# Home Assignment

Objects Detection, Mobileye
Proprietary and confidential. Do not copy or distribute.

July 2021

## 1 Introduction

You are given a short clip taken from Mobileye data (the file clip.mp4).
Each vehicle in the clip has the following labels:

- 3 vertical sticks which form a box bounding the vehicle.
  Note that the box is tighter than a bounding-rectangle which is commonly mentioned in the literature as a "bounding-box". Throughout this document the term "box" refers to the geometric figure containing 3 sticks.
- For each visible face of the vehicle, which type is it - back/front/left/right.
- Each vehicle has a unique ID that is identical in all of the frames where the vehicle is present.
  The ID is simply a positive integer.

As an example you can see the following frame with the labels. Note the ID which is presented in the top-left corner of the vehicle.

## 2 Training Data

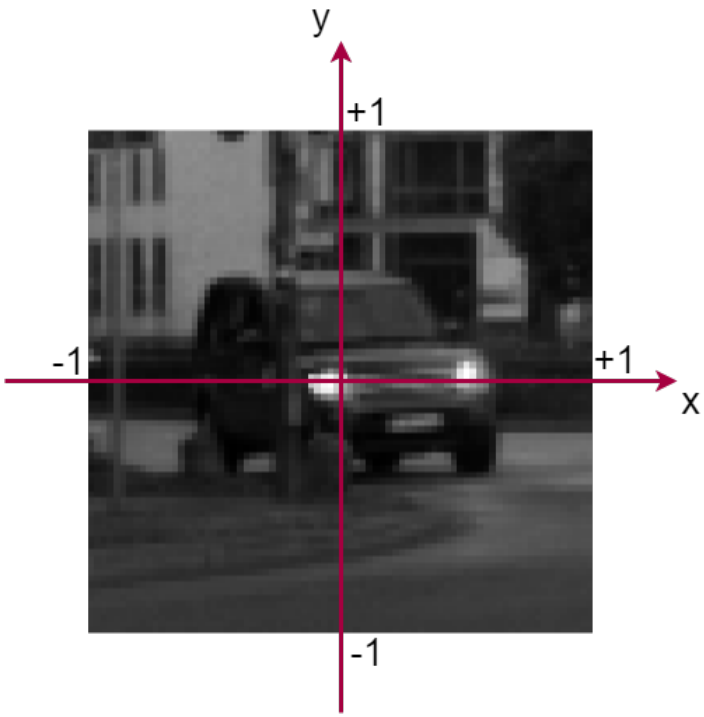
### 2.1 Processed Clip

The clip was processed to create training data for a classifier:

- For each vehicle, the portion of the image that contains it was cropped and resized to an image of fixed size $80 \times 80$. As an example, from the frame shown before we get 3 fixed-size images:

- For each vehicle, the sticks values were converted to the following coordinate system:
  - The origin is at the center of the image.
  - The $x$ axis starts from $-1$ in the left image edge and $+1$ in the right image edge.
  - The $y$ axis starts from $-1$ in the bottom image edge and $+1$ in the top image edge.

  An illustration of the coordinate system:




- For each vehicle, the types of the visible faces is saved as 4 numbers - the fraction of the face width from the total width of the vehicle, for each one of the four possible faces (back/front/left/right).

## 2.2 Technical Details

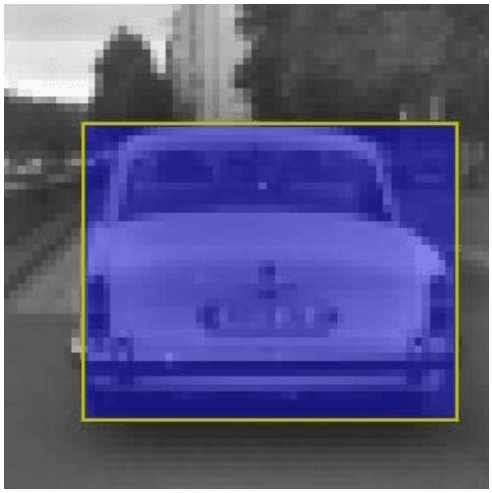
### 2.2.1 Data Format

The dataset is saved in binary files. Each attribute of the data is saved in its own binary file named *attribute1.bin*, and all of the binary files are aligned with one another. For example, if attribute *attribute1.bin* has shape $H \times W \times C$ and the numbers are stored as float32, each number is stored in 4 bytes so the total number of bytes is $H \cdot W \cdot C \cdot 4$. Reading the file can be done with the NumPy function np.fromfile. A description of each binary file shape and data type is saved along the binary files in a JSON file named data_structure.json. You should use it to parse the bytes read from the binary file.

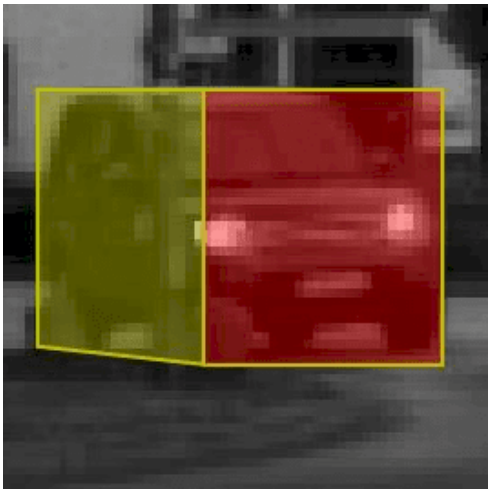
### 2.2.2 Data Attributes

The dataset contains the following attributes for each training sample:

- The image itself, which has shape $80 \times 80$ (the number of channels is 1). Numbers are stored as uint8 values (since grayscale images contain integers in the range $[0, 255]$). Note that the data type uint8 is stored in 1 byte (i.e. 8 bits).



(a) A vehicle with back face only.
The sticks are
$(-0.7, -0.7, 0.5, 0.9, -0.7, 0.5, 0.9, -0.7, 0.5)$
and the ratios are
$(1, 0, 0, 0)$



(b) A vehicle with right and front faces.
The sticks are
$(-0.9, -0.3, 0.7, -0.1, -0.4, 0.7, 0.8, -0.4, 0.7)$
and the ratios are
$(0, 0.6, 0, 0.4)$

Figure 1: Examples for vehicles with the box drawn on them. The sticks provide the bounding polygon, and the ratios provide the semantics of each face.
Blue corresponds to back face, red to front, yellow to right and green for left face.

- The sticks are stored as 9 numbers with data type float32 (4 bytes).
  The sticks are organized from left to right in the image, and numbered stick1, stick2 and stick3.
  The result is
  (stick1_x, stick1_b, stick1_t, stick2_x, stick2_b, stick2_t, stick3_x, stick3_b, stick3_t)
  where x is the x coordinate, b stands for "bottom" y coordinate and t for "top" y coordinate.
- The ratios are stored as 4 numbers with data type float32 ordered as
  (ratio_back, ratio_front, ratio_left, ratio_right).
- The frame and ID are both stored as a single number with data type uint32.

A few examples that can help understand the dataset are shown in Figure 1.

# 3 The Task

## 3.1 Background

We have trained a neural network. The network's input is an image cropped around the vehicle, and it predicts the sticks and the ratios describing the box. To increase the network's performance, we now want to give the network information about previous frames.

As a start, we want to reorganize the dataset so each training sample will contain a consecutive pair of images (and their corresponding labels). When our dataset will be organized this way, each training sample will be a pair of images containing the vehicle in two consecutive frames.

## 3.2 Instructions

Your task is to reorganize the dataset so that each data attribute will contain 2 channels instead of 1. The first channel will be the attribute of the vehicle in the current frame, and the second channel will be the attribute of the vehicle in the previous frame. In more details:

- Each image will now be of shape $80 \times 80 \times 2$ (width=height=80 as before, and channels=2 instead of 1). The first channel will be the image of the vehicle in the current frame, and the second channel will be the image of the vehicle in the previous frame.
- The sticks will now be a matrix of shape $9 \times 2$ where the first column will be the sticks associated with the current frame and the second will be the sticks associated with the previous frame.

- The same goes for the ratios - they will now be a matrix of shape $4 \times 2$.
- The frame will be two numbers instead of just 1, where the first one is the frame index of the current frame image, and the second will be the frame index of the previous frame image.
- The ID will stay the same - a single number (since the ID of both vehicles is the same).

### 3.3 Supplementary Code

To help you visualize the data and debug your code, we provide a python function that given an image, sticks, ratios, frame and ID, plots everything - the image with the bounding box, with colored faces indicating the types of the faces - blue corresponds to back face, red to front, yellow to right and green for left. The frame and ID are written below the image.

The function can be used to visualize what you will get as an input. The combined dataset that you will create (organized as pairs) can be visualized for debugging.

### 3.4 Guidelines

- You should write modular, readable and efficient code. Keep these things in mind:
  - Your script will be run by someone else, so it has to be dummy-proof and run as smoothly as possible.
  - Think of the future developer who will not be familiar with your code, and make their life easier by keeping the code self-explanatory.
  - Someone will probably need to do something similar but slightly different than you did. Your code should be written in a way that small behavioral changes can be achieved by minor code changes.
    An example for doing something similar but different is given in "Food For Thought" part 2.
  - Occasionally, your code might be useful in other scenarios, which is why it is essential to break it up into functions (and to generally write modular code).
- Regarding efficiency - write vectorized code. This task can be written **without any loops over the samples/IDs**, and this will be the most preferable solution. Of course, looping over the 5 different files is perfectly fine since it's a small constant number, but generally the number of samples can grow and become extremely large (already here in this 8 seconds clip there are 600-700 samples). Looping over the different IDs should also be avoided since it can also be quite large (although smaller than the number of samples, because a single ID often has multiple frames).

### 3.5 What To Submit?

You should submit a python script which given the directory containing the dataset, creates a new directory with the dataset organized in the aforementioned multi-framed structure.
The expected command-line should be something like:

```
python create_multi_framed_data.py --single_frame_dir INPUT_DIR --output_dir OUTPUT_DIR
```

where INPUT_DIR is the directory containing the binary files describing the dataset, and OUTPUT_DIR is the directory that the script will dump the new binary files describing the multi-framed dataset.

### 3.6 Tip & Tricks

A few python packages you will find useful - numpy, pandas, argparse, matplotlib.
More specifically, you could find the following functions/subjects useful: np.fromfile, array.tofile, indexing.

## 4 Food For Thought

1. What kind of model (i.e. neural network) will you use to train with the dataset you created? Think about the trade-off between model expressiveness and inference run-time.
2. Suppose we want to change the model to use more vehicle history.
   For example, make use of $k$ previous frames and not just the single previous one.
   (a) How will you organize the data?

(b) How will you change your code accordingly?

(c) What kind of model (i.e. neural network) will you use?

3. Suppose we want to change the model to use all of the vehicle history, and not just a bounded number of previous frames.

   (a) How will you organize the data?

   (b) How will you change your code accordingly?

   (c) What kind of model (i.e. neural network) will you use?

4. In each data organization you suggested in the previous questions, think about the efficiency of the training process - in each training step we sample a mini-batch of $N$ samples, how many I/O operations will be needed?