[ENPM673] Project 4

Due Date: April 19, 2019

1 Introduction

In this project you will implement the Lucas-Kanade (LK) template tracker. Then you will evaluate your code on three video sequences from the Visual Tracker benchmark database: featuring a car on the road, a human walking, and a box on a table. The short video that you will process are available in this folder.

To initialize the tracker you 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 will update an affine transform that warps the current frame so that the template in the first frame is aligned with the warped current frame.

For extra credit, you may look at ways to make tracking more robust, by incorporating steps to increase illumination invariance.

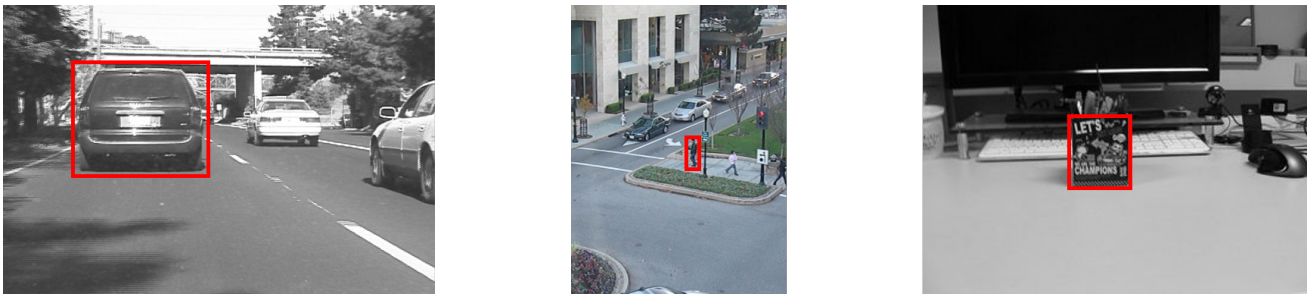

Figure 1: Tracking sequences

2 Lucas Kanade Template tracker (80points)

2.1 Implementation of the Tracker

Technical background and the implementation of the LK tracking algorithm are described in Section 2 of Simon and Matthew's paper.

Initialize manually the coordinates of a bounding box surrounding the object to be tracked.

At the core of your algorithm is a function affineLKtracker(img, tmp, rect, p_{prev}). This function will get as input a grayscale image of the current frame (img), the template image (tmp), the bounding box (rect) that marks the template region in tmp, and the parameters p_{prev} of the previous warping.

The function will iteratively compute the new warping parameters p_{new}, and return these parameters. You can either use a fixed number of gradient descent iterations or formulate a stopping criteria for the method.

Your algorithm should compute the affine transformations from the template to every frame in the sequence and draw the bounding boxes of the rectangles warped from the first frame.

2.2 Evaluation of the Tracker

Evaluate your tracker on the three sequences: the car sequence, the human walking, and the table top scene. Use only the grayscale, not the color.

What sort of templates work well for tracking? At what point does the tracker break down? Why does this happen?

Submit your code, for each scene a video of the object being tracked, and your discussion. Please submit your output videos in a Google Drive folder that is accessible to us. The code and report can be submitted via ELMS.

3 Robustness to Illumination (extra credit 10+10 points)

The LK tracker as it is formulated, breaks down when there is a change in illumination because the sum of squared distances error that it tries to minimize is sensitive to illumination changes. There are a few things you could try to do to fix this. The first is to scale the brightness of pixels in each frame so that the average brightness of pixels in the tracked region stays the same as the average brightness of pixels in the template. The second is to use a more robust M-estimator instead of least squares (e.g. a Huber loss) that does not let outliers adversely affect the cost function evaluation.

Note that doing this will modify the least squares problem to a weighted least squares problem, i.e. for each residual term you will have a corresponding weight Λ_{ii} and your minimization function will look like

$$L = \sum_x \Lambda_{ii}[T(W(x;\Delta p)) - I(W(x;p))]^2$$

leading your Jacobian computation to be a weighted Jacobian

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

Thus

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

Here Λ is a diagonal matrix of weights corresponding to the residual term computed as per the choice of the robust M-estimator used, A is the Jacobian matrix for each pixel of the template considered in the cost function, and b is the corresponding vector of residuals. Implement these two methods and test your new tracker on the car video sequence again.

Credit This project has been inspired by homework assignment 7 from CMU's Computer Vision course in Spring 2018.