The project was divided in two parts:

* Object detection
* Multiobject tracking

**Object detection:**

This part was done by Andrade Selene and Ludwig Harald, both students contributed equally for this part.

**Model selection:**

For the model we chosed “Super Fast and Accurate 3D Object Detection based on LiDAR”, we decided on this model based on its fast inference time, that we can use the pre-trained weights and it’s based on LiDAR.

The network architecture is a ResNet based Keypoint Feature Pyramid Network (KFPN) which works as a one-stage and multi-scale network for 3D keypoint detection which provide the accurate project points for multi-scale object. For this architecture to work a RGB image is taken as input based on the Bird's Eye View (BEV) transformed into a map using the intensity provide for each point and the respective coordinates.

The outputs of this model are heatmap for main center, center offset, heading angle, dimension, and the z coordinate. The model is capable of detect cars, pedestrians, and cyclists. This network was trained in the KITTI dataset which is a 3D object detection benchmark which consists of 7481 training images and 7518 test images as well as the corresponding point clouds, comprising a total of 80.256 labeled objects.

**Model configuration:**

First, we cloned the repository in order to use the model and the functionalities within, the link of the repository is <https://github.com/maudzung/SFA3D>

We use **dataset\_2** which has to be present in the repo folder. Simply running the [object\_detection.py](https://github.com/hmludwig/MultiObjectTracking/blob/objectDetection/object_detection.py) produces the [output\_video.avi](https://github.com/hmludwig/MultiObjectTracking/blob/objectDetection/output_video.avi) showing our detections in action as well as a pickled dictionary [output\_data.pkl](https://github.com/hmludwig/MultiObjectTracking/blob/objectDetection/output_data.tar.xz) where the key is the frame number and the corresponding value is a (detections, bev\_map) tuple.

**Overview of Bird's Eye View (BEV) calculation.**

To use the dataset provided we decode the lidar data in the dataset. For the Bird's Eye View (BEV) calculation we took the first channel of the Lidar decoding as “x” axis, the second channel as “y” axis, the third channel as “z” axis.

Instead of creating a BEV of every single point captured by the Lidar, it is useful to just focus on a rectangular region of the data when looked at from the top. For this reason, a filter was created that only keeps points within the forward rectangle.

**Metrics**

The [objectDetection\_metric.ipynb](https://github.com/hmludwig/MultiObjectTracking/blob/objectDetection/objectDetection_metric.ipynb) notebook shows the performance of the model compared to the ground truth.