ENPM 673 – P4

Name: Hrishikesh Tawade Name: Krishna Bhatu Name: Kapil Rawal

UID: 116078092 UID: 116304764 UID:116133357

1) Lucas Kanade Template Tracker

1.1) Implementation of the tracker

- To initialize the template, we have used the mouse callback function wherein we can draw a box around the object of interest. The callback function gives the start and end coordinate of the rectangle.
- We crop the image based on these coordinates from the first frame and use it as the template.

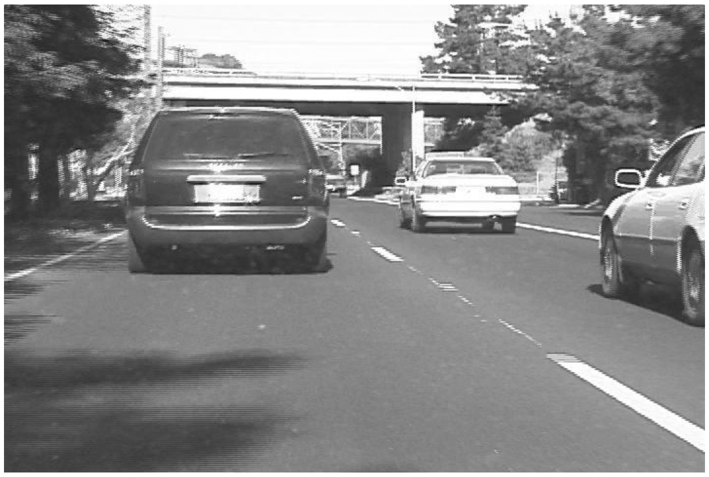
Figure 1 Input Frame

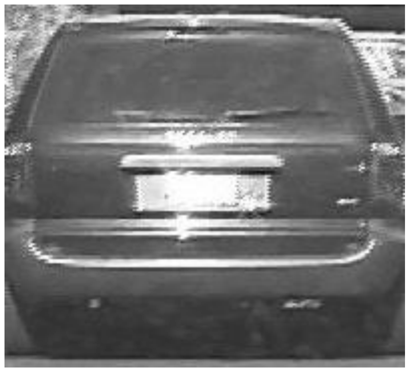
Figure 2 Template

- We have then read all the images from the corresponding folder i.e. car or human or vase using glob function.
- We then store all the coordinates of the template in a matrix also store its intensities separately.
- Further, we start reading all the frames from the folder and scale its intensities based on the formula given in section 2.
- Then we find the x and y gradient of the image using Sobel filter of kernel size 3.
- We then calculate the affine matrix from the all the p values and warp the current image, and the gradients using it and calculate the error between the warped current image and the template.
- The matrix implementation we did earlier helps in expediating the process of warping.
- We then find the Jacobian and multiply it with the gradients obtained above. We then calculate the steepest descent images and the hessian matrix using matrices which expediates the process.
- Then we find delta p using the formula given in the pdf and find its norm using the "np.linalg.norm" function.
- In case of car if it is less than or equal to 0.035 then the we assume that the delta p has converged and move to next frame of repeat the process from affine matrix calculation.
- If the iterations exceed 150 and the delta p has still not converged, then we simply assume that the delta p won't converge and then use the last converged delta p and p matrix for further frame.
- We then find the bounding boxes of the tracking box using the 4 corners of the template.
- Following is the output of our LK algorithm tracking the objects. The red box is the final tracked box and the yellow box which changes is the combinations the algorithm tries to track the car.
- Similar process is repeated for vase and human with different thresholds and iterations.

ENPM 673 – P4

Name: Hrishikesh Tawade Name: Krishna Bhatu Name: Kapil Rawal

UID: 116078092 UID: 116304764 UID:116133357

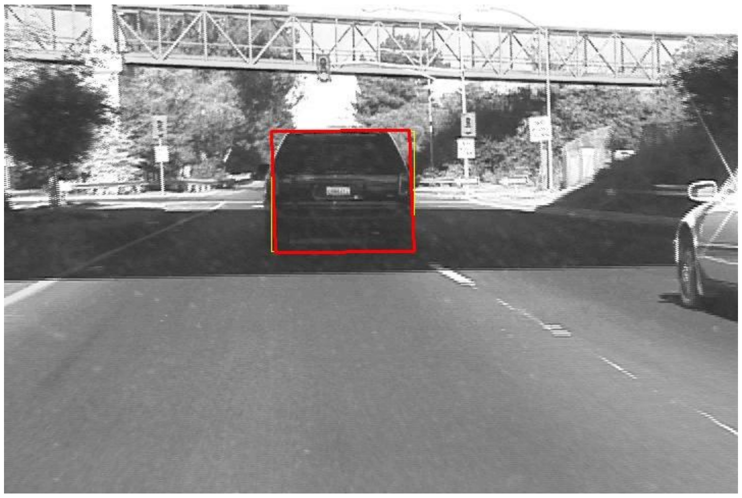

Figure 3 Tracking car

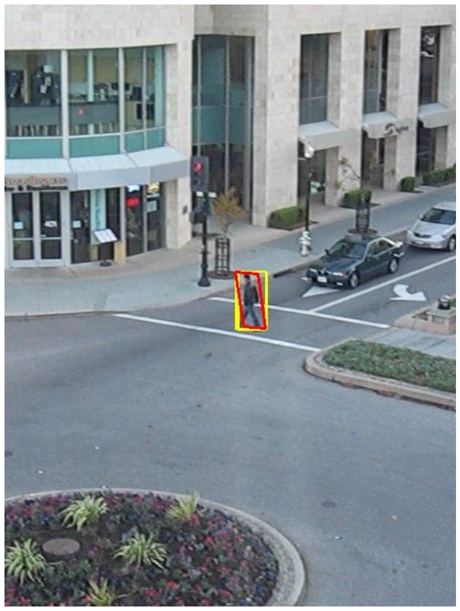

Figure 4 Tracking Person

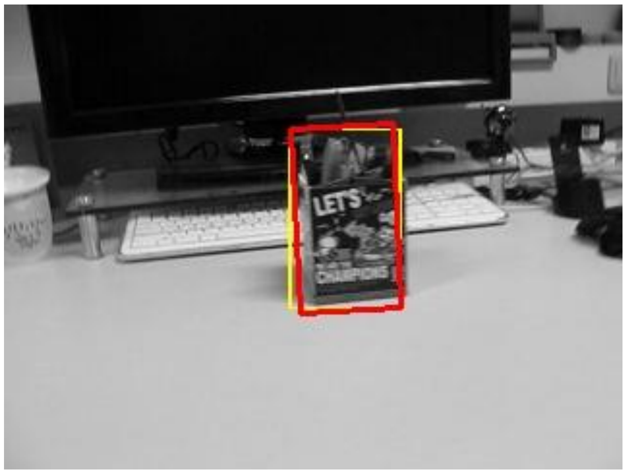

Figure 5 Tracking Vase

ENPM 673 – P4

Name: Hrishikesh Tawade Name: Krishna Bhatu Name: Kapil Rawal

UID: 116078092 UID: 116304764 UID:116133357

- In the data frames given in the vase detection, the change of the object position in between two consecutive frames is very large. This cannot be compensated by the basic lucas-kanade algorithm. Therefore, after some frames, the algorithm does not converge to the minimum threshold. Thus, we have submitted the video submission till that frame.
- Solution to this problem would be to use the pyramid method with the existing algorithm. By using down pyramid, we can reduce the resolution of the image and the change in position of the object to be tracked will be less at the top layer of the pyramid than the bottom layer. By computing the change in parameter thorough all such layers we can compensate the larger change in motion and detect the object.

1.2) Evaluation of the Tracker

- During our implementation we found that for car and person we need tight bounding boxes because these objects are moving, and the changes in background creates noise. Hence, we want the least portion of the background.
- In the case of vase since the object itself is not moving and the surrounding are pretty like the vase, thus, this strategy fails because the algorithm can't distinguish well between the background and the object. Hence, in this case we select a bigger template so that it includes the background, which the algorithm can then use to detect changes.
- The tracker fails when the object is moving at high speeds or there is change in illumination. In case of fast-moving objects, the optical flow vector becomes too large in subsequent frames and hence the LK tracker can't track it. In case of illumination changes the algorithm fails because the sum of squared distances error that it tries to minimize is sensitive to illumination changes.

2) Robustness to Illumination

2.1) Scaling of Intensities

- To make our algorithm robust to illumination, we have calculated the mean of the template frame and the current frame.
- We then use the following formula to get the intensities of the current frame.

Formula:

$$\boldsymbol{frame_{current}} = \frac{\boldsymbol{mean\ of\ template\ intensities}}{\boldsymbol{mean\ of\ the\ current\ frame\ intensities}} * \boldsymbol{frame_{current}}$$

- In this case we thus scale the intensities of the current frame according to the template frame then pass to LK and the LK can now track the template.
- Following is the output of this technique.

2.2) Weighted Error

- Here we will discuss the implementation of the second of illumination robustness.
- Instead of giving same weightage to all the errors we gave weights to the errors by which the areas with more error will be penalized more.
- And thus, the effect of outliers in the output will greatly reduce.
- We calculated the mean and variance of error and then if its in one standard deviation then it was given a higher value and lower value was given to the outliers.
- Following is the output of this technique.

Formula:

$$\boldsymbol{\nabla p} = (\boldsymbol{A^T \wedge A})^{-1} \boldsymbol{A^T \wedge b}$$

ENPM 673 – P4

Name: Hrishikesh Tawade Name: Krishna Bhatu Name: Kapil Rawal

UID: 116078092 UID: 116304764 UID:116133357

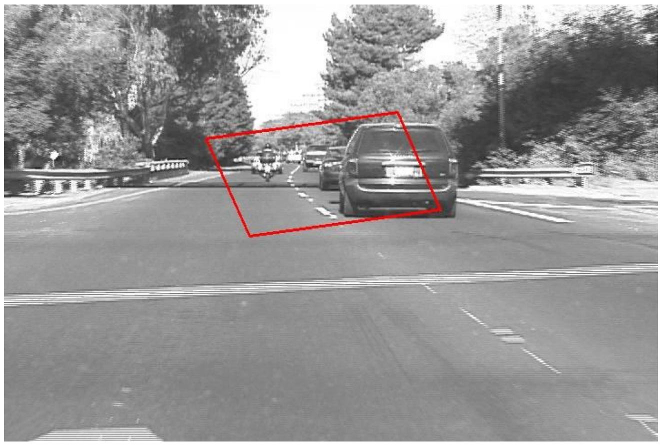

Figure 6 Tracking before illumination correction

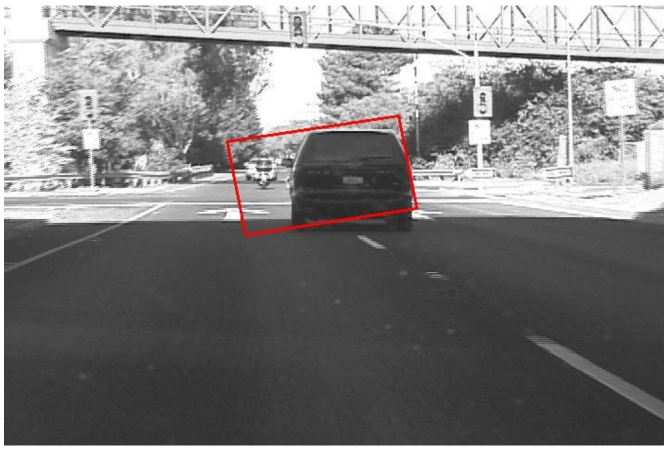

Figure 7 Tracking before illumination correction

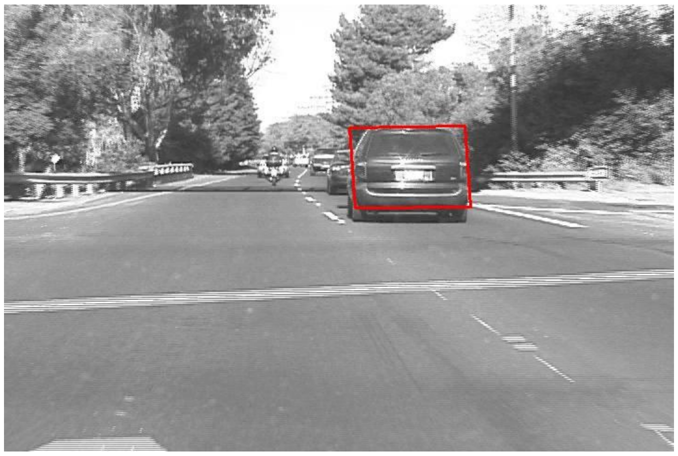

Figure 8 Tracking after illumination correction using scaling

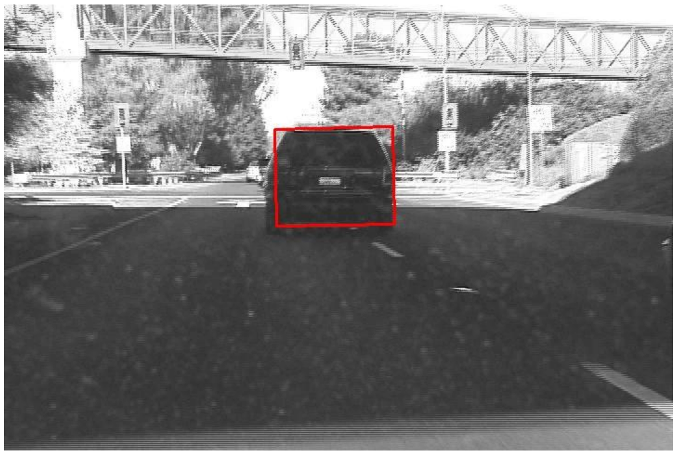

Figure 9 Tracking after illumination correction using scaling

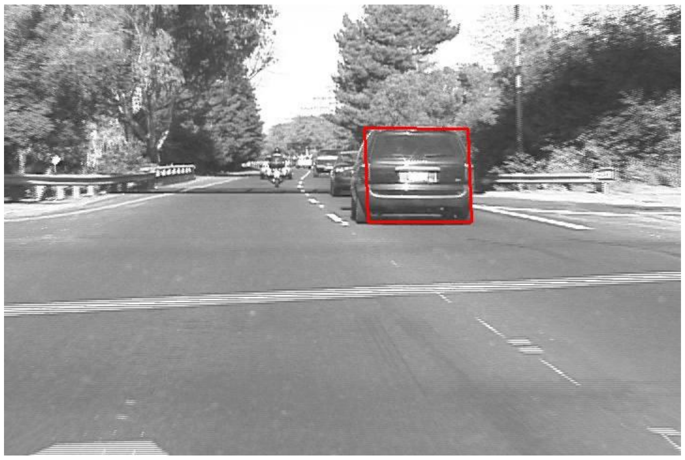

Figure 10 Tracking using the weighted error method

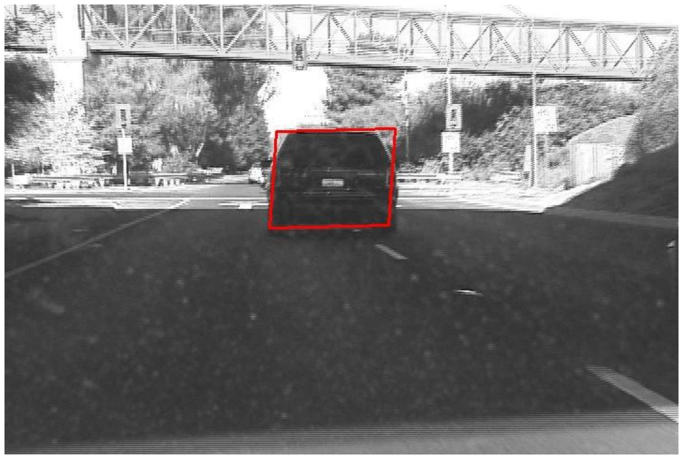

Figure 11 Tracking using the weighted error method

Link to Video and Data: LINK