COMP 558: Assignment 2
Available: Monday, October 25th, 2021
Due Date: Monday, Nov 8th, 2021 (before midnight) via mycourses.

Introduction

This assignment covers material in Lectures 11-12 and 14-16. The questions concern 2-frame image registration using the Lukas-Kanade algorithm, and models of projection with rotation and translation of 3D points involved. Please use the mycourses discussion board forum for assignment 2, for any assignment related questions. This will help us reach the entire class when we respond. However, follow appropriate protocol, e.g., do not reveal the answer to a particular question or ask if your proposed solution is correct. Save these kinds of questions for one-on-one interactions during office hours. Do not post pictures of your solutions because these could give ideas to the rest of the class. You are free to discuss the general aspects of questions with each other, but not to the point where you are revealing answers or sharing solutions. This is an individual assignment and the solutions that you submit must reflect your own work and must be written by you. You are not permitted to submit code or text that you have copied from one another or from any third party sources, including internet sites, such as the Matlab file exchange forum. We expect that you will write your code. By submitting this assignment you agree to abide by these rules. Keep an eye on the discussion board where we might post hints or reply to questions that come up.

Instructions

- All submissions should be made to the assignment 2 folder.
- For the first question you should submit a single .zip file q1.zip containing: 1) a PDF file explaining your solution to each part in some detail, and providing illustrative examples and/or figures as you need them, 2) your Matlab code, with comments, so that we can run it and 3) sample movie sequences of your "optical flow" results. Be sure to follow the README file and to build on the sample helper code we have provided.
- For the second question you should also submit a single .zip file q2.zip containing: 1) a PDF file explaining your solution to each part in some detail, and providing illustrative examples and/or figures as you need them, 2) your Matlab code, with comments, so that we can run it and 3) sample movie sequences of your "rotation and translation" results. We have provided helper code with functions that you can build on, as well as README file explaining how to proceed.
- In order to receive full points, your solution code must be properly commented, and you must provide a clear and concise description of your computed results in the PDF. The TAs have limited time and will spend at most 20-30 minutes grading each submission.

Late assignment policy: Late assignments will be accepted up to only 3 days late and will be penalized by 10% of the total marks, per day late. For example, an assignment that is submitted 2 days late and that receives a grade of 90/100 would then receive an 18 point penalty. Submitting one minute after midnight on the due date means it is 1 day late (not a fraction of a day), and the same for the second day.

Question 1: Lukas-Kanade Image Registration (50%)

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 part of the 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:

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

This motion field can be solved either in one shot, or via an iterative process, as discussed in class. However, there are a few parameters that you will need to play with for the actual implementation. The first has to do with the σ of the 2D Gaussian you will use to smooth each frame spatially. This is required because otherwise the image gradients (I_x, I_y) that will be used to construct the second moment matrix will not be well defined. The second has to do with the window size in which one averages (computes sums) for the second moment matrix parameters (see Lecture 11). The third has to do with the fact that derivatives in time on the right hand side of the equation might not be well defined, so for those you might also want to "smooth", but with a 1D Gaussian.

At the end of the day, once you have settled on those parameters, you can compute frame to frame motion field estimates for v_x, v_y. Attempt to do this for a selection from the image sequence(s) we have given and to overlay the results as a quiver plot for a particular frame. For simplicity you can start by overlaying a flow vector only for the central pixel in a window and you can assume that the windows do not overlap spatially. Once that is working, you can then compute flow vectors more densely (by allowing the windows to overlap spatially). Try to make some sense of your results once you have them and provide a discussion of them. For example, you would expect that at homogenous regions the flow vectors would be somewhat incoherent geometrically (but would be small) and at background regions the flow vectors would also be small, so you can threshold on flow vector length to get a more useful (qualitatively) quiver plot.

- For a particular image sequence, it might be helpful to first preprocess the frames. We show an example of how to smooth in time using built n functions in one of the helper functions. You could supplement this by smoothing spatially as well. Then, if you save

these preprocessed frames you can proceed to solve for the frame to frame motion field estimation.

- We will be grading this question holistically, looking for evidence that you have understood the sub-parts and have organized your results well. Therefore it is important that you ask questions on the discussion board if you need clarification.
- We are not asking you to implement the method using a pyramid, or to consider the iterative version. This is something you could consider later, since it is not a big additional step, to improve efficiency and quality.
- You will find that some sequences are easier to solve for than others and that the estimates you get are affected by parameter choices such as smoothing amounts and window sizes. Try to discuss this when you present your results or your movies.
- If you wish, when you generate movies or figures, you can overlay the quiver plots on the original frames in the sequence so that the context of motion is clear.
- When all is working feel free to try the algorithm on a longer sequence of your own.

Question 2: Projection, Translation, Rotation (50%)

1. In class we considered 3 cases of rotations of a camera's axes system, i.e., a pure rotation about its Z axis (roll), a pure rotation about its X axis (tilt) and a pure rotation about its Y axis (pan). For each case we sketched out an idea for computing the image motion field (v_x, v_y) seen by the camera. For each case, complete the calculations to show that the motion fields are indeed what we claimed they are (slides 16, 17, 19 in Lecture 15). [**10 marks**]
2. We will now elucidate the mapping from world to camera to pixel coordinates, using the matrix formulation we have used in class, as well as explore the consequences of certain extrinsic operations, in particular, rotations as well as translations. You are provided with 3 data sets, each of which, for a particular 3D scene, contains the following:
   (a) A greyscale image of a particular size where at each pixel location one has a value proportional to the depth of the projected $3D$ scene point. Thus brighter points are further away from the camera.
   (b) A color image of the same size, which is registered to the above depth image. This should be interpreted as the appearance image which contains the red, green and blue channels of image irradiance for each corresponding 3D scene point.
   (c) A text file containing the intrinsic parameters, i.e., the camera calibration matrix $\mathbf{K}$ we shall discuss in class.
   (d) A text file containing the camera extrinsic parameters, i.e., the $\mathbf{R}[\mathbf{I}|-\mathbf{C}]$ part of the finite projective camera model we shall discuss in class.

Based on the extrinsic and intrinsic parameters we have provided, your first task is to write a Matlab function which, for any pixel in the depth image, provides the coordinates of the associated 3D scene point (X, Y, Z) and the associated color channel values. You can assume that the world coordinate frame corresponds to the standard Cartesian frame with basis vectors $(1, 0, 0), (0, 1, 0), (0, 0, -1)$ corresponding to the X, Y, and Z directions. [**10 marks**]

3. You will now consider the 3 explicit rotations (along with a translation) of the camera coordinate system we discussed in class: a rotation about the Y-axis (pan), a rotation about the X-axis (tilt) and a rotation about the Z-axis (roll). These rotations are modeled by Ωt where Ω is a parameter representing the rotation amount per step and t is the number of steps. Write a Matlab function which rotates a 3D scene point (X, Y, Z) about the X, Y, or Z axes (as desired) by an amount Ωt, while also translating the points with a certain step size in a certain direction (T_x, T_y, T_z). This function should output the new coordinates of the 3D scene point with respect to the camera reference frame. [**10 marks**]

4. Now use the above function along with the intrinsic parameters to consider a sequence of rotations $\Omega t \in (0, \pi/2)$ about the X, Y, or Z axes of each 3D scene point along with translations, followed by a projection to create two images: 1) an image that contains the projected depth value (greyscale) and 2) an image that contains the projected color of the 3D scene point. You can choose the step size for translation (how much the scene point moves at each iteration in the direction of the translation you provide) and the rotation amount as long as they are visible in your movies (small amounts may be undetectable whereas if the values are too large this may cause numerical instability). However, you must restrict your presented results to the following 3 rotation and translation pairs: XZ (you rotate about X, but translate along Z), YX (rotate about Y and translate along X), and ZY (rotate about Z and translate along Y). You will be creating two sequences of images for each of these 3 cases. Matlab provides a simple tool for you to create a movie out of the projected images (see $getframe$ and $writeVideo$). When viewed as movies these sequences should look like the expected cases of panning, tilting or rolling the camera, but with an additional effect due to the translation. [**20 marks**]

   Hint: When you rotate and translate, some points on the initial image frame may go out of frame (and actually might end up really far from the initial frame). You should disregard these and show only the projected points that remain in the initial image frame.