Optical Flow

A. JAMES CLARK
SCHOOL OF ENGINEERING

Pooja Kabra

May 2020

Table of Contents

Plot motion vectors 3
Remove background to isolate moving objects 5
References: 6

Plot motion vectors

Here, we are given a video of cars being driven on a highway. The aim is to use the concept of Optical Flow to track the motion of vehicles on the highway.

For the video, we plot vector field of the optical flow for every frame. This can be done by comparing subsequent frames and checking how a pixel has progressed. We space the vectors 25 pixels apart from each other (both vertical and horizontal).

Optical flow is the set of displacement vectors showing movement of an object between consecutive frames. Optical flow makes 2 main assumptions:

1. Brightness constancy, i.e. object pixels don't change intensity in consecutive frames
2. Neighboring pixels in a small region have the same velocities

From 1,

$$I(x,y,t)=I(x+dx,y+dy,t+dt)$$

Using Taylor Series Expansion for RHS, we get:

$$f_x u+f_y v+f_t=0$$

where:

$$f_x=\partial f/\partial x \quad \text{and} \quad f_y=\partial f/\partial y$$

$$u=dx/dt \quad \text{and} \quad v=dy/dt$$

fx and fy, they are image gradients

This is the equation for one object pixel. However, this is insufficient information to solve for u and v as we have one equation and two unknowns. Here is where the second assumption comes into play. We use a small 3x3 block of neighbor pixels to solve for our unknowns. Now we have 9 equations and 2 unknowns. This is an overdetermined system. Thus Total Least Squares gives a better solution:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_i f_{x_i}{}^2 & \sum_i f_{x_i} f_{y_i} \\ \sum_i f_{x_i} f_{y_i} & \sum_i f_{y_i}{}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i f_{x_i} f_{t_i} \\ -\sum_i f_{y_i} f_{t_i} \end{bmatrix}$$

Because this approach only works for small motions, we use the Lucas-Kanade Method that deals with large motions using pyramids.

Code:

We read one frame from the video, as we need an initial frame to work upon. We call this old_frame. We extract a grid of evenly spaced (25-pixels apart) points into p0. Now we start our while loop. For every iteration, we read a new frame and find corresponding points p1 to p0 by calling cv2.calcOpticalFlowPyrLK(). It takes in p0, old_frame, *current* frame as arguments. Now on a black mask of zeros, we draw arrows from p0s to corresponding p1s and add this mask on the current frame. We use cv2.imshow() to display.

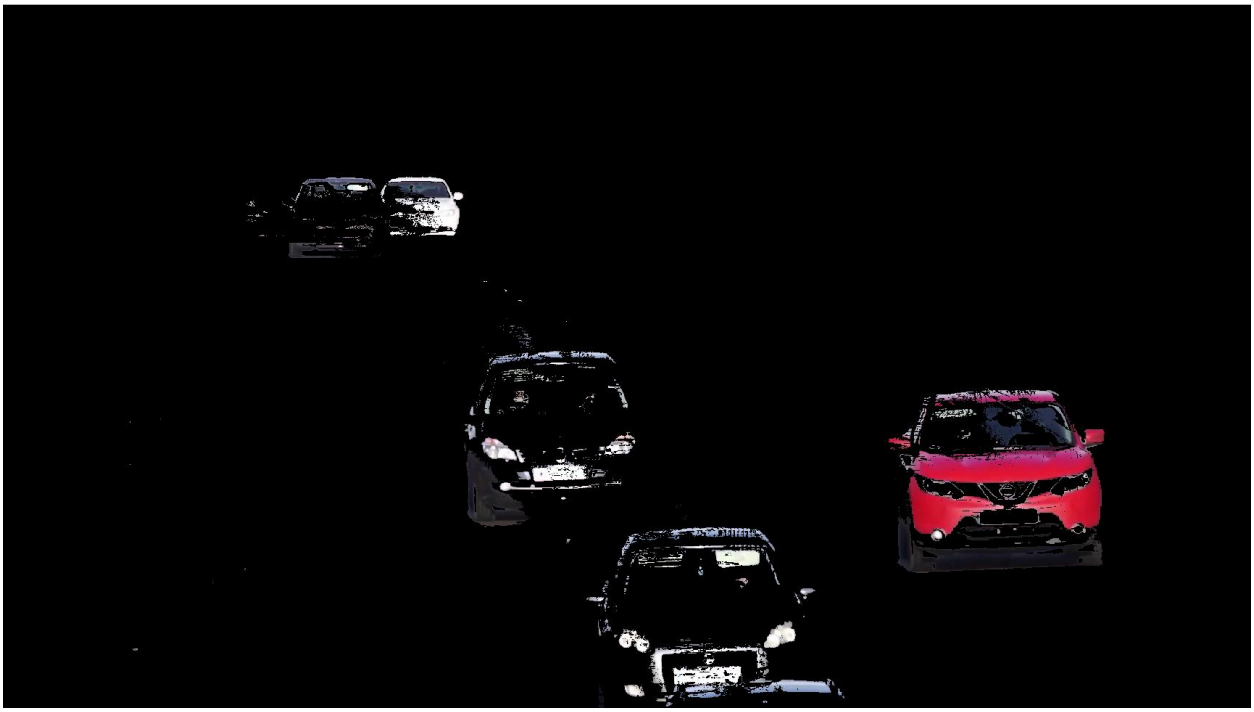
Figure 1: Optical Flow

Remove background to isolate moving objects

In a separate video, we remove the background and show only the movement of cars on the highway.

Optical flow can also be used to detect moving objects from stationary background. Flow vectors can be matched to detect pixels belonging to same objects. However, here we will use inbuilt functions to achieve the same.

The process is fairly simple. We create a background subtractor object fgbg using cv2.bgsegm.createBackgroundSubtractorMOG(). We apply this on our current frame to extract a foreground mask and bitwise_and() the mask with the color frame. Of, course we convert the mask to RGB color space before the operation. cv2.imshow() is used to display the result.



References:

1. Optical Flow
   https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html

2. How to Use Background Subtraction Methods
   https://docs.opencv.org/master/d1/dc5/tutorial_background_subtraction.html