ENPM673
Project 4
Perception for Autonomous Robots

Project Group -

Amrish Baskaran (116301046)
Arpit Maclay (116314992)
Bala Murali Manoghar Sai Sudhakar (116150712)

Implementation of Lucas Kanade Algorithm for template tracking:

The Lucas-Kanade optical flow algorithm is a simple technique which can
provide an estimate of the movement of interesting features in successive
images of a scene. One of the assumptions of Lucas Kanade is the two images are separated by a small-time increment Δt, in such a way that objects have not displaced significantly.
We start by converting the image into a grayscale image to make it easier to detect edges and features of the image.

Methodology:

Consider an Image I(x,y). For a very small displacement, the new image can be represented by H(x,y) = I(x+u, y+v).
Here (u,v) represents the displacement of the pixel.
To get the u and v, we will solve the following equation.

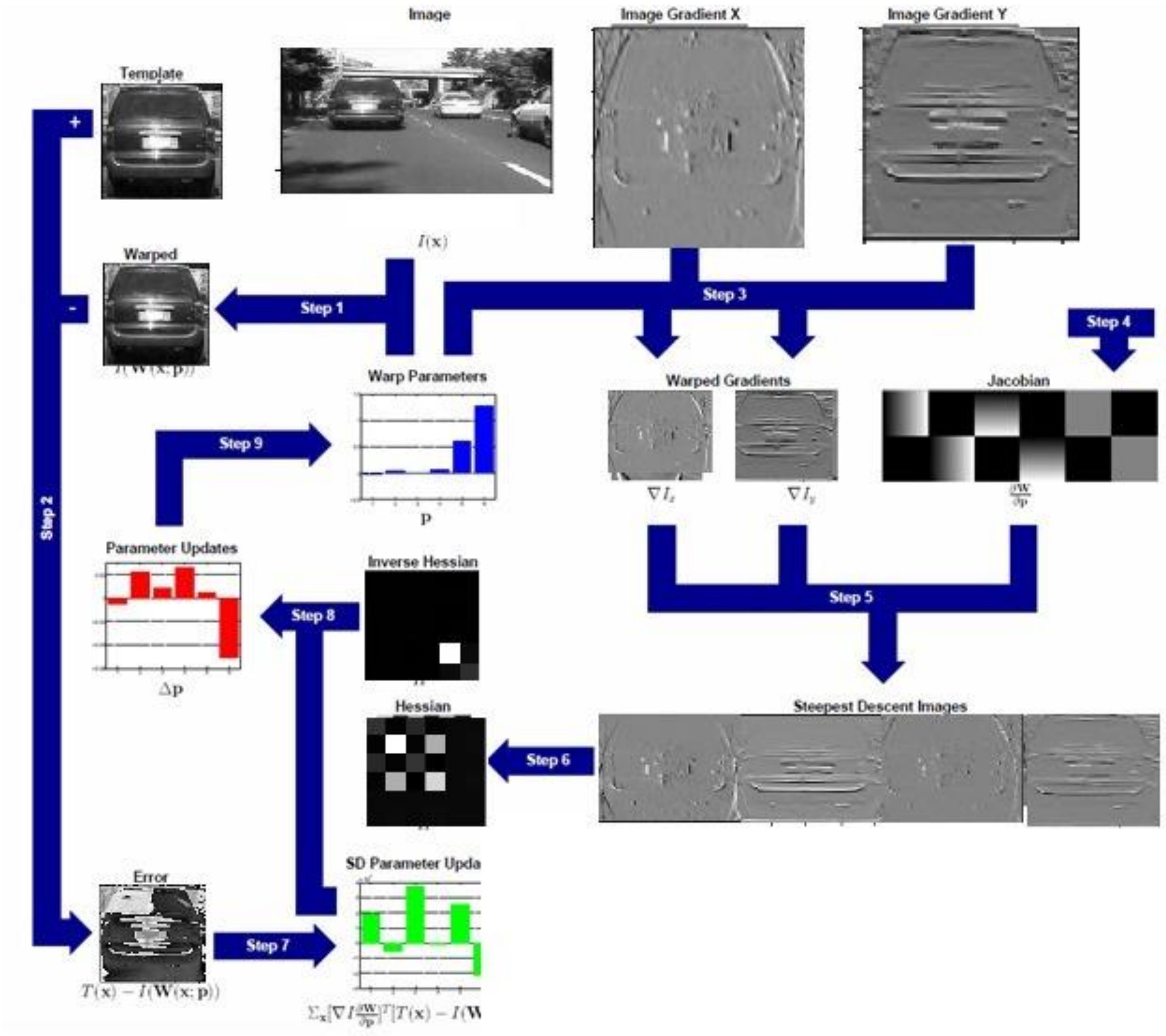
$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

A B

The summation is of all pixels in a KxK window.
This equation is solvable only when –

1.) A is invertible.
2.) Eigen values should not be too small.
3.) A should be well conditioned i.e. eigen values should not be too large.

We implement the Lucas Kanade algorithm by following steps-



Car Data Set LK Flow

Lucas Kanade Steps:

Step1: Warp I with W(x;p) to compute I(W(x;p))

Step 2: Computing the error images T(x) – I(W(x;p))

Step 3: Warping the gradient ∇I with W(x;p)

Step 4: Evaluate the Jacobian $\frac{\partial W}{\partial p}$ at (x;p)

Step 5: Compute the steepest descent images $\nabla I \frac{\partial W}{\partial p}$

Step 6: Compute hessian matrx

Hessian matrix = H = $\sum_x [\nabla I \frac{\partial W}{\partial p}]^T [\nabla I \frac{\partial W}{\partial p}]$

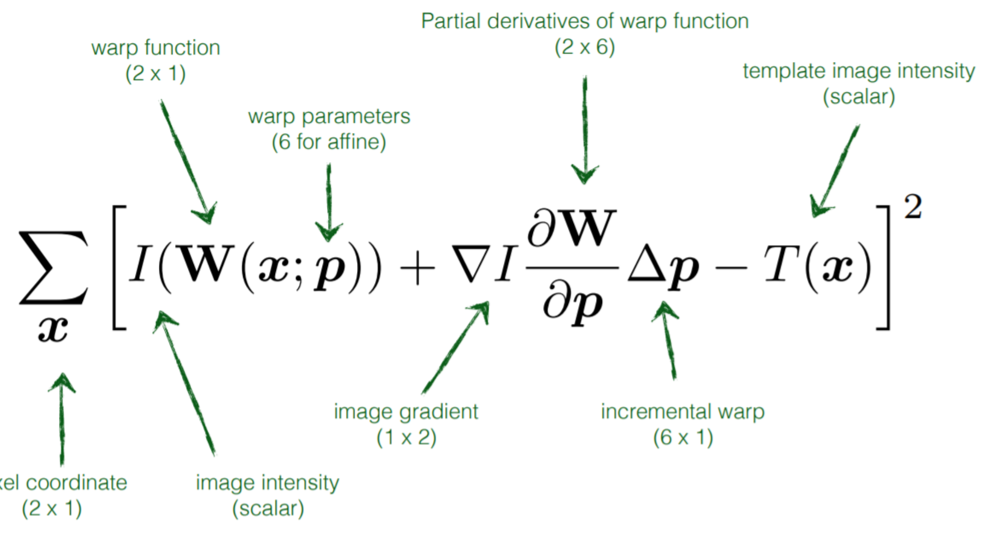
Step 7: Compute $\sum_x \left[\nabla I \frac{\partial W}{\partial p}\right]^T \left[T(x) - I\left(W(x;p)\right)\right]$

Step 8: Compute Δp using the equation below:

$$\Delta p \ = \ H^{-1} \sum_x \left[\nabla I \frac{\partial W}{\partial p}\right]^T \left[T(x) - I\left(W(x;p)\right)\right]$$

Step 9: Update parameters p → p + Δp

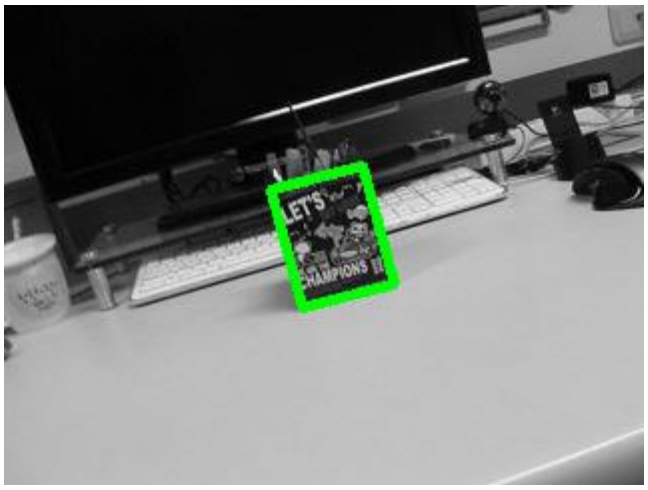
Where,

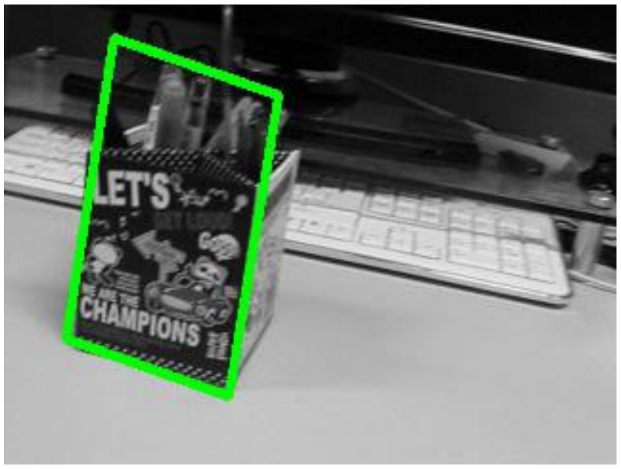

Partial derivatives of warp function (2 x 6)
warp function (2 x 1)
template image intensity (scalar)
warp parameters (6 for affine)
$\sum_{x} \left[I(\mathbf{W}(x;p)) + \nabla I \frac{\partial \mathbf{W}}{\partial p} \Delta p - T(x) \right]^2$
image gradient (1 x 2)
incremental warp (6 x 1)
pixel coordinate (2 x 1)
image intensity (scalar)

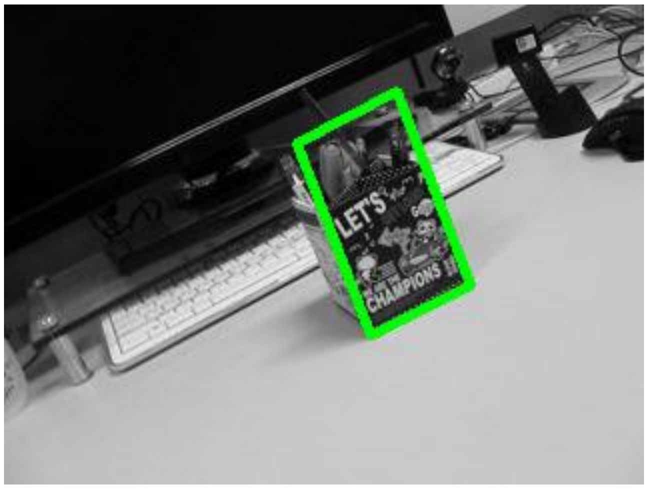
Lukas Kanade may fail in the following cases

1.) Brightness consistency is not satisfied.
2.) The motion is not small.
3.) A point does not move like its neighbors.

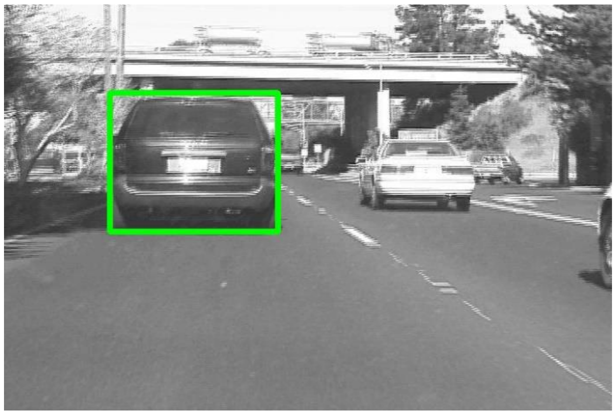
Output for Template Tracking on Vase:

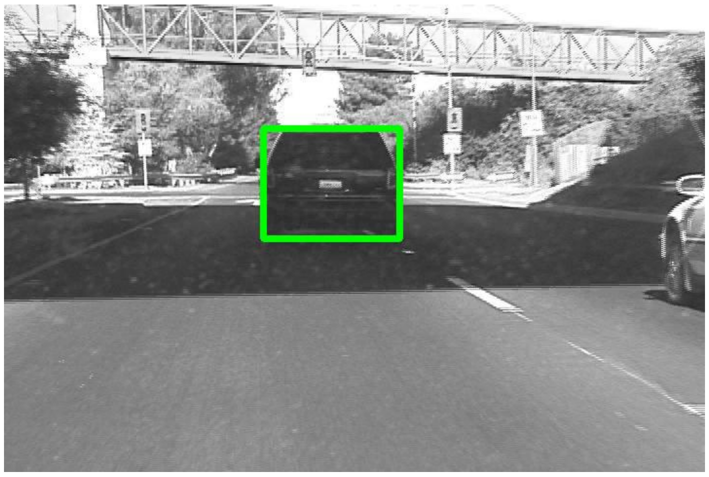




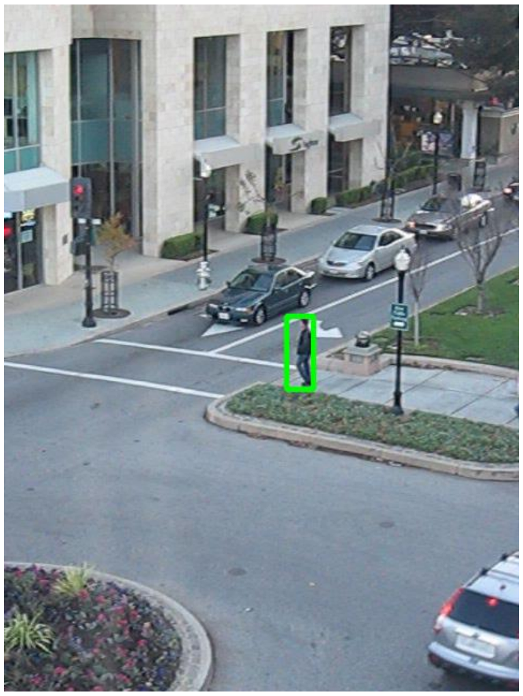


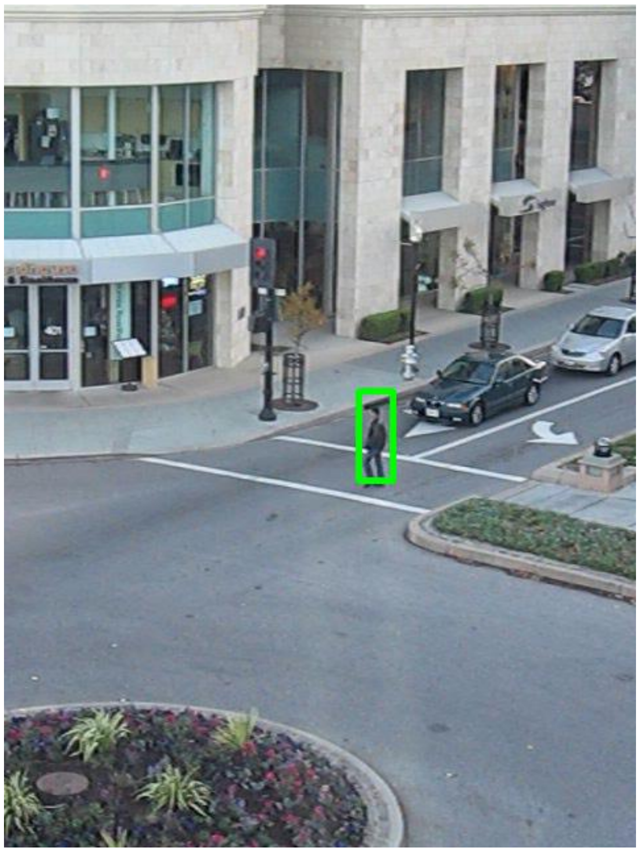
Output for Template Tracking on Car:

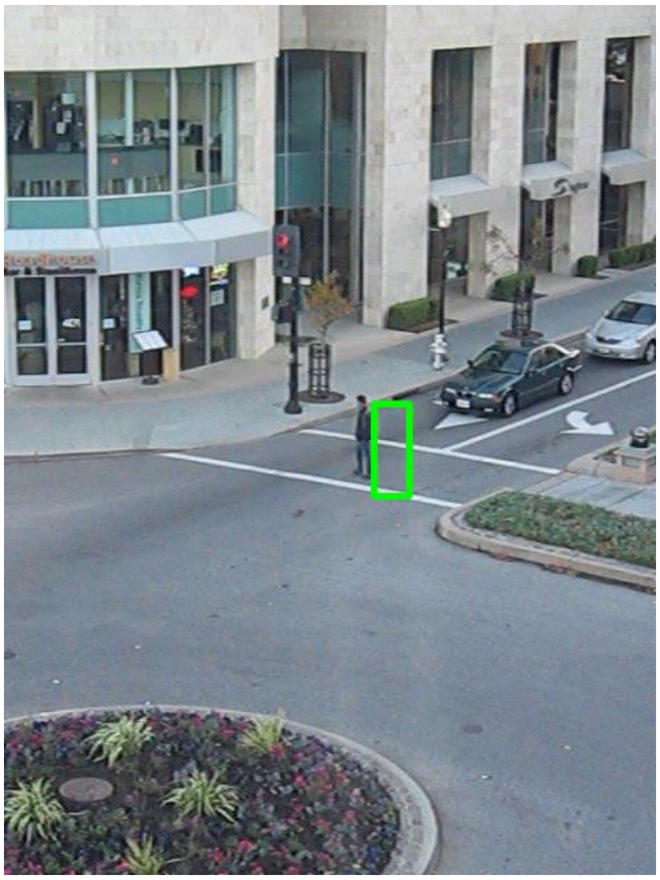




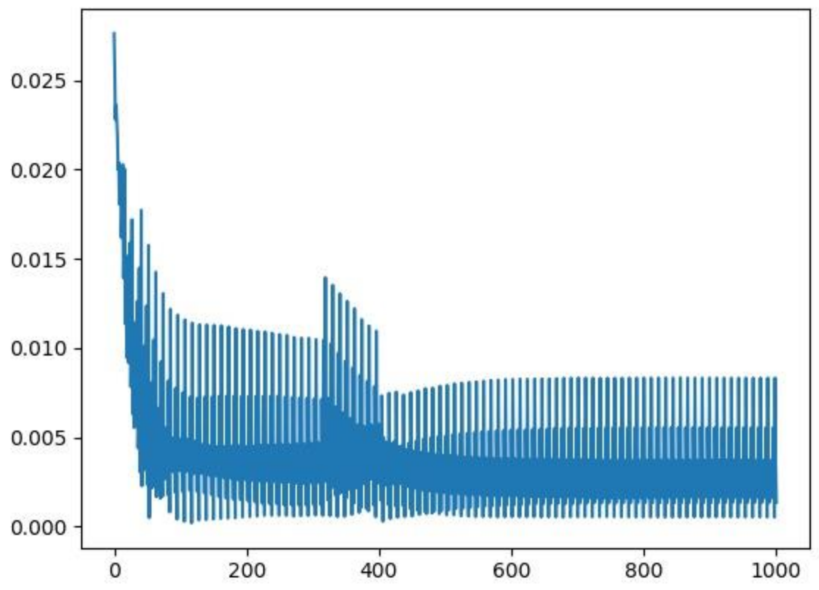
Output for Template Tracking on Human:

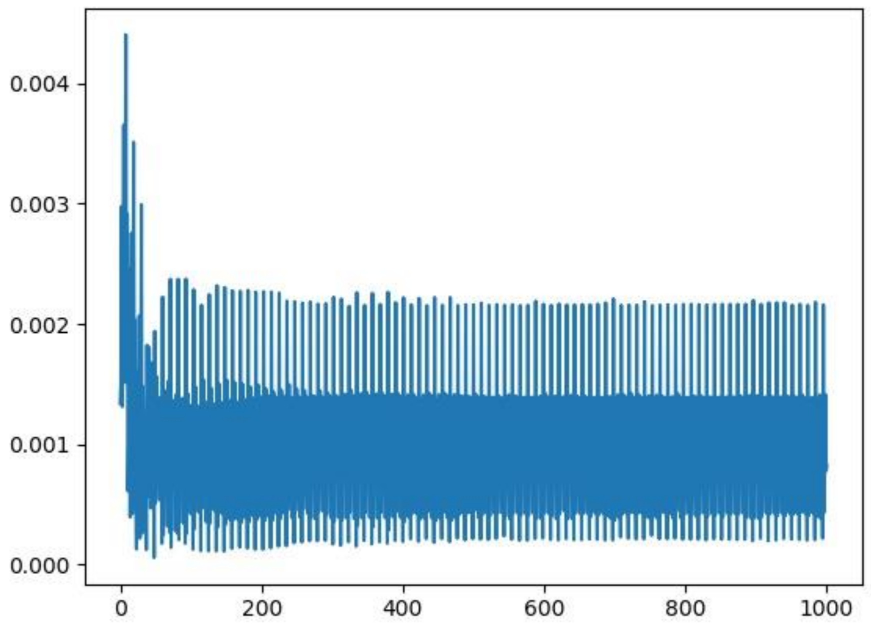




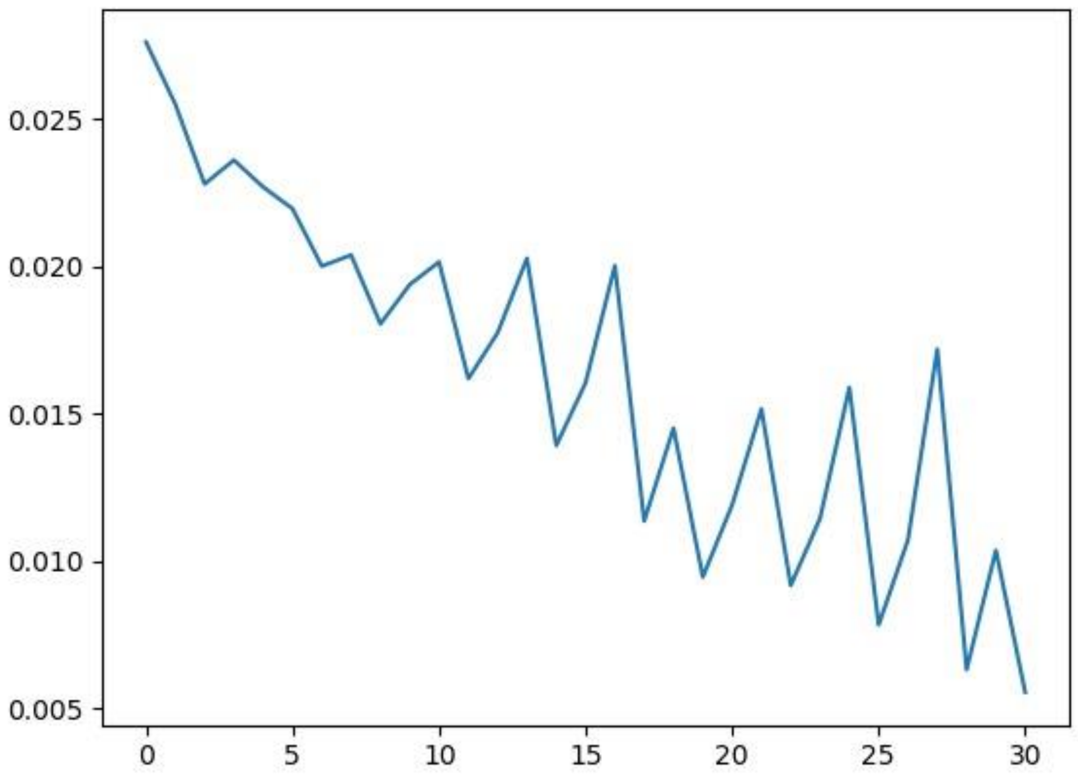


Error plot:


0.025
0.020
0.015
0.010
0.005
0.000
0
200
400
600
800
1000


0.004
0.003
0.002
0.001
0.000
0
200
400
600
800
1000

Norm of delta p Plot:



Parameters that affect the performance Lucas Kanade algorithm (LK algorithm):

Template:

The template should represent the entire dataset over which the tracking has to be done. Thus, resistance to illumination or brightness can be encoded in the template itself. The template should contain enough features so that the image gradients can be matched with the template. Also, the change between template and the object frames must be incremental since we do TAYLOR SERIS approximation to make the problem LINEAR and minimize the error. There is a necessity that the warp function used must be CONTINOUS since we calculate the JACOBIANS. Affine transformation is one such warp that satisfies the conditions of Lucas Kanade algorithm.

Blur:

When there is noise, the gradients are disrupted in all the directions. Since the LK algorithm primarily works based on Jacobians and image derivatives, noise reduction in necessary component in pipeline. The basic noise reduction technique is gaussian blur and it is faster. This will smoothen the edges which is not a problem since blur gives us thicker and smoother gradients which are the features on which LK algorithm works.

Countering Intensity Variation:

This is a major issue when we have a fixed template for the entire tracking dataset and the dataset has good variation in brightness thus affecting intensity. This is a major concern since we work on intensity space in LK algorithm. To counter this effect one solution is to do brightness correction such as histogram equalization or gamma correction. GAMMA CORRECTION is a better option since it gives us more control to increase or decrease the brightness in the image.

Pipeline corrections:

Affine transform vs Inverse affine for LK:

Initially when we developed LK algorithm, we use normal affine transform. This approach was not converging, and the error was constantly increasing irrespective of learning rate. Also, the standard affine transformation that comes with open CV cannot be applied since there few cases which need to be handled explicitly. So, we used inverse affine transform from template to image and use it instead. This allowed is to use the standard affine transformation of open CV package. Thus, the LK algorithm converges to the required tolerance and the algorithm runs much faster.

Broadcasting Advantage:

The hessian calculation is time consuming and must run for evert iteration, the algorithm runs very slow. There are handful of parameters that need to be tweaked for robust tracking which takes long time to tune since the algorithm run slow. The entire algorithm code is rebased to take advantage of BLAS library of python and the algorithm doesn't have any for loops now. We are able to achieve real time tracking speeds when the algorithm converges within 10 iterations. Learning rate is tweaked to achieve the convergence with minimum iterations.

Blur Correction:

The algorithm still converges, and acceptable tracking performance is achieved for Car data base even without noise reduction. But in case of vase and human data set, the performance is not as expected. In our case gaussian blur with kernel size of 3x3 is giving good enough results for the purpose of tracking. Even with car dataset, a tighter boundary is obtained over the entire data set after gaussian blur. One more advantage of choosing gaussian kernel is that it is separable, and kernel can be applied much greater speed than averaging or median filter. Median filter is not needed since we are not using any edge features for tracing.

Sobel filter:

To find the image gradients we make use of sobel filter if size 3x3 in x and y direction separately. We take gradients on the full-size image and then warp to required rectangle. This makes sure that we do not loose the edge data rather than warping the image and taking gradients.

Pyramid correction:

In case of vase data set, we have a dynamic data set that moves faster and jerky. One constraint from the LK algorithm is that there should not be a large deviation between frames. In case of vase data set, this is the problem and the error increases rapidly. Once way to counter this is to reduce the image into smaller pyramids and run LK progressively on each pyramid by using warp parameters from previous layer. A coarse form of pyramid is constructed on the input image. LK Algorithm is first applied on the lowest resolution image and the converged parameters are used as seeding values LK algorithm on the next bigger layer. This greatly improved the accuracy of the vase dataset. The accuracy of tracking also depends on the number of layers. We achieved best results with 3 layers of pyramid. This is because when the pyramid layer is too small, the parameters converge differently and since the same is seeded to further layers, the error increases.

Comments on Tracking in each data set:

Car Data Set

For the car data set a rectangular template with a tight fit is used. This template was a good choice as the car segment is close to a rectangle and the shift between frames is not drastic hence the Lucas Kanade tracker works well.

This data set breaks down when there is an illumination change right under the bridge. Lucas Kanade works on the assumption that there is no change in illumination and a good estimate of P is available, both of which are lost.

Vase Data Set:

For the vase data set there is both panning and rotation of the camera. The tracker breaks down due to these reasons. For such a motion we are taking an expanded rectangular template around the vase. It breaks down when the image croses the sides of the cropped image. Hence a larger template is taken to combat this.

Human Data Set:

For the human data set a rectangular template is used. Due to the area to be tracked varying in shape and having a small size it is harder to track. The variation between frames is also high. The camera panning is also erratic leading to a large shift in P. Hence the tracker breaks down.

Video Links:

1. Car Data Set - https://drive.google.com/file/d/110Ev2COEsHGR0fv94A9QQIAEsZDkFK2z/view?usp=sharing
2. Vase Data Set - https://drive.google.com/open?id=1jaQq3-PnMC1DVUtk9P4he5w5H8I2WFZ8
3. Human Data Set - https://drive.google.com/file/d/1W7Ib1NZPmEEoU-iRZPubeyeK_Edf_3-C/view?usp=sharing

References:

1.) OpenCV Gamma Correction. (2018, August 02). Retrieved from https://www.pyimagesearch.com/2015/10/05/opencv-gamma-correction/

2.) Baker, S., & Matthews, I. (2004). Lucas-Kanade 20 Years On: A Unifying Framework. International Journal of Computer Vision, 56(3), 221-255. doi:10.1023/b:visi.0000011205.11775.fd

3.) [Carnegie Mellon]. (n.d.). Retrieved from http://www.cs.cmu.edu/~16385/s17/Slides/14.4_Alignment__LucasKanade.pdf

4.) Robust template tracking with drift correction. (2007, March 27). Retrieved from https://www.sciencedirect.com/science/article/pii/S0167865507001018

5.) Lucas Kanade affine template tracking - File Exchange - MATLAB Central. (n.d.). Retrieved from https://www.mathworks.com/matlabcentral/fileexchange/24677-lucas-kanade-affine-template-tracking

6.) Image Pyramids¶. (n.d.). Retrieved from https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_pyramids/py_pyramids.html