# Homework 01

due March 19, 2020

Please submit your assignments through BlackBoard. To get you ready for your final project report, you need to use LaTeX to generate the PDF. Also, include all your code and a README that describes how your code can be run. We prefer running one script that generates all your results. Explicitly add any dependencies you may need to your submission.

## 1 Photometric Stereo/Shape-from-Shading [60 pts]

As discussed in class, photometric stereo is the process of estimating 3D normals and points of a surface, when that surface is imaged under different lighting conditions. See example images of five surfaces in Figure 1. These images, as well as, their corresponding light directions are included in the attached MATLAB environment, which contains a `code` and `data` directory. The main script in the environment is `evalCode.m`. You will use that skeleton code to implement functions `photometricStereo.m` and `getSurface.m`, which compute the 3D normals and albedo of a surface and reconstruct this surface's 3D point cloud, respectively.

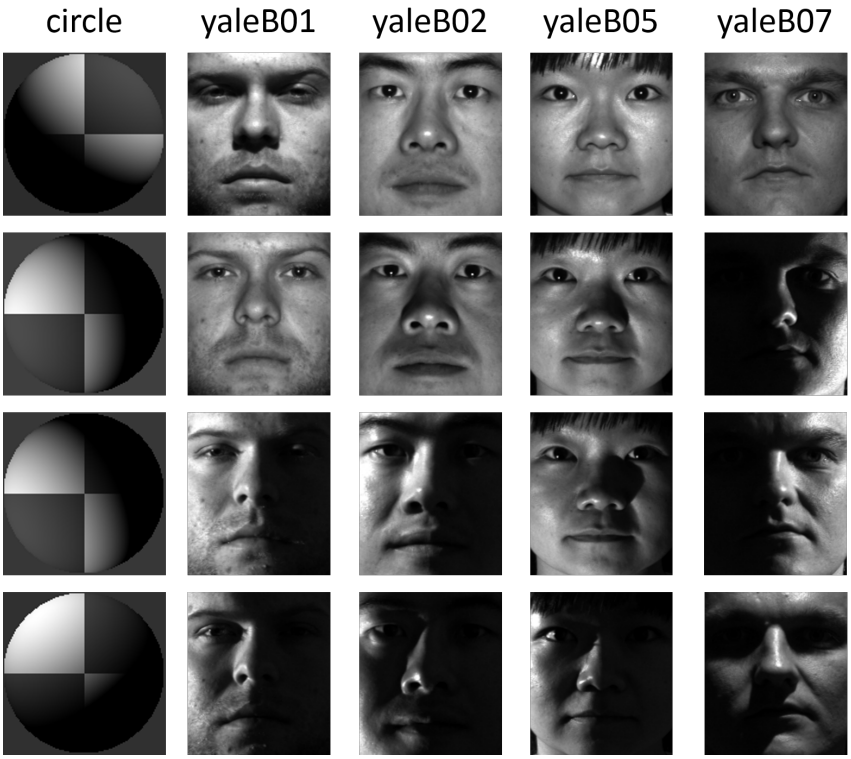


Figure 1: Images of the same object viewed under different lighting conditions (source at infinity)

(a). **[15 pts]** For the `photometricStereo()` function, you should write the most efficient code (*e.g.* can you solve for the albedo and normals in one line of code using simple matrix op-

erations?). Apply this code to the five surfaces: `circle`, `yaleB01`, `yaleB02`, `yaleB05`, and `yaleB07`.

(b). **[15 pts]** As an image, plot the estimated albedo for each of the five surfaces and comment on its quality. Remember that the albedo is a function of the material of a surface and not its geometry. Since the normals are in 3D ($\mathbf{N}_x$, $\mathbf{N}_y$, and $\mathbf{N}_z$), they can be displayed as a colored RGB image, where each channel corresponds to a shifted and scaled dimension of the original normal. Given that normals are of unit length, convert each normal to the range $[0, 1]^3$ and then plot the estimated normals as an RGB image for each surface. You can use the code provided in the environment for this.

(c). **[15 pts]** Use the albedo and normals you computed to generate the image of each surface for three light directions: $\mathbf{s}_1 = [0 \quad -1 \quad 1]^\top$, $\mathbf{s}_2 = [0 \quad 1 \quad 1]^\top$, and $\mathbf{s}_3 = [1 \quad 1 \quad 1]^\top$. Plot these synthetic images and comment on why they should look like this.

(d). **[15 pts]** For the `getSurface()` function, write code to estimate the 3D points of the surface using the line integration method in the textbook (Algorithm 2.1 in Chapter 2). Apply your code to the normals you generated previously and plot your reconstructed point cloud, *e.g.* using the *surf* command in MATLAB. Also, write the 3D points (with the corresponding intensity from one of the surface's images) to an `obj` file and visualize it using MeshLab, a popular 3D viewer. Generate snapshots from MeshLab of your reconstructed point cloud (for each surface) and add them to your report.

## 2 Discrete Fourier Transform [10pts]

Match natural images (a)-(e) to their Fourier amplitude spectra images (f)-(j) *without* computing Fourier transforms. Explain choices.

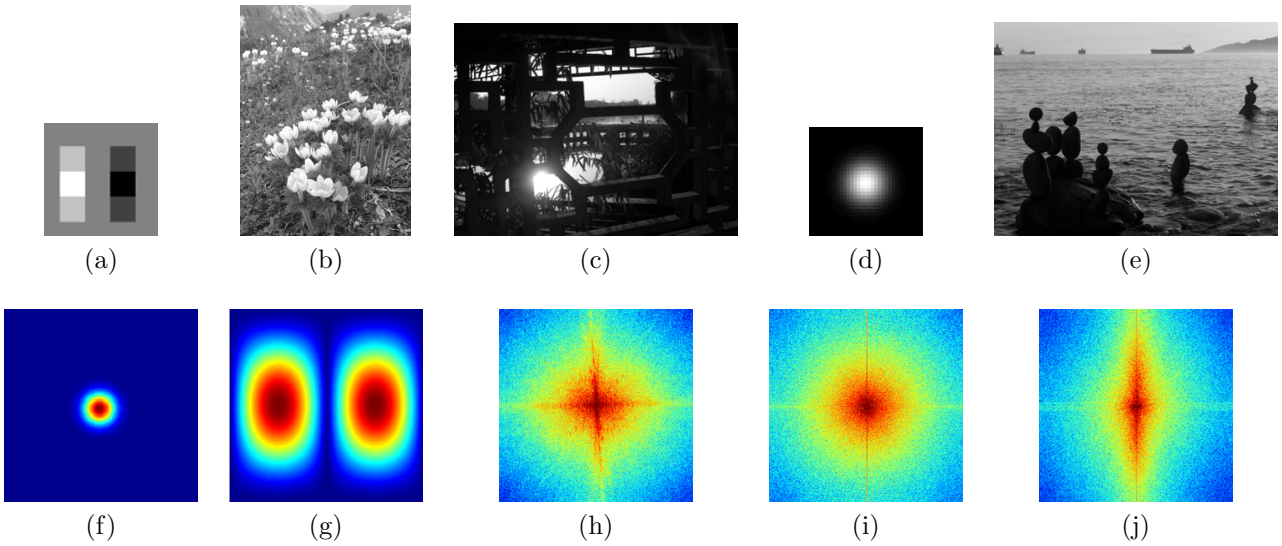


Figure 2: First row shows natural images. Second row shows their (shifted) FFT amplitudes (log scale).

# 3 Image Reconstruction from Phase Information Only [30pts]

In this problem, we will investigate the effect of phase information on signal reconstruction, specifically how it can be used to reconstruct gray-scale images.

(i). **[15 pts]** Read Oppenheim's seminal work on phase and signal reconstruction, specifically the method used to reconstruct a signal from its phase component only (refer to Figure 9 in the paper). Write code to implement this method and include its pseudocode in your report using the *algorithm* environment.

(ii). **[10 pts]** Apply your code using the provided *lake* and *house* images (refer to Figure 3 for an illustration). Take the *lake* image to be the source of the phase response. To initialize the amplitude response in Oppenheim's method, use the *house* image. By combining the phase of the *lake* with the most recent amplitude response, plot the reconstructed image at iterations 1, 10, and 100, alongside the *lake* image that we aim to reconstruct. Comment on the quality of the reconstruction result.

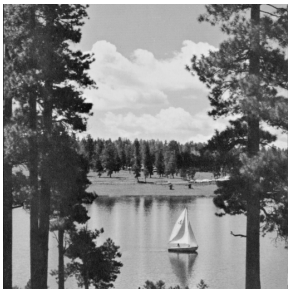

Figure 3: (*left*) Phase image: the image whose Fourier phase response you will use to reconstruct it back again. (*right*) Amplitude Image: the image whose Fourier amplitude response you will use to initialize Oppenheim's phase-only reconstruction method.

(iii). **[5 pts]** Repeat the previous step but with a random generated positive image (*e.g.* using the *rand* command in MATLAB) as the initial amplitude response. Compare the quality of the reconstruction to that in the previous step.