ENPM673: Perception for Autonomous Robots

Project 4

Umang Rastogi UID: 116773232

Sayani Roy UID: 116818766

Prateek Bhargava UID: 116947992

Date: 20 April 2020

1. Introduction

This project implements the Lucas Kanade tracker on a set of three given video sequences: featuring a car on the road, a baby fighting a dragon, and Usain Bolt's race respectively. The Lucas Kanade tracker is set to find the optical flow of events inside a frame which is initialized by the help of a rectangle bounding the objects to be tracked. The objects to be tracked are collected in a frame with the help of the bounded and adjacent pixels. For each frame the intensity between the pixels of the two frames are compared to find the vector flow or the flow of motion of the objects. Finally, the optical flow equations are used for the pixels and the centered window.

2. Lucas Kanade Tracker

2.1 Implementation of tracker

Implementing the Lucas-Kanade tracker involves finding the motion (u, v) that minimizes the sum-squared error of the brightness constancy equations for each pixel in a window.

Here, u and v are the x and y components of the optical flow, I_1 and I_2 are two images taken at times t=1 and t=2 respectively. Additionally, a threshold τ should be added, where if τ is larger than the smallest eigenvalue of $A^T A$, then the the optical flow at that position should not be computed.

$$A^T A = \begin{pmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_y I_x & \sum I_y^2 \end{pmatrix}$$

- First, we define a function which gets the location for all the images from the stored location and then sorts them in a list.
- The start and end points of the bounding rectangle are defined, then interpolating points from top left to bottom right corner of the bounding box.
- As we have multidimensional data, f(x, y) we only know the values at points (x[i], y[i]) that do not form a regular grid, thus we use the Multivariate data interpolation (grid data) function, to find the mesh.
- The Bi-variate spline function is then used to compute the spline value by passing in the two co-ordinates as the two arguments.
- Then we have used warp W(x,p) which takes values of the pixels in the co-ordinate frame x and then maps the pixels area in the co-ordinate frame I for the template.

$$W(x;p) = \begin{pmatrix} x + p_1 \\ y + p_2 \end{pmatrix}$$

- Then we use Hessian matrix which describes the second order local image intensity variations around the selected voxel.This is used to extract the orthonormal coordinate matrix that is aligned with the second order structure of the image.
- The norm of the error is then calculated using the Hessian matrix and the difference between template and current image. The warping parameters are then changed based on these errors.
- We have taken two convergence criterion. We break out of loop if the norm of the error goes below decided threshold i.e.,0.001 or the number of iterations go over 200.

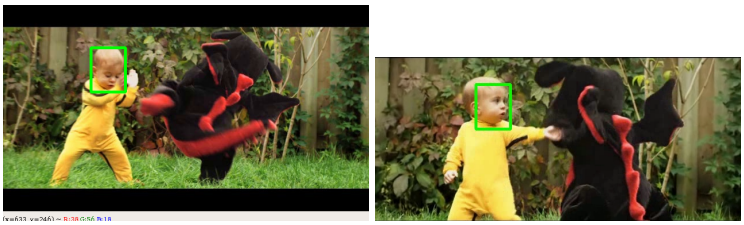
2.2 Evaluation of tracker

The templates for LK tracking for each data-set was chosen from the first frame. A tightly bounded rectangle was drawn on the first frame around the object to be tracked, i.e., the car, the face of the baby and Usain Bolt in their respective video sequences. For each of the subsequent frames, the tracker then updates the affine transform that warps the current frame, so that, the template chosen from the first frame is aligned with the current frame.

(a) (b) (c)

Figure 1: Templates chosen for (a) baby, (b) car and (c) Bolt

The face of the baby was tracked well almost throughout the video. In few of the frames, the face was not tracked properly. This was because the gradient was unable to adapt to the drastic changes in the object to be tracked between consecutive frames.



(a) (b)

Figure 2: Tracking of baby

In the car video, the tracker tracks the car smoothly till the bridge arrives in the video. After the car passes under the bridge, the tracker slowly starts to fail. This is due to the fact the formulation of LK tracker breaks down when there is a change in illumination.

(a) (b)

Figure 3: Tracking of car

In Usain Bolt's video, we can see that Bolt is only tracked for some time with the chosen template. The reason why the tracker fails is that there is sudden change in posture of Bolt which the gradient fails to adapt to. Also, there is no distinguishable feature that the template can use to align itself to the warped image.

(a) (b)

Figure 4: Tracking of Bolt

3. Robustness to illumination

As the tracker failed to track the car after it passes the bridge, few different approaches were taken to rectify that error.

- One of the method applied was Histogram Equalization. Histogram Equalization is applied for darker

images to scale the brightness of pixels. This was performed as it helps in equalizing the darker intensities.

- Another approach was Adaptive brightness method. This method evaluates the difference in pixel values within region of interest between the template and current image. The standard deviation of these pixel values is then calculated. The standard deviation represents the index by which the current pixel values differ. The pixels within the region of interest in the current frame are then shifted by that index.

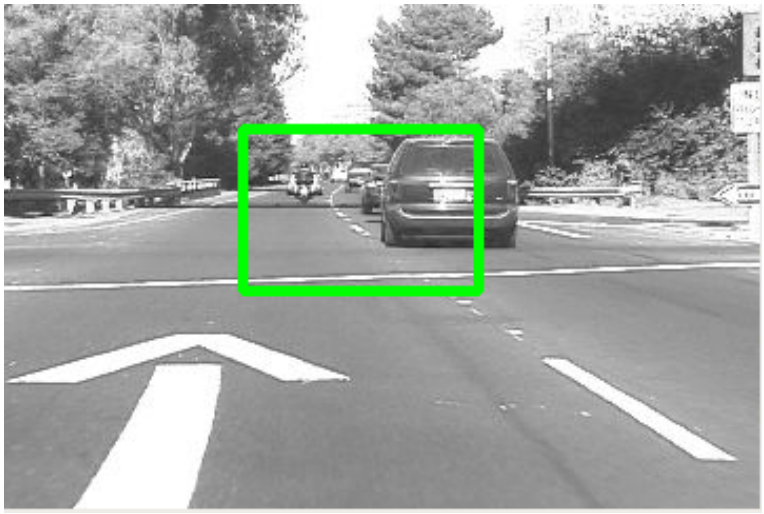


Figure 5: Output after applying Adaptive brightness method

- Pseudo Huber loss was also performed. In the equation given below, 'a' is the difference between the template and current image, 'δ' is the decided Huber threshold which was obtained iteratively to get better results.

$$L_\delta(a) = \delta^2 \left(\sqrt{1 + (a/\delta)^2} - 1 \right)$$

Figure 6: Pseudo Huber loss equation

We observed that when the Huber threshold was below the convergence threshold, the final output was not good as the coordinates of the rectangular tracker became random. Whereas, when the Huber threshold was above the convergence threshold, the output was better. However, in this case, there was no proper tracking after the car passes under the bridge.

Figure 7: Output after applying Huber loss

- Additionally, we performed gamma correction to minimize the dark intensities present in the image. This gave us a much better output, that is, the car was tracked even after it crosses under the bridge.

(a) (b)

Figure 8: Tracking of car after crossing bridge (a) before and (b) after applying Gamma correction

4. Problems encountered

- First, we implemented Sobel and warping functions using affine transformations. But during inverse computation, there were singular matrix error. So, we tried a different approach of using pixel intensities using Bi-variate spline approximation, thus, implementing Lucas Kanade tracking.
- Secondly, the problem to choose the correct number of linearly spaced values for interpolation of intensities along rows and columns of the image. This problem was tackled by using trial and error method to get the values 87 and 36. These values were further ascertained when similar values were found on various posts on stack-overflow and GitHub repositories.
- Finally, as can be seen in the output video of the tracking of the car, the bounding rectangles between the two subsequent frames of the car track slightly away from the car motion. Gamma Correction is applied to rectify this error, resulting in the convergence between the parameters and thus we obtain a lesser error value.
- Gamma correction was also applied to the Bolt video sequence to make the tracking much better than the current. However, still no good results were obtained.

5. Links

This section includes links to the google drive where the video outputs are stored and the GitHub repository.

- Link to folder containing all the output videos: Project outputs
- GitHub repository link : lucas-kanade-tracker

6. References

- Understanding Lucas Kanade algorithm
- Wikipedia - Huber Loss
- Wikipedia M-estimator