Project No.3 Part one Car Tracking

Zejiang Zeng 05/15

Overview:

Detecting and tracking vehicles are important in self-driving system. In this project, a car tracking techniques are implemented using MATLAB and its computer vision toolbox. Car tracking requires detecting cars first and then tracking them in real-time. Haar Cascade Classifier was used to distinguish cars and other non-car objects. And those cars are tracked based on the detected bounding boxes or feature points from frame to frame.

Detecting

The Computer Vision System Toolbox cascade object detector can detect object categories whose aspect ratio does not vary significantly. Objects whose aspect ratio remains fixed include faces, stop signs, and cars viewed from one side [1].
The cascade classifier consists of stages, where each stage is an ensemble of weak learners. Each stage of the classifier labels the region defined by the current location of the sliding window as either positive or negative. Positive indicates that an object was found and negative indicates no objects were found. Positive images that include different views of cars are provided. Those positive images with region of interested (whole image in this project) are used as positive samples. Negative images are bunch of irrelevant non-car objects, which are also provided from the dataset. More negative images are cropped and added during testing. Negative samples are generated automatically by the function. To achieve acceptable detector accuracy, set the number of stages, feature type, and other function parameters. More details can be found in [1, 2]. In this project, I set the FalseAlarmRate as 0.2, stage as 5 and feature type as HOG. A XML file was generated using based on those samples and parameters. In the end, a car detector was created to identify cars on the road like figure showed below. During the process, the positive and negative images are added and XML file may be regenerated. The cars (bounding box) will be filtered based on its centroid location and aspect ratio. Cars won't be top half part of the image and won't be too close to both sides of the image, so bounding boxes that located in that regions should be filtered. And the width or height of the bounding box shouldn't be too large or too small. Those bounding boxes will be filtered as well.

Figure 1. Detected Cars

Tracking

After those cars are detected with bounding box in previous and current frame, it's time to match/ track each of them to the appropriate bounding box. There are two ways to find the matches. First one was using the closest matching bounding box between each frame. The second one is using KTL tracker. Both ways are used in my project to find an efficient way to track the car.

Like what described in provided material, Closest Bounding Box was to find the closet bounding box in the previous frame for each bounding box in current frame. For the first of the image, cars are detected by the detector we just created. After filtering all those false positive bounding boxes, each centroid of those solid car (bounding boxes) are calculated. Then in next frame, do the same thing. Then compare those centroids to find the closest appropriate bounding boxes based on a threshold (20 pixels in my project) and match each other to find the same car in previous frame to current frame. If the current frame lost one solid car or detect a new potential car, those bounding box will be checked to verify if it's need to add the lost car or the new car. Do the same thing for the whole video. It will get a car tracking result.

Same thing for using the KLT tracker. After find the solid cars (bounding boxes), a vision.PointTracker system is used. The point tracker object tracks a set of points using the Kanade-Lucas-Tomasi (KLT), feature-tracking algorithm. More details can be find

in [3]. For each bounding box, feature points are extracted and will be matched from those source points to destination points by the KLT tracker. The geometric transform is calculated to find the transformation of those bounding box. Repeat the process for the whole video.

Result

Both of the method can give a feasible result of tracking cars. However, for the simple.avi, the closest bounding box works a little better than the KTL tracker. Since for some reason, as showed in KTL video, the bounding box will deform a little bit and one bounding box is getting out of the car. The CBB gives a perfect result though.

Reference

1. "Documentation." *Train a Cascade Object Detector - MATLAB & Simulink*. N.p., n.d. Web. 15 May 2017.
2. "Training Image Labeler." *Train Cascade Object Detector Model - MATLAB TrainCascadeObjectDetector*. N.p., n.d. Web. 15 May 2017.
3. "NumPyramidLevels." *Track Points in Video Using Kanade-Lucas-Tomasi (KLT) Algorithm - MATLAB*. N.p., n.d. Web. 15 May 2017.