LUCAS KANADE TRACKER

IMPLEMENTATION:

1.OWN IMPLEMETATION USING GAUSS NEWTON

First, I initialize the parameters for Shi Tomasi corner detection. Then I took first frame and found corners in it. Then I created mask data structure to create a mask image for drawing purposes. Then I took next frame and found gradient using CV2.sobel() function for future purposes. Then I made a function get_New_Coordinate() to get new position of the pixel in next frame.
Within the function: First I created data structure T which contains original frame data. Then I implemented the algorithm which was described in Simon Baker Paper.

Algorithm:

Pre-compute:

Evaluate the gradient ∇T and the second derivatives $\frac{\partial^2 T}{\partial \mathbf{x}^2}$
Evaluate the Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ and the Hessian $\frac{\partial^2 \mathbf{W}}{\partial \mathbf{p}^2}$ at $(\mathbf{x}; \mathbf{0})$
Compute $\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$, $[\frac{\partial \mathbf{W}}{\partial \mathbf{p}}]^{\mathrm{T}}[\frac{\partial^2 T}{\partial \mathbf{x}^2}][\frac{\partial \mathbf{W}}{\partial \mathbf{p}}] + \nabla T[\frac{\partial^2 \mathbf{W}}{\partial \mathbf{p}^2}]$, and $[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}]^{\mathrm{T}}[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}]$

Iterate:

Warp I with $\mathbf{W}(\mathbf{x};\mathbf{p})$ to compute $I(\mathbf{W}(\mathbf{x};\mathbf{p}))$
Compute the error image $I(\mathbf{W}(\mathbf{x};\mathbf{p})) - T(\mathbf{x})$
Compute the Hessian matrix $\sum_{\mathbf{x}} \frac{\partial^2 G}{\partial \mathbf{p}^2}$ using Equation (82)
Compute $[\sum_{\mathbf{x}} \frac{\partial G}{\partial \mathbf{p}}]^{\mathrm{T}} = \sum_{\mathbf{x}} [\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}]^{\mathrm{T}} [T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]$
Compute $\Delta \mathbf{p}$ using Equation (85)
Update the warp $\mathbf{W}(\mathbf{x};\mathbf{p}) \leftarrow \mathbf{W}(\mathbf{x};\mathbf{p}) \circ \mathbf{W}(\mathbf{x};\Delta \mathbf{p})^{-1}$

Until $\|\Delta \mathbf{p}\| \leq \epsilon$.

Here Equation (82) is:

$$\frac{\partial^2 G}{\partial \mathbf{p}^2} = \left(\left[\frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^{\mathrm{T}} \left[\frac{\partial^2 T}{\partial \mathbf{x}^2} \right] \left[\frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right] + \nabla T \left[\frac{\partial^2 \mathbf{W}}{\partial \mathbf{p}^2} \right] \right) [T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))] + \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^{\mathrm{T}} \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]$$

And Equation (85) is:

$$\Delta \mathbf{p} = - \left[\sum_{\mathbf{x}} \frac{\partial^2 G}{\partial \mathbf{p}^2} \right]^{-1} \left[\sum_{\mathbf{x}} \frac{\partial G}{\partial \mathbf{p}} \right]^{\mathrm{T}}$$

I have made Warp_Inverse() function to get inverse of W(x;p). After converging the above iteration, I get my new coordinate of original points in next frame by using W(x;p). Thus, after getting the points fuction returns it and now I have coordinate of features in current frame and in next frame, so I joined them using a line to track the object path.
Now after that, new features is again extracted using Shi Tomasi Corner detection and it again goes back to loop and repeats the process.

2.USING OPENCV LIBRARY

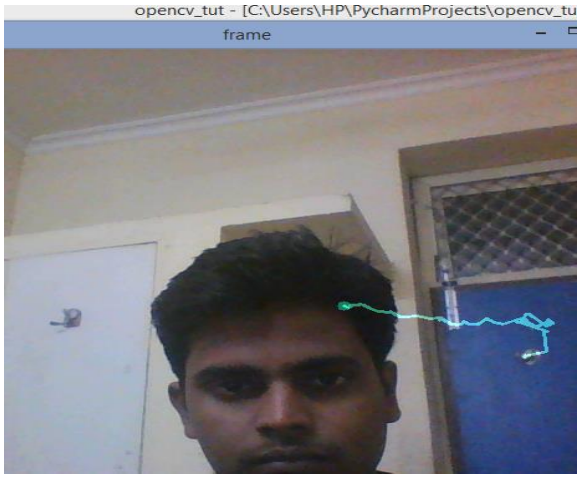
This is done using the inbuilt library function for Lucas Kanade tracking. First I used function for feature detection that is: cv2.goodFeaturesToTrack(). Then I used the inbuilt function for getting the new coordinate that is: cv2.calcOpticalFlowpyrLK().
Now after getting the coordinate in both current and next frame. I join them using the line and hence the objects.

EXPERIMENTAL ANALYSIS:

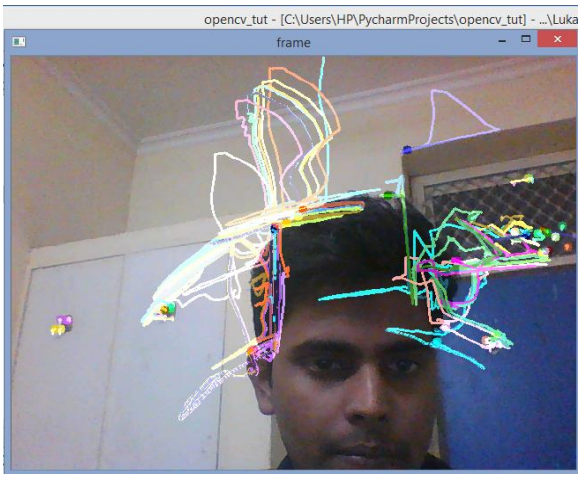
I have experimented on both the implementation and got following results:

Video1:

OWN IMPLEMENTED ALGORITHM



USING OPENCV



Video2:

OWN IMPEMENTED ALGORITHM

USING OPENCV