Assignment 1 Report
Introduction to Computer Vision
Prof. Lee Kyoung Mu
TA: Gyeongsik Moo, Baik Sung Yong David

Nguyen Tuan Nghia
Student ID: 2018-21525
Email: nghianguyentuan@snu.ac.kr
Seoul National University

1 Calibrated Photometric Stereo

In this part, albedo and normal map are computed from given images with known light source directions. In the dataset, there are subsets in each of which there are multiple images of one person taken under various lighting conditions. Each subset also contains an ambient image which can be used as baseline to compute illumination intensity on each point of the images. I used MATLAB to complete this part.

1.1 Preprocessing

At first, I make sure that the working environment is clean by executing following commands:

```
close all;
clear;
clc;
```

Then images are loaded and necessary preprocessing is carried out.

```
size_after_crop_y = 176; % CHANGE THIS: Size y of cropped image
size_after_crop_x = 176; % CHANGE THIS: Size x of cropped image

crop_y = 117; % CHANGE THIS: Start pixel y
crop_x = 257; % CHANGE THIS: Start pixel x

crop_range_y = crop_y:crop_y + size_after_crop_y - 1;
crop_range_x = crop_x:crop_x + size_after_crop_x - 1;

[ambimage, imarray, lightdirs] = LoadFaceImages();

imarray = permute(imarray,[2 3 1]);

imarray = imarray(crop_range_y,crop_range_x,:);
ambimage = ambimage(crop_range_y,crop_range_x);

[width, height, nImages] = size(imarray);

% Subtract the ambient image from other images
% Threshold so that no pixel value is negative
imarray = max(imarray - repmat(ambimage,1,1,nImages),0);

imarray = permute(imarray,[3 1 2]);
```

1.2 3-image method

In this part, I demonstrate computing albedo and normal from exactly 3 images with different known light source directions. I will select a combination of 3 random images to do so. The other light source directions will be used to reconstruct images from computed albedo and normal.

```
% Choose random 3 images to calculate Albedo and Normal
% 3 first indexes will be used
```

```
ridx = randperm(nImages);
train = ridx(1:3);
test = ridx(4:end);

im_compute = imarray(train,:,:);
li_compute = lightdirs(train,:);

im_compute = reshape(im_compute,[3, width * height]);

kdn = li_compute \ im_compute;
albedo = zeros(size(kdn,2),1);
normal = zeros(size(kdn));
for i = 1:size(kdn,2)
albedo(i) = norm(kdn(:,i));
if (albedo(i) == 0)
normal(:,i) = kdn(:,i);
else
normal(:,i) = kdn(:,i) / albedo(i);
end
end

% Recover Albedo and Normal maps
alb_map = reshape(albedo,[width, height]);
nrm_map = reshape(normal',[width, height, 3]);
nrm_map = 0.5 * nrm_map + 0.5;

% Show results
figure(1);
subplot(1,2,1);
imshow(alb_map);
title('3 Images: Albedo');
subplot(1,2,2);
imshow(nrm_map);
title('3 Images: Normal');

% Reconstruction Test 1;
li_rcs = lightdirs(test,:);
im_rcs = li_rcs * kdn;
im_rcs = reshape(im_rcs',[width,height,8]);

figure(2);
set(figure(2),'units','normalized','outerposition',[0 0 1 1]);
for i = 1:8
subplot(2,8,i);
imshow(reshape(imarray(i,:,:),[width,height]));
title(sprintf('Ground-truth %d',i))
subplot(2,8,i+8);
imshow(im_rcs(:,:,i));
title(sprintf('Reconstructed %d',i))
end
```

The computed albedo and normal are shown in Figure 1 and images reconstructed from them are shown on the second row of Figure 2. You can easily compare the reconstructed images and the ground-truth ones, which is on the first row of the same Figure. Please note that re-running the script will yield different results since images selected for the computation are not consistent.

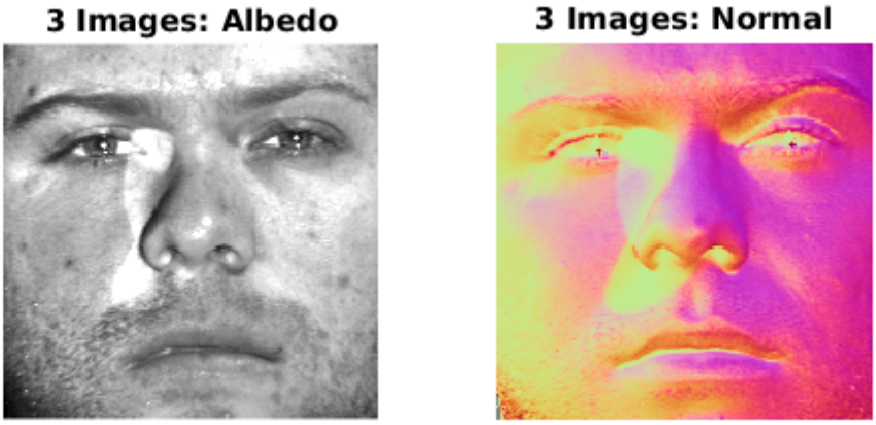

Figure 1: Computed albedo and normal from 3 images

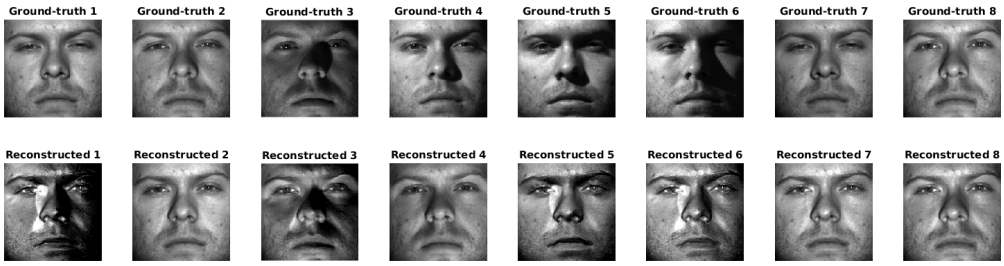

Figure 2: Reconstructed images from computed albedo and normal

1.3 Least-square method

I will use more than 3 images to compute albedo and normal this time. Concretely in this report, I used 7 random images in the dataset to do so.

```
ridx = randperm(nImages);
n_train = 10; % CHANGE THIS: Number of images will be used to find Normal and Albedo
train = ridx(1:n_train);
test = ridx(n_train+1:end);

im_compute = imarray(train,:,:);
li_compute = lightdirs(train,:);

im_compute = reshape(im_compute,[n_train, width * height]);

kdn = (li_compute' * li_compute) \ (li_compute' * im_compute);
albedo = zeros(size(kdn,2),1);
normal = zeros(size(kdn));
for i = 1:size(kdn,2)
albedo(i) = norm(kdn(:,i));
if (albedo(i) == 0)
normal(:,i) = kdn(:,i);
else
normal(:,i) = kdn(:,i) / albedo(i);
end
end

% Recover Albedo and Normal maps
alb_map = reshape(albedo,[width, height]);
nrm_map = reshape(normal',[width, height, 3]);
nrm_map = 0.5 * nrm_map + 0.5;

% Show results
figure(3);
subplot(1,2,1);
imshow(alb_map);
title(sprintf('%d Images: Albedo',n_train));
subplot(1,2,2);
imshow(nrm_map);
title(sprintf('%d Images: Normal',n_train));

% Reconstruction Test 2;
li_rcs = lightdirs(test,:);
im_rcs = li_rcs * kdn;
im_rcs = reshape(im_rcs',[width,height,nImages - n_train]);

figure(4);
set(figure(4),'units','normalized','outerposition',[0 0 1 1]);
for i = 1:nImages - n_train
subplot(2,nImages - n_train,i);
imshow(reshape(imarray(i,:,:),[width,height]));
title(sprintf('Ground-truth %d',i))
subplot(2,nImages - n_train,i+nImages - n_train);
imshow(im_rcs(:,:,i));
title(sprintf('Reconstructed %d',i))
end
```

The computed albedo and normal are shown in Figure 3 and images reconstructed from them are shown on the second row of Figure 4. You can easily compare the reconstructed images and the

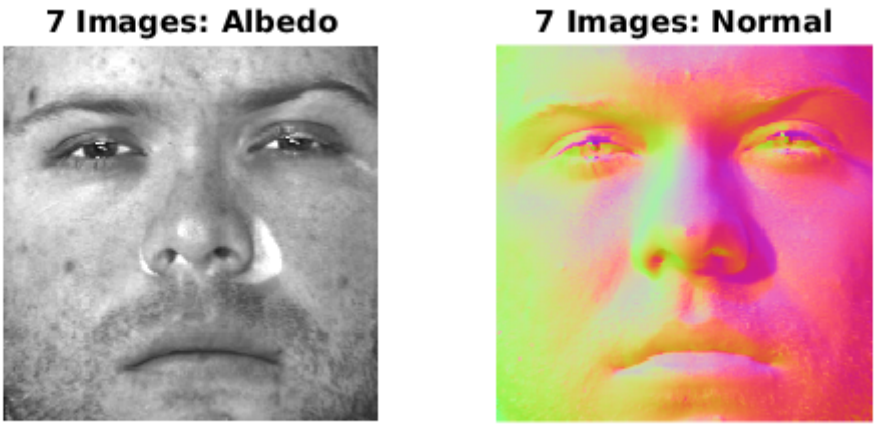

Figure 3: Computed albedo and normal from 7 images

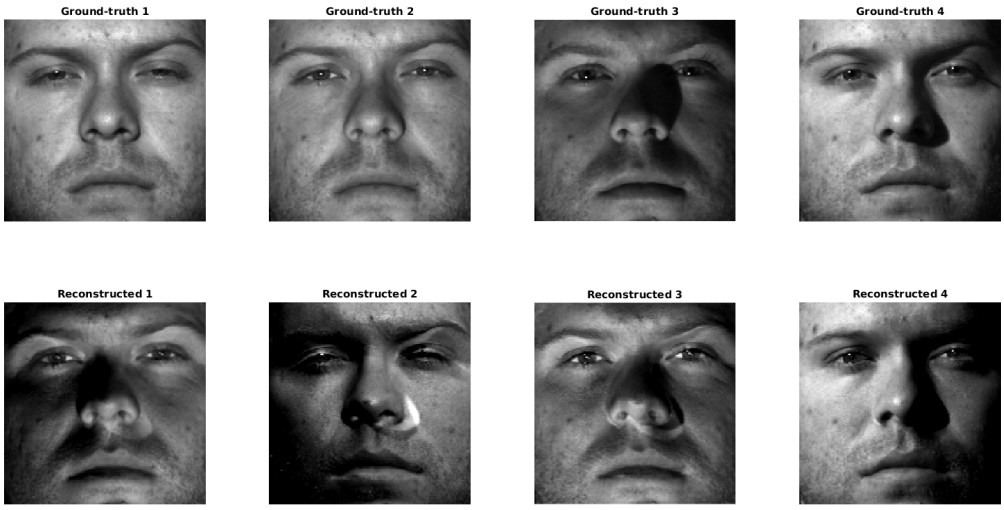

Figure 4: Reconstructed images from computed albedo and normal

ground-truth ones, which is on the first row of the same Figure. Please note that re-running the script will yield different results since images selected for the computation are not consistent.
You might notice a better result when using 7 images to compute albedo and normal. The bright artifact around the nose of subject is reduced in size. Normal map also looks more realistic than the previous one computed from 3 images.

2 Uncalibrated Photometric Stereo

Previously, we have dealt with the photometric stereo problem with known light source directions. This time, lighting information is not given, thus we need to estimate surface normal and albedo based on Generalized Bas Relief. In detail, I will use Singular Vector Decomposition (SVD) to make an estimation on light source directions.

2.1 Preprocessing

Since SVD is a computationally expensive operation, during preprocessing, I will resize images by a factor of 0.8. In addition, I am keeping the previous working environment, thus all variables remain.

```
% Preprocessing
resize_factor = 0.8; % CHANGE THIS: Resize image to work with SVD
imarray = permute(imarray,[2 3 1]);
imarray = imresize(imarray,resize_factor);
[width, height, nImages] = size(imarray);
imarray = permute(imarray,[3 1 2]);
```

2.2 Estimating normal surface and light

Eigenvalues in S component resulted from SVD computation are sorted in the descending order (from largest to smallest value). For the best estimation, I will use 3 largest eigenvalues, as described in [1]. For choosing s* used in the reconstruction part, I will random pick 3 components of V which are different from ones that I used to estimate normal.

```
% We will use all 11 images for svd
im_compute = imarray;
im_compute = reshape(im_compute,[nImages, width * height]);
im_compute = im_compute';

% SVD
[U,S,V] = svd(im_compute);

% Since S is sorted, three best eigenvalues are at index 1, 2, 3
u = U(:,1:3);
s = S(1:3,1:3);
v = V(:,1:3);

lightdirs = v;
kdn = u*s;

kdn = permute(kdn,[2,1]);
albedo = zeros(size(kdn,2),1);
normal = zeros(size(kdn));
for i = 1:size(kdn,2)
albedo(i) = norm(kdn(:,i));
if (albedo(i) == 0)
normal(:,i) = kdn(:,i);
else
normal(:,i) = kdn(:,i) / albedo(i);
end
end

% Recover Albedo and Normal maps
alb_map = reshape(albedo,[width, height]);
nrm_map = reshape(normal',[width, height, 3]);
nrm_map = 0.5 * nrm_map + 0.5;

figure(5);
imshow(nrm_map);
title('Estimated Normal by SVD');

% Reconstruction from s*
rand_s_star = randperm(nImages-3) + 3;
```

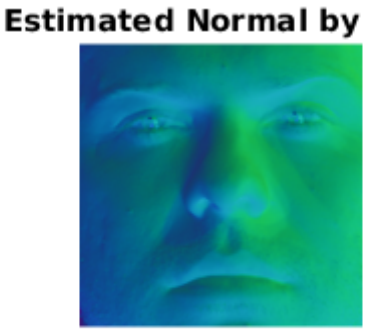

Figure 5: Estimated normal by SVD

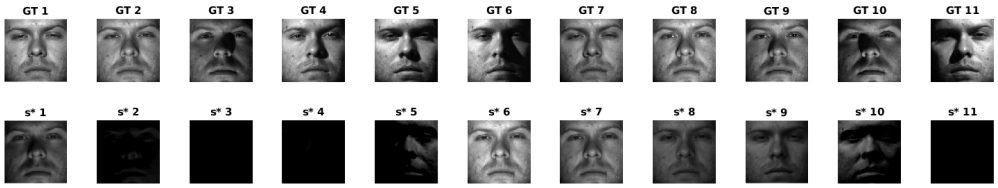

Figure 6: Reconstructed images from computed albedo, normal and light source directions s*

```
lightdirs_star = V(:,rand_s_star(1:3));
im_rcs_star = lightdirs_star * kdn;
im_rcs_star = reshape(im_rcs_star',[width,height,nImages]);

% Show results
figure(6);
set(figure(6),'units','normalized','outerposition',[0 0 1 1]);
for i = 1:nImages
subplot(2,nImages,i);
imshow(reshape(imarray(i,:,:),[width,height]));
title(sprintf('GT %d',i))
subplot(2,nImages,i+nImages);
imshow(im_rcs_star(:,:,i));
title(sprintf('s* %d',i))
end
```

The estimated normal is illustrated in Figure 5. Images reconstructed from the normal using light source direction s* are shown on the second row of Figure 6. You can easily compare the reconstructed images and the ground-truth ones, which is on the first row of the same Figure. Please note that re-running the script will yield different results since s* selected for the reconstruction are not consistent. The estimated normal looks totally different from one computed in the previous section. However, details of subject can still be displayed on it, to an extent that we can recognize that it is showing a face.

For the reconstruction, 4 images are completely in black. Some of them can barely show the face of subject, while the rest seems to be realistic.

To get a better result, I think we need to deal with shadow as well, since the illumination intensity on each pixel is affected by shadow as well. But I am not sure how.

References

[1] A. Yuille and D. Snow, "Shape and albedo from multiple images using integrability," in *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pp. 158–164, IEEE, 1997.