Motion detection and tracking,
With stationary camera

Péter Bence

Széchenyi István University Győr

Abstract This is a documentation about detecting motion then tracking it, using a stationary camera. Background subtraction and Optical Flow algorithm are used for detecting tracking motion. Written in python, with the use of OpenCV Computer vision library.

1. Introduction

A static camera observing a scene is a common case of a surveillance system. Detecting intruding objects is an essential step in analyzing the scene. An usually applicable assumption is that the images of the scene without the intruding objects exhibit some regular behavior that can be well described by a statistical model. If we have a statistical model of the scene, an intruding object can be detected by spotting the parts of the image that don't fit the model. This process is usually known as "background subtraction". Then we can track the detected objects, optical flow can calculate the motion vector of the moving object, this can show us the direction, the object was moving in.

2. Background subtraction algorithms

Background subtraction (BS) is a common and widely used technique for generating a foreground mask (namely, a binary image containing the pixels belonging to moving objects in the scene (pixel intensity values: 0 = black, 127 = shadow, 255 = moving object) See: Foreground Mask below) by using static cameras. As the name suggests, BS calculates the

foreground mask performing a subtraction between the current frame and a background model, containing the static part of the scene or, more in general, everything that can be considered as background given the characteristics of the observed scene. Every frame is used both for calculating the foreground mask and for updating the background.

Foreground Mask *Output image*

If there is an image of background alone, like an image of the room without visitors, image of the road without vehicles etc., it is an easy job. Just subtract the new image from the background. We get the foreground objects alone. But in most of the cases, we may not have such an image, so we need to extract the background from whatever images we have. It becomes more complicated when there are shadows of the vehicles. Since shadows also move, simple subtraction will mark that also as foreground. It complicates things.

The background model is estimated from a training set X. In practice, the illumination in the scene could change gradually (daytime or weather conditions in an outdoor scene) or suddenly (switching the light off or on in an indoor scene). A new object could be brought into the scene, or a present object removed from it. In order to adapt to these changes, we can update the training set by adding new samples and discarding the old one. There are models in the literature that consider the time aspect of an image sequence and then the decision depends also on the previous pixel values from the sequence. However, these methods are usually much slower and adaptation to changes of the scene is difficult. The pixel-wise approaches assume that the adjacent pixels are uncorrelated. Markov random field can be used to model the correlation between the adjacent pixel values (Kato et al., 2002) but leads to slow and complex algorithms. Some additional filtering of the segmented images often improves the results since it imposes

some correlation (Elgammal et al., 2000; Cemgil et al., 2005). Another related subject is the shadow detection. The intruding object can cast shadows on the background. Usually, we are interested only in the object and the pixels corresponding to the shadow should be detected (Prati et al., 2003).

Gaussian mixture model: In order to adapt to possible changes, the training set should be updated. We choose a reasonable time adaptation period T. At time t we have $X_r = \{x^t, \ldots, x^{(t-T)}\}$. For each new sample we update the training data set X_r and reestimate the density. These samples might contain values that belong to the foreground objects. Therefore, we should denote the estimated density as $\hat{p}(\vec{x}^{|t|} | X_r, BG+FG¿$. We use a GMM with M components:

$$\hat{p}(\vec{x}| X_r, BG+FG¿ = \sum_{m=1}^{M} \hat{\pi}_m N(\vec{x}; \hat{\vec{\mu}}¿, \hat{\sigma}_m^2 I)¿$$

Where $\hat{\vec{\mu}}_1, \ldots, \hat{\vec{\mu}}_M$ are the estimates of the means and $\hat{\sigma}_1^2, \ldots, \hat{\sigma}_M^2$ are the estimates of the variances that describe the Gaussian components. For computational reasons the covariance matrices are kept isotropic. The identity matrix I has proper dimensions. The estimated mixing weights denoted by $\hat{\pi}_m$ are non-negative and add up to one.

3. Filtering out false positives

Based on the number of not black (intensity = 0) pixels N in the foreground mask $F(x, y)$: $N = Count¿$), it can be decided, whether true motion detected, or it is just false positive. For further filtering, Gaussian Blur, thresholding, morphological transformation being applied on the foreground mask. For easier visualization, around the moving objects contours or rectangles can be drawn, using contour detection algorithm. (See: Output image below)

Gaussian blur: $K(u, v) = \frac{1}{2\pi\sigma^2} e^{\frac{-u^2+v^2}{\sigma^2}}$

$$J(x.y) = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} * I(x, y)$$

Mophology:

Filtered mask

Raw mask

4. Optical Flow

Optical flow is the pattern of apparent motion of image objects between two consecutive frames caused by the movement of object or camera. It is 2D vector field where each vector is a displacement vector showing the movement of points from first frame to second. Optical flow works on several assumptions:

1. The pixel intensities of an object do not change between consecutive frames.
2. Neighbouring pixels have similar motion.

Consider a pixel $I(x, y, t)$ in first frame (Check a new dimension, time, is added here. Earlier we were working with images only, so no need of time). It moves by distance (d_x, d_y) in next frame taken after *dt* time. So, since those pixels are the same and intensity does not change, we can say,

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

Then take taylor series approximation of right-hand side, remove common terms and divide by *dt* to get the following equation:

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

where:

$$f_x = \frac{\partial f}{\partial x}; f_y = \frac{\partial f}{\partial y}$$

$$u=\frac{dx}{dt};v=\frac{dy}{dt}$$

Above equation is called Optical Flow equation. In it, we can find f_x and f_y, they are image gradients. Similarly, f_t is the gradient along time. But (u,v) is unknown. We cannot solve this one equation with two unknown variables. So, several methods are provided to solve this problem and one of them is Lucas-Kanade.

Lucas-Kanade method: We have seen an assumption before, that all the neighbouring pixels will have similar motion. Lucas-Kanade method takes a 3x3 patch around the point. So all the 9 points have the same motion. We can find (f_x,f_y,f_t) for these 9 points. So now our problem becomes solving 9 equations with two unknown variables which is over-determined. A better solution is obtained with least square fit method. Below is the final solution which is two equation-two unknown problem and solve to get the solution.

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

(Check similarity of inverse matrix with Harris corner detector. It denotes, that corners are better points to be tracked.)

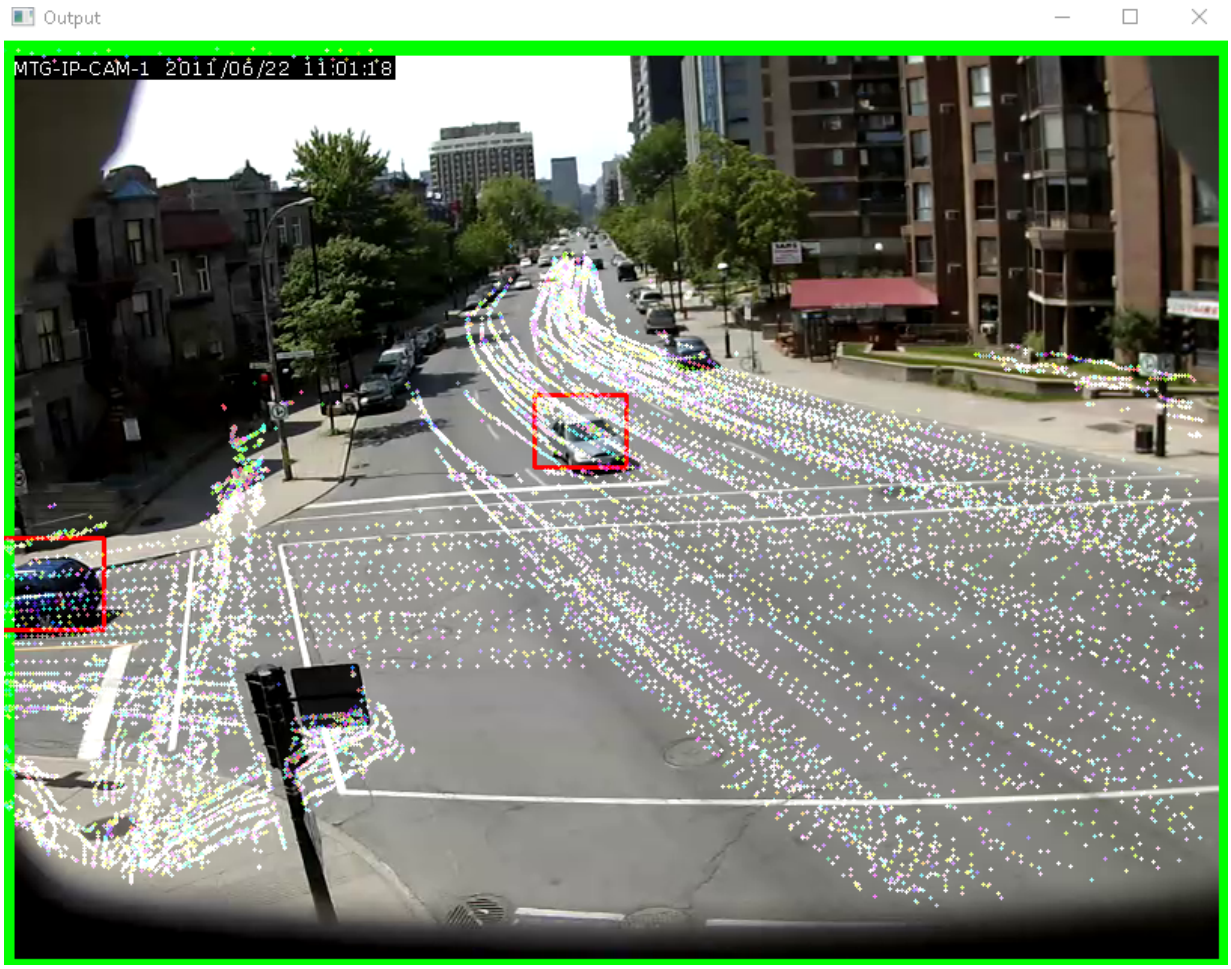
The idea is simple we give some **points to track (Harris corner detector)**, we receive the optical flow vectors of those points. But again, there are some problems. Until now, we were dealing with small motions, so it fails when there is a large motion. To deal with this we use pyramids. When we go up in the pyramid, small motions are removed, and large motions become small motions. So, by applying Lucas-Kanade there, we get optical flow along with the scale.

Harris corner detector

For each pixel (x,y) it calculates a 2×2 gradient covariance matrix $M(x,y)$ over a $blockSize\times blockSize$ neighborhood. Then, it computes the following characteristic:

$$dst(x,y)=det\,M^{(x,y)}-k\cdot\left(tr\,M^{(x,y)}\right)^2$$

Corners in the image can be found as the local maxima of this response map.



The Drawn Optical flow