**Team Bosch Documentation**

**Deployment:**

In the main directory there is a file named “run.bat”, when this file is run our program will open. Alternatively, the program can be executed using a Conda environment with the modules specified in our install guide. If you choose the latter you will not need the directory “release” in the main directory and can remove it if you choose.

**App.ui, App\_ui.py:**

The GUI for our application main window generated from PyQt editor.

**App.py:**

The main window of our application. The root for our program that contains functions for all of our buttons. Uses PyQt5.

**Classifier.py:**

This code uses LaneLineDetectorERFNet, VehicleDetector, VehicleTrackerSORT, and LabelGenerator to perform feature extraction and label prediction on a given video.

**ClassifierRunner.py:**

The portion of our code that manages the execution of the feature extraction for each video in a given list of videos. Feature extraction is executed in a separate python process. Only one video is processed at a time. It is possible to increase the number of concurrent processes to increase GPU utilization, however this has not been tested thoroughly.

**CONFIG.py:**

Useful for testing/debugging. MAKE\_PRECOMPUTED\_FEATURES allows the features to be saved so they do not need to be recomputed in the future. USE\_PRECOMPUTED\_FEATURES tells the program to use those saved features. If you are using precomputed features you will need to change the file path to the features in lines 65 and 134 in App.py and line 50 of Classifier.py

**DataPoint.py:**

Class that we save the features and labels to. There is a DataPoint object for each video that is processed.

**InfoDialog.py:**

The info window of our program. Activated by the “Info” button. Contains relevant functions to calculate and display a histogram of predicted labels.

**Infodialog.ui, infodialog\_ui.py:**

The GUI for our application info window generated from PyQt editor.

**LabelGenerator.py:**

Contains the function that generates all of the labels for a video. Each frame’s features are passed through this function right after they are generated.

**LaneLineDetectorERFNet.py:**

The portion of the code that generates the lane lines. Input is one video frame and output is a list of points that we expect a lane line in

**sort.py:**

Simple Online and Realtime Tracking algorithm. Gives a unique identifier to each bounding box that can be used to track the box from frame to frame.

**Storage.py:**

Contains functions related to reading and writing files.

**VehicleDetectorYOLO.py:**

You Only Look Once algorithm. Used to get bounding boxes for each vehicle in a video frame.

**VehicleTrackerSORT.py:**

Combines the SORT algorithm and the bounding boxes to make a “Vehicle” class, and returns a list of all the “Vehicles” for a video frame

**Video.py:**

Contains relevant functions for processing video files. Mainly used to get each frame of a video file as an image. Uses OpenCV.

**VideoOverlay.py:**

Contains relevant functions for overlaying vehicle bounding boxes, lane lines, and current label on an already processed video in the VideoWidget.

**VideoWidget.py:**

Contains relevant functions for playback of a video file with PyQt5.