Assignment 3: CS 763, Computer Vision

Due 15th March before 11:55 pm

Remember the honor code while submitting this (and every other) assignment. You may discuss broad ideas with other student groups or ask me for any difficulties, but the code you implement and the answers you write must be from members of the group. We will adopt a zero-tolerance policy against any violation.

Submission instructions: You should ideally type out all the answers in Word (with the equation editor) or using Latex. In either case, prepare a pdf file. For assignment submission, follow the instructions for arrangement of folders and subfolders as given in `https://www.cse.iitb.ac.in/~ajitvr/CS763_Spring2017/HW3/HW3_VideoStabilization_RobustAlignment.rar`. Put the pdf file and the code for the programming parts all in one zip file. The pdf file should contain instructions for running your code. Name the zip file as follows: A3-IdNumberOfFirstStudent-IdNumberOfSecondStudent.zip. If you are doing the assignment alone, the name of the zip file should be A3-IdNumber.zip. Upload the file on moodle BEFORE 11:55 pm on 15th March. Late assignments will be assessed a penalty of 25% per day late. Please preserve a copy of all your work until the end of the semester.

1. In this exercise, you will implement a software routine to stabilize a video. Videos acquired by handheld cameras or smartphones often appear jerky or shaky due to inevitable motion of the hand during acquisition. This artifact is exacerbated in videos acquired from handheld devices while inside a moving vehicle. The processing of removing the unwanted motion between consecutive frames (while maintaining or preserving the intended motion) is called as video stabilization. In some papers, video stabilization also comprises removal of motion blur, but we will not consider that in this exercise. Your task here is as follows: [50 points for this assignment in total - see details below]

   (a) Download the sample videos from `http://www.cse.iitb.ac.in/~ajitvr/CS763_Spring2017/HW3/`. The videos can be read in MATLAB using the 'mmread' routine from the package `http://www.mathworks.in/matlabcentral/fileexchange/8028-mmread` (Local copy inside appropriate folders at `http://www.cse.iitb.ac.in/~ajitvr/CS763_Spring2017/HW3/`). The routine supports .mp4, .mpeg, .avi formats besides others. Generate shaky versions of these videos using the following MATLAB routines provided in the homework folder: 'generate_shaky_video_TranslationOnly.m' (for 2D translations), 'generate_shaky_video_Rigid.m' (for 2D translations + in-plane rotations). These routines assume the first frame of the video is already 'stable' and apply random motion only to the subsequent frames. Also, the folder contains a MATLAB function called 'displayvideo.m' which takes a video in the form of a 3D array and displays it at a specified rate, and another function called 'writeVideo' which writes a video (in the form of 3D array) to an uncompressed .avi file with a specified frame rate.

   (b) You now have to estimate the motion between frames n and $n-1$ ($1 < n \leq T$) of the shaky video. For this, you should use the SIFT algorithm to (1) detect salient feature points in both the frames and (2) determine matches in between the points of those frames. The code for both these tasks is available at `http://www.cs.ubc.ca/~lowe/keypoints/`. Now, given this set of matching pairs of points produced by the SIFT package, your first task is to estimate the motion in between them - which will (hopefully) be the same as the motion between the frames. You should perform this using two methods: (1) Least squares, and (2) RANSAC. You should repeat this for all pairs of consecutive frames and generate a motion sequence. A motion sequence is a sequence containing the motion parameters at every frame. For instance, assuming a translation+rotation model, the motion sequence acquires the form $\{(t_x^{(n)}, t_y^{(n)}, \theta_x^{(n)})\}_{n=2}^{T}$. [6 points - 3 points each for pure translation, translation + rotation]

(c) For a shaky video, this motion sequence will be very noisy. You should smooth the sequence using a simple averaging filter to generate a smoothed motion sequence. The width of the averaging filter is a user-choice. Make sure your averaging filter is wide enough or the amount of smoothing may be inadequate. Plot two examples of noisy and smoothed sequences in your report for every motion model you experiment with. [4 points]

(d) Now given the smoothed sequence, re-warp the different frames of the shaky video to generate a more stable version of the video. (My hunch is that most of you will find this part to be trickier than what it may initially appear!). View the shaky and stabilized videos together by (1) putting them in a single array of size $2H \times W \times T$ where (H, W) is the size of each frame and T is the number of frames, and (2) using the routine 'displayvideo' mentioned earlier. Also your MATLAB program should write the combined video to a file and give it a sensible, self-explanatory name that you print on screen. [15 points for successfully dealing with 2D translation + 10 points for rotation]

(e) After patting yourself on the back for your hard work :-), it's now time to act as your own critic: What are the limitations of what you have developed? For example, can you think of certain types of videos or motion models or other situations your program is or will be unable to handle? [15 points]

(f) **Your code should prompt the user to specify the path to a folder containing the input data. For this, use the MATLAB command uigetdir.**

(g) **In the final submission, do not submit the shaky videos or stabilized outputs - just submit code which produces such videos, and we will grade accordingly.**

(h) **Tips**

i. In the initial stages, do not process the entire video. Work with just about 25-50 frames (or maybe even 5-10 frames). This will save you a lot of time.

ii. Also work only with translations in the beginning - in fact, just set the Y translation to 0 and work with only the X translation.

iii. When you write the video to an avi file, make sure you specify the same frame rate that the original shaky video had.

iv. The final outputs must be on the complete videos - so that you appreciate the output properly.

v. Sample outputs have been provided for reference in the homework folder - for a video sequence with two different smoothing parameters. Note that you may get better outputs with smoothing parameters of around 30-50.

2. In this task, you will use the RanSaC algorithm to estimate the homography in order to make the estimate resistant to the presence of incorrect point correspondences. The code for RanSaC for various problems including estimation of homographies is available at `http://www.peterkovesi.com/matlabfns/`. You should work with the images in the homework folder and also on any one pair of pictures of an approximately planar scene taken with a real camera. (You should acquire these images yourself and make sure they have small non-overlapping areas to make this more interesting.) Also, in each case, you should warp one of the images so that it aligns with the other one. However, you should not crop the image so that no parts of either image are deleted, and a true mosaic may be obtained. While you should use the RanSaC code as is, I recommend stepping through it to get a feel for what is going on inside. Display all the resulting mosaics in your report. State the number of iterations and all thresholds you used. The code performs a normalization step before computing the homography. Can you guess the reason for this normalization step? **Your code should prompt the user to specify the path to a folder containing the input data. For this, use the MATLAB command uigetdir.** [10 (given images) + 10 (your own acquisitions) + 5 + 5]

3. This is a straightforward exercise to make sure you understand the basic update equations in the Horn-Shunck algorithm for optical flow. As seen in class, we seek to minimize the quantity $J(\{(u_{ij}, v_{ij})\})$ w.r.t. the optical flow vectors $(u_{i,j}, v_{i,j})$ at all pixels (i,j), where $J(\{(u_{i,j}, v_{i,j})\}) = \sum_{i=1}^{N}\sum_{j=1}^{N}(I_{x;i,j}u_{i,j} + I_{y;i,j}v_{i,j} +$

$I_{t;i,j})^2 + \lambda((u_{i,j+1} - u_{i,j})^2 + (u_{i+1,j} - u_{i,j})^2 + (v_{i,j+1} - v_{i,j})^2 + (v_{i+1,j} - v_{i,j})^2)$. Setting the partial derivatives w.r.t. $u_{k,l}$ and $v_{k,l}$ to 0, prove that

$$u_{k,l} = \bar{u}_{k,l} - \frac{I_{x;k,l}(I_{x;k,l}\bar{u}_{k,l} + I_{y;k,l}\bar{v}_{k,l} + I_{t;k,l})}{I^2_{x;k,l} + I^2_{y;k,l} + 4\lambda} \tag{1}$$

$$v_{k,l} = \bar{v}_{k,l} - \frac{I_{y;k,l}(I_{x;k,l}\bar{u}_{k,l} + I_{y;k,l}\bar{v}_{k,l} + I_{t;k,l})}{I^2_{x;k,l} + I^2_{y;k,l} + 4\lambda} \tag{2}$$

where $\bar{u}_{k,l}$ and $\bar{v}_{k,l}$ are as defined in the lecture slides. Also verify the Jacobi update equations given by the following:

$$u^{(t+1)}_{k,l} = \bar{u}^{(t)}_{k,l} - \frac{I_{x;k,l}(I_{x;k,l}\bar{u}^{(t)}_{k,l} + I_{y;k,l}\bar{v}^{(t)}_{k,l} + I_{t;k,l})}{I^2_{x;k,l} + I^2_{y;k,l} + 4\lambda} \tag{3}$$

$$v^{(t+1)}_{k,l} = \bar{v}^{(t)}_{k,l} - \frac{I_{y;k,l}(I_{x;k,l}\bar{u}^{(t)}_{k,l} + I_{y;k,l}\bar{v}^{(t)}_{k,l} + I_{t;k,l})}{I^2_{x;k,l} + I^2_{y;k,l} + 4\lambda} \tag{4}$$

[10 points]

4. You know that both the Horn-Shunck as well as Lucas-Kanade methods bank on the brightness constancy assumption. Given a pair of images, let us suppose that this assumption holds good for most physically corresponding pixels, but not for some $p\%$ of the pixels. Briefly explain how you will modify the Horn-Shunck method and Lucas-Kanade method to deal with this. [5+5 = 10 points]