# Introduction

This assignment addresses some of the topics we covered in class. There are two parts to the assignment:

1. Understanding the role of color in image formation and image alignment by translation { Align three separate images representing the R, G and B channels of an image to form the corresponding color image}

2. Calculating the surface normals and estimating the depth from a set of images taken under different lighting conditions.

Please download the homework1.zip file from the course piazza page as it contains images and code needed for the assignment.

# Requirements

You should perform this assignment in Matlab and is due on **Friday October 3rd by 5pm**. You are strongly encouraged to start the assignment early and don't be afraid to ask for help from either the TA or the Instructor. You are also welcome to ask questions about and have discussions about the homework on the course piazza but please do not post your solutions or any closely related material. If there are parts of the assignment that are not clear to you, or if you come across an error or bug please don't hesitate to contact either of the TAs or the Instructor. Chances are that other students are also encountering similar issues.

You are allowed to collaborate with other students as far as discussing ideas and possible solutions. However you are required to code the solution yourself. Copying others' code and changing all the variable names is not permitted! You are not allowed to use solutions from similar assignments in courses from other institutions, or those found elsewhere on the web. If you access such solutions YOU MUST refer to them in your submission write-up. Your solutions should be uploaded to the CSE server using the CSE_SUBMIT command (There is a tutorial on how to use CSE_SUBMIT) on the course piazza page.

Your submitted zipped file for this assignment should be named **LastnameFirstname_hw1.zip**. Failure to follow this naming convention will result in your file not being being picked up by the grading script. Your zipped file should contain: (i) a PDF file named Lastname-Firstname_hw1.pdf with your report, showing output images for each part of the assignment and explanatory text, where appropriate; (ii) the source code used to generate the solutions (with code comments), along with a running script (named runfile_hw1.m). We should be able to cut and paste from your runfile to execute the code for each part of the assignment in turn.

Figure 1: A sample from the Prokudin-Gorskii photo collection: three separate plates for each color channel and the resulting color image.

## Problem 1. Colorizing the Prokudin-Gorskii photo collection[1]

Sergei Mikhailovich Prokudin-Gorskii (1863-1944) was a photographer who, between the years 1909-1915, traveled the Russian empire and took thousands of photos of everything he saw. He used an early color technology that involved recording three exposures of every scene onto a glass plate using a red, green, and blue filter. Back then, there was no way to print such photos, and they had to be displayed using a special projector. Prokudin-Gorskii left Russia in 1918, following the Russian revolution. His glass plate negatives survived and were purchased by the Library of Congress in 1948. Today, a digitized version of the Prokudin-Gorskii collection is available on- line at http://www.loc.gov/exhibits/empire/gorskii.html.

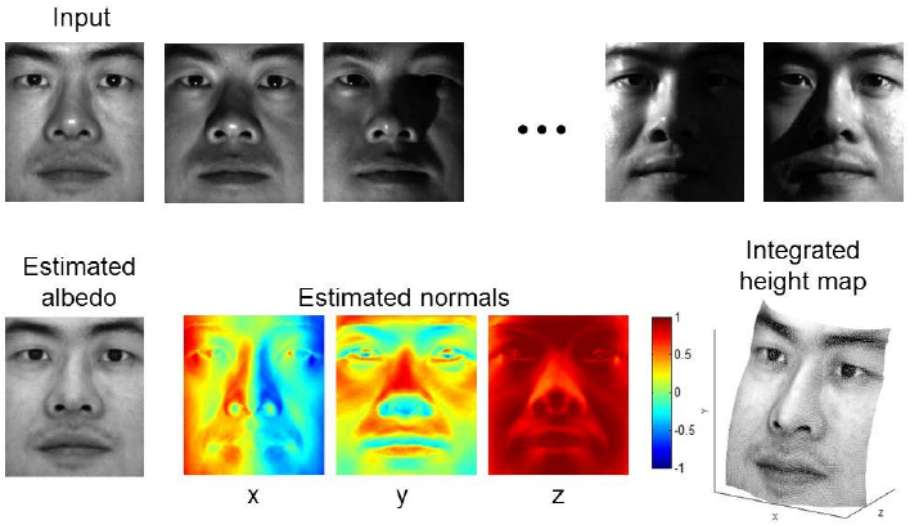
The goal of this assignment is to learn to work with images in Matlab by taking the digitized Prokudin-Gorskii glass plate images and automatically producing a color image with as few visual artifacts as possible. In order to do this, you will need to extract the three color channel images, place them on top of each other, and align them so that they form a single RGB color image. Some starter code is available in part1.m, included in the main zip file. You are welcome to use it, if you find it helpful. The zip file contains six images (part1 *.jpg) that you should run your algorithm on. Note that the filter order from top to bottom is BGR, not RGB!

Your program should take a glass plate image as input and produce a single color image as output. The program should divide the image into three equal parts and align the second and the third parts (G and R) to the first (B). For each image, you will need to print the (x,y) displacement vector that was used to align the parts. The easiest way to align

[1]We adapted this problem from Svetlana Lebsnick with permission, though originally defined by Aloysha Efros at CMU

the parts is to exhaustively search over a win- dow of possible displacements (say [-15,15] pixels), score each one us- ing some image matching metric, and take the displacement with the best score. There is a number of possible metrics that one could use to score how well the images match. The simplest one is just the L2 norm also known as the Sum of Squared Differences (SSD) distance which is simply sum(sum((image1-image2)$^2$)). Another is normalized cross-correlation (NCC), which is simply a dot product between two normalized vectors: (image1./‖image1‖ and image2./‖image2‖). See the Matlab function normxcorr2.

You report should show: (i) the output color image for each of the six images in the zip file (please be sure to avoid compressing the images un- necessarily); (ii) the displacement vectors used to align the G and R color channels for each image and (iii) a brief description of your approach.




## Problem 2. Estimating Shape from Shading[2]

The goal of this assignment is to implement shape from shading as described in Lecture 5 from September 5 (more details in Section 2.2.4 of Forsyth & Ponce 2nd edition), and to start becoming comfortable with MATLAB and matrix processing functions.

(a) Download the sample code and data. The data, in the `croppedyale` directory, consists of 64 images each of four subjects from the Yale Face database. The light source directions are encoded in the file names. The code consists of several .m functions. Your task will be to add some code to the top-level script, run_me.m, and to fill in the code in `photometric_stereo.m` and `get_surface.m`, as explained below. The remaining files are utilities to load the input data and display the output.

(b) For each subject (subdirectory in `croppedyale`), read in the images and light source directions. This is accomplished by the function LoadFaceImages.m provided in the zip file (which, in turn, calls getpgmraw.m to read the PGM files). LoadFaceImages returns the images for the 64 light source directions and an ambient image (i.e., image taken with all the light sources turned off).

(c) 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 (they are originally between 0 and 255).

(d) Estimate the albedo and surface normals. For this, you need to fill in code in `photometric_stereo.m`, which is a function taking as input the image stack corresponding to the different light source directions and the matrix of 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 x w`, the surface normal matrix should be `h` $\times$ `w` $\times$ `3`, where the third

[2]Again, we adapted this problem from Svetlana Lebsnick at UIUC with permission

dimension corresponds to the `x-, y-`, and `z-`components of the normals. To solve for the albedo and the normals, you will need to set up a linear system as shown in Lecture 5.

(e) Compute the surface height map by integration by summing up the discrete values of the partial derivatives over a patch. Your code implementing the integration should go in the `get_surface.m` file. To get the best results, you should compute integrals over multiple paths and average the results.

You should turn in both your code and a report discussing your solution and results. For full credit, your report should include a section for each of the following questions:

(a) Briefly describe your implemented solution, focusing especially on the interesting parts of the implementation. What are some artifacts and/or limitations of your implementation, and what are possible reasons for them?

(b) For every subject, display your estimated albedo maps and screenshots of height maps (feel free to include screenshots from multiple viewpoints).

(c) Discuss how the Yale Face data violate the assumptions of the shape-from-shading method covered in the slides and discussed in the textbook. Feel free to include specific input images to illustrate your points.