The first step is to extract features from each bounding box of every frame, and associate those bounding boxes with similar features together. In this experiment, the extracted feature is selected to be Optical Flow Vector. There are several main reasons of this selection. Initially, the optical flow vectors can accurately represent the motion information of the objects, which can be very useful features to perform data association in situations where various types of objects move in different directions and speeds. One of such situations can be that some pedestrians, buses and cars move at a busy road intersection. Besides, optical flow features can be computed with high efficiency. Since this computation can be performed in parallel, it is even faster in an running environment where a GPU is enabled. This can be a huge advantage when certain run-time requirements need to be achieved in practise. In addition, the optical flow can be calculated quite accurately even from a pair of relatively blurred images. Some of our testing videos are taken with low resolution cameras, with extreme weather conditions such as heavy rain, or with diverse time conditions such as twilight and night time. The optical flow vectors can still succeed in describing the object motion information to some extent under such condition.

Viewpoint invarianet?

This process requires at least three continuous frames together. For example, if the first three frames are indexed with frame 0, frame 1, and frame 2, the program computes the optical flow vectors of each pair of bounding boxes from sequential frames separately (between frame 0 and frame 1, then frame 1 and frame 2) and associate the bounding boxes with similar optical flow vectors together. In detail, after receiving the frame 0 as an input, the YOLO detector would directly scan the image and label bounding boxes around any areas where it thinks there could be some objects. Then, the program will save these bounding boxes’ coordinates and draw the bounding boxes with the same width and height at the same location in frame 1. If an object moves in any directions at frame 1, there will be a slight shift in terms of pixels inside the bounding box, which reflects the motion of this object. If an object stays stationary, there would be no changes in pixels inside the bounding box, therefore the bounding box would be the exact same as the one in frame 0. In either situation, it is possible to compute a list of optical flow vectors to measure the pixel differences between the bounding box pair from frame 0 and frame 1. The optical flow vectors computed based on bounding box pairs are able to represent the motions of objects. The degree parameter inside the optical flow vector measures in which direction an object moves, and the magnitude parameter measures the speed of the motion in terms of pixels. After computing the optical flow for the bounding boxes from frame 0 and frame 1, we can repeat this process for bounding boxes from frame 1 and frame 2, which results in gaining another set of optical flow vectors for each bounding box pair. Then, we can compare the bounding boxes to see if their optical flow vectors are within certain range.

Optical Flow Computation

The important assumption here is that the object does not move far away from its position in the last frame. In terms of pixels, this means the object should not move beyond TEST pixels.

Minor change in illumination

There are two commonly used approaches for computing optical flow vectors, which are based on features or dense pixels. In order to find the most appropriate method for our application, we pick one method from each approach and test them separately. As for the feature-based approach, we choose Lucas-Kanada method; as for the dense pixel approach, we select Farnback method. These two methods are famous for their high efficiency and robustness, and the APIs for both of methods are available in OpenCV library. After testing each method against the same number of image pairs, their results are compared with each other in terms of accuracy and computation speed. We finally chose to apply the Farnback method in the final version of the application because it is more robust and able to generate more accurate results, which is more representative to describe the object motion.

Lucas-Kanada Method (EQUATION)

This is an iterative pyramid method to compute optical flow vectors based on a sparse features set.

Farnback Method (EQUATION)

The pictures below [PICTURE] are examples of optical flow vectors computed from the same image pair using two methods. In Lucas-Kanada method example, the same number of corner features are extracted around the edges of cars in the image pair. Next, these corner features from each frame are matched based on the Lucas-Kanada method. The optical flow vectors are the pixel differences between each feature pair. In this example, most of corner features are correctly extracted around the edge of the moving car. However, one feature pair is extracted near the road, which is not expected. In fact, this unexpected feature extraction problem would get more severe as the image dimension becomes larger. The larger the bounding box dimension gets, the lower chance of that bounding box precisely covering or including the target object becomes. In other words, some unwanted surrounding objects (such as roads and walls) are more easily to be included inside larger bounding boxes, which results in features computed around unwanted these objects. This becomes a serious problem during the process for computing an optimal value among optical flow vectors to correctly represent the object motion. Please read the Optimal Optical Flow Vector Selection for details.

On the other hand, optical flow vectors are densely calculated based on pixels in Farnback method example. More specificly, for every certain number of pixels in each frame, we choose a number of pixels. Afterwards, we calculate the dense pixel movements by matching pixels with their neighbors, and then assign these movements as the optical flow vectors. As it can be shown in the example, pixel movements are computed everywhere inside the bounding box. Since every component of the car is moving in the same direction, the pixel movements around the car are all in the same directions. In contrast, all of road pixels would have zero movements since the road is stationary. By doing this way, it is very fast to identify and separate moving regions and stationary regions. For most of cases, this means separating foreground regions and background regions from the input image, thus it is much easier to compute a single optimal optical flow vector later.

Special case 1: Pedestrian

Special case 2: large bounding boxes

Optimal Optical Flow Vector Selection