COMS30121: Assignment 03 ('Gesture Recognition')

Guidelines

- This is the second and final piece of assessed coursework in the unit.
- It is to be completed in groups of 2 or 3 (please report any change to the current team structure to the course director BEFORE starting your assignment)
- It is worth 25% of the unit mark
- Every student is required to upload their full piece of work (that is a 1page PDF file and all code and executables in a project folder including all your source code files and project files as a single ZIP file to SAFE before 23:59:59, Wed 18th December 2013. Make sure you upload it early enough (not last minute!) to avoid upload problems. (Each member of a team has to upload an identical copy of the teams work.)
- In addition to the online submission, you will present your submitted program running in the labs on 19th December 2013. You will need to attend this lab session to get a mark. At these labs, we will ask you questions about your work and you will be able to showcase the merits of it.
- Your work will be marked on an individual basis considering both the submission and the lab presentation.
- Do not attempt to plagiarise or copy code between subgroups etc. It is not worth it, be proud of your own work! We will individually ask you questions about your work in the labs - so you must understand the code your team developed in any case.

Introduction

This piece of coursework involves three subtasks. On the completion of all tasks you will have created a rudimentary gesture recognition system based on the Lucas-Kanade (LK) optical flow method. As with all the previous tasks, you have been provided with working code - video.cpp - to get you going. In this case the code either displays the video from a web cam (supplied in the lab!) or loads a video and displays the output. Fire it up and have fun.

Task One

The LK method relies on image derivatives, both in the spatial and temporal dimensions, i.e. I_x, I_y and I_t. The aim of this task is to calculate these derivatives from an image sequence as shown on slide 14 of lecture 8. You will implement a function that will return 3 matrices containing I_x, I_y and I_t. In addition, display these derivatives alongside the original video.

Hint: computing the temporal derivative requires two frames. Think carefully about the frames you use and in which order.

Task Two

The second task requires you to implement the LK algorithm. Once you have completed task one, there is little here that you have not done before. To help you, this task has been split into smaller sections.

Hint: One important thing to mention is that the LK algorithm is slow to compute dense optical flow (flow at every pixel). Because of this, it is highly recommended to implement your algorithm in such a way that only a subset of pixels are considered.

Part 1

The LK method works over a given region size. Changing the size will produce different results. Implement an **LKTracker** function. In this function you will need to go to the set of positions that you wish to compute the motion at, and then iterate over the pixels in a region about those points. Implement this function and make sure it works for any region size. The implementation is similar to that of a convolution, so pay attention to things like padding, etc.

Part 2

The main section of the LK algorithm is the *least squares approximation* as given on slide 13 of lecture 8. To prepare for this, you will need to create the following matrices:

$$A = \sum_{region} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \qquad \mathbf{b} = \sum_{region} \begin{bmatrix} -I_x I_t \\ -I_y I_t \end{bmatrix}$$

The motion estimation is then computed using : $\mathbf{v} = A^{-1}\mathbf{b}$. Given some handy OpenCV functions, this is the easiest part of the exercise. See here for inverting a matrix, and here for matrix multiplication. The vector $\mathbf{v}$ will now contain the motion vectors at the pixel p.

Part 3

With motion vectors computed, you will want to display the output. This task requires you to display a vector plot superimposed over the original image; the x component of the motion as a gray scale image; and the y component of the motion as a gray scale image. For the vector plot, you will need to use the OpenCV drawing function here.

Hints: 1. Think carefully about the normalisation of the x and y components when you come to display them. 2. You may want to scale the magnitude of the vectors to make the motion more visible. 3. You may also want to scale the size of the image, in which case, you will need to think about adjusting the positions of the vectors.

Task Three

Given a correctly implemented LK algorithm from task three, this task requires you to implement a simple gesture recognition algorithm that can distinguish movement to the left and movement to the right. This task is intentionally left open for you to experiment. However, a starting point might be to use a threshold on the magnitude of the flow vectors and remove motion below a certain size. You could then compute the mean direction of the remaining vectors.

Mark Breakdown

The marker will take into account your lab presentation and your source code. Find below a guideline for the assessment

- ***For a mark above 40*** *you need to have implemented task one and be able to answer basic questions about your program and the derivative images*

- ***For a mark above 50*** *you also need to implement the first part of task two and be able to answer questions about your program*
- ***For a mark above 60*** *you also need to implement all of task two and be able to answer questions about the motion fields you have generated and its performance on the sample videos*
- ***For a mark above 70*** *you also need to be able to explain in detail the concepts behind the LK algorithm. You also need to have completed task three and be able to discuss the reasons for your implementation choices*