Computer vision activity report

Haichuan Chang (changhaichuan@gmail.com)

1. Introduction

This report provides the solution to the given computer vision problem, i.e., locating vehicles in bird-eye-view (BEV) images using a video captured by a fixed traffic camera as the input. This report is organised as follows. Section 2 introduces the adopted methodology, with a focus on object detection and perspective transform, and Section 3 provides results and discussions on the adopted method and other potential methods, followed by Section 4 providing some ideas to improve the model.

2. Methodology

The general algorithm to locate vehicles in bird-eye-view images is illustrated in Figure 1. Overall, each frame of the input video was processed via three main steps: (i) object detection, (ii) post-processing, and (iii) perspective transform, to be converted to a BEV image overlayed with vehicle detection results.

2.1. Object detection

A pretrained object detection CNN, i.e., SSD Mobilenet V2 trained on COCO 2017 dataset, was chosen for the vehicle detection task. The model is able to detect up to 90 classes of common objects, including cars, bus, person, etc, in an image with a fast inference speed and a moderate detection accuracy [ref].

With an image of variable size (typically 320x320), the model outputs (i) detected objects, (ii) detection confidence from 0 to 1, and (iii) bounding boxes that locate detected objects. When the input image size increases, normally the detection accuracy will be improved due to increased primary features, but the inference speed will be compromised.

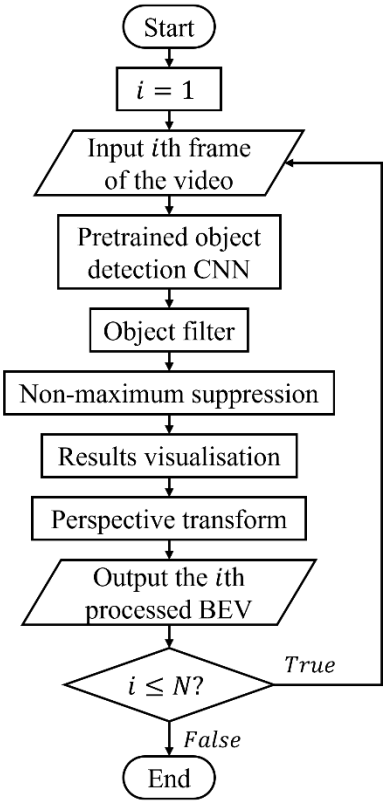

Figure 1: The algorithm diagram for locating vehicles in bird-eye-view images. N is the total number of frames of the input video.

2.2. Post-processing

The raw outputs of the object detection model could be problematic for this given task. The first immediate issue is that the model not only detect vehicles but also other non-vehicle objects in the input video, such as persons. To address this issue, a filter was implemented to only keep vehicles and their corresponding confidence score and bounding boxes. Specifically, only four types of vehicles were kept: (i) car, (ii) motorcycle, (iii) bus, and (iv) truck.

The second issue is common for object detection, i.e., multiple overlapping bounding boxes for one actual object. To select the most appropriate bounding box, non-maximum suppression was selected with a threshold of 0.2 (the ratio of overlapping).

The last step of post-processing is visualising the detection results, i.e., plotting bounding boxes for detected vehicles and encode different vehicle types with different colours of the bounding boxes. In this activity, yellow ([255, 255, 0]) was used for motorcycles, and green ([0, 255, 0]) was used for cars, with blue ([0, 0, 255]) for buses and red ([255, 0, 0]) for trucks.

2.3. Perspective transform

After post-processing, the images are still perspective views, i.e., from the traffic camera, and they need to be transformed to BEV images. The transformation of perspective was realised via two steps: (i) obtaining the perspective transformation matrix (PTM) by matching multiple points in the perspective view and the corresponding BEV, and (ii) applying perspective transform to processed video frames to generate BEV frames.

The PTM is a 3x3 matrix with 8 unknowns and can be calculated using 4 matching points in the perspective view and the BEV. Those points need to be at the same height in the world coordinate system. However, 13 points were used in this activity to improve the transform accuracy (see Figure 2), and a RANSAC-based robust method (cv2.RANSAC) was adopted to utilise the 13 matching points to refine the PTM.

Figure 2: Selected matching points (A-M) in the perspective view image (left) and the BEV image (right).

3 Results and discussions

3.1 Results

The output video can be found in the shared repository with a name of “output video.mp4”. Note that is made via screening recording because writing the output video using OpenCV failed due to an unsolved error.

The full output video can also be viewed in real time by running the “vehicle_detection.py” script, which only requires packages of cv2 and numpy. The output frame rate was around 12 fps using an Intel Core i7-4710MQ CPU @2.5GHz with the input image size of 320x320. The

frame rate could be significantly increased using GPU inference. However, CUDA support for OpenCV was not finished due to limited time.

Figure 3 shows an example output BEV frame with detection results. Overall, severe distortion can be observed, especially for objects above the ground, e.g., buses and buildings on the right. Most vehicles were detected but wrong detections can be observed, including wrong detections (e.g., the bottom post recognised as truck), wrong vehicle types (e.g., cars on the left recognised as buses), and overlapping predictions (e.g., the top car recognised as two cars).

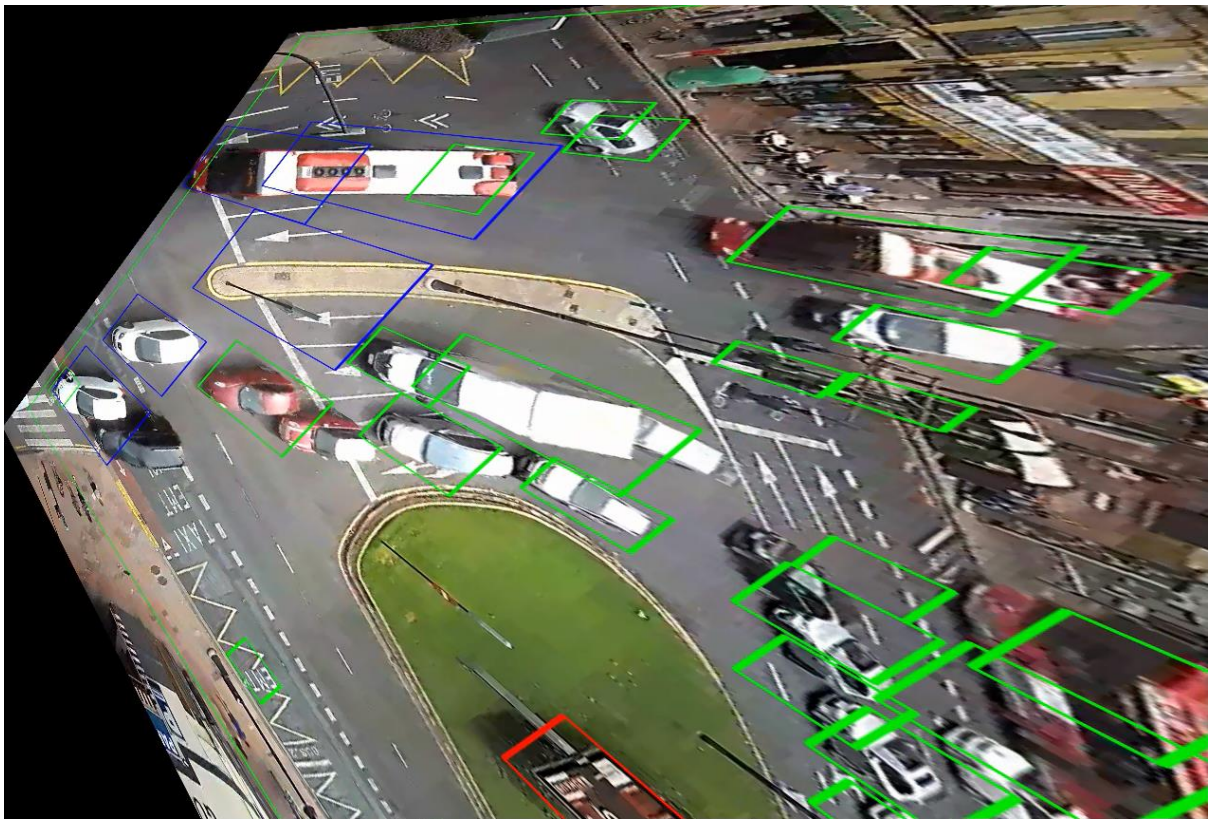

Figure 3: An output BEV frame with detection results.

3.2 Object detection methods

It can be seen from the results that the chosen object detection model, i.e., SSD MobileNet V2, delivered a compromised detection accuracy compared to its performance shown in other tests. A main reason could be the traffic camera perspective being different from those of most training samples for the model. By examining the COCO 2017 dataset, it can be found that most car images were captured from a pedestrian's perspective or a similar one. Cars in

different perspective views have different prominent features, and it could partly explain the comprised detection accuracy of SSD MobileNet V2 in this task.

Figure 4: Examples images of cars in COCO 2017 dataset

Except the chosen model, there are many other pretrained object detection models that can be utilised, such as EfficientDet and YOLOv5. Attempts have been made to convert a TensorFlow EfficientDet D0 model to OpenCV-compatible model via extracting its frozen inference graph and configuration file, but errors occurred during graph transform. Directly using TensorFlow Object Detection API [ref] could avoid the issue of model conversion, also providing access to more mature models, but it requires many dependencies, causing potentially too much trouble for inexperienced users. Due to the time limit, I did not use TensorFlow models and its object detection API.

Except using deep learning models, object detection could also be realised via image processing techniques. Since the traffic camera is fixed and a background image (the empty street junction), background subtraction (BS) could extract foreground objects from the input video, i.e., vehicles, people, etc., and then vehicles could be separated from other objects assuming they strictly occupy different regions. Then, morphological operations such as image opening could be employed to remove noises. This approach could provide advantages such as fast execution but also could bring issues such as low robustness.

3.3 Perspective transform methods

As shown in the output video, there are large distortions in BEV images, and it could be caused by two issues. The first and the most important one is that the adopted method for perspective transform only works for object points on the same plane, which is the street surface in this task. Other object points such as car roofs, trees, posts, and buildings that are higher than the street will be inevitably distorted. The second and the minor one is camera distortion which is typically associated with wide-angle lenses, and camera calibration is needed to remove the distortion. However, the common way to calibration a camera involves using the camera to capture multiple images that contain chessboard patterns, which cannot be realised for this task. Theoretically, the matching patterns (lane marks) in the original image and the BEV image could be utilised for camera calibration, but a working implementation was not developed within the given time.

In order to preserve 3D information in the images, 3D reconstruction using perspective images could be used and proper BEV images could be obtained by reprojection. Based on research on this topic, deep learning models are commonly trained to achieve high-quality 3D reconstruction. Due to time limit, this approach was not attempted. A potential problem of using such a model is a significant increase of inference time and a decrease of the output frame rate.

If the actual images of cars are not essential in BEV images, i.e., the output video showing only bounding boxes moving on an empty street, perspective transform of the entire image is not needed anymore. But the image coordinates of detected vehicles need to be transformed to world coordinates. It can be realised by multiplying one point of each detected vehicle (preferably a point that is close to the ground) with the PTM obtained in section 2.3. In this way, detected vehicles are located in the ideal BEV (see Figure 2), and then imaginary bounding boxes can be plotted on the BEV with different colours representing different types of vehicles. This approach large avoids the distortion problem, but the actual images of vehicles are missing, which makes it impossible to humans to evaluate the detection accuracy when inspecting the output video. Considering the moderate detection accuracy of the selected model in this task, this work-around was not adopted.

4 Ideas for model improvement

- Train a customised object detection CNN that only detects vehicles using transfer learning. Pre-trained object detection CNNs such as YOLOV5 and EfficientDet with

the final few layers being relaxed could be trained on a small dataset of images captured by traffic cameras.

- Deploy a 3D reconstruction model that converts perspective images to 3D models and use re-projection to generate BEV images.