It's time for object detection! Project the 3d bounding boxes of all given vehicles, cyclists and pedestrians within the scene of **detection_data.p** to the Cam 2 image. A reference solution can be seen in Fig.2c.
Hint: A 3D bounding box is essentially just 8 points!

[2]https://opencv-python-tutroals.readthedocs.io/

Problem 3. ID Laser ID (1 point(s))

will be released next week.

Problem 4. Remove Motion Distortion (2 point(s))

will be released next week.

Problem 5. Theoretical Questions (2 point(s))

will be released next week.