Vehicle Detection in Color Images

The problem

Having a video sequence, taken from a static camera, of a road, detect the vehicles on each frame.

The solution

1. Compute the optical flow of all the pixels in the image (this step is done for each of the RGB channels of the image).

2. Segment the image in pixels with high optical flow (moving objects) and pixels with small optical flow (static pixels – background).

3. Cluster the pixels with high optical flow.

4. Find bounding boxes of the clusters.

5. Remove clusters that are too small.

1 Optical Flow - The Farnebäck algorithm

Optical flow means computing the displacement of a pixel between two consecutive frames. This implies **finding a match** in the second image for each pixel in the first image.

Let's assume that we have an initial guess of the optical flow. How do we know how good it is? How can we measure similarities between the pixels in the first frame and their matches?

In computing the optical flow, the essential part is data representation. The Farnebäck algorithm uses polyonmial expansion: “The idea of polynomial expansion is to **approximate some neighborhood of each pixel with a polynomial**.” [1] That way, we can measure how good the similarities are by computing the error of the two polynomials.

Let $f_1(x)$ be a local approximation of the first frame around point x_0, and $f_2(x)$ be a local approximation of the second frame around point x_0. The task is to find the displacement

vector d such that $f_1(x - d) = f_2(x)$ in a neighborhood around x_0. The task is accomplished using mathematical operations.

In my implementation, I used the *calcOpticalFlowFarneback* function provided by OpenCV to compute the optical flow. The function was called, in turn, for each channel of the image. So there will be 3 optical flows for each pair of frames.

The Farnebäck algorithm was chosen because it computes the optical flow in each pixel, i.e. it is a **dense** optical flow algorithm.

2 Image Segmentation

Having the displacement of each pixel in the first image, we can find the pixels that belong to moving objects by static thresholding. The thresholded was set on the optical flow **magnitude** (how big the displacement is) in each pixel.

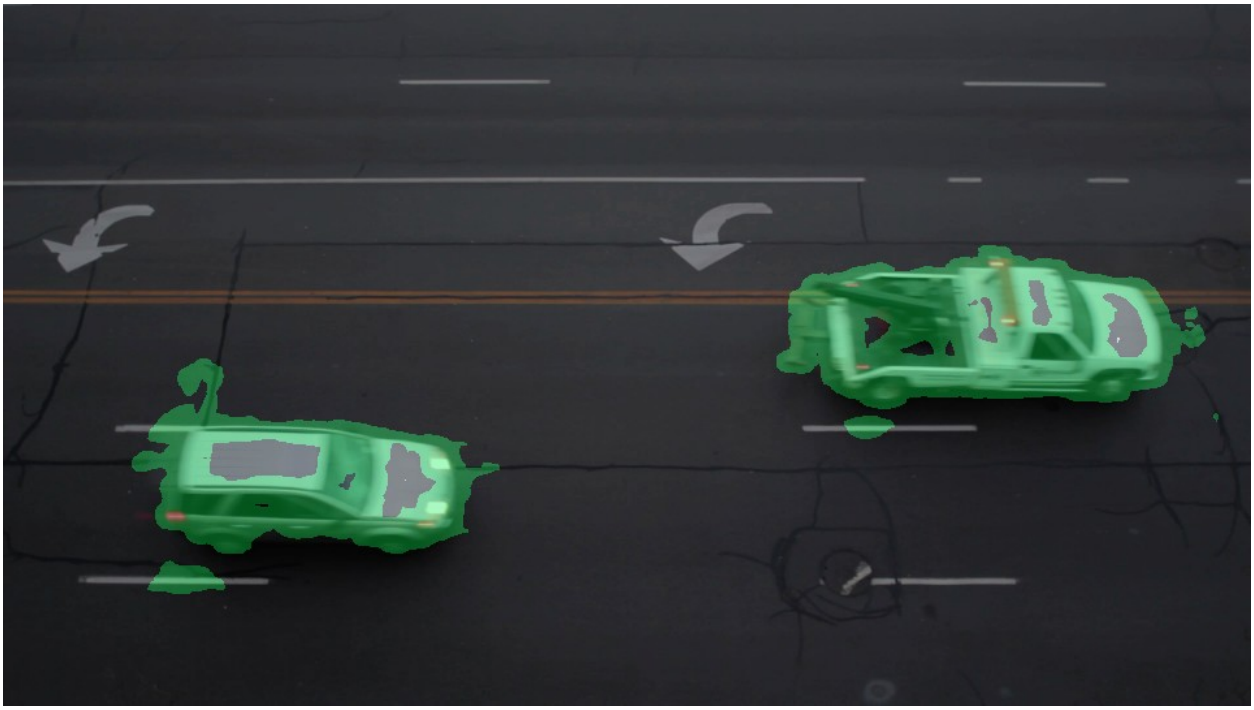

Figure 1: The pixels with high optical flow are highlighted in green

3 Clustering

In this step, we create regions that have high optical flow. For that purpose, a bread-first-search algorithm was used. Starting from each pixel with high optical flow, create a queue and add its neighbors (left, up, right, bottom) to it, then add its neighbors' neighbors and so on, until the region cannot be expanded.

4 Bounding Boxes

For each cluster, find its extents for each coordinate, to display the result as seen in other papers. That is, find minimum and maximum x and y coordinate for each cluster.

5 Noise Filtering

In figure 1, there are two regions that should not have been detected (noise). To rule them out, I've used a simple filtering: all the regions that are smaller than some fixed width and height limits are removed.

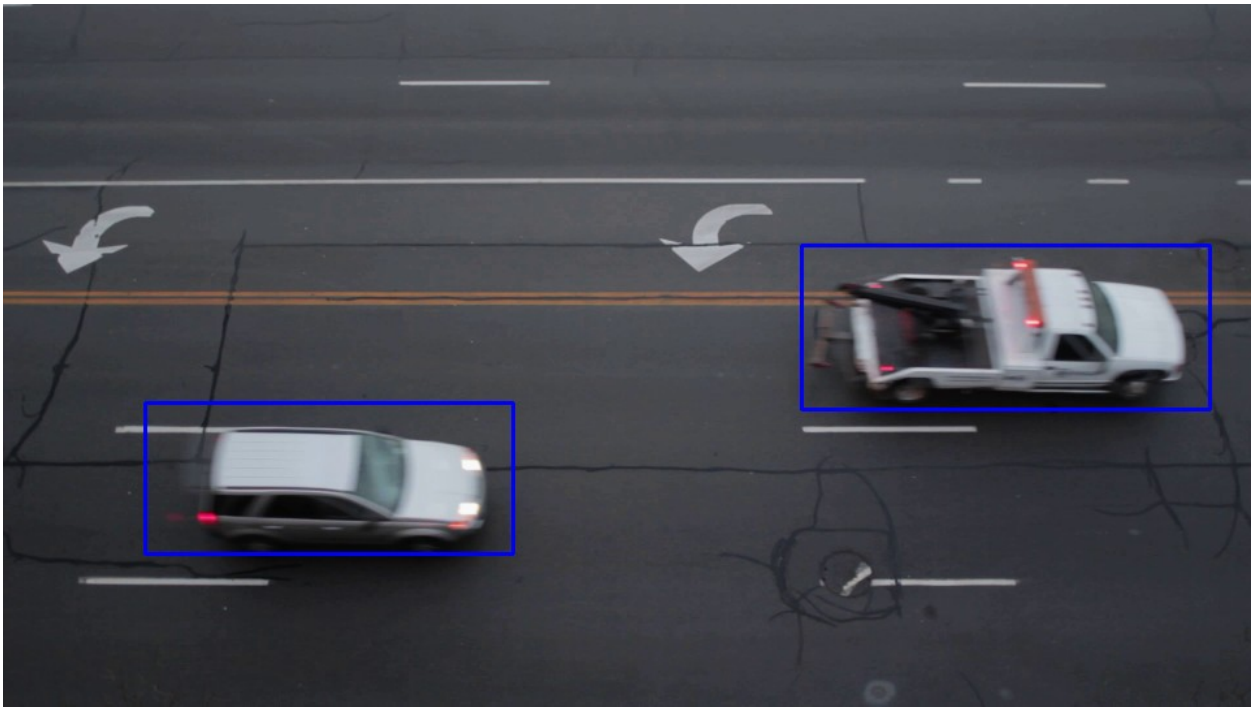

Figure 2: the final image, with bounding boxes on the two cars.

References

[1] Gunnar Farnebäck: “Two-Frame Motion Estimation Based on Polynomial Expansion”