

Assignment 3: CS 763, Computer Vision

Due 19th 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. 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 11:59 pm on 19th 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. Consider a surface with Lambertian reflectance map, known geometry (example: sphere) and unknown but constant albedo. Given an image of such a surface taken with a point light source of unknown power, show how you will determine the lighting direction. Assume there are no shadows. Write down all necessary equations. [4 points]

ANSWER:

We have $I(x_i, y_i) = L\rho\mathbf{N}_i^T\mathbf{d}$ for all M points with coordinates (x_i, y_i) and with normal vector \mathbf{N}_i . The lighting direction \mathbf{d} (a unit vector) is unknown, as are L and ρ . Let us define the vector $\mathbf{f} = L\rho\mathbf{d}$, giving us $I(x_i, y_i) = \mathbf{N}_i^T\mathbf{f}$ at each point. We can write this as a matrix/vector equation $\mathbf{I} = \mathbf{N}^T\mathbf{f}$ where \mathbf{I} is a $M \times 1$ vector, $\mathbf{N} \in \mathbb{R}^{3 \times M}$ and \mathbf{f} has 3 elements. As the geometry of the object is known, we know all entries in \mathbf{N} . Hence using a pseudo-inverse, we can easily determine \mathbf{f} . Dividing \mathbf{f} by its magnitude, we get \mathbf{d} .

2. A Lambertian object illuminated by a point source has a reflectance map of the form given by

$$R(p, q) = \frac{1 + pp_s + qq_s}{\sqrt{1 + p_s^2 + q_s^2}\sqrt{1 + p^2 + q^2}} \quad (1)$$

where the surface normal is $(-p, -q, 1)$ and the light source direction is $(-p_s, -q_s, 1)$. What value(s) of (p, q) will maximize $R(p, q)$? For what values, will you get $R(p, q) = 0$? [4 points]

ANSWER:

When the vector $(p, q, 1)$ and $(p_s, q_s, 1)$ are maximally aligned, their dot product will be maximum. That will maximize $R(p, q)$. Hence the first answer is $p = p_s, q = q_s$. The denominator is always positive, so if $R(p, q) = 0$, we can say that $pp_s + qq_s + 1 = 0$, i.e., when the vectors $(p, q, 1)$ and $(p_s, q_s, 1)$ are perpendicular. Also, if the angle between them is obtuse, the light does not reach the point with normal $(p, q, 1)$ and hence $R(p, q)$ is set to 0.

3. Consider K images of a stationary Lambertian object captured from an orthographic camera at a fixed viewpoint, under K different lighting directions and lighting intensities respectively. For the i^{th} lighting condition, we write $I_i(x, y) = L_i\rho(x, y)\mathbf{N}^t(x, y)\mathbf{d}_i$, where $1 \leq i \leq K$, \mathbf{d}_i is a lighting direction, L_i is the power of the light source, $\rho(x, y)$ is the albedo and $\mathbf{N}(x, y)$ is the unit surface normal vector at pixel (x, y) (we will assume that there were no shadows at any pixel in any image). We can combine all K equations to write the relation $\mathbf{I} = \tilde{\mathbf{N}}\tilde{\mathbf{D}}$. Here $\mathbf{I} \in \mathbb{R}^{M \times K}$ is a matrix whose i^{th} column ($1 \leq i \leq K$) contains the i^{th} image

in vectorized form. $\tilde{\mathbf{N}} \in \mathbb{R}^{M \times 3}$ is a matrix whose j^{th} row contains the product of the unit surface normal vector at some point and the albedo at that point, i.e. $\tilde{\mathbf{N}}_j$ (the j^{th} row of $\tilde{\mathbf{N}}$) is given by $\tilde{\mathbf{N}}_j = \rho(x, y)\mathbf{N}(x, y)$ where (x, y) will go to row j when the image is vectorized. $\tilde{\mathbf{D}} \in \mathbb{R}^{3 \times K}$ is a matrix whose i^{th} column contains $L_i \mathbf{d}_i$. Now answer the following:

- (a) Show that \mathbf{I} has rank 3 in the absence of noise. [4 points]

ANSWER: Given $\mathbf{I} = \tilde{\mathbf{N}}\tilde{\mathbf{D}}$, we see that \mathbf{I} has rank 3 as both the matrices $\tilde{\mathbf{N}}$ as well as $\tilde{\mathbf{D}}$ have rank 3 (assuming that all the lighting directions are not coplanar, and assuming that the surface normals do not all point in the same direction - in which case the rank will be lower than 3).

- (b) We have seen (or will soon see) the photometric stereo problem in class where we estimate surface normals from the images under different lighting conditions, assuming the lighting directions were known. Now suppose, we did not know the lighting directions $\{\mathbf{d}_i\}$, the light source intensities $\{L_i\}$, the albedo at each point on the surface of the object $\{\rho(x, y)\}$, and the surface normals $\{\mathbf{N}(x, y)\}$. Given \mathbf{I} , we can perform the SVD to ‘find’ $\tilde{\mathbf{N}}$ and $\tilde{\mathbf{D}}$. But the decomposition will be unique only up to an unknown 3×3 invertible transformation \mathbf{A} . (There is a very interesting similarity between this problem and the Tomasi-Kanade factorization in structure from motion.) Now, consider that the albedo at some $m > 1$ points was known. Show how this information can help you make the decomposition unique up to an unknown orthonormal transformation \mathbf{R} . What is the minimum m needed? What will happen if you didn’t know the actual albedo values at these m points, but only knew that they were all equal? [4 points]

ANSWER: The decomposition is unique up to an unknown 3×3 invertible matrix \mathbf{A} . The magnitude of every row of $\tilde{\mathbf{N}}\mathbf{A}$ is equal to the albedo of the corresponding point. Since we know the albedo at $m > 1$ points, we have m equations of the form $\tilde{\mathbf{N}}_{i_k} \mathbf{A} \mathbf{A}^T \tilde{\mathbf{N}}_{i_k}^T = \rho_{i_k}^2$ where i_k indicates the k^{th} location whose albedo is known. Given at least 9 such equations, we can solve for \mathbf{A} using Newton’s method as described in class when we studied structure from motion. But all such solutions for \mathbf{A} are unique up to an unknown orthonormal transformation \mathbf{R} because $\mathbf{A}\mathbf{A}^T = \mathbf{A}\mathbf{R}\mathbf{R}^T\mathbf{A}^T$. Once you know \mathbf{A} , you can easily compute the surface normals and hence the albedos, as well as the lighting directions and the lighting intensity. (Note that one can also solve for $\mathbf{B} = \mathbf{A}\mathbf{A}^T$ directly, and since \mathbf{B} is symmetric, there are only 6 independent entries. However, as $\mathbf{B} = \mathbf{A}\mathbf{A}^T$, it must also necessarily be positive semi-definite. This constraint will not be imposed in the solution for \mathbf{B} and the solution for \mathbf{B} may end up not being positive semi-definite - especially if there is noise.) Now, if you didn’t know the albedo at those m points, but knew that the albedo values were equal, you can estimate A only up to an unknown constant. You will still be able to compute the surface normals, but the albedos and the lighting intensity will be estimated only up to an unknown constant.

- (c) Instead of marking out points with equal albedo, suppose you were told that the intensity of the light source in $m > 1$ images was known. Show again how this information can help you make the decomposition unique up to an unknown orthonormal transformation \mathbf{R} . What is the minimum m needed? What will happen if you didn’t know the actual intensities, but only knew that they were all equal? [4 points]

ANSWER: The answer is very similar to the previous part, except that the equations have the form $\tilde{\mathbf{d}}_{i_k} \mathbf{A}^T \mathbf{A} \tilde{\mathbf{d}}_{i_k} = L_{i_k}^2$. Again $m > 9$.

4. 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: [30 points for this assignment in total - see details below]

- Download the sample videos from http://www.cse.iitb.ac.in/~ajitvr/CS763_Spring2016/HW3/SampleVideos/. The videos can be read in MATLAB using the ‘mmread’ routine from the package <http://www.mathworks.in/matlabcentral/fileexchange/8028-mmread> (Local copy at <http://www.mathworks.in/matlabcentral/fileexchange/8028-mmread>)

www.cse.iitb.ac.in/~ajitvr/CS763_Spring2016/HW3/MMread/). 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) or 'generate_shaky_video_Affine.m' (for affine motion in 2D). 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.

- 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 (which we are currently studying in class). 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 - 2 points each for pure translation, translation + rotation, and affine]
- 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]
- 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 is 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 logical name that you print on screen. [8 points for successfully dealing with 2D translation + 7 points for rotation + 5 BONUS points for the affine case]
- 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? [5 points]
- Tips: In the initial stages, do not process the entire video. Work with just about 25-50 frames. This will save you a lot of time. Also work only with translations in the beginning - in fact, just set the Y translation to 0 and work with only the X translation. When you write the video to an avi file, make sure you specify the same frame rate that the original shaky video had.

ANSWER: The code is uploaded in the homework folder. Problems with our implementation are: (1) We are too dependent on the first frame. If there are errors in the first frame, the motion estimates will be erroneous and those errors will build up over time. (2) The feature-point based motion estimation will fail in the event that there are too few feature points detected in each frame. In such a case, an image similarity based method would need to be used. (3) There is a small loss of field of view while rendering the stabilized video. This causes the borders of each frame to flicker. This situation can be remedied by performing some sort of image mosaicing.