Problem 1. Colorizing the Prokudin-Gorskii photo collection

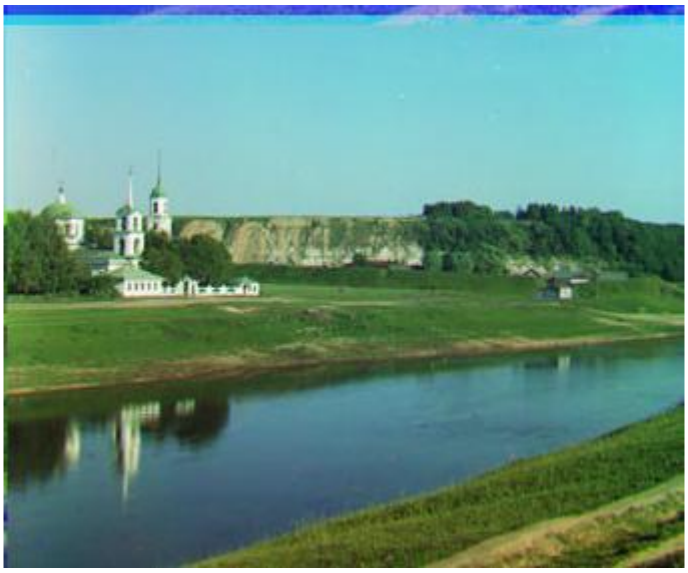
OBJECTIVE: To work with glass plate images and automatically produce a color image with as few visual artifacts as possible.

IMPLEMENTATION: The input to the program is a glass plate image and the output is a colored image. The approach used to align the images is SSD (Sum of squared distances).

- Initially, the image is read and the height is obtained. The image is divided into three equal parts namely R, G and B.
- Each of the divided image is cropped and sent through the canny edge detector in order to remove the unwanted border and align the images in a more effective way.
- The red and green images are aligned keeping the blue image constant. The red and green image along with the blue image is passed through the align function where the offset between the two images is computed by making a call to the offset function. In the offset function, iterating over a window size of 30, the displacement is computed using SSD.
- The image is then shifted by the offset computed in the previous step.
- The red, green and blue images are combined into one using the cat function and the displayed. Hence, a colored image is obtained from the glass plate.

RESULT: The following sets of images show the colored image of each of the 6 images given along with their displacement vectors.

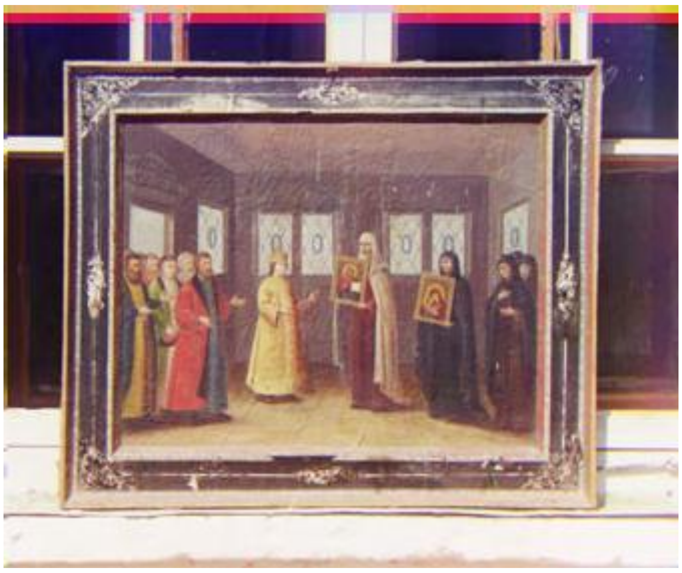
Image 1:



Displacement(Red) : (10,1)

Displacement(Green) : (5,2)

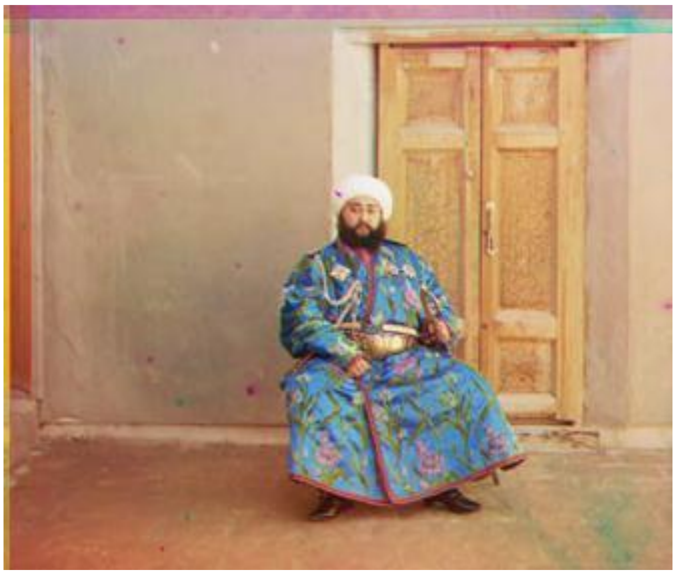
Image 2:



Displacement(Red) : (9,2)

Displacement(Green) : (4,2)

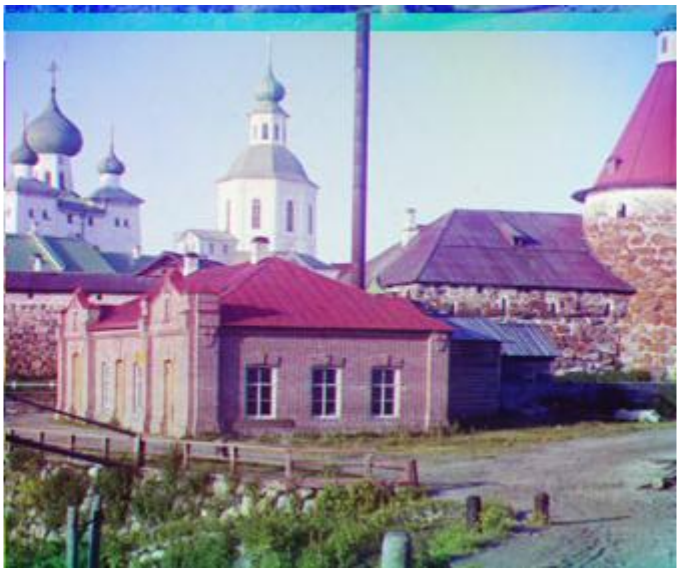
Image 3:



Displacement(Red) : (14,4)

Displacement(Green) : (7,3)

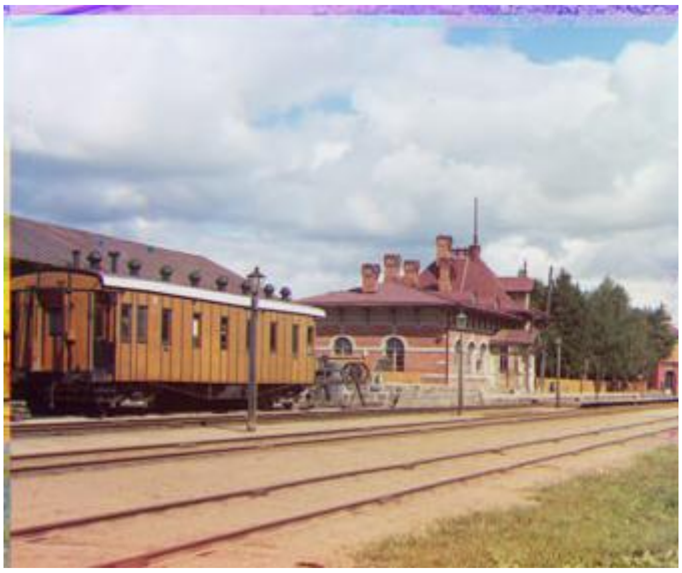
Image 4:



Displacement(Red) : (13,1)

Displacement(Green) : (4,1)

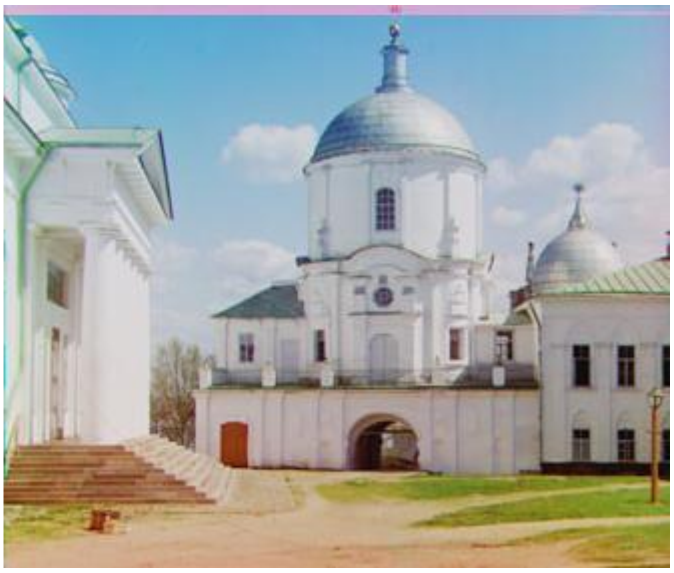
Image 5:



Displacement(Red) : (11,4)

Displacement(Green) : (5,3)

Image 6:



Displacement(Red) : (5,1)

Displacement(Green) : (0,0)

Problem 2. Estimating Shape from Shading

OBJECTIVE: To implement shape from shading by computing the albedo and surface normal of the image.

IMPLEMENTATION: The input is a set of 64 images each of four subjects from the Yale Face database and the output is the shape of the subject.

1. The data is preprocessed by subtracting each image in the set of 64 images by the ambient image in order to remove the ambient illumination. We also ensure that none of the values in the resulting matrix is below and rescale the image.
2. The albedo of the image is estimated using Lambert's law. In the formula given below, Matrix I(x,y) denotes the intensity matrix. Matrix V denotes the light directions and g(x,y) is the surface matrix. Therefore, we divide the intensity matrix by the light directions to obtain the surface matrix. The albedo is the absolute value of the surface matrix.

- For each pixel, set up a linear system:

$$\underbrace{\begin{bmatrix} I_1(x,y) \\ I_2(x,y) \\ \vdots \\ I_n(x,y) \end{bmatrix}}_{\substack{(n \times 1) \\ \text{known}}} = \underbrace{\begin{bmatrix} \mathbf{V}_1^T \\ \mathbf{V}_2^T \\ \vdots \\ \mathbf{V}_n^T \end{bmatrix}}_{\substack{(n \times 3) \\ \text{known}}} \underbrace{\mathbf{g}(x,y)}_{\substack{(3 \times 1) \\ \text{unknown}}}$$

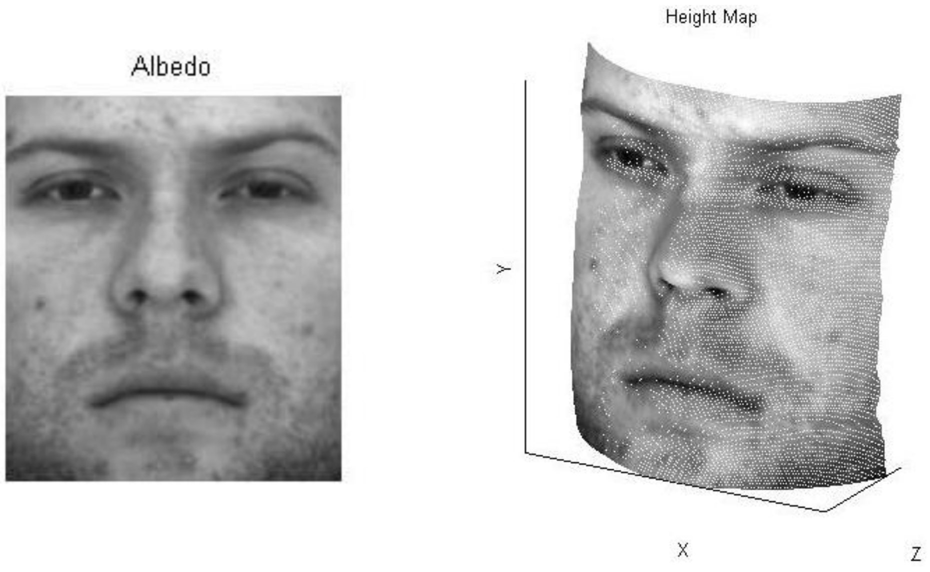
(Taken from: Class notes, LightNShading, Slide 36)

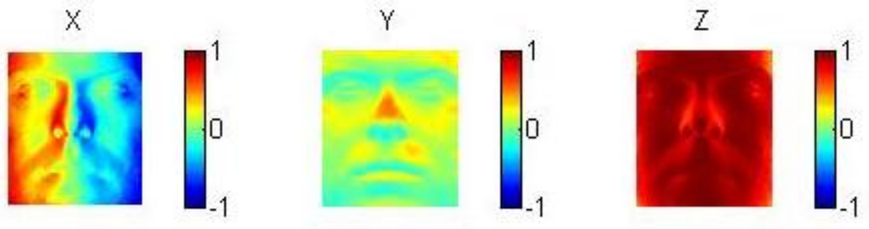
3. Once the surface matrix and the albedo have been obtained, we compute the surface normal by dividing the surface matrix by the albedo because surface normal is a unit vector. Hence, we get the direction i.e., the surface normal.
4. Once the surface normals have been estimated, we can build the height map by integrating by summing up the discrete values of the partial derivatives over a patch using the algorithm given below:

```
Integration:
Top left corner of height map is zero
For each pixel in the left column of height map
   height value = previous height value + corresponding q value
end
For each row
   For each element of the row except for leftmost
      height value = previous height value + corresponding p value
   end
end
```

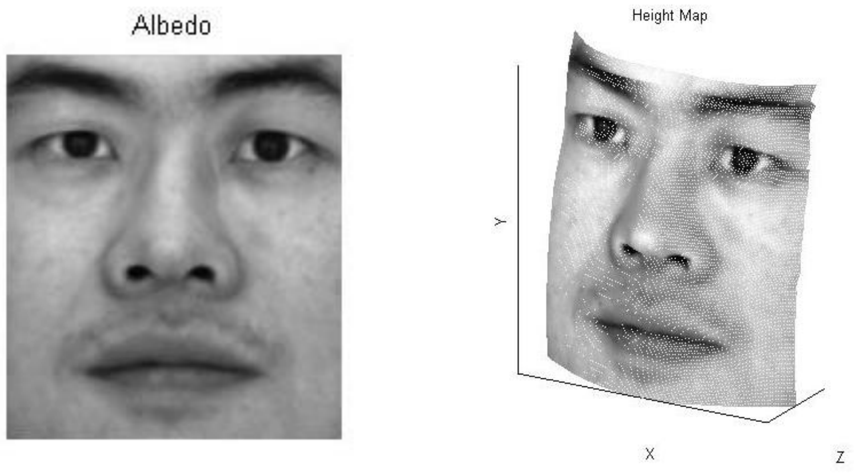
(Taken from: COMPUTER VISION, A MODERN APPROACH, second edition by David A. Forsyth and Jean Ponce, University of Illinois at Urbana-Champaign)

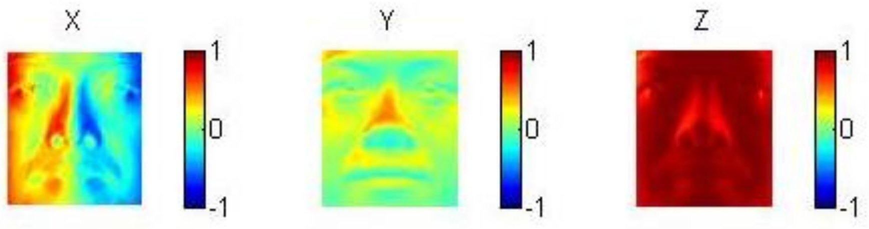
RESULT: **yaleB01**



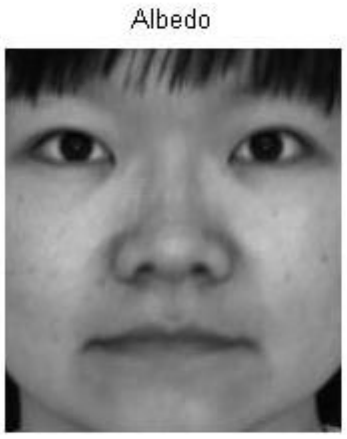


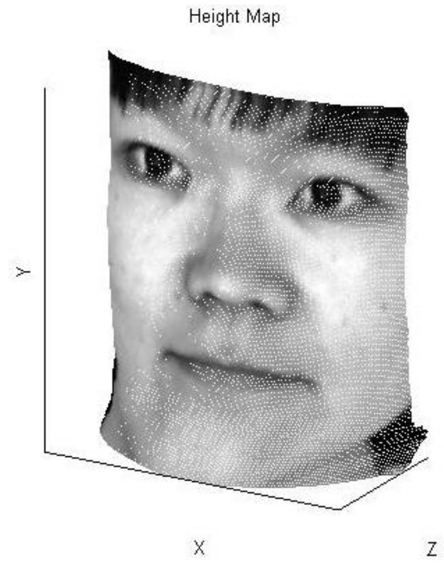
yaleB02

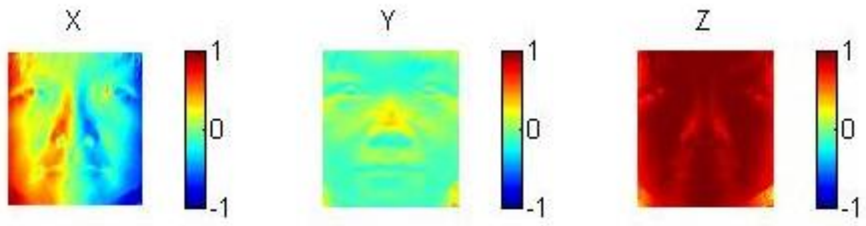




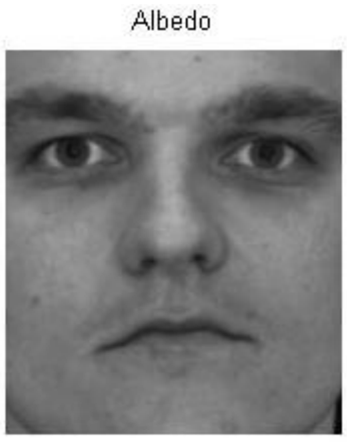
yaleB05

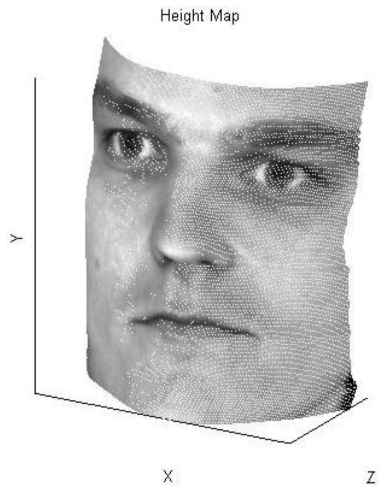


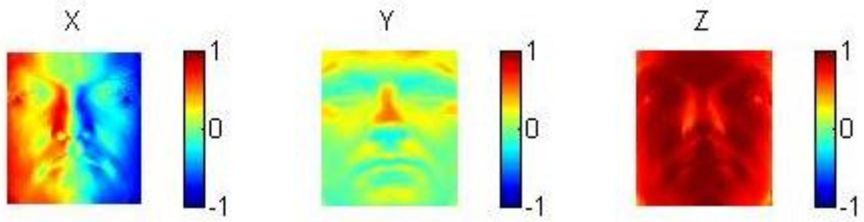




yaleB07







(c) The Yale Face database assumes that the light sources are pointing towards the object and has not taken the other possibilities into consideration. Also, the model also does not take shadows into consideration.