ECE 549 Computer Vision: Homework 1

Xianming Liu
NetID: xliu102

February 17, 2015

1 Lighting

Answers:

A.

1. Light from Right Above
2. It is because of Reflection, and more specifically, nearly specular reflection.
3. The cause of hard streaks is albedo.
4. It is not possible that the glasses cast a shadow on it. If it is completely specular reflection, no light will be reflected to camera sensors and the whole table will be black.

B.

a. If the surface has a specular component, the observed insensitive will change. Since for diffuse reflection, intensity is not dependent on view angle. But for specular reflection, the intensity will change. When the camera's view angle is on the specular reflection direction of a plane, the intensity of the point on the plane will be enhanced because more light is reflected (specularly) to camera sensor.

b. If the surface is Lambertian, then $I(x) = \rho(x)(\mathbf{S} \cdot N(x))$, in this case,

$$I_3 = I_2 \cdot \cos(\pi - \theta_{23}) = I_2 \sin(\theta_{23})$$

, and

$$I_1 = I_2 \cdot \cos(\pi - \theta_{12}) = I_2 \sin(\theta_{12})$$

. So, to compute the angles, should use $\sin(\theta_{12}) = \frac{I_1}{I_2}$, and $\sin(\theta_{12}) = \frac{I_3}{I_2}$. Given intensity vector $(0.5, 0.9, 0.8)$, we know $\sin(\theta_{12}) = 0.5/0.9$, $\theta_{12} = \arcsin(0.5/0.9) = 0.59$, and $\sin(\theta_{23}) = 0.8/0.9$, $\theta_{23} = \arcsin(0.8/0.9) = 1.09$ (in rad).

2 Image Pyramids

The Matlab codes are shown in the following:

```
function [gaussian_pyramid, laplacian_pyramid] = gaussian_pyramids(img, levels, ...
    sigma, hsize)

%Parameters:
if nargin < 4
    hsize = [5,5];
end
if nargin < 3
    sigma = 2;
end

% Creat size arrays
wlevels = [size(img, 1)];
hlevels = [size(img, 2)];
for i = 2:levels
    wlevels(end+1) = floor(wlevels(end) / 2);
    hlevels(end+1) = floor(hlevels(end) / 2);
end

% Creat Gaussian Filter
G = fspecial('gaussian',hsize, sigma);

gaussian_pyramid = {img};
laplacian_pyramid = {};

for idx = 1:(levels - 1)
    cimg = gaussian_pyramid{end};
    gimg = imfilter(cimg, G, 'same');
    gimg = imresize(gimg, [wlevels(idx+1), hlevels(idx+1)]);
    gaussian_pyramid{end+1} = gimg;

    % Calucate laplacians
    laplacian_img = cimg - imresize(gimg, [wlevels(idx), hlevels(idx)]);
    laplacian_pyramid{end + 1} = laplacian_img;
end
laplacian_pyramid{end+1} = gaussian_pyramid{end};

% Show Gaussian and Laplacian Pyramid
ha = tight_subplot(2,5,[.01 .03],[.1 .01],[.01 .01]);
for figidx = 1:levels
    gaussian_fig_idx = figidx;
    laplacian_fig_idx = figidx+levels;
    axes(ha(gaussian_fig_idx)); imshow(gaussian_pyramid{figidx} * 255);
    axes(ha(laplacian_fig_idx)); imagesc(laplacian_pyramid{figidx}); colormap gray;
end

% Plot the frequence of Gaussian / Laplacian Pyramids: using FFT2
figure(2);
hb = tight_subplot(2,5,[.01 .03],[.1 .01],[.01 .01]);
for figidx = 1:levels
    gaussian_fig_idx = figidx;
    laplacian_fig_idx = figidx+levels;
    % Resize to original size to keep spectrums comparable
    g_pyramid_img = imresize(gaussian_pyramid{figidx}, [wlevels(1), hlevels(1)]);
    l_pyramid_img = imresize(laplacian_pyramid{figidx}, [wlevels(1), hlevels(1)]);
    % Performing fft and calculate the magnitute
    FG = fft2(g_pyramid_img); FG = fftshift(FG);
    FL = fft2(l_pyramid_img); FL = fftshift(FL);
    axes(hb(gaussian_fig_idx)); colormap gray; imagesc(log(abs(FG)));
    axes(hb(laplacian_fig_idx)); colormap gray; imagesc(log(abs(FL)));
end
```

and the main script is:

```
% I/O: Read image and turn it to gray, and turn from unit8 to double
img = rgb2gray(imread('./test.jpg'));
img = im2double(img)/255;

[g_pyramid, l_pyramid] = gaussian_pyramids(img, 5);
```

Figure 1 and Figure 2 show the results generated from the above code.

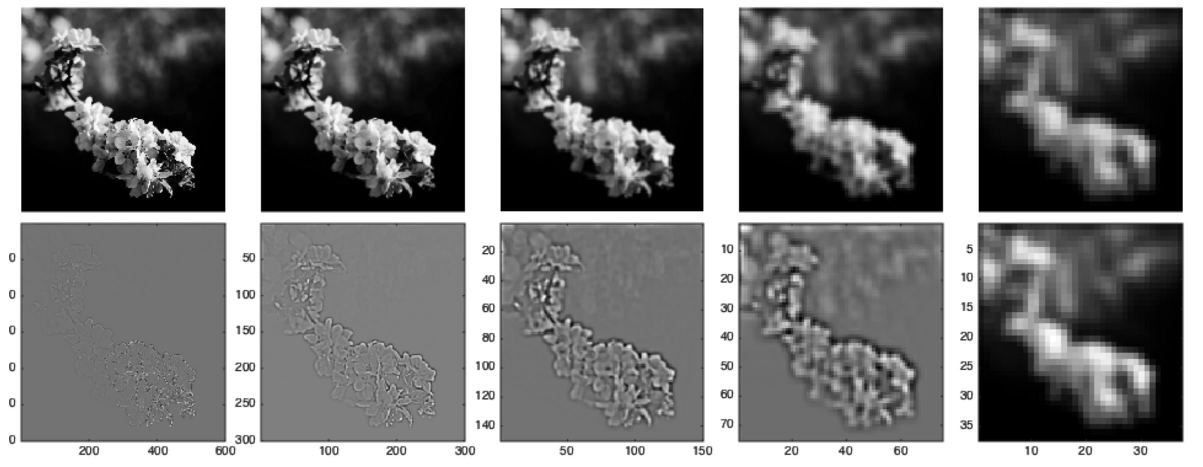

Figure 1: Gaussian and Laplacian Pyramid

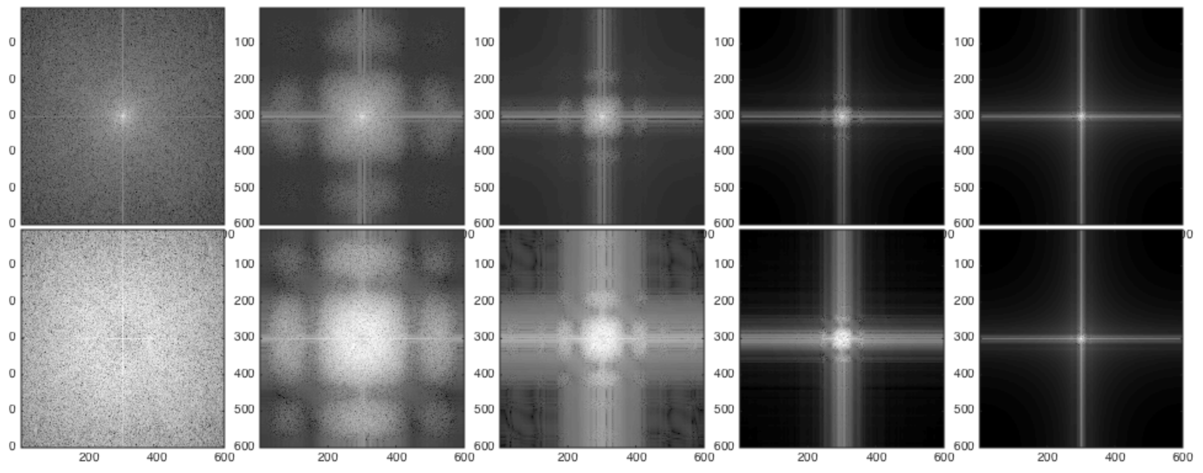

Figure 2: FFT Magnitude of Gaussian and Laplacian Pyramid

Conclusion: The *Gaussian Pyramid* removes high frequencies signals in the image gradually, while *Laplacian Pyramid* tries to preserve the lost high-frequency signals in the Gaussian Pyramid, in each level.

3 Edge Detection

a) *Gradient Edge*: In this implementation, the Gaussian Kernel size is 1, and Gaussian filter size is $9 * 9$.

b) *OrientedGradientMap*: In my implementation, I adopt Steerable Filters, on *six* orientations. The filter bank is shown in Figure 3. For each filter, the kernel size is $\sqrt{2}$.

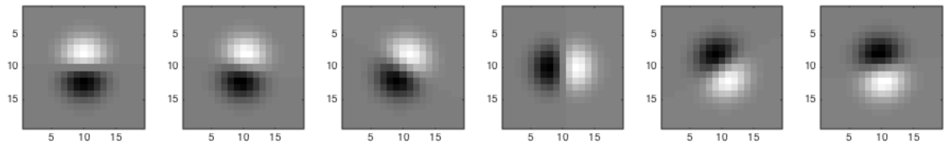

Figure 3: Steerable Filter bank used in orient boundary detection.

Qualitative Results: As shown in Figure 4, the boundary detected by orient filters are less noisy. Especially in the second example, the flower is detected by orient filters, but the edge response of gradient based method is not significant.

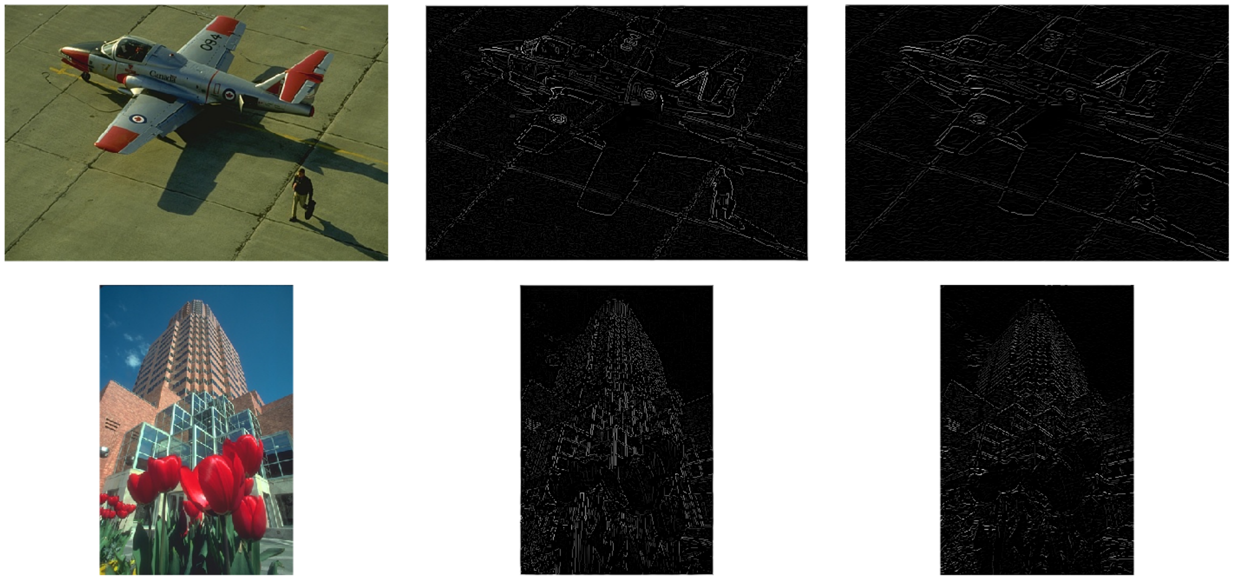

Figure 4: Two examples of gradient based and orient filter based boundary detection. The second column is gradient based algorithm, and the third one is orient filter based results.

Quantitative results: PR curves for gradient-based and orient filter based methods are shown in Figure 5 and Figure 6. Quantitatively, orient filter based method performs better than gradient based method. More detail, Method gradient: overall F-score = 0.537, average F-score = 0.564; Method oriented: overall F-score = 0.536, average F-score = 0.587.

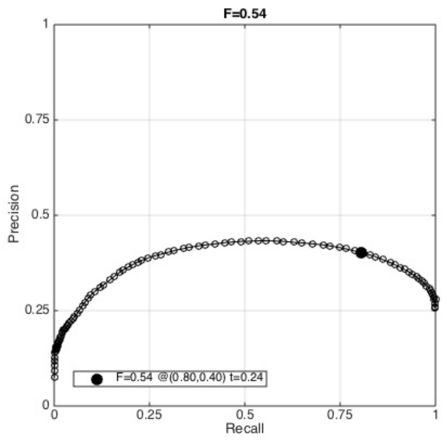

Figure 5: PR curve of gradient based method

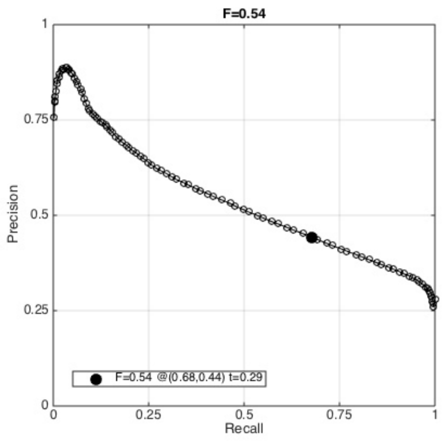

Figure 6: PR curve of orient filter based method

c) *Potential Improvement*: Edge responses may vary greatly on different scales. So, one possible improvement is to select the best scale for edge responses. Previous work [1] has already demonstrated the improvement of involving multiple scale filters in boundary detection. However, it is still computational heavy to extract features on so many scales. Inspired by Tony Lindeberg [2] and my own work on scale selection for edges [3], I implemented a simplified version of scale selection. The basic idea is to convolve the image with filters on multiple scales (5 in my implementation), and according to [2], a Gaussian normalizer should be used to obtain the normalized gradient response, by $L_\sigma = \sqrt{\sigma} \cdot L$, where L is the oriented Laplacian of Gaussian responses.

d) As mentioned above, I implemented a scale-selective version of oriented filter based method. The F score improves slightly: Method multi scale oriented: overall F-score = 0.552, average F-score = 0.595. Figure 7 shows the PR-Curve of the newly proposed method. In my complex implementation in [3], I achieved average F-score = 0.66, with much less computation compared with [1].

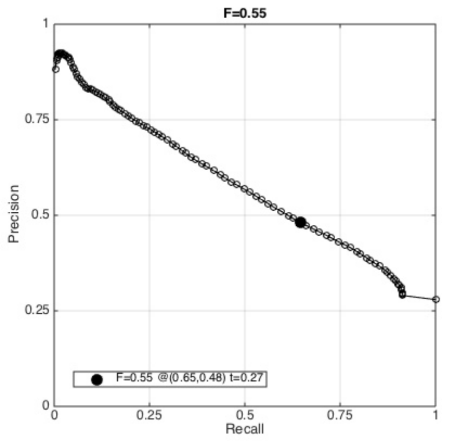

Figure 7: PR curve of multi-scale orient filter based method

References

[1] Xiaofeng Ren, "Multi-scale improves boundary detection in natural images," in *Computer Vision–ECCV 2008*, pp. 533–545. Springer, 2008.

[2] Tony Lindeberg, "Feature detection with automatic scale selection," *International journal of computer vision*, vol. 30, no. 2, pp. 79–116, 1998.

[3] Xian-Ming Liu, Changhu Wang, Hongxun Yao, and Lei Zhang, "The scale of edges," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 462–469.