POINT FEATURE TRACKER

Meghana Ravirala
Department of Computer Science and Engineering
University of South Florida
Tampa, Florida, US

I. Algorithm Description

A feature is technically a piece of information which is used to solve tasks related to a certain application. Features can take many forms such as points, objects or edges. In this assignment, points are considered as features which are then used for tracking. The first step is to determine the points in the first frame of the video. This is done by finding the gradient images and computing the auto-correlation matrix as well as eigen values of the matrix. Once the points are learnt, the next step is to track those points through the video using summed square difference metric in all the subsequent frames.

II. Algorithm Implementation

The algorithm is implemented in python. The first stage of the algorithm includes edge detection of the object in the first frame. This is done by computing the x gradient, y gradient and xy gradient using the gaussianfilter1d function under the scipy library. The value of sigma was taken as 1. Then, the x gradient and y gradient are squared to get Ixx and Iyy.

Gaussian function:

$$g(x) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-x^2/2\,\sigma^2}$$

Ixx and Iyy are used to calculate the auto-correlation matrix, a 2x2 matix in which Ixx and Ixy constitute the first row whereas Ixy and Iyy constitute the second row.

Auto-correlation matrix:

$$A = \begin{bmatrix} I_{xx} & I_xI_y \\ I_xI_y & I_{yy} \end{bmatrix}$$

Next, eigen values of the auto-correlation matrix are computed using the function eigens() under the numpy library. A local maximum of the eigen values is computed and every other eigen value less than the local maximum is discarded i.e. made zero. The eigen values are used to determine the points in the first frame which can then be used for tracking in the subsequent frames. The determined points are sorted and the first k maximum points are considered. k value was chosen as 15. These points will be located on the object in the frame. Some might be located on the edges while the others may be somewhere on the object's surface.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \cdots & a_{kk} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{bmatrix},$$

where lambda is the eigen value

The second stage is the tracking phase. Each point determined in the first stage, is considered in a window within a window. Then, the summed square difference of the point is computed using the following equation:

$$\sum_i [\, I_1(x_i + u) - I_1(x_i)]^2$$

The point is updated with the least summed square difference which becomes the new point in the next frame. After a certain number of frames, some of the points may coincide with each other thereby reducing the number of tracking features in the frame. In such cases, a threshold is considered. If the number of points drops below the threshold, the duplicates are removed and new points are added by repeating the first stage.

III. Results

The algorithm was tested on two sample video inputs. The video lengths are quite small consisting of less than 200 frames. For the gaussian function, the sigma value was chosen to be 1. Initially, the number of points was assumed to be 15 in the first frame. For the summed square difference calculation, the size of the outer window and inner window was taken as 3x3 and 17x17 respectively. Subsequently when the number of points drop below 10, new points were added. The results are shown in Figures 1,2,3 and 4.

IV. Discussions and Conclusion

The first stage is the most difficult and crucial stages of the two. Determining the points in every frame individually is cumbersome and inefficient. Hence, in this algorithm, points are determined only in the first frame and those points are tracked in the subsequent frames. As the points overlap after a certain number of frames, new points are added corresponding to a similar intensity which consumes a little more time. The complexity of the algorithm depends on the window sizes i.e., while calculating the eigen values and the summed square difference.

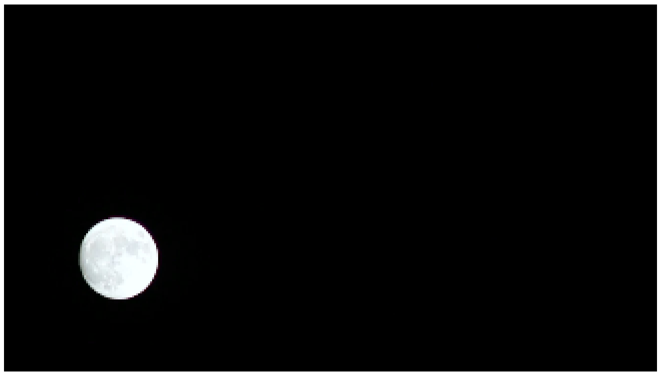

Fig. 1. First frame of the input sequence

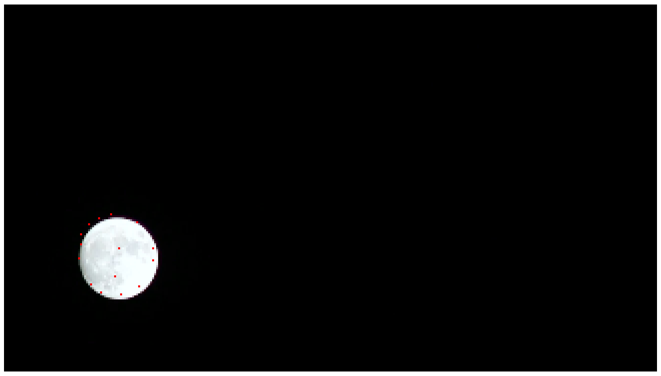

Fig. 2. Output after running the algorithm

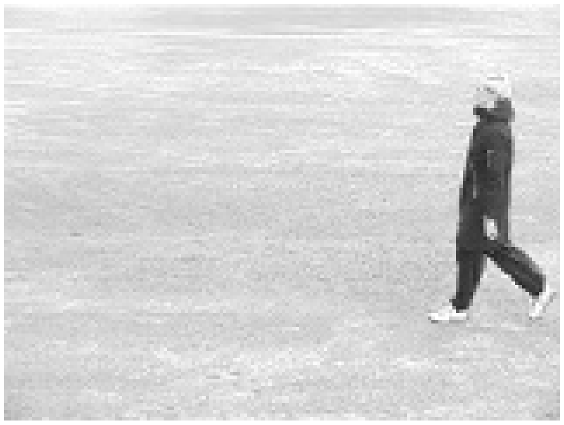

Fig. 3. First frame of the input sequence

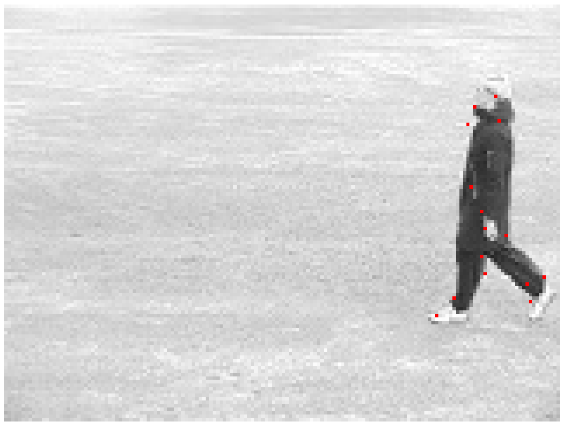

Fig. 4. Output after running the algorithm