Computer Vision I - Sheet 5
Group 2

Jonas Otto
jonas@jonasotto.com

Dominik Authaler
dominik.authaler@uni-ulm.de

July 24, 2019

1 Gaussian Pyramids

The source code for this task can be found in the files `sh05ex01.m` and `gaussianPyramid.m`. To avoid opening Matlab only for viewing the code, it's included below. The results of this exercise are shown in the figures 1 and 2. Comparing the two pyramids, one can see a few differences, but in the overall comparison they look quite similiar. Nevertheless there should be quite huge differences if one would compare the reconstructed images. Because of the missing gaussian filtering, the reconstructed image of the second pyramid should show a low of distortions.

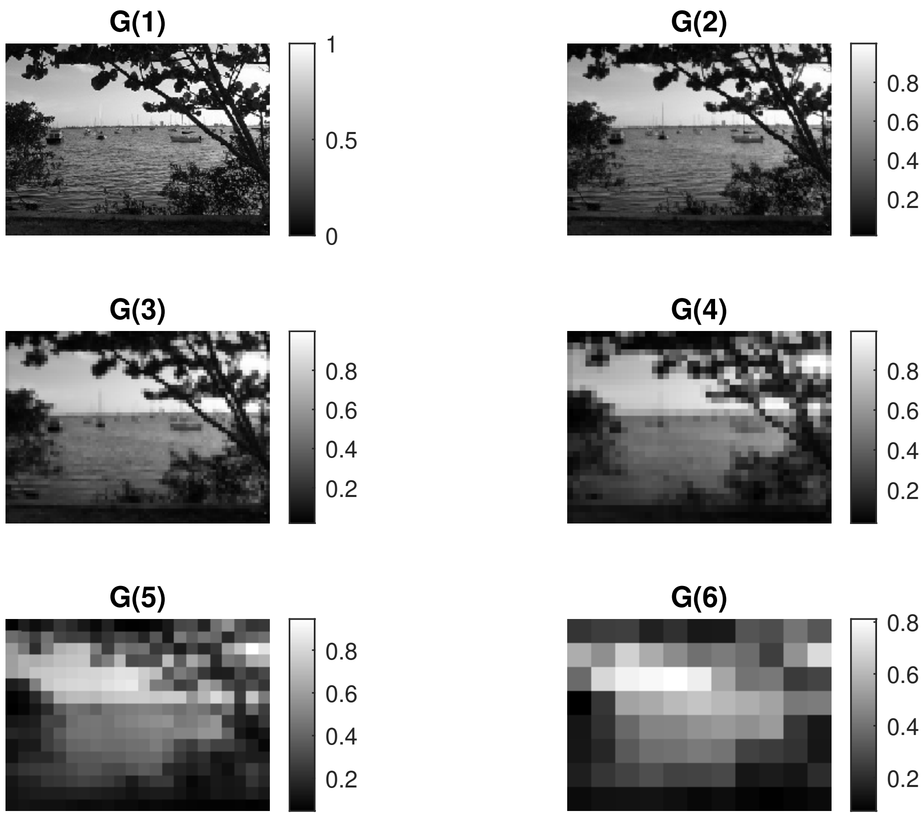

Figure 1: Gaussian pyramid with Gaussian filtering

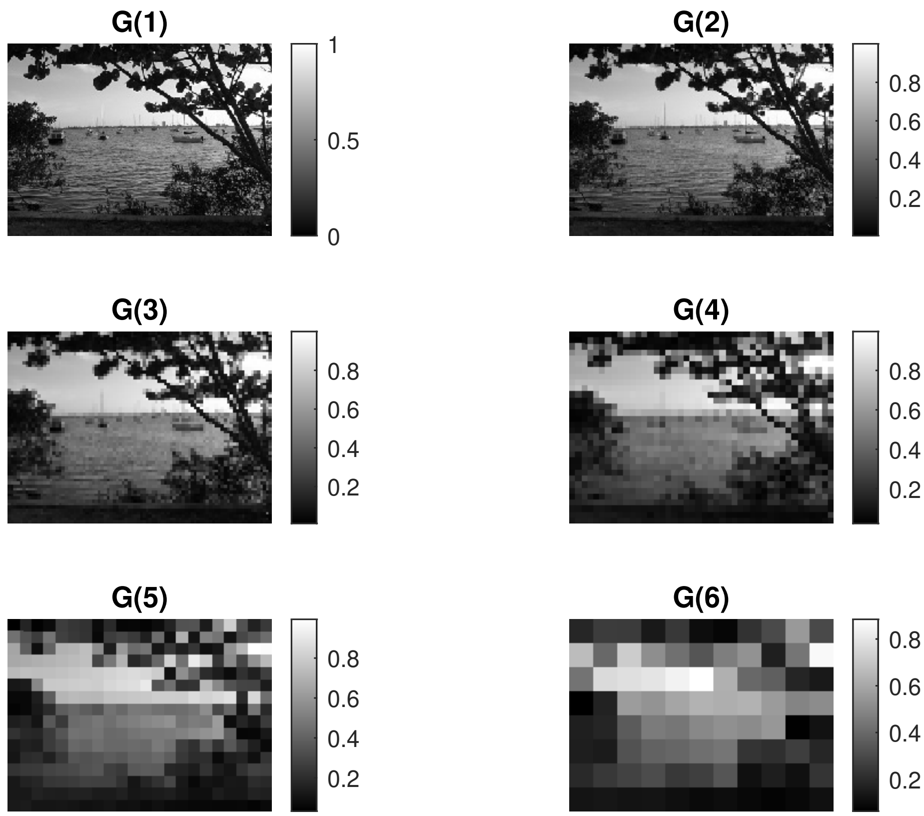

Figure 2: Gaussian pyramid without Gaussian filtering

```
image = im2double(imread("boats.tif"));

n=6;

images_gauss = gaussianPyramid(image, n, 1);
images = gaussianPyramid(image, n, 0);

%% Visualization
close all;

figure('Name','With gaussian filtering');
for i = 1:n
  subplot(3,n/3,i);
  imshow(images_gauss{i}, []);
```

```
  colorbar();
  title(['G(' num2str(i) ')']);
end
saveas(gcf,'ex01_with.eps','epsc')

figure('Name','Without gaussian filtering');
for i = 1:n
  subplot(3,n/3,i);
  imshow(images{i}, []);
  colorbar();
  title(['G(' num2str(i) ')']);
end
saveas(gcf,'ex01_without.eps','epsc')
```

```
function images = gaussianPyramid(image, n, useGaussian)
  clear images;
  images{1} = image;

  % TODO: Parameter f r gauss bestimmen
  gauss = fspecial('gaussian');

  for i = 2:n
    if useGaussian == 1
          image = imfilter(image, gauss);
    end
    image = 0.5 * image(1:2:end, 1:2:end) + 0.5 * image
        (2:2:end, 2:2:end);
    images{i} = image;
  end
end
```

2 Laplacian Pyramids

The source code for this task can be found in the files `sh05ex02.m` and `reconstruct.m` or below. Figure 3 shows the generated Laplacian pyramid, the corresponding reconstructed image is shown in figure 4. The figures 5 and 6 show the same Laplacian pyramid where compression was applied. Obviously the compression wasn't lossless, as the image no has quite huge artifacts in it. Figure 7 shows the resulting error of the compression for different threshold levels. As one would expect, the error raises with the threshold value, because more values are treated as zero.

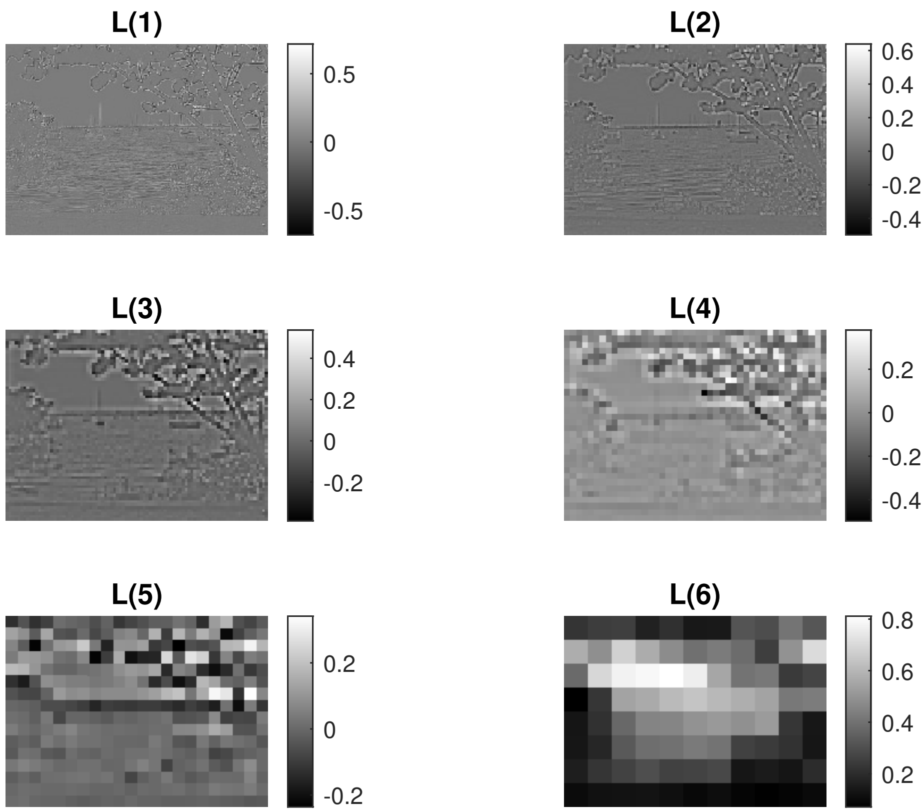

Figure 3: Laplacian pyramid

Figure 4: Reconstructed image

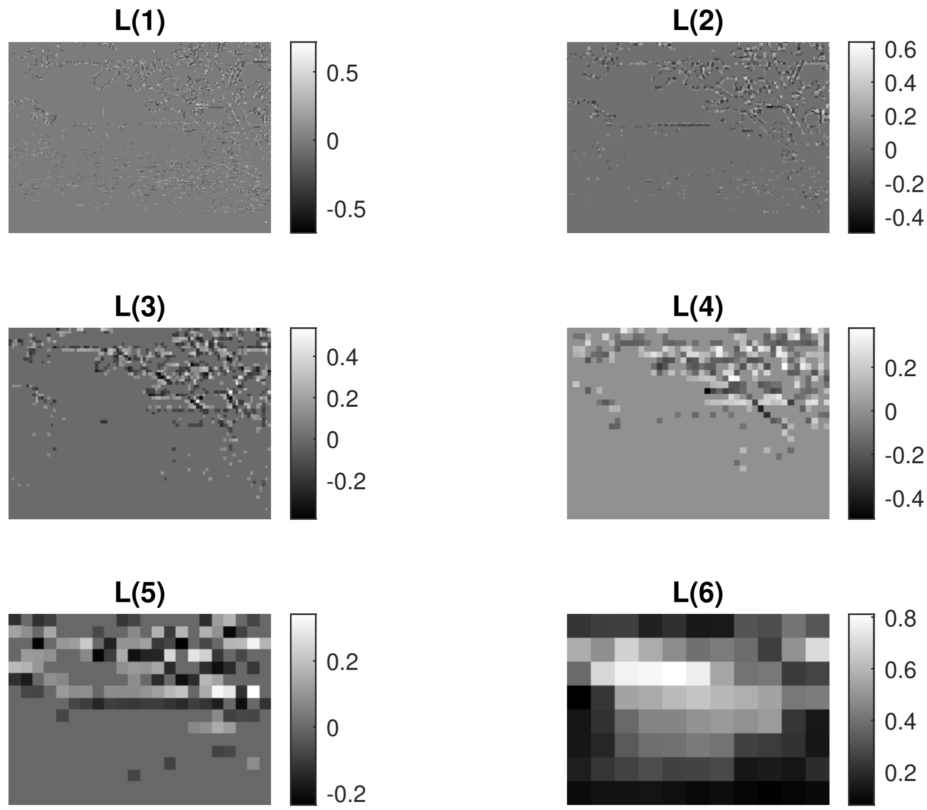

Figure 5: Laplacian pyramid with applied compression

Figure 6: Reconstructed image from the compressed pyramid

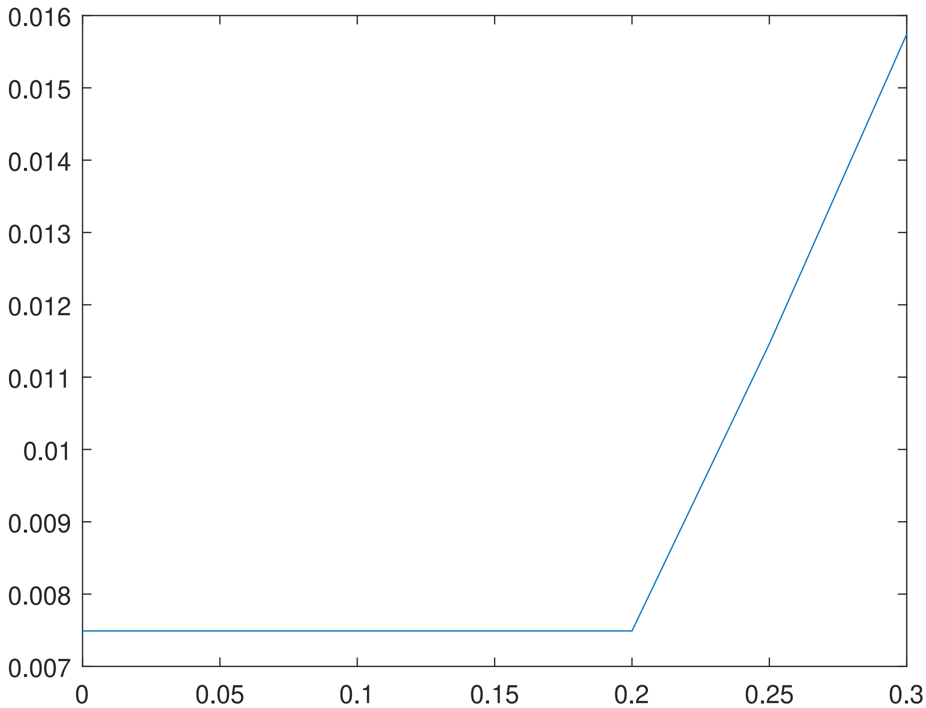

Figure 7: Error for different threshold values

```
image = im2double(imread('boats.tif'));
n = 6;
G = gaussianPyramid(image, n, 1);

% Calculate Laplacian Pyramid
for i = 1:n
  % calc L{i}
  if i ~= n
    L{i} = G{i} - imresize(G{i+1}, 2);
  else
    L{i} = G{i};
  end
end

%% Plot L
close all;
figure('Name', 'Laplacian Pyramid');
```

```
for i = 1:n
  subplot(3,n/3,i);
  imshow(L{i}, []);
  colorbar();
  title(['L(' num2str(i) ')']);
end
saveas(gcf,'ex02_pyramid.eps','epsc')


% Reset G
clear G;

% Reconstruct image from L
figure('Name', 'Reconstructed Image from L');
imshow(reconstruct(L),[]);
saveas(gcf,'ex02_reconstructed.eps','epsc')

% Compression
lambda = 0.2;
for i = 1:n-1
  L{i}(abs(L{i}) < lambda * max(max(abs(L{i})))) = 0;
end

%% Plot L
figure('Name', 'Laplacian Pyramid after Compression');
for i = 1:n
  subplot(3,n/3,i);
  imshow(L{i}, []);
  colorbar();
  title(['L(' num2str(i) ')']);
end
saveas(gcf,'ex02_pyr_compress.eps','epsc')


% Reset G
clear G;

% Reconstruct image from L
figure('Name', 'Reconstructed Image from L after Compression');
imshow(reconstruct(L),[]);
saveas(gcf,'ex02_rec_compress.eps','epsc')

lambda = 0:0.05:0.3
error = zeros(1, size(lambda,2));

```

```
for i = 1:size(lambda,2)
  LT = L;
  for k = 1:n-1
    LT{k}(abs(LT{k}) < lambda(i) * max(max(abs(LT{k})))) = 0;
  end
  error(i) = immse(image, reconstruct(LT));
end

figure('Name', 'Error');
plot(lambda, error)
saveas(gcf,'ex02_error.eps','epsc')
```

```
function image = reconstruct(L)
  n = size(L, 2)

  G{n} = L{n}
  for i = n-1:-1:1
    G{i} = imresize(G{i+1}, 2) + L{i};
  end
  image = G{1};
end
```

3 Gabor wavelets

The source code for this task can be found in the file `sh05ex03.m` or below. The Matlab script for the `getGabor` function was given in the exercise. Figure 8 shows the results of the energy calculations. On can easily stand, that the main orientations of the texture in the image are vertical and horizontal, because this orientations showed the biggest results.

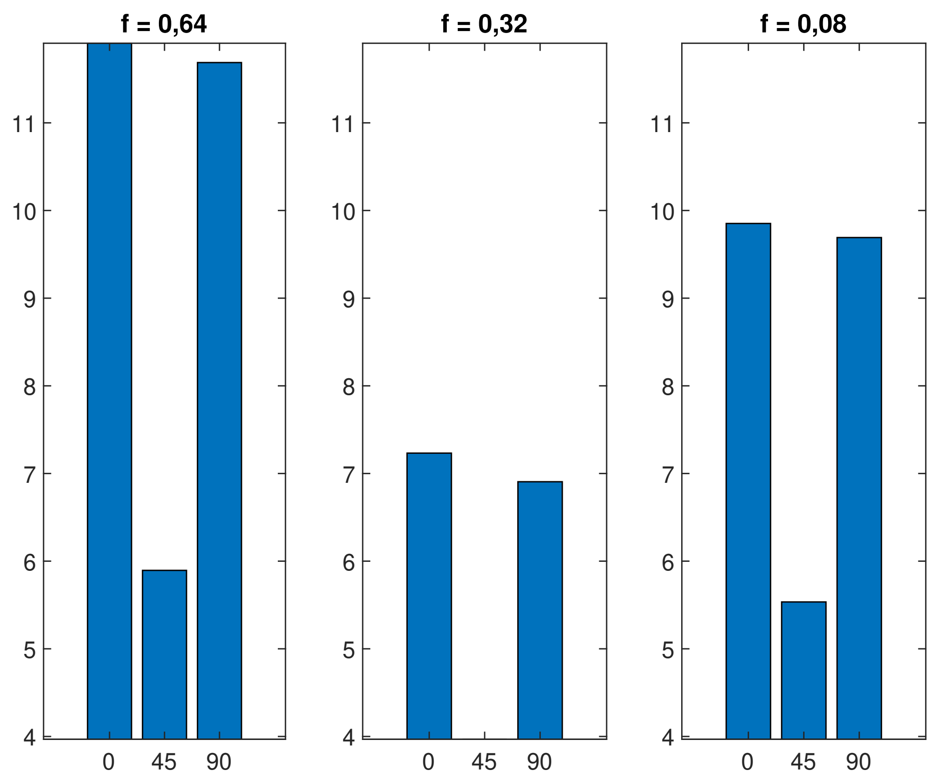

Figure 8: Filter energy of the different Gabor wavelets

```
clear; clc; close all;

image = im2double(imread('basket.jpg'));

orientations = [0, pi/2, pi/4]*(180/pi);
frequencies = [0.64, 0.32, 0.08];

b = 2.32;
```

```
Gabors = {};

for f = frequencies
  for theta = orientations
    % b = f * sigma = const
    Gabors{end+1} = getGabor(f, b/f, theta);
  end
end

energy = zeros(1,length(Gabors));
for k = 1:length(Gabors)
  energy(k) = sqrt(sum(sum(abs(imfilter(image, Gabors{k},
      'conv').^2))));
end

close all;
figure();
axis auto;
subplot(1,3,1);
bar(orientations, energy(1:3));
title('f = 0,64');
ylim([min(energy), max(energy)]);

subplot(1,3,2);
bar(orientations, energy(4:6));
title('f = 0,32');
ylim([min(energy), max(energy)]);


subplot(1,3,3);
bar(orientations, energy(7:9));
title('f = 0,08');
ylim([min(energy), max(energy)]);
saveas(gcf,'ex03.eps','epsc')
```