

Computer Vision I
CSE 252a, Fall 2012
David Kriegman

Name:
Student ID:
Email:

Assignment #4

(Due date: 12/07/2012)

Overview

In this assignment you will implement the Lucas-Kanade algorithm for computing a dense optical flow field and template tracking. You will then implement a corner detector and combine your corner detector and optical flow algorithms to compute a flow field only at reliable corner points. Next, you will implement a template tracking algorithm based on Lucas Kanade template tracking. Three sets of test images and a video sequence are available from the course website. The first set of images contains a synthetic (random) texture, the second a rotating sphere, and the third a corridor at Oxford university; the video sequence is a highway chase scene. Before running your code on the images, you should first convert your images to grayscale and map intensity values to the range $[0,1]$.

Problem 1. Single-Scale Optical Flow [5pts]

Implement the single-scale Lucas-Kanade optical flow algorithm. This involves finding the motion (u,v) that minimizes the sum-squared error of the brightness constancy equations for each pixel in a window. As a reference, you should read pages 191-198 in *Introductory Techniques for 3-D Computer Vision* by Trucco and Verri. Your algorithm will be implemented as a function with the following inputs, `[u,v] = optical_flow(I1,I2>windowSize)`. Here, u and v are the x and y components of the optical flow, $I1$ and $I2$ are two images taken at times $t=1$ and $t=2$ respectively, and $windowSize$ is the width of the window used during flow computation. In addition to these inputs, you should add a threshold τ such that if the smallest eigenvalue of $A^T A$ is smaller than τ , then the optical flow at that position should not be computed. Recall that the optical flow is only valid in regions where

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

has rank 2 (why?), which is what the threshold is checking. A typical value for τ is 0.01; if you use something different please specify this in your report.

What to turn in:

Submit a hardcopy of your code and figures on the due date with the following:

- Your matlab code.
- Quiver plots of (u,v) for the following image pairs / window sizes (6 quiver plots total)
 - SYNTH : $I1 = \text{synth 000.png}$, $I2 = \text{synth 001.png}$ $windowSize = 9 \times 9$, $windowSize = 15 \times 15$
 - SPHERE : $I0 = \text{sphere.0.png}$, $I1 = \text{sphere.1.png}$ $windowSize = 15 \times 15$, $windowSize = 21 \times 21$
 - CORRIDOR : $I0 = \text{bt.000.png}$, $I1 = \text{bt.001.png}$ $windowSize = 15 \times 15$, $windowSize = 21 \times 21$
- Any modifications to the window size and/or thresholds you used.
- Does this algorithm work well for these test images? If not, why?
- Try experimenting with different window sizes. What are the tradeoffs associated with using a small vs. a large window size? Can you explain what's happening?

Problem 2 - Corner Detection [2pts]

Use your corner detector from Assignment #3 to detect corners in the provided images. If you do not have working corner detector, email the TA for a set of corners. You will only receive half credit for this part.

Problem 3 - Sparse Optical Flow [3pts]

Combine Parts A and B to output an optical flow field only at detected corner points. **What to turn in:**

Submit a hardcopy of your code and figures on the due date with the following:

- f. Your matlab code.
- g. Show 3 quiver plots on the same images from problem 2, displaying a flow vector at each corner point.
- h. Select the values for windowSize and that give you the best results, and include those numbers in your report.

Problem 4 - Template Tracking [5pts]

One of the most classical work in template tracking is the Lucas-Kanade Tracker. Basically, between two consecutive images, it assumes constant motion in a small region. This assumption helps solve the aperture problem and give a well-defined solution.



Figure 1: Car Sequence Images

Starting with a rectangle R_t on frame I_t , the Lucas-Kanade Tracker aims to move it by an offset (u, v) to obtain another rectangle R_{t+1} on frame I_{t+1} , so that the pixel squared difference in the two rectangles is minimized:

$$\min J(u, v) = \sum_{(x, y) \in R_t} (I_{t+1}(x + u, y + v) - I_t(x, y))^2$$

Starting with a initial guess of (u, v) (usually $(0, 0)$), we can compute the optimal (u, v) iteratively. In each iteration, the objective function is locally linearized by first-order Taylor expansion and optimized by solving a linear system.

Implement a function that computes the optimal local motion from frame I_t to frame I_{t+1} that minimizes $J(u, v)$ ¹. Your function should have signature `[u, v] = LucasKanade(It, rect, It1)`. I_t is the image at frame t and I_{t1} is the following frame, `rect` is a 4x1 vector with $[x_1 y_1 x_2 y_2]$ where the first xy pair is the top left corner of a rectangle and the second pair is bottom right coordinates.

¹See Lucas-Kanade 20 Years On: A Unifying Framework Section 2 for more details

Load the video sequence (**carSequence.mat**) and run the Lucas-Kanade Tracker to track the speeding car. The rectangle in the first frame is $[x1, y1, x2, y2] = [328, 213, 419, 265]$. In other words, the rectangle starts from (328, 213) (row 213 and column 328 in the image) and ends at (419, 265).

What to turn in:

Submit a hardcopy of the code and figures on the due date with the following:

- i. Your matlab code.
- j. Print the tracking result (image + rectangle) at frames 20, frame 50 and frame 100.

In addition, please email to jmerkow@ucsd.edu a copy of your code and figures. Please send as a zip or tar file. In the subject line, please put the string CSE252AHW4.