ECE549 / CS543 Computer Vision: Assignment 1

Instructions

1. Assignment is due at **11:59:59 PM on Thursday Feb 20 2020**.
2. Course policies: http://saurabhg.web.illinois.edu/teaching/ece549/sp2020/policies.html.
3. Submission instructions:
   (a) A single `.pdf` report that contains your work for Q1, Q2 and Q3. For Q1 and Q2 you can present your responses in the space provided on pages 2, 3 and 4 (directly typing on PDF, or via LaTeX, or by hand-writing on a print out and scanning it). For Q3 your response should be electronic (no handwritten responses allowed). You should respond to the questions 3(c), 3(d)(i), 3(d)(ii), 3(e), and 3(f) individually and include images as necessary. Your response to Q3 in the PDF report should be self-contained. It should include all the output you want us to look at. You will not receive credit for any results you have obtained, but failed to include directly in the PDF report file. PDF file will need to be submitted to https://www.gradescope.com (Entry Code: **MVZDNW**), and you will need to tag your PDF with where your response to each of the question is.
   (b) You also need to submit code for Q3 in the form of a single `.ipynb` file (with output cleared). Code will need to be submitted to compass2g.
   (c) The LaTeX source is available: http://saurabhg.web.illinois.edu/teaching/ece549/sp2020/mp/mp1-latex.tgz.
   (d) We reserve the right to take off points for not following submission instructions.

Change log

v0	02/06/2020	Creation.

1. **Vanishing Points and Vanishing Lines [10 pts][1].** Consider a plane defined by $\mathbf{N}^T\mathbf{X} = d$, that is undergoing perspective projection with focal length f. Show that the vanishing points of lines on this plane lie on the vanishing line of this plane.

[1] Adapted from Jitendra Malik.

2. **Sphere Under Perspective Projection [30 pts].**[2]

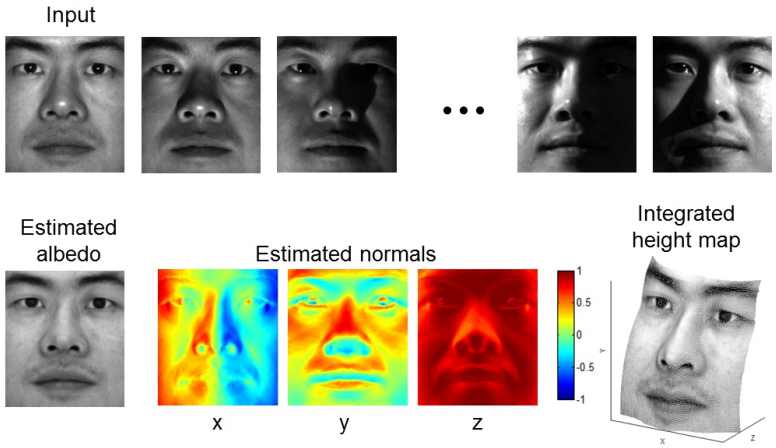
   (a) **[20 pts]** Under typical conditions, the silhouette of a sphere of radius r with center $(X, 0, Z)$ under planar perspective projection is an ellipse. Show that the eccentricity of this ellipse is $\frac{X}{\sqrt{X^2 + Z^2 - r^2}}$. Recall that, under perspective projection a point (X, Y, Z) in 3D space maps to $\left(f\frac{X}{Z}, f\frac{Y}{Z}\right)$ in the image, where f is the distance of the image plane from the pinhole.
   Hint: There are different ways you can solve this. One line of attack would be to compute the lengths of major and minor axes of the projected ellipse, and compute eccentricity via $e = \sqrt{1 - \left(\frac{\text{length of minor axis}}{\text{length of major axis}}\right)^2}$, but there could be other simpler alternatives as well.

[2] Adapted from Jitendra Malik.

(b) **[10 pts]** Are there circumstances under which the projection could be a parabola or hyperbola? If yes, write down the conditions on X, Z, r and f, for parabola and hyperbola respectively; if no, explain why.

3. **Shape from Shading [50 pts]**.[3]

The goal of this problem is to implement shape from shading as described in Lecture 4 and to start becoming comfortable with python image and matrix processing and display functions.



You will need the following python libraries: `numpy`, `matplotlib`, `jupyter`, `Pillow`.

Download the data (http://saurabhg.web.illinois.edu/teaching/ece549/sp2020/mp/mp1-data.tgz), and use this ipython notebook (http://saurabhg.web.illinois.edu/teaching/ece549/sp2020/mp/mp1.ipynb). The data consists of 64 images each of four subjects from the Yale Face database. The light source directions are encoded in the file names. We have provided utilities to load the data and display the output. Your task will be to implement the functions `preprocess`, `photometric_stereo` and `get_surface` in the ipython notebook, as explained below.

(a) For each subject (subdirectory in `croppedyale`), read in the images and light source directions. This is accomplished by the function `LoadFaceImages` which returns the images for the 64 light source directions and an ambient image (i.e., image taken with all the light sources turned off).

(b) Preprocess the data: subtract the ambient image from each image in the light source stack, set any negative values to zero, rescale the resulting intensities to between 0 and 1 by dividing 255 (they are originally between 0 and 255).

Hint: These operations can be done without using any loops. You may want to look into the concept of array broadcasting in numpy.

(c) Estimate the albedo and surface normals. For this, you need to fill in code in `photometric_stereo`, which is a function taking as input the image stack corresponding to the different light source directions and the matrix of the light source directions, and returning an albedo image and surface normal estimates. The latter should be stored in a three-dimensional matrix. That is, if your original image dimensions are $h \times w$, the surface normal matrix should be $h \times w \times 3$, where the third dimension corresponds to the x-, y-, and z-components of the surface normals. To solve for the albedo and the normals, you will need to set up a linear system as shown in slide 19 of Lecture 4.

[15 pts] For each of the four subjects, display your estimated albedo maps and surface normals using `plot_albedo_and_surface_normals`. When inserting results images into your report, you should resize/-compress them appropriately to keep the file size manageable – but make sure that the correctness and quality of your output can be clearly and easily judged. For each subject, you should also report the mean residual of the least square fitting (a single number defined as the squared reconstruction error averaged over all pixels and images).

Hint: To get the least-squares solution of a linear system, use `numpy.linalg.lstsq` function. If you directly implement the formulation of slide 19 of the lecture, you will have to loop over every image pixel and separately solve a linear system in each iteration. There is a way to get all the solutions at once by stacking the unknown g vectors for every pixel into a $3 \times$ npix matrix and getting all the solutions with

[3] Adapted from Svetlana Lazebnik.

a single call to numpy solver. You will most likely need to reshape your data in various ways before and after solving the linear system. Useful numpy functions for this include `reshape`, `expand_dims` and `stack`.

(d) Compute the surface height map by integration. The method is shown in slide 22 of Lecture 4, except that instead of continuous integration of the partial derivatives over a path, you will simply be summing their discrete values. Your code implementing the integration should go in the `get_surface` function. As stated in the slide, to get the best results, you should compute integrals over multiple paths and average the results. You should implement the following variants of integration:

- Integrating first the rows, then the columns. That is, your path first goes along the same row as the pixel along the top, and then goes vertically down to the pixel. It is possible to implement this without nested loops using the `cumsum` function.
- Integrating first along the columns, then the rows.
- Average of the first two options.
- Average of multiple random paths. For this, it is fine to use nested loops. You should determine the number of paths experimentally.

i. **[10 pts]** Discuss the differences between the different integration methods. Specifically, you should choose one subject, display the outputs for all of four variants above with `display_3d` (be sure to choose viewpoints that make the differences especially visible), and discuss which method produces the best results and why. You should also compare the running times of the different approaches.

ii. **[10 pts]** For the remaining subjects, it is sufficient to simply show the output of your best method, and it is not necessary to give running times. Display the 3D screenshots of height maps using `display_3d`. Be sure to choose a viewpoint that makes the structure as clear as possible (and/or feel free to include screenshots from multiple viewpoints).

(e) **[15 pts]** Discuss how the Yale Face data violate the assumptions of the shape-from-shading method covered in the slides. What features of the data can contribute to errors in the results? Feel free to include specific input images to illustrate your points. Choose one subject and attempt to select a subset of all viewpoints that better match the assumptions of the method. Show your results for that subset and discuss whether you were able to get any improvement over a reconstruction computed from all the viewpoints.

(f) **[Upto 10 pts] Extra Credit**

On this assignment, there are not too many opportunities for easy extra credit. This said, here are some ideas for exploration:

- Generate synthetic input data using a 3D model and a graphics renderer and run your method on this data. Do you get better results than on the face data? How close do you get to the ground truth (i.e., the true surface shape and albedo)?
- Investigate more advanced methods for shape from shading or surface reconstruction from normal fields.
- Try to detect and/or correct misalignment problems in the initial images and see if you can improve the solution.
- Using your initial solution, try to detect areas of the original images that do not meet the assumptions of the method (shadows, specularities, etc.). Then try to recompute the solution without that data and see if you can improve the quality of the solution.

In your report describe the improvements you tried along with relevant implementation details. Describe the results you obtained using images, visualizations, and supporting quantitative metrics, as necessary.