Project 4 – Lucas Kanade Affine Tracker

Submitted in partial fulfilment of the requirements for the course of

ENPM673 – Perception for Autonomous Systems

By

Adheesh Chatterjee

Andre Ferreira

Pablo Sanhueza

Course Faculty
Prof Cornelia Fermuller

A. James Clark
SCHOOL OF ENGINEERING

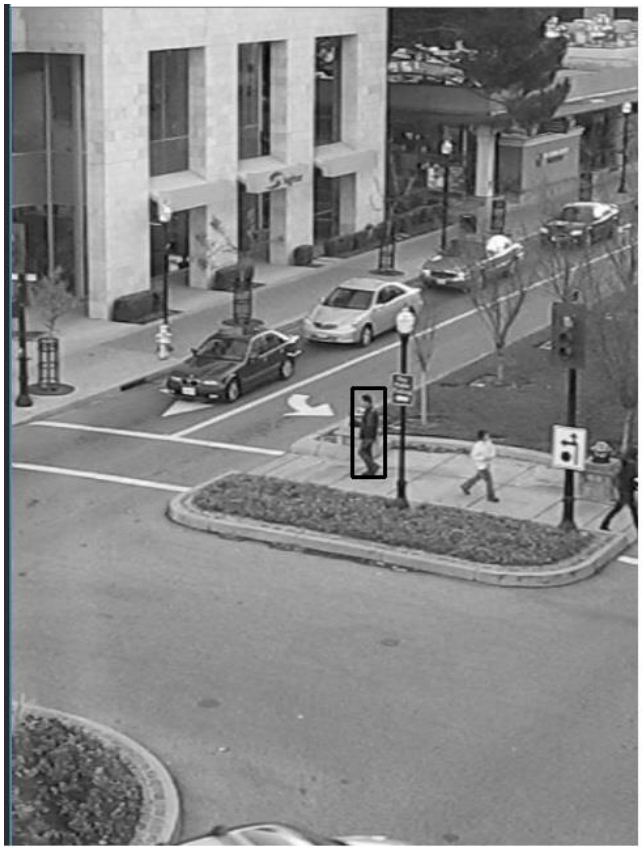
INTRODUCTION

The aim of this project is to implement the Lucas Kanade(LK) Affine Template tracker. The concepts learned in class will allow us to write a script that solves the problem of object detection. To initialize the tracker, we need to define a template by drawing a bounding box around the object to be tracked in the first frame of the video. For each of the subsequent frames the tracker updates an affine transform that warps the current frame so that the template in the first frame is aligned with the warped current frame. Hence, this is tested on 3 videos of the Visual Tracker benchmark database and provided in the pipeline of the project guidelines. Below you will see explanations of what we did to accomplish such an arduous task.

IMPLEMENTATION OF THE TRACKER

We first make a bounding box around the object to be tracked in the first frame and define that as our region of interest. Essentially, we will be matching this template to every frame and using the affine transform we try to detect it as it moves.

Bounding Box for Car



Bounding Box for Human

Bounding Box for Vase

The LK algorithm is a minimization type algorithm where we try to minimize the sum of the squared error between two images – the template T and the warped image I.

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

We warp the frame out and crop out the ROI using our template.

We apply the gradient along x and y, we then warp the images out, and then crop out the ROI manually using the template

We then do the error calculation between the original template and the warped out image.

We form the jacobian of shape (2x6) for each pixel in the warped out image. We then multiply this jacobian formed with the 2 gradient images generated

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

We then form the hessian, using the outputs generated by the previous step.

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

Now we calculate delta p which is the change in position of the ROI in the frame as compared to the previous frame.

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

We then check if the delta p converges to a reasonable limit by taking the norm.

We now have the new positions and the new parameters of the ROI, this gets updated for every frame and the object is successfully tracked.

So to summarize –

The Lucas-Kanade Algorithm

Iterate:

(1) Warp I with $\mathbf{W}(\mathbf{x};\mathbf{p})$ to compute $I(\mathbf{W}(\mathbf{x};\mathbf{p}))$
(2) Compute the error image $T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))$
(3) Warp the gradient ∇I with $\mathbf{W}(\mathbf{x};\mathbf{p})$
(4) Evaluate the Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ at $(\mathbf{x};\mathbf{p})$
(5) Compute the steepest descent images $\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$
(6) Compute the Hessian matrix using Equation (10)
(7) Compute $\sum_{\mathbf{x}}[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}]^{\mathrm{T}}[T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]$
(8) Compute $\Delta\mathbf{p}$ using Equation (9)
(9) Update the parameters $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$

until $\|\Delta\mathbf{p}\| \leq \epsilon$

Reference
https://www.ri.cmu.edu/pub_files/pub3/baker_simon_2003_3/baker_simon_2003_3.pdf

EVALUATION OF THE TRACKER

We implement the tracker on the 3 video sequences – for the car, for the human and for the vase.

Now we know that the LK tracker doesn't work with fast movement between frames, it fails to account for rapid shaking of the camera, and it can't account for changes in brightness well. The least squares method doesn't properly account for outliers. It essentially assumes that the errors are a gaussian with zero mean, using maybe a better M-Estimation technique like Huber loss, we can give appropriate weights to the errors. The LK method can also give an error due to a local minima being formed, which can attenuate over multiple iterations.

Car

We notice, that with just the basic tracker, we lose the car during a change of illumination. Basically, when the car goes under the bridge, the tracker loses the car. To solve this issue, we utilize some basic statistics involving the mean, standard deviation, and z-score.

ADAPTIVE BRIGHTNESS FIX

The implemented algorithm can be interpreted as a shift in data, where we take the mean of the template image as our main focus. The goal is to shift the pixel values to adapt to the brightness by changing them by a certain z-score relative to the mean of the template. Please see equations and explanations shown below.

Standard deviation calculation of the cropped image. This is done for every frame of the video.

$$\boldsymbol{\sigma_{cropped}} = \frac{\sum_{i=0}^{N}\left(\boldsymbol{x_i} - \boldsymbol{\bar{x}_{cropped}}\right)^2}{\boldsymbol{N}}$$

Using the standard deviation we can obtain a z-score relative to the template by using its mean. The z-score is simply measures how many standard deviations the pixel value is from the mean of the template. Obviously, if I want to change/adapt pixel values I need to take into account the standard deviation of the data being analyzed, in this case the cropped images. The z-scores are being calculated for each pixel value in the cropped image. In this case, the z-scores represent how many standard deviations the pixel value is away from the mean of the template image relative to the standard deviation of the cropped image. Please see equation below.

$$\boldsymbol{z_i} = \frac{\boldsymbol{x_i} - \boldsymbol{\bar{x}_{template}}}{\boldsymbol{\sigma_{cropped}}}, where\ x_i\ is\ for\ every\ pixel\ value$$

This z-score is then utilized to recalculate the shifted value of every pixel, but now we actually use the actual mean of the cropped image. See equation below.

$$\boldsymbol{x_{i\_new}} = \boldsymbol{z_i} * \boldsymbol{\sigma_{cropped}} + \boldsymbol{\bar{x}_{cropped}}$$

To implement this algorithm, we change all equations to matrices to improve the speed of the code. These are displayed below.

$$\boldsymbol{Z} = \frac{\boldsymbol{X} - \bar{\boldsymbol{X}}_{\boldsymbol{template}}}{\boldsymbol{\sigma}_{\boldsymbol{cropped}}}$$

$$\bar{X}_{template} = \begin{pmatrix} \bar{x}_{template} & \cdots & \bar{x}_{template} \\ \vdots & \ddots & \vdots \\ \bar{x}_{template} & \cdots & \bar{x}_{template} \end{pmatrix}, \quad template\ image\ mean\ (all\ values\ are\ the\ same)$$

$$X = \begin{pmatrix} Pixel_{(0,0)} & \cdots & Pixel_{(X,0)} \\ \vdots & \ddots & \vdots \\ Pixel_{(0,Y)} & \cdots & Pixel_{(X,Y)} \end{pmatrix}, \quad pixel\ values\ of\ cropped\ image$$

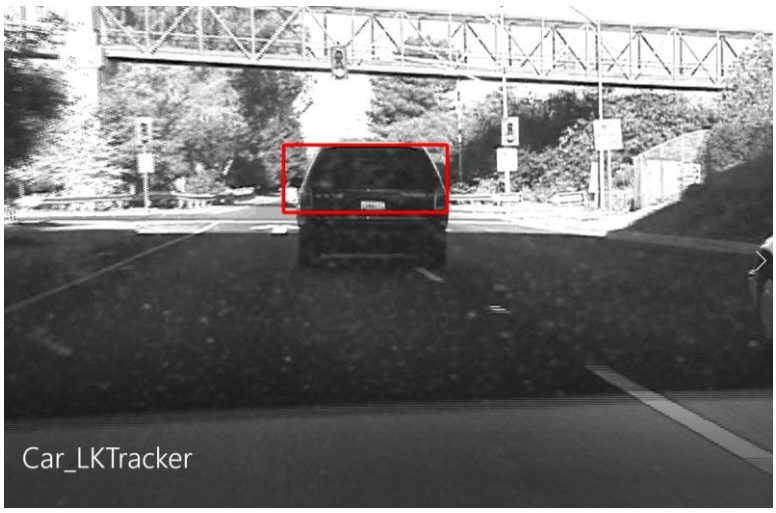
$$Z = \begin{pmatrix} z_{(0,0)} & \cdots & z_{(X,0)} \\ \vdots & \ddots & \vdots \\ z_{(0,Y)} & \cdots & z_{(X,Y)} \end{pmatrix}, \quad every\ z-score\ for\ each\ pixel$$

$$\boldsymbol{X_{new}} = \boldsymbol{Z} * \boldsymbol{\sigma_{cropped}} + \bar{\boldsymbol{X}}_{\boldsymbol{cropped}}$$

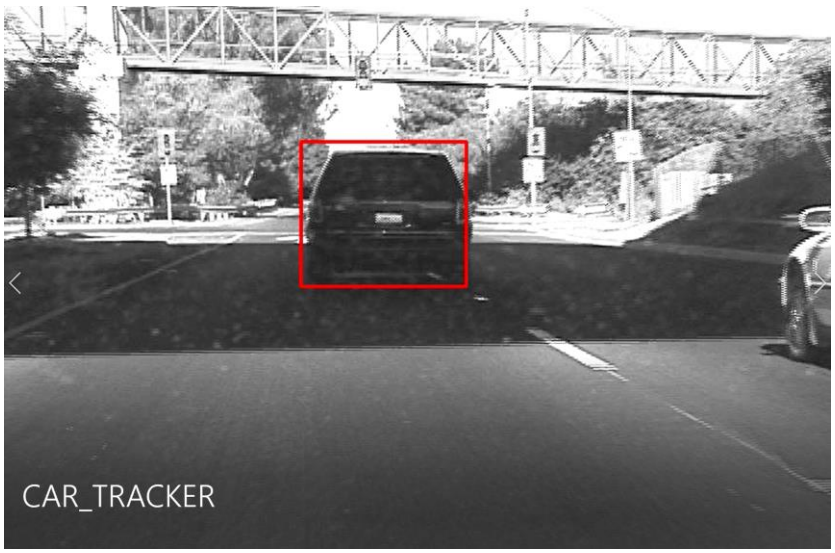
$$\bar{X}_{cropped} = \begin{pmatrix} \bar{x}_{cropped} & \cdots & \bar{x}_{cropped} \\ \vdots & \ddots & \vdots \\ \bar{x}_{cropped} & \cdots & \bar{x}_{cropped} \end{pmatrix}, \quad cropped\ image\ mean\ (all\ values\ are\ the\ same)$$

These simple matrix equations are implemented in a function called shift_data(), which returns a new image with adapted values relative to the template mean. The implementation of these equations provided very good results as shown in the video and images shown below.

During Bridge – Without adaptive brightness algorithm



During Bridge – With adaptive brightness algorithm

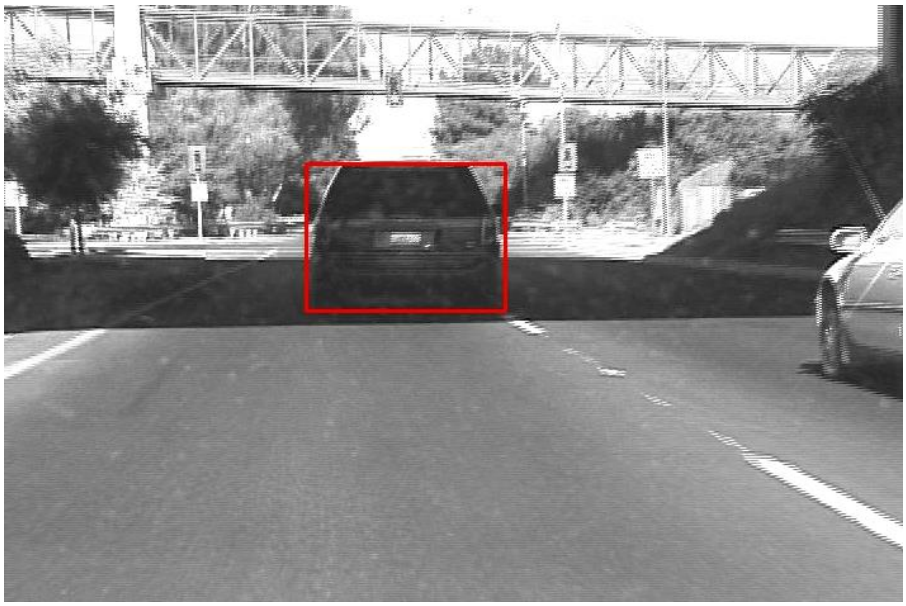


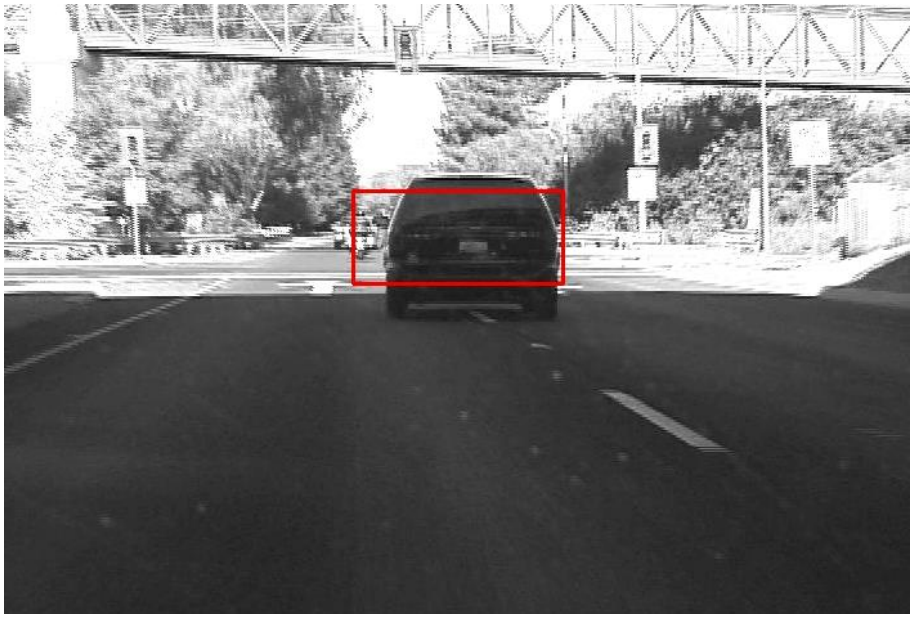
GAMMA CORRECTION

We also implement the gamma correction method as an alternative for extra credit 2nd part. To make it more adaptive, we found the norms of the template and of the current frame. If the norm value of the current frame was lesser than that of the template, a gamma correction was applied.

Even though it is not as robust as the previous method, it still manages to track the car under the bridge, however the bounding box isn't a perfect fit. After the bridge though, we are able to track the car perfectly with a good fitted bounding box.

During Bridge – With gamma correction





After Bridge – With gamma correction

Human

For the human, we notice when the camera shakes and the movement is too rapid for the tracker to detect it and hence it loses the human. To solve this issue, we tried to do the temporal smoothing so as to make it more robust, but the structure of our pipeline made it difficult to do so. Hence, we had to take another approach. We tried to tune the parameters that would give us the best result for the end of the video. This however, posed a problem because towards the end of the video, the tracker lost the human for a moment and tried to detect the wheel of the black car.

However, once it returns to the human, we are able to properly track him until the end of the video, even with the shakiness of the camera.

With our old parameters the code was working when he starts to cross the street but in the end, we were losing the human because of the camera shakes.

We hence chose to keep the parameters that we felt worked better as it would detect the human even in the presence of disturbances due to the camera shaking. We display the differences in the following images. The Green rectangle represents our Tracker and the Red rectangle is fixed to one position on the screen.

We thus notice, that even though the red one is always in the same place, our green rectangle tries to always track the human. Hence, our tracker works.

Vase

For the vase, we observe that the camera rotates and zooms in every other frame. This makes it almost impossible to track the vase accurately. We considered using the Pyramid approach to try to solve this issue, however, we found an easier and possibly a more robust method to solve it. The way our tracker works, is that we form the new bounding box using 2 diagonal points. However, because of this, the other two points are randomly generated and the bounding box no longer covers the whole object. Also, the affine doesn't account for the scaling and the bounding box doesn't rotate with the image. So we basically, initialized using the other 2 points along the diagonal and implemented our tracker using those points. This made sure the box was formed using the opposite points.

Since we had all 4 points of the bounding box which could be tracked well, we used cv2.polylines to join all 4 points. This accounted for the rotation of the tracker as well as the scaling. Hence, we were able to get good detection.

As we see in the image, using only 2 points, we can fix the top left and bottom right of the bounding box, but it doesn't account for the other two randomly generated points, hence the whole box isn't covered.

Similar to the previous image, here the bounding box is defined using the top right and bottom left points and hence, the other 2 diagonal points are randomly generated and and whole box isnt covered.

We thus combine the 2 methods, whereby 4 points will be used to define the bounding box. We get those 4 points and use cv2.polylines to define the bounding box that will always cover the vase.

EXTRA CREDIT – ROBUSTNESS TO ILLUMINATION (10+10)

1) Since we only had to implement the extra credit part for the car, we optimized the code to account for the changing brightness under the bridge. This process is explained in the previous section, where we explain how the Z-score and mean was used to check for changing brightness.

2) Another possible method of solving the brightness issue is using a gamma correction, if we use a high gamma correction, we can brighten images. However, this may cause potential errors as it is not adaptive in nature making it less robust than the previous method. To make it more adaptive, we found the norms of the template and of the current frame. If the norm value of the current frame was lesser than that of the template, a gamma correction was applied.

If you look at the code titled ‘extra credit.py’, we created 2 functions, one which accounts for method 1. This shifts the means as explained and enables tracking in changing brightness. The other function accounts for method 2. This applies a gamma correction onto the image to increase brightness when under the bridge.