COMP 558: Assignment 3
Available: Friday, November 2nd, 2018
Due Date: Friday, November 16th, 2018 (before midnight) via mycourses.

Notes: Please use mycourses->Discussions for questions related to this assignment. You are free to discuss the questions with one another. However, the solutions that you submit must represent your own work. You are not permitted to copy code from each other, or from the internet. Present and discuss your results carefully and submit documented Matlab code that you have written to generate your results as a separate file. There are aspects of this assignment that are a bit open-ended, having to do with parameter choices to getting results that are qualitatively acceptable for the sequences we provide. This is something you can assess visually. Overall we will look not only at your implementations, but also how you present and discuss your results. Make sure that your figures and plots are in the PDF you submit for all questions.

Lucas-Kanade Image Registration (100 marks)

For this question, you will be provided a starter code package. Please extract the package and follow the instructions in the README file before implementing anything.

In the Lucas-Kanade construction, the motion field is obtained by finding, for each pixel in an image, a corresponding pixel in the next frame of the sequence of images, in such a way that it minimizes the sum of squared intensity differences computed over a window. We saw in class in our discussion of 2D image registration that this leads directly to the use of the second moment matrix. In this assignment we will explore the use of this strategy for frame to frame optical flow (motion field estimation). Using essentially the same notation as in the notes, but treating I^n and I^{n+1} as successive frames, the motion field (v_x, v_y) for frame I^n satisfies:

$$\begin{bmatrix} \sum W(\frac{\partial I^n}{\partial x})^2 & \sum W(\frac{\partial I^n}{\partial x})(\frac{\partial I^n}{\partial y}) \\ \sum W(\frac{\partial I^n}{\partial x})(\frac{\partial I^n}{\partial y}) & \sum W(\frac{\partial I^n}{\partial y})^2 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = - \begin{bmatrix} \sum (I^n(x,y) - I^{n+1}(x,y))\frac{\partial I^n}{\partial x} \\ \sum (I^n(x,y) - I^{n+1}(x,y))\frac{\partial I^n}{\partial y} \end{bmatrix}$$

This is essentially the same formula we saw in the Lecture 11 notes on registration (page 3), except that (h_x, h_y) is being treated now as a frame-to-frame velocity vector (v_x, v_y) and $I(x,y)$ is being replaced by $I^n(x,y)$ and $J(x,y)$ by $I^{n+1}(x,y)$. To simplify the notation, we have not mentioned the neighborhood, but is understood that the sums are being computed over a chosen local neighborhood for each image point. Also, there is one further subtlety. When you compute the sums you should weight the individual entries by a Gaussian centered at the pixel under consideration. In the notation this is captured by W. In other words, the contribution to the sum should fall off with distance from (x,y), using a Gaussian profile. This will help to localize your estimates.

1. **a)** (30 marks) We ask you to first estimate the motion field for a set of frames, using a one shot process. By this we mean, choose a pair of frames, and solve for (v_x, v_y) at each location $I(x,y)$ in $I(x,y)$. In order to do this you can assume a square window of size $m \times m$ about each (x,y) within which to compute the sums for each entry in the

second moment matrix. To make this problem well posed, you will need to also blur the images with a 2D Gaussian of a certain σ prior to computing the gradient terms. This is because in theory the second moment matrix may not be everywhere invertible, as we saw in class. Illustrate and discuss your results for a few frames in the *Basketball* and *Backyard* sequence, using the plotting functions we have provided. Discuss the performance and limitations of this one-shot algorithm.

2. **b)** (20 marks) Once you have an implementation of the one-shot frame-to-frame approach, we shall now try to refine it by using the iterative version of this algorithm. Here we ask that you implement the version of the algorithm at the bottom of the Lecture 11 notes on registration (page 3). The basic idea is to implement a step by step version of the solution to part **a)**, i.e., having obtained an initial displacement vector (v_x, v_y) we use it to update I and then we find a new displacement vector. This process continues until the displacements are considered to be small. Implement this iterative version and discuss how the results improve upon the results you obtained in part **a**, again, using the *Basketball* and *Backyard* sequence. Ideally some of your motion vectors should be more refined.

3. **c)** (40 marks) Once you have part **a)** and **b)** working well qualitatively for the Basketball or Backyard sequence, we ask you to implement a coarse-to-fine version of the algorithm. The basic steps of a course to fine approach for estimating a motion field (v_x, v_y) from frame to frame, are described on page 4 of the Lecture 11 notes on registration. The idea is to generate a Gaussian pyramid for the two image frames being registered and to use the motion field vectors estimated at the coarsest scale (most blur) to initialize the estimate at the next coarsest scale. One repeats this process, upto the finest scale. Within each level of the pyramid there is also an iteration, having to do with the update described in part **b)**. Implement this coarse-to-fine approach and discuss its advantages over the fixed scale approach in part **b)**. Do this by comparing results and discussing the tradeoffs. You are free to choose the parameters that give you good results. For evaluation, try this advanced method on the *Basketball* and *Backyard* sequence and the *Schleffera* sequence.

4. **d)** (10 marks) The Lucas-Kanade algorithm suffers from the limitation that it can only recover translations. In some of the sequences the local motion is much more complex than that, e.g., in the *Mequon* sequence you see a rotational component on the bottom left. Discuss in some detail how you might modify the approach you took in part **b)** to recover a more general affine motion field, one that could include local translation, rotation, and shearing. You don't actually have to provide an implementation, just a discussion of how you would go about doing this.