ENPM673 - Perception for Autonomous Robots
Project 4 - Lucas Kanade Tracking

Abstract
The Lucas Kanade method is mostly widely used in computer vision applications as a differential method for estimating optical flow using brightness constancy. This method works based on the assumption of constant flow in the local neighborhood of the pixel and solving the optical flow equations for all the neighboring pixels using least squares criterion. We intend to leverage this information of several such nearby pixels and resolve the inherent ambiguity of the optical flow equations extending it to tracking.

1. Lucas Kanade Template Tracker

We implement the Lucas Kanade with the inverse compositional algorithm for efficient image alignment and extending it to a consistent framework without significantly better efficiency. The approached we have adhered here is using gradient descent which is the defacto standard and can be performed in a variety of different ways where we estimated using an additive increment to the parameters and another method where we used an incremental warp computed together with the estimated current warp. The Lucas Kanade method assumes that the displacement of the image contents is too small and almost constant with a neighborhood point under consideration. The approach of the Lucas Kanade method is as below.

Starting with an initial Affine transform parameter p we can compute optimal Δp iteratively, where in each iteration the objective function is linearized by 1st order Taylor expansion and solved by a linear system with form AΔp where Δp is the template offset.

$$\sum_{\mathbf{x}} [\, I(\mathbf{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p})) - T(\mathbf{x})\,]^2$$

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

with respect to Δp and update the estimates of the parameters such that

$$\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$$

1.1 Implementation of the Tracker

Our goal is to minimize the sum of squared error Δp between two images, the template T and the image I and Δp = [p0, p1, p2, p3, p4, p5, p6]^T. The following steps are iteratively applied till Δp converge.

Template preparation - T(x): The tracking starts from on initial bounding box in a start frame. Object that has to be tracked is cropped from the first original frame. Then, it is

updated after ROI of the next frame is found. It is later used as the reference to calculate the error image. The corner point of the image is used as reference to calculate the Affline transform Parameters p.

Fig 1. Template of the Car data set

Warped Image: The original frame is warped using affine transform parameter p. The part of the image that comes within the corner points of the step 1 is cropped and used as the warped image. It is later used to calculate the error image.

Fig 2. Warped image

Error Computation: Error is calculated by subtracting the warped image from the template image. It is used to calculate the SD parameters.

$$[T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]$$

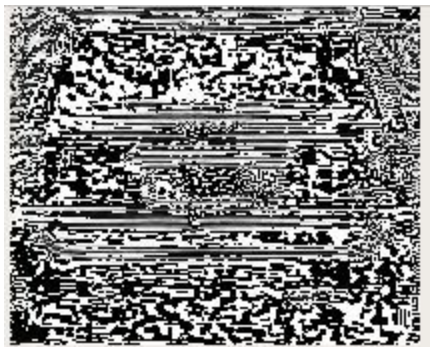

Fig 3.1 Computed Error for car

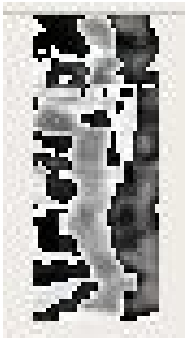

Fig 3.2 Computed Error for human

Fig 3.3 Computed Error for vase

Warped Gradient - I(W(x;p)): Gradient along the x and y of the original frame is calculated using sobel filter. 5x5 kernel is used in sobel filter. It is warped using the affine transform parameter p. The formula is shown below.

$$\mathbf{W}(\mathbf{x};\mathbf{p}) = \begin{pmatrix} (1+p_1)\cdot x & + & p_3 \cdot y & + & p_5 \\ p_2 \cdot x & + & (1+p_4)\cdot y & + & p_6 \end{pmatrix} = \begin{pmatrix} 1+p_1 & p_3 & p_5 \\ p_2 & 1+p_4 & p_6 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

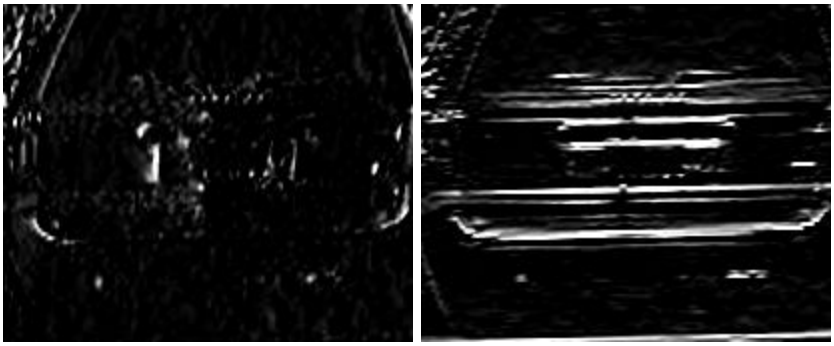

Fig 4. i) Gradient along x. ii) Gradient along y

Jacobian: Based on the number of rows and columns in the template image, a meshgrid is calculated. Jacobian is calculated using the formula below.

$$\mathbf{W}(\mathbf{x};\mathbf{p}) = (W_x(\mathbf{x};\mathbf{p}), W_y(\mathbf{x};\mathbf{p}))^{\mathrm{T}}$$

$$\frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \begin{pmatrix} \frac{\partial W_x}{\partial p_1} & \frac{\partial W_x}{\partial p_2} & \cdots & \frac{\partial W_x}{\partial p_n} \\ \frac{\partial W_y}{\partial p_1} & \frac{\partial W_y}{\partial p_2} & \cdots & \frac{\partial W_y}{\partial p_n} \end{pmatrix}$$

$$\frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \begin{pmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{pmatrix}$$

Steepest Descent Image: The Wrapped gradient is flattened and multiplied to get steepest descent image. It is of order (m*n)x6. It is calculated using the formula below.

$$\left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]$$

Hessian - H: Hessian is calculated by multiplying transpose of the steepest descent and steepest descent. The resultant matrix has the order of 6x6.

$$H = \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^{\mathrm{T}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]$$

Fig 5.1 Hessian for car data set

Fig 5.2 Hessian for human data set

Fig 5.3.Hessian for vase data set

SD Parameter updates: The error image (order (n*m)*1) is flattened. It is multiplied with the transpose of steepest descent images (order 6*(n*m)). The resultant matrix has the order of (6*1).

Parameter update: The inverse of Hessian is multiplied with the resultant matrix to get Δp.

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

Then the Affline transform parameter is is updated with the Δp.

$$\mathbf{p} \leftarrow \mathbf{p} + \Delta \mathbf{p}$$

The steps from 1 to 9 is repeated until norm of Δp is less than ε (User defined threshold)

After the Δp has converged, inverse affine transform is found out for p. It is used to find the transformed corner points. A polygon is fitted on the transformed corner points to track the object.

1.2 Evaluation of the Tracker

The Lucas Kanade method assumes constant flow / pure translation for all the pixels in a larger window which tends to be unreasonable for longer periods of time. Although, we can generalize the Lucas Kanade approach to other parametric models like affine or projective using a warp function. Another technique is to reduce the resolution of the images before applying the Lucas Kanade method.

Fig 6.1 LK Tracker - Car data set

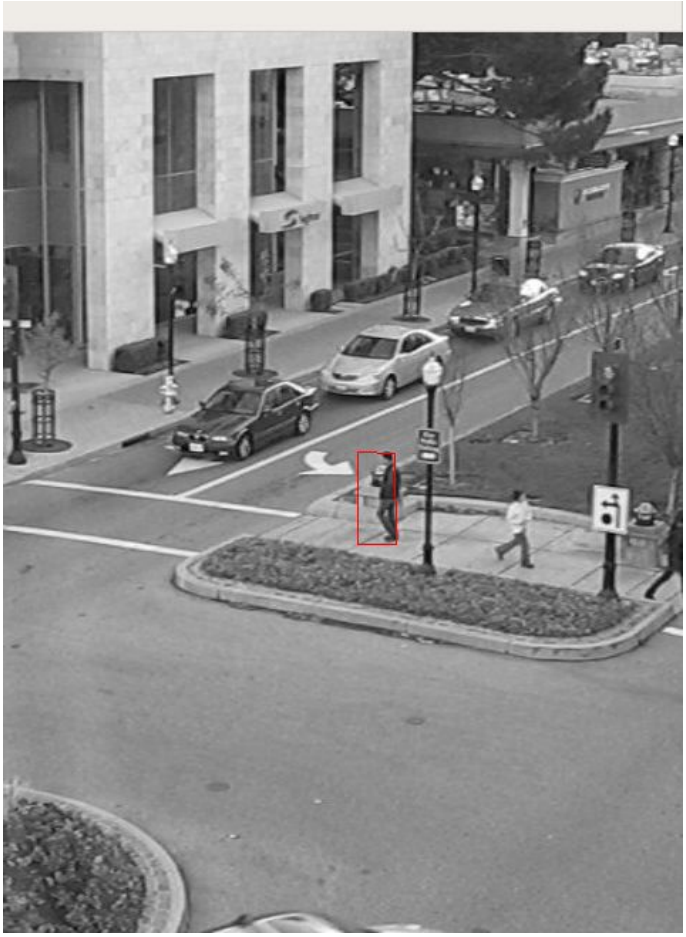

Fig 6.2 LK Tracker - Human data set

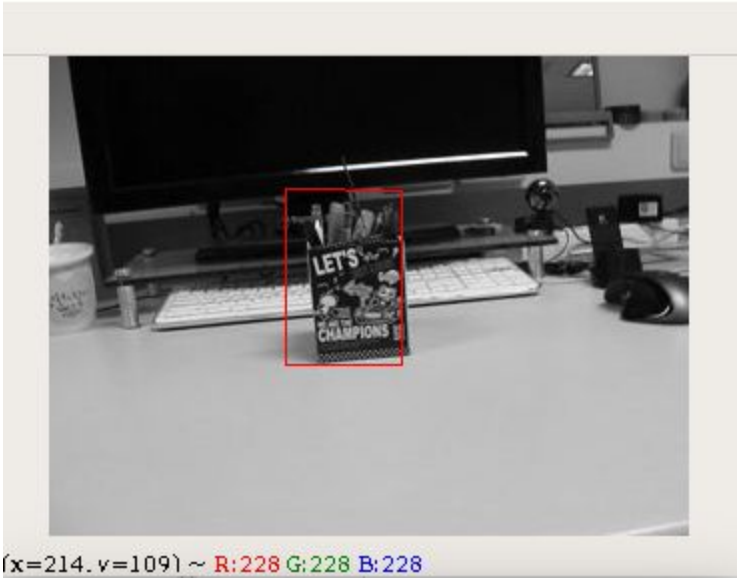

Fig 6.3 LK Tracker - Vase data set

1.3 Convergence

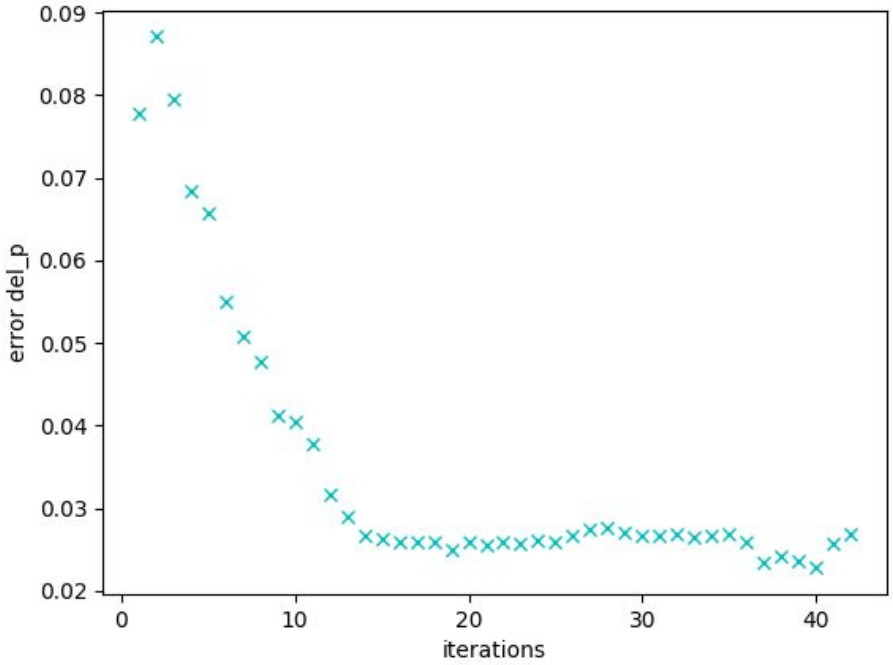

Fig 7.1 Error convergence for human data set

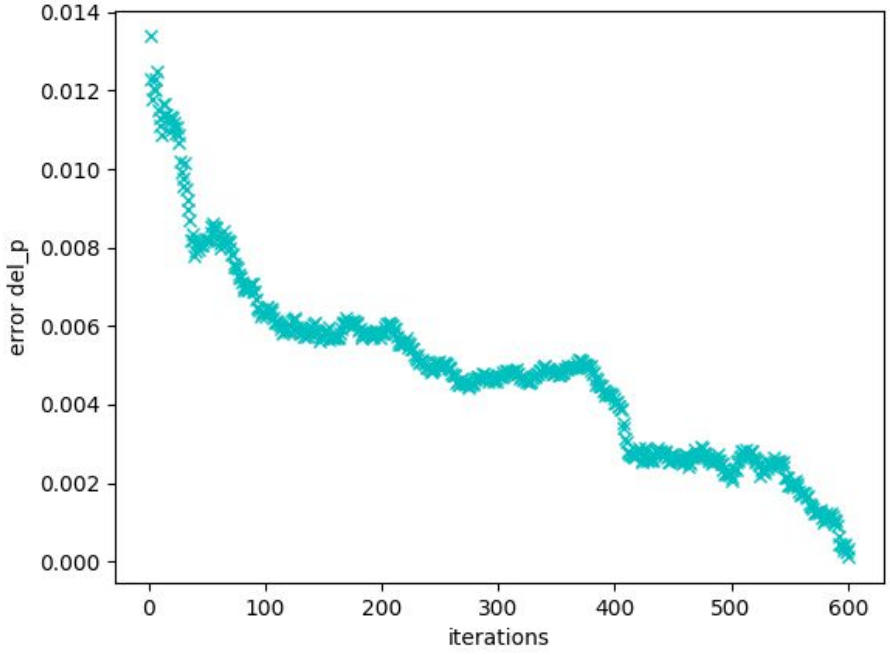

Fig 7.2 Error convergence for car data set

2. Robustness to Illumination

The least-squares approach of the Lucas Kanade tracker assumes that the errors obtained in the image data have a Gaussian with mean zero. Although, if the window contains outliers, we opt to statistical analysis to detect them and reducing their weights respectively. Also, the Lucas Kanade tracker does not hold good when there is a change in illumination. This is mainly because the sum of all the squared distance errors we try to minimize have become sensitive to these illumination changes. We implement the following the approaches to handle this scenario.

2.1 Scaling By Average Brightness

The traditional Lucas Kanade tracker is sensitive to brightness changes. We computed the average brightness of the pixels in the template and scale the brightness of the pixels in each frame such that average brightness of the tracked region is the same as the template's average brightness. By doing this, we remove some potential error scenarios for changes in illumination as we first normalized the template and then the tracked window by subtracting off mean and divided by standard deviation. This improves the Lucas Kanade tracker's efficiency in scenarios where there is a sudden illumination change.

2.2 Huber Loss

Huber loss is a better robust M-estimator compared to the least squares approach as it is less sensitive to the outliers in the data than the squared error loss and both represented as a function of y - f(x) is represented below to contrast and compare.

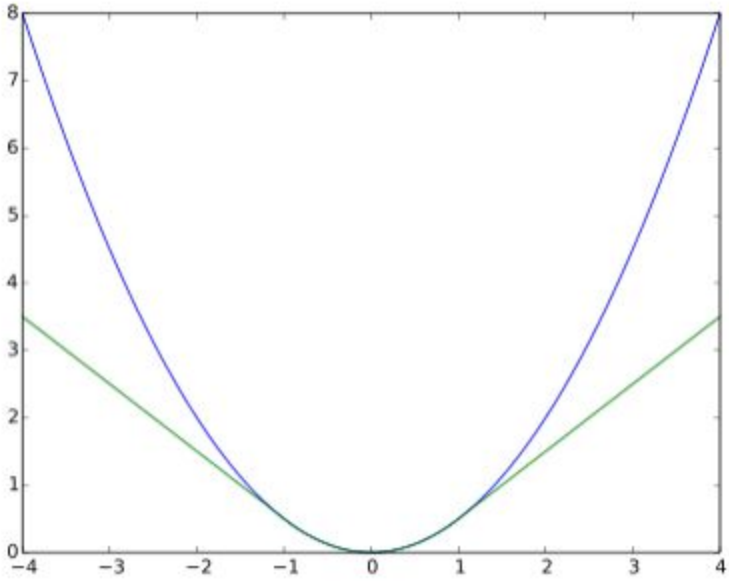

Fig 8. Huber loss vs Least Square Error Loss

Huber loss has a differentiable extension to an affine function and its properties allow it to combine most of the sensitivity and robustness of the M-estimation. Now this becomes a weighted least square problem where each residual term has a corresponding weight and the minimization function is given as follows.

$$A^T \Lambda A \Delta p = A^T \Lambda b$$

$$\Delta p = (A^T \Lambda A)^{-1} A^T \Lambda b$$

Also our Jacobian becomes weighted and Λ is our diagonal matrix of weights for respective residual term for the robust M-estimator.

How to run code

1. Unzip the folder which has the code, input sequences and the datasets.
2. Each of the following code parts needs to be separately run.
3. For lucas kanade tracking run

   python3 Part1.py

Project Team

1. Nantha Kumar Sunder (UID: 116104777)
2. Nithish Kumar (UID: 116316958)
3. Rama Prashanth (UID: 116428941)

References

1. Computer Vision, A Modern Approach, Forsyth and Ponce (http://cmuems.com/excap/readings/forsyth-ponce-computer-vision-a-modern-approach.pdf)
2. https://www.ri.cmu.edu/pub_files/pub3/baker_simon_2002_3/baker_simon_2002_3.pdf

Output Video Link:

https://drive.google.com/open?id=1RBBSpxIfen3_MmVeiPm1XsP7TesmA4hd