16:180:537 Intelligent Transportation Systems
Final Report and User Manual

Huixiong Qin

Anjiang Chen

May 5th, 2022

1. Video Object Detection

Features
object detection
object racing
roadway lane detection, see: https://github.com/voldemortX/pytorch-auto-drive
vehicle lane detection
Oracle uploading

Get Start
Clone the source code with submodule **git clone --recursive https://github.com/CoomaQin/its.git**

Follow the: https://github.com/voldemortX/pytorch-auto-drive
to download the weight of lane detection models

Modify the visualization function and run it to generate lane keypoints. In pytorch-auto-drive/utils/runners/lane_det_visualizer.py, check the run function of Class LaneDetVideo to:

```
def run(self):
        # Must do inference
        output = []
        for imgs, original_imgs in tqdm(self.dataloader):
            keypoints = None
            cps = None
            if self._cfg['pred']:
                imgs = imgs.to(self.device)
                original_imgs = original_imgs.to(self.device)
                cps, keypoints = self.lane_inference(imgs,
original_imgs.shape[2:])
            results = lane_detection_visualize_batched(original_imgs,
                                                       masks=None,
                                                       keypoints=keypoints,
                                                       control_points=cps,
                                                       mask_colors=None,

keypoint_color=self._cfg['keypoint_color'],
                                                       std=None, mean=None,
style=self._cfg['style'])
            results = results[..., [2, 1, 0]]
            output.append(keypoints[0])
            np.save("./kp", output)
            for j in range(results.shape[0]):
                self.writer.write(results[j])
```

Run the object detection and tracking (ODT) python3 ./object_detection_tracking.py. Note that you need to modify the video_path, YOLO weight etc.

Fuse the lane keypoints and object bounding boxes + trakcingID to determine the vihecle position by lane python3 ./determine_vehicle_lane.py. Note that you need to specify the path of the keypoint file, the ODT file and the forward-facing csv file.

2. Data Table Processing

Step 1. Concatenate frames to the most close-in epoch time of the video
After getting the coordinates of lane detection and vehicle bounding box information, run the **ITSproject.py** file to concatenate the frames and timestamps of video. The code will save the csv file called 'Timestamp_Frame.csv'.
In the table, column 'bounding box' includes the tracking ID, vehicle type and pixel coordinates of bounding box in form Array (ID, type, left top x, left top y, right bottom x, right bottom y); columns 'key point + number' are the pixel coordinates of lane detection in form Array (coordinate 1, coordinate 2…)

Step 2. Determine the relative position of the vehicles and the video vehicle
Use the csv file as the input from step 1, run the **fuse_lane_object.py** file to get the vehicle location information for each frame. The code will save the csv file called 'its.csv'.
In the table, columns 'right', 'left', 'current' means the relative position of the vehicles, they store an array of the tracking ID of vehicles in each column

Step 3. Separate the tables in a proper format for dumping to database
By combining the tables from step 1 and 2, disassemble them into three tables and store them in the database respectively, which is convenient for query and operation.
Run the code **Table.py** to get the tables. This code will output three tables in csv: '2021-08-27 17-15-45.csv', '2021-08-27 17-15-45 key point.csv', '2021-08-27 17-15-45 bounding box.csv'. Among the table, the date of the video is used to identify the corresponding video, set the column 'device name' as the date in the table as well.

Step 4. Load tables into Oracle database
The query file final **project.sql** is used to create table spaces for the tables and provide some case query operation codes. Also, three control files are created for uploading the tables into database.

3. Further Ideas and Future Work

Distance Estimation

1. Use the areas of vehicle bounding box directly to estimate the distance between vehicles.
2. Use a 'Detecting vanishing point' method to estimate the distance between vehicles. Reference: Huang, D. Y., Chen, C. H., Chen, T. Y., Hu, W. C., & Feng, K. W. (2017). Vehicle detection and inter-vehicle distance estimation using single-lens video camera on urban/suburb roads. Journal of Visual Communication and Image Representation, 46, 250-259.

Vehicle lane change detection

The midpoint of the lower boundary of the bounding box is used as the reference point of the vehicle, and the position is compared with the coordinate point of the lanes. When the coordinates of the midpoint and the lane are almost coincident, it is used as a key frame for analysis, and a suitable calculation method is found.

Vehicle position detection when the lane cannot be detected

1. The deep application of computational vision can be used to determine the shape of the vehicle appearing in the bounding box to determine the position: if there is only the rear of the vehicle in the box, it is the current lane, if there are doors, windows, and other shapes in the box, it is the left side or right
2. If there are vehicles in front of the current lane, directly use the bounding boxes of these vehicles as a reference, determine the approximate width and position of the lane through algorithms and visual applications, and then judge the vehicles on the left and right sides.