Project 4 ENPM 673

Rene Jacques and Zachary Zimits

April 19, 2019

1 Image Processing

For each data set we started with the first frame and selected a template that contained the object we wanted to track. After the first frame we began to process the image with the Lukas Kanade (LK) algorithm

Figure 1: Template Image

1.1 Error

To calculate the error between the template and the new frame we initially warped the new image using an affine matrix that is initialized as an identity matrix. After the image has been transformed we selected the ROI using the pixel locations from the template. We subtracted the new image from the original template to find the error. We had cast each image as int type before the subtraction operation to keep python from rounding the negative values back up to 255. The equation representation of the error can be seen in (1)

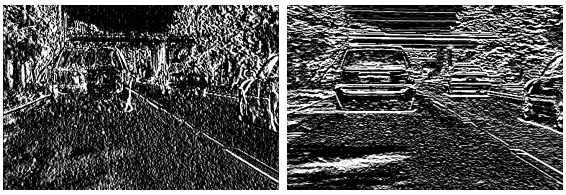
$$T(x) - I(W(x;p)) \tag{1}$$

1.2 Steepest Decent and Jacobian

Finding the steepest decent began by taking the gradient of the new image in both the x and y direction. We then warped the image back to the original frame and selected the ROI using the same method from the error calculation. The Jacobian is defined as equation (2), derived from equation (8) in (Baker and Matthews.

$$\frac{\delta W}{\delta p} = \begin{bmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{bmatrix} \tag{2}$$

We go through each pixel in the gradient ROIs row by row. We combine the intensity value from each ROI and format it into a 1x2 matrix and multiply it



(a) Gradient X (b) Gradient Y

Figure 2: Gradient images

by the Jacobian to get a 1x6 vector that represents the value of the steepest descent images at that given point. The vector is added to a array to create an (n*m)x6 array where n and m are the dimensions of the template. This array is represented by equation (3) with the resulting images shown in Figure 4.

$$\nabla I \frac{\delta W}{\delta p} \tag{3}$$

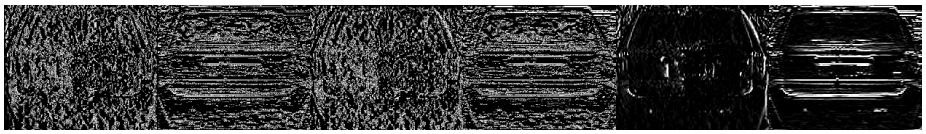


Figure 3: Steepest Descent Images

1.3 Hessian

Using equation (4) the 6x6 hessian matrix is calculated

$$H = \sum_x \left[\nabla I \frac{\delta W}{\delta p} \right]^T \left[\nabla I \frac{\delta W}{\delta p} \right] \tag{4}$$

1.4 Update W

Plugging in the values we calculated from equations (1) through (4) we can calculate the change to our affine matrix using equation (5).

$$\Delta p = H^{-}1 \sum_x \left[\nabla I \frac{\delta W}{\delta p} \right]^T [T(x) - I(W(x;p))] \tag{5}$$

We then multiplied the Δp by a learning rate α to help the algorithm cope with situations where the objects move large distances between frames.

We update the transformation matrix with the Δp and calculate the norm of Δp. If the norm is below a determined threshold value then the algorithm ends and returns the updated transformation matrix. If the norm is to high then the

algorithm begins again but this time starting with the updated W matrix. We added a hard limit on the number of loop the algorithm can take in order to speed up the algorithm and to keep the program from entering an infinite loop in the case of a non converging solution.

2 Results

The template used for each of the three sequences: car, human, and vase, was obtained from the first frame of each video. This worked well for all three sequences and did not seem to have any adverse affects on the tracking.

In the car sequence the tracker breaks down when the car goes under the bridge. This is because the lighting changes drastically on the car in the current frame and therefore cannot be reliably compared to the template image. A method for addressing this issue is discussed in Section 3. Once the car is under the bridge the tracker begins to fail and eventually begins to track a section of the sky that the car is not in. The tracker never finds the car again.

In the human walking sequence the tracker works well until the camera shakes. At this point the tracker loses the human, but might find the human again if the video were longer due to the movement of the camera. The tracker loses the human because the Lukas Kanade algorithm is designed to track small changes in pixel locations, and not large changes. Large changes can mean that the target being tracked will not be in the ROI for the current frame, making tracking impossible.

In the vase sequence the tracker breaks down at multiple points in the video, but then finds the vase again later because the video stays mostly centered on the vase for the entire sequence. The tracker fails due to rapid changes between the frames, the same way the tracker broke down in the human walking sequence.

3 Illumination Robustness

In the case of the car going under the bridge we implemented a simple robust illumination correction to help the algorithm to continue to track the car despite the shadow cast by the bridge. In the error step before we perform the subtraction algorithm we calculate the ratio of the brightness values of the template to the new ROI and multiply the ROI by this value.

(a) Template (b) Original ROI (c) Lightened ROI

Figure 4: Illumination Correction Results

This method did not work perfectly as can be seen in figure 4 but it did significantly improve the trackers ability to follow part of the car through the bridge and to the other side

4 Video Output

Video clips of Lukas Kanade tracking of the car, human, and vase can be seen by following this link: https://drive.google.com/open?id=1BRTn5sksLKZSSY6FM8qns-28jbDrS87N)