CSE 573 Assignment 1 – Report

Problem 1: Colorizing the Prokudin-Gorskii photo collection1

Solution:

Images:

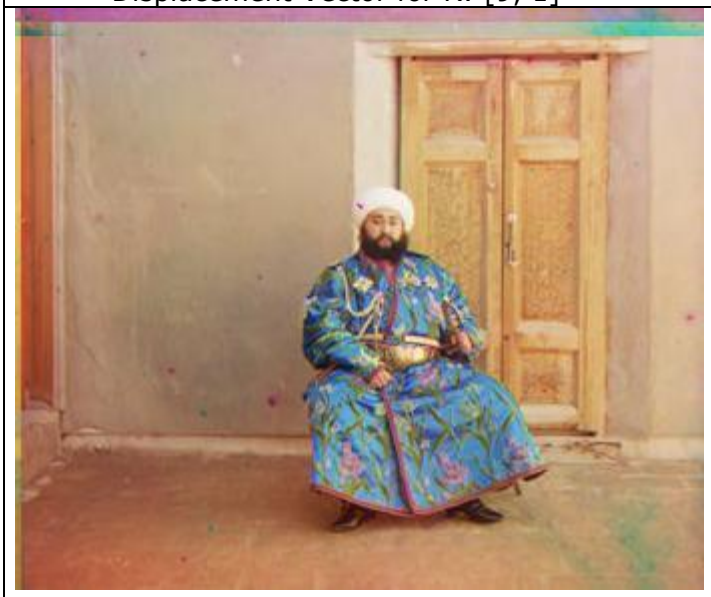
<table>
<tr>
<td> 
Figure 1:Result Image for part1_1</td>
<td> 
Figure 2:Result Image for part1_2</td>
</tr>
<tr>
<td>Displacement Vector for G: [5, 2]
Displacement Vector for R: [9, 1]</td>
<td>Displacement Vector for G: [4, 2]
Displacement Vector for R: [9, 2]</td>
</tr>
<tr>
<td> 
Figure 3:Result Image for part1_3</td>
<td> 
Figure 4:Result Image for part1_4</td>
</tr>
<tr>
<td>Displacement Vector for G: [7, 3]
Displacement Vector for R: [15, 5]</td>
<td>Displacement Vector for G: [4, 1]
Displacement Vector for R: [13, 1]</td>
</tr>
</table>

 *Figure 5: Result Image for part1_5*	 *Figure 6: Result Image for part1_6*
Displacement Vector for G: [5, 3] Displacement Vector for R: [11, 4]	Displacement Vector for G: [0, 0] Displacement Vector for R: [5, 1]

part1.m explanation:

a) Created an array named "imname" which contains all the input file names by using function "cellstr(imname)" which converts array to cell array.
b) Initialized a loop from 1 to 6 so that it can do the processing for all images at once.
c) Read the image using "imread" which converted the image to double precision.
d) Now split the image into 3 parts to get B, G and R image.
e) Clipped all three images by 25 pixel like "B(26:end-25,26:end-25)" and saved in the same image B,G and R respectively.
f) Called the function "getdispvect" saved in file getdispvect.m, created for separately calculating the displacement vector. This function returns displacement vector (DispVec_G) for G.
g) Calculated G1 using "circshift" by shifting the image G by the displacement vector (DispVec_G).
h) Same way called "getdispvect" again to get the displacement function DispVec_R for R.
i) Calculated R1 by shifting R by the displacement vector DispVec_R.
j) Now used "cat" on R1, G1 and B to get the final colored image.
k) Wrote the result file using "imwrite".

getdispvect.m [function [disp_vect] = getdispvect(firstImage, secondImage)]

a) This function is created to calculate the displacement vector with minimum SSD value. It takes two images as input, out of which second image is the base image with respect to which we will shift the first image and calculate the displacement vector for the same.

b) Initialized minSSD (minimum SSD) with infinite and disp_vect (Displacement Vector) with [0,0].
c) Started two loops "i" and "j" from -15 to 15 increasing by one unit to check all the possible values for displacement vector [i,j] and get the same with minimum SSD.
d) Inside the loop, shifting the first image by the displacement [i,j] and calculating SSD value by using "sum(sum((secondImage-temp).^2))". If SSD is less than the minSSD then update minSSD with ssd and disp_vect by the current [i,j].

How did I improve the resulting images?

a) Started with without clipping the images and followed the above mentioned process. All the images were coming blurred.
b) Then clipped the images by 50 pixel and all images except the fat man were coming properly.
c) Again iterated the step b) and clipped the images by 30 pixel. Same result all images except fat man were coming properly.
d) In between noticed that by decreasing the clipping size the fat man image was getting better. This may be due to cutting of the original images from R , G or B. So decreased size further by 5 pixel and all images started to come properly.

Problem 2: Estimating Shape from Shading

Solution:

1. 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?

Ans.

run_me.m

a) Initialized loop from 1 to 64 to process "imarray".
b) Subtracted ambient_image from all images of the array imarray.
c) Then found out the index at which the value is negative using "find" function.
d) Replace all the negative values with zero.
e) Rescale the values in imarray between 0-1 by dividing all the values by 255 (values can be between 0-255).
f) Update imarray with the rescaled image.
g) Call photometric_stereo to get albedo image and surface normal.
h) Call get_surface to get height map by passing surface normal and image size.
i) Call display_output to display albedo image and height map.
j) Call plot_surface_normals to plot the surface normal.

photometric_stereo.m

a) Initialized albedo_image and surface_normals array with zeros.
b) Initialized two loops one inside other, out of which outer one is for rows and the inside one for columns.
c) Getting the intensity values form imarray in 64 X 1 format using reshape function.
d) Convert intensity to a vector for further processing.
e) Now calculating albedo and surface_normals using the equation I = Vg where I is the intensity (point d) and V is light directions already given.
f) Calculate g by solving the equation I = Vg by using mldivide function.
g) Calculate modulus of g using norm(g) function, which is the albedo value for particular (i,j).
h) Check if modulus of g is greater than zero then divide g by |g| to get the surface normal values.
i) Finally return the albedo image and surface normals which are of dimension hXw and hXwX3 respectively.

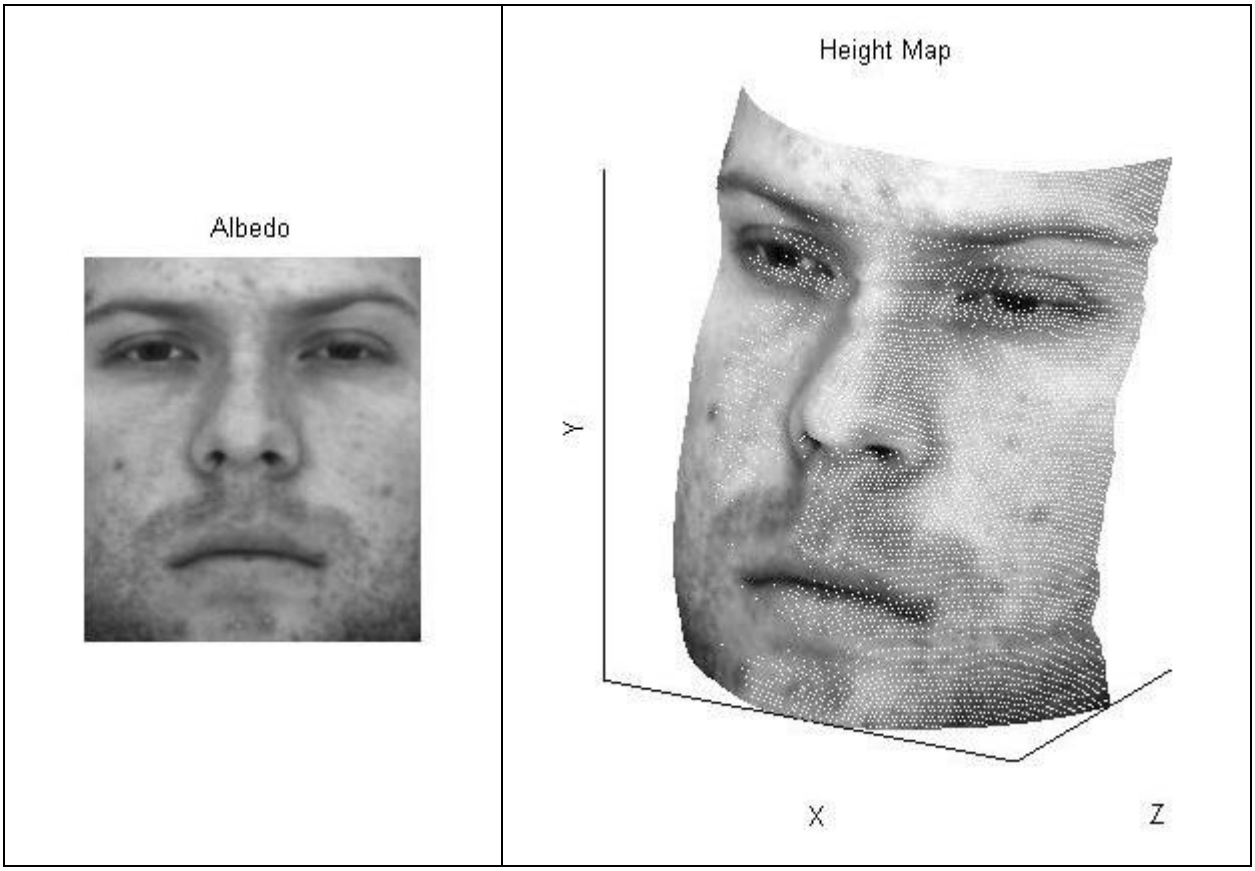
get_surface.m

a) Initialized height map with zeros.
b) Initialized for loop from 2 to 192 for rows considering the values at (1,1) as zero.
c) Calculate q at each point except (1,1) by dividing surface normal value N2 by N3 and add up with the values at last index.
d) Same way calculate p at each point except at (1,1) by dividing surface normal value N1 by N3 and add up with values at last index.
e) Finally return height map by using the above steps for all the indexes.

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

Ans.

yaleB01



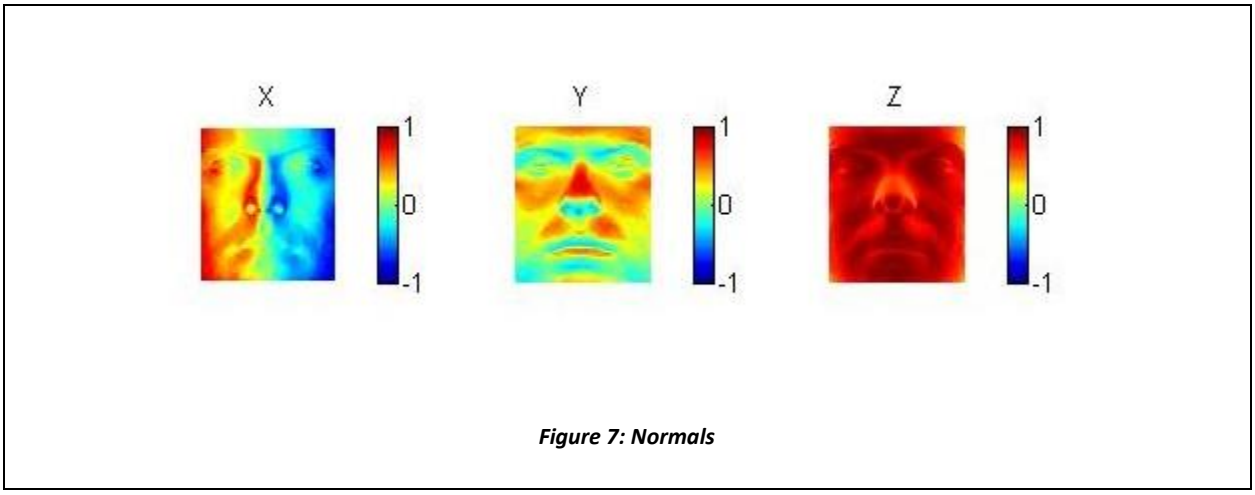
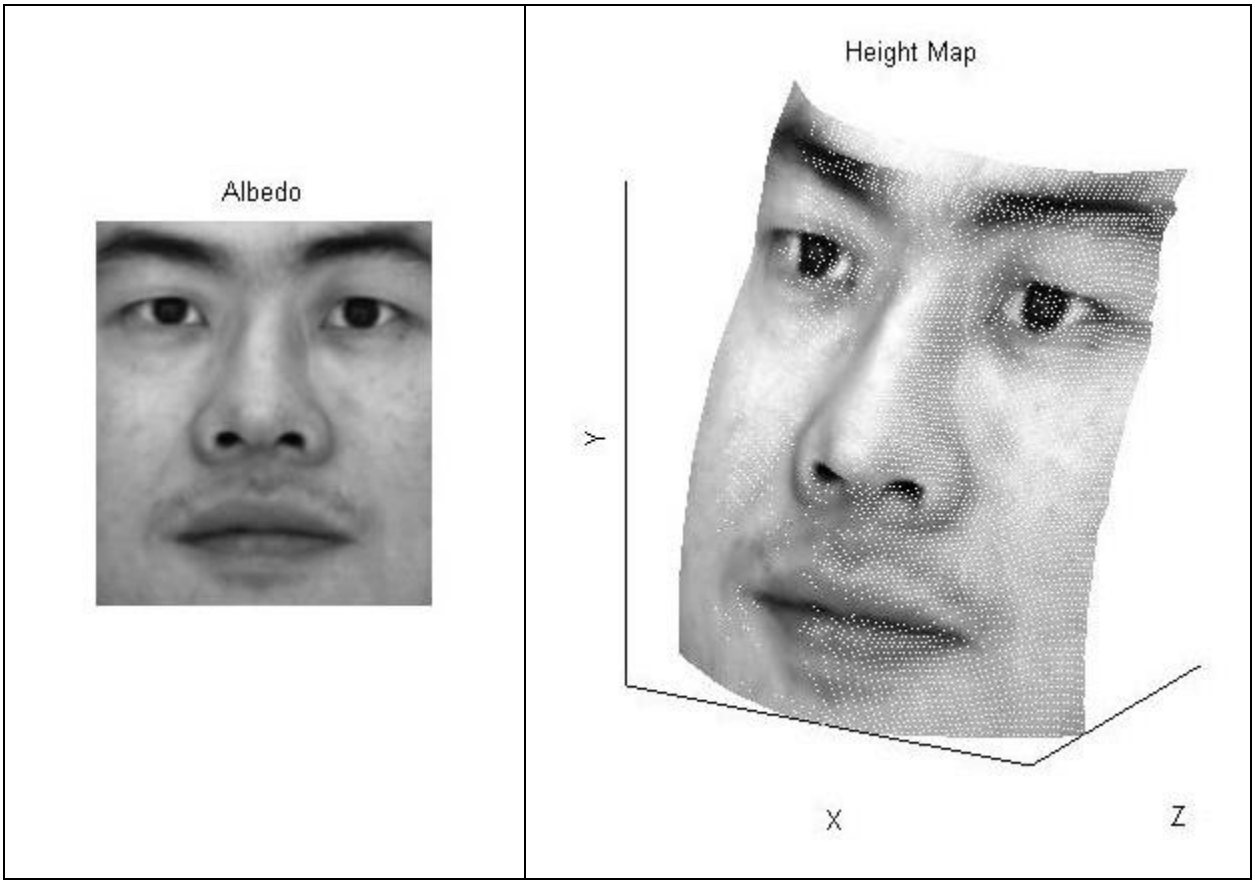


Figure 7: Normals

yaleB02



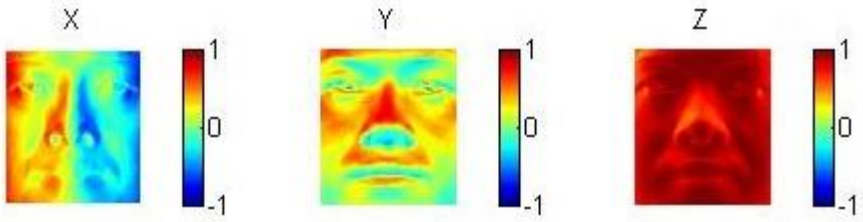
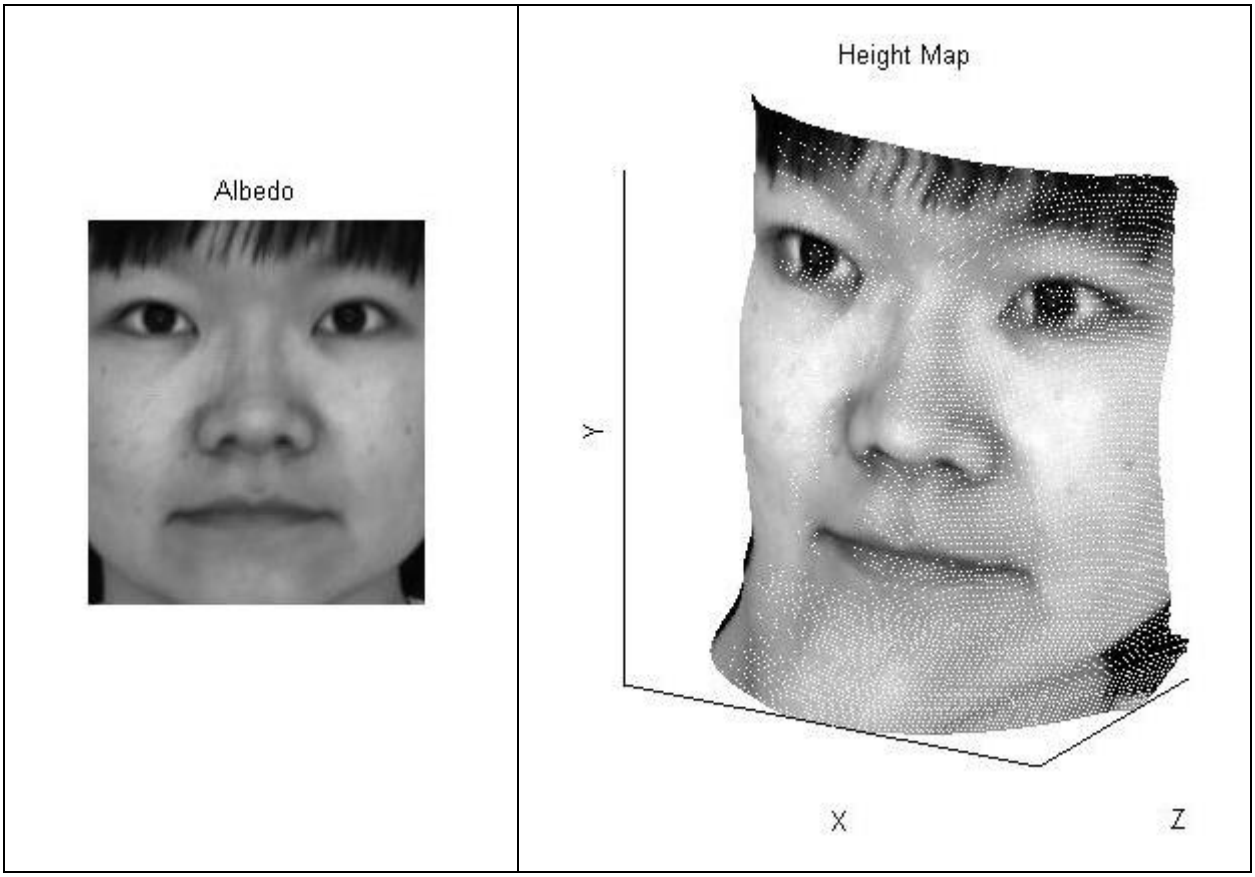


Figure 8: Normals

yaleB05



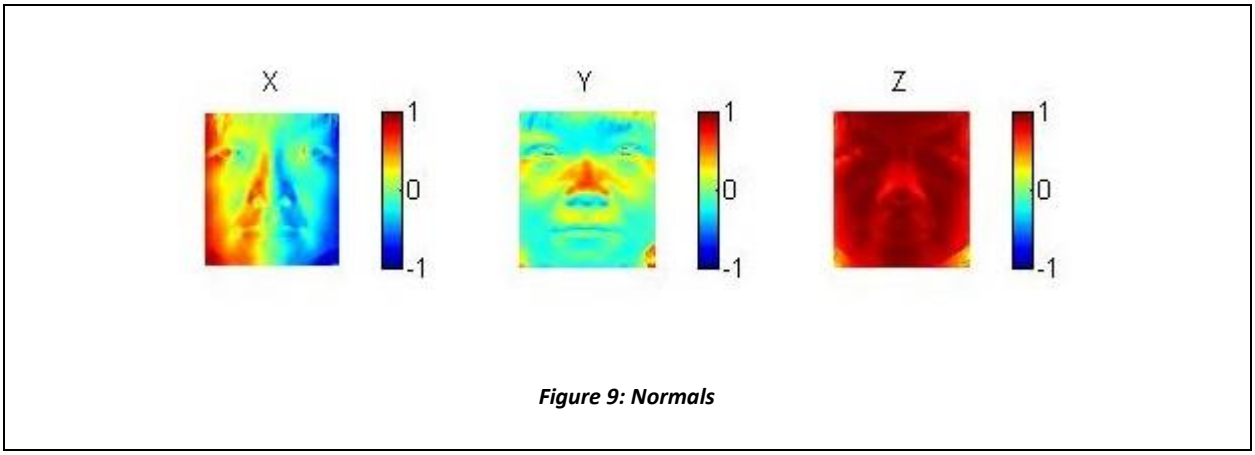


Figure 9: Normals

yaleB07

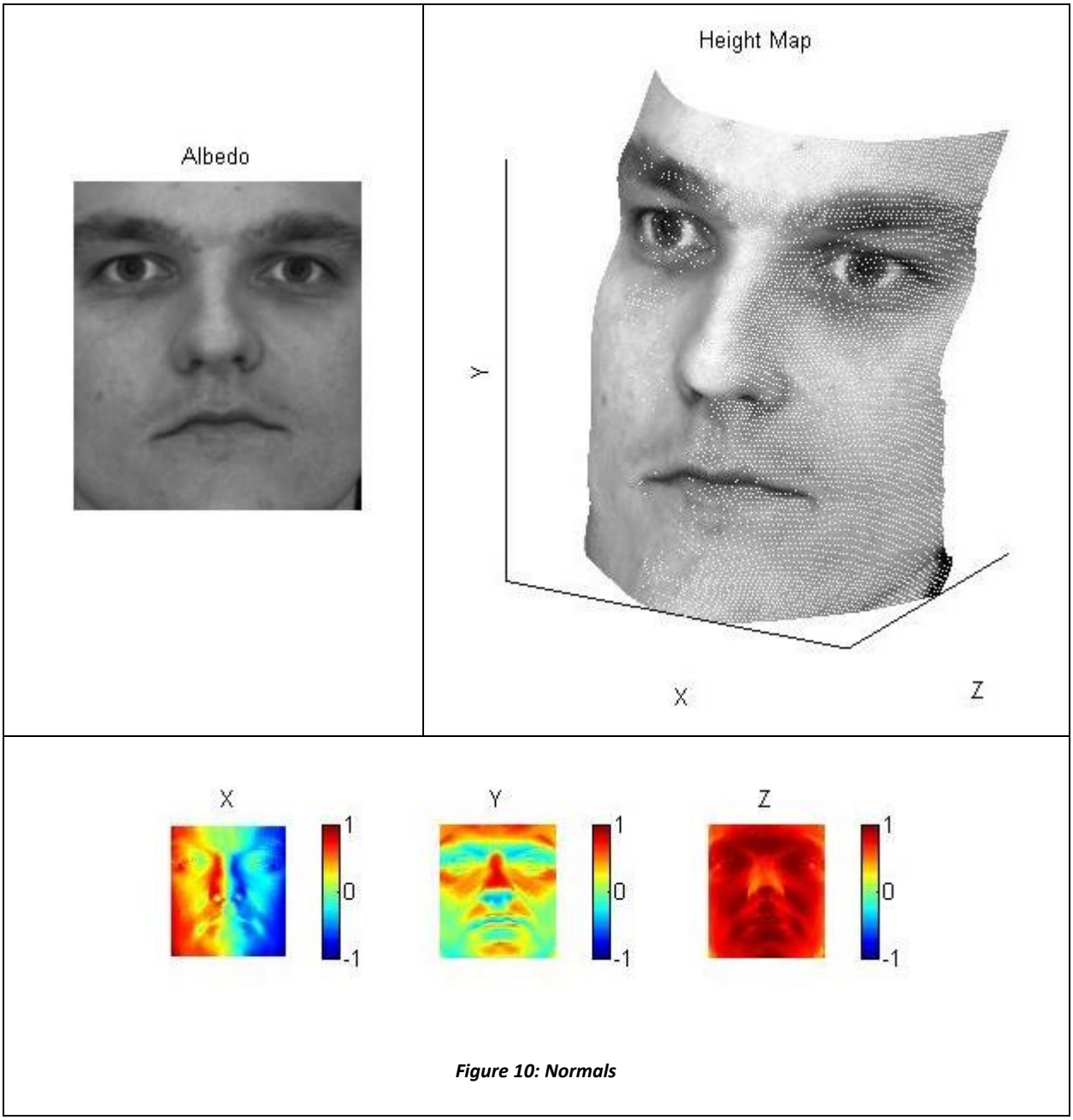
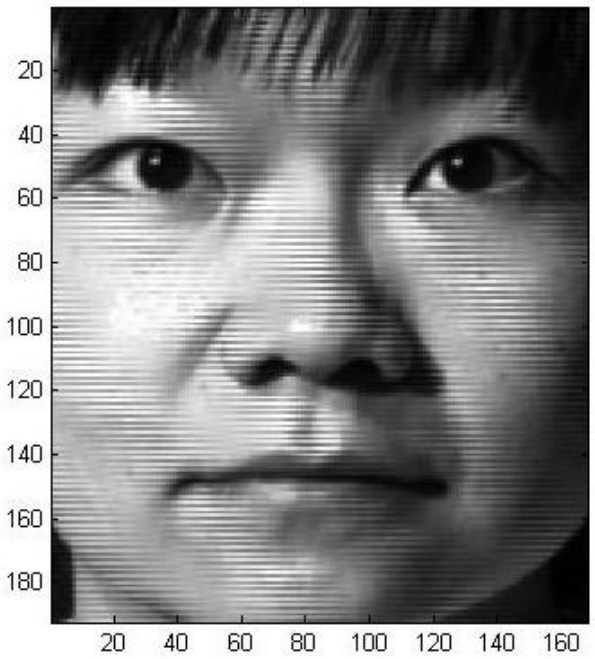


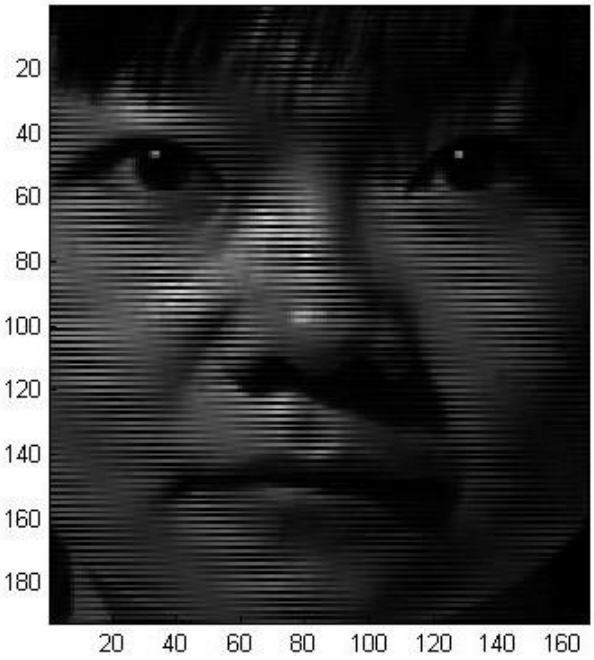
Figure 10: Normals

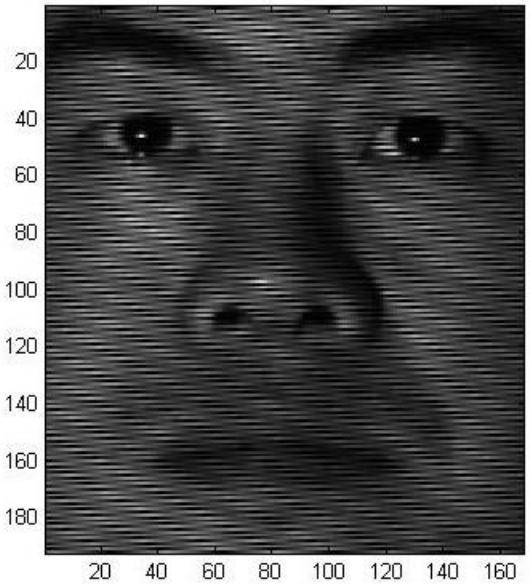
3. Discuss how the Yale Face data violate the assumptions of the shape-from-shading method covered in the slides and discussed in the textbook. Feel free to include specific input images to illustrate your points.

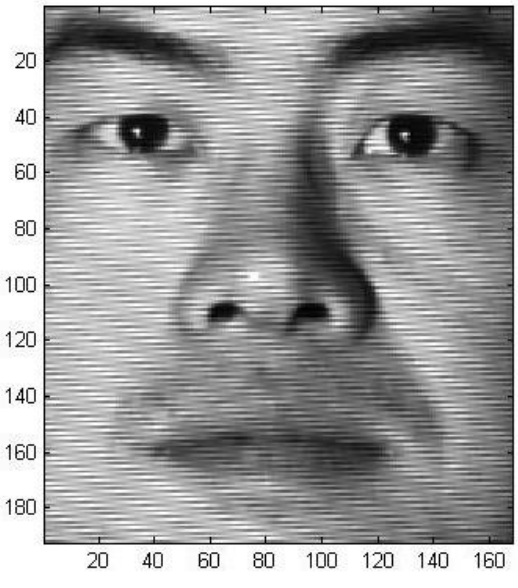
Ans.

a) Noise Images



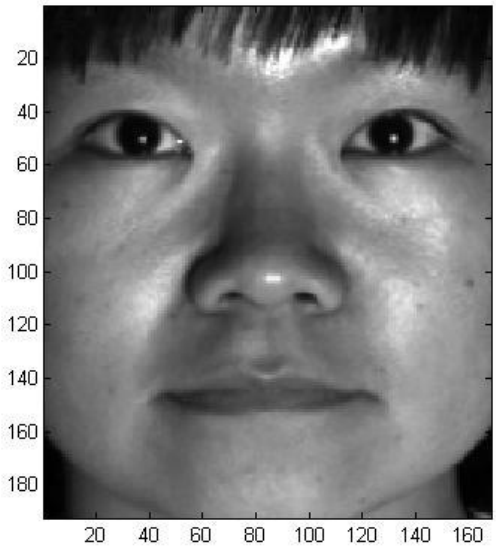






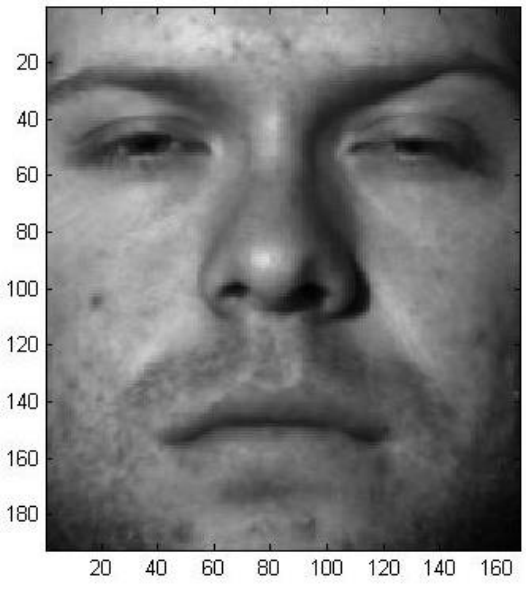
This noise propagates throughout the process which affects the final outcomes for example white dots in the final image.

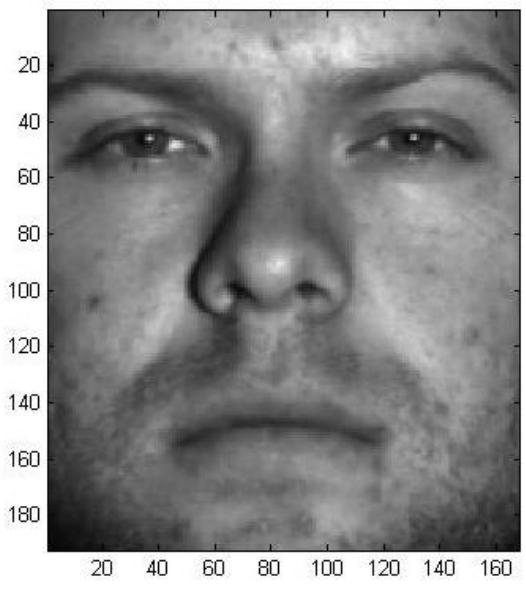
b) Background



Chin in the final image yaleB05 is not coming properly may be because of the background coming in the input images as presented in the above picture.

c) Moving object





The above case can be considered as the moving object just with respect to eyes. Because eyes are in quite different position in both of the above given images.