

# Computer Vision I

Tim Luchterhand, Paul Nykiel (Group 17)

5. Juni 2018

## 1 Filter Algebra

### 1.1

Without loss of generality choose an image  $I \in [0, 255]$  with size  $3 \times 3$ .

The maximum value for  $I_{22}$  is achieved by weighting all positive values in the filter with 255 and all negative values with 0. This results in image (1). With this image the maximum output value is  $2 \cdot 255 + 2 \cdot 255 + 1 \cdot 255 = 1275$ .

$$I_{max} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 255 \\ 0 & 255 & 255 \end{pmatrix} \quad (1)$$

The minimum value for  $I_{22}$  is achieved by weighting all positive values in the filter with 0 and all negative values with 255. This results in image (2). With this image the maximum output value is  $-2 \cdot 255 + (-2) \cdot 255 + (-1) \cdot 255 = -1275$ .

$$I_{max} = \begin{pmatrix} 255 & 255 & 0 \\ 255 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad (2)$$

### 1.2

$$\begin{aligned} I &= \begin{pmatrix} 1 & 2 \end{pmatrix} \\ H &= \begin{pmatrix} 1 & -1 \end{pmatrix} \\ \alpha &= 255 \end{aligned}$$

With these values calculate the convolution (the values are continued with 0):

$$\begin{aligned}(\alpha \cdot I) * H &= (255 \ 255) * (1 \ -1) = (255 \ 0 \ 0) \\ \alpha \cdot (I * H) &= 255 \cdot (1 \ 1 \ 0) = (255 \ 255 \ 0) \\ &\Rightarrow (\alpha \cdot I) * H \neq \alpha \cdot (I * H)\end{aligned}$$

As shown by the contradiction above the linearity does not hold for clamped values.

### 1.3

Matlab Code:

```

1 I = imread("lena.tif");
2 sigma = 3;
3 G = fspecial("gaussian", 2*ceil(2*sigma)+1, sigma);
4 H = [-1 -2 0; -2 0 2; 0 2 1];
5
6 T1 = imfilter(I, G, 'replicate', 'conv');
7 R1 = imfilter(T1, H, 'replicate', 'conv');
8
9 figure();
10 subplot(1,3,1);
11 imshow(I);
12 title("Original image");
13 subplot(1,3,2);
14 imshow(T1);
15 title("Blurred Image");
16 subplot(1,3,3);
17 imshow(R1);
18 title("Diagonal edges of the blurred image");
19
20 print("lenaEdge1.eps", "-depsc");

```

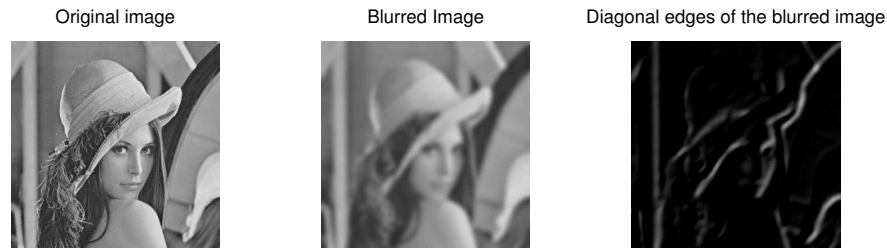


Abbildung 1: Output of the Matlab script

The filter highlights strong edges in the diagonal axis (north-west to south-east) by calculating an approximation of the derivative in this direction.

#### 1.4

```

22 T2 = imfilter(I, H, 'replicate', 'conv');
23 R2 = imfilter(T2, G, 'replicate', 'conv');
24
25 figure();
26 subplot(2,3,1);
27 imshow(I);
28 title("Original image");
29 subplot(2,3,2);
30 imshow(T1);
31 title("Blurred Image");
32 subplot(2,3,3);
33 imshow(R1);
34 title("Diagonal edges of the blurred image");
35 subplot(2,3,4);
36 imshow(abs(R2-R1));
37 title("Difference of both images");
38 subplot(2,3,5);
39 imshow(T2);
40 title("Diagonal edges of the original image");
41 subplot(2,3,6);

```

```

42 imshow(R2);
43 title("Blurred image of the diagonal edges");
44
45 print("lenaEdge2.eps", "-depsc");

```

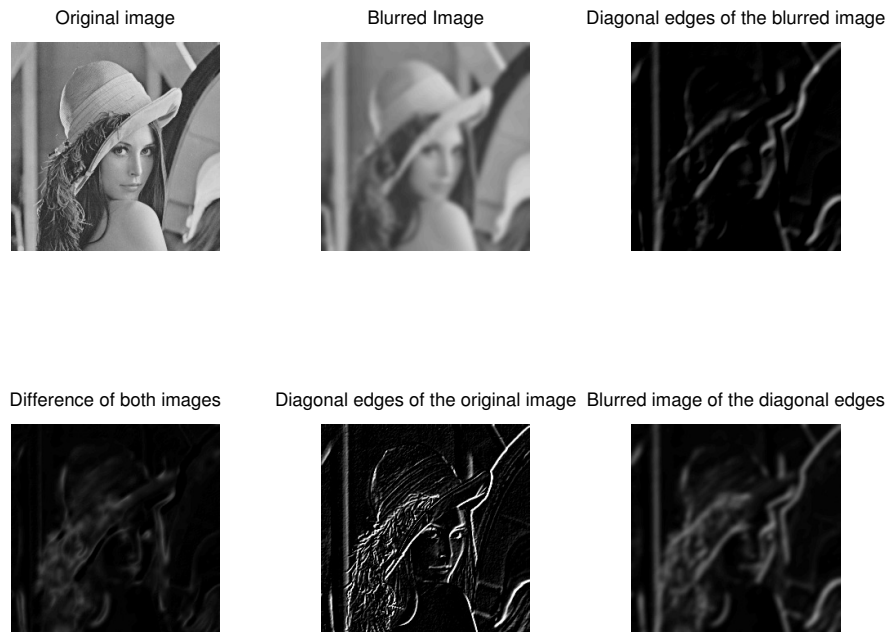


Abbildung 2: Output of the second part of the Matlab script

The second image looks slightly more detailed. When using clamped images the convolution product is no longer commutative.

## 2 Discrete Fourier Transform

### 2.1

```

1 %% Part a
2 N = 4;
3 ti = [0 1/30 2/30 3/30];
4 si = [2 3 0 1];
5
6 S = zeros(size(ti));
7 f = zeros(size(ti));
8

```

```

9  for u=0:(N-1)
10     for t=0:(N-1)
11         S(u+1) += si(t+1) * exp(-2*pi*i*u/N*t);
12     end;
13     f(u+1) = 2*pi*u/N;
14 end;
15 S ./= sqrt(N);

```

## 2.2

The  $u$ th coefficient (starting at zero) corresponds to a frequency of

$$f(u) = 2 \cdot \pi \cdot \frac{u}{N} = \frac{\pi \cdot u}{2}$$

```

17 %% Part b
18 x = 0:0.01:1;
19 y = sin(2*pi*x*1/N*3);
20 figure();
21 plot(x,y);
22 title("A sine-wave with the frequency of S(3)");
23 xlabel("t");
24 ylabel("y(t)");
25 print("frequency.eps", "-depsc");

```

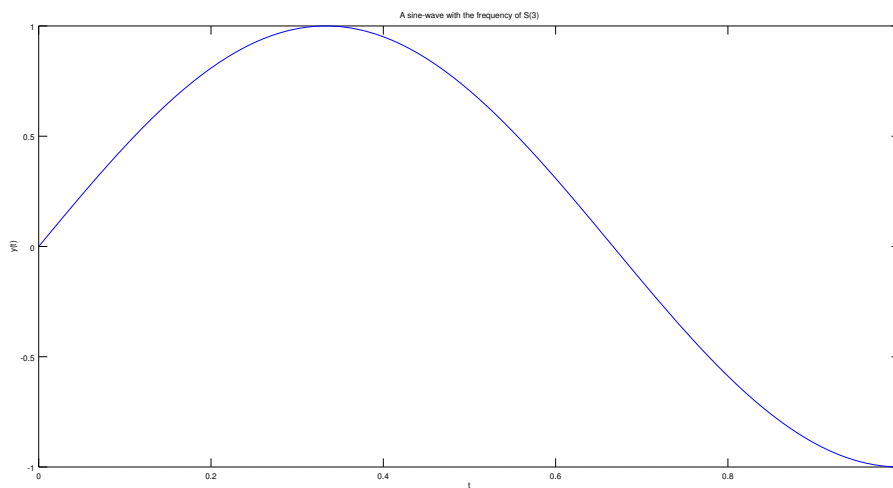


Abbildung 3: Output of the second part of the Matlab script

## 2.3

```
27 %% Part c
28 sr = zeros(size(ti));
29 for u=0:N-1
30     for t=0:(N-1)
31         sr(u+1) += S(t+1) * exp(2*pi*i*u/N*t);
32     end;
33 end;
34 sr ./= sqrt(N);
```

## 2.4

```
36 %% Part d
37 figure();
38 subplot(3,1,1);
39 plot(ti, si);
40 xlabel("t");
41 ylabel("s_i(t)");
42 subplot(3,1,2);
43 plot(f, S);
44 xlabel("f");
45 ylabel("S(f)");
46 subplot(3,1,3);
47 plot(ti, sr);
48 xlabel("t");
49 ylabel("s_r(t)");
50 print("transforms.eps", "-depsc");
```

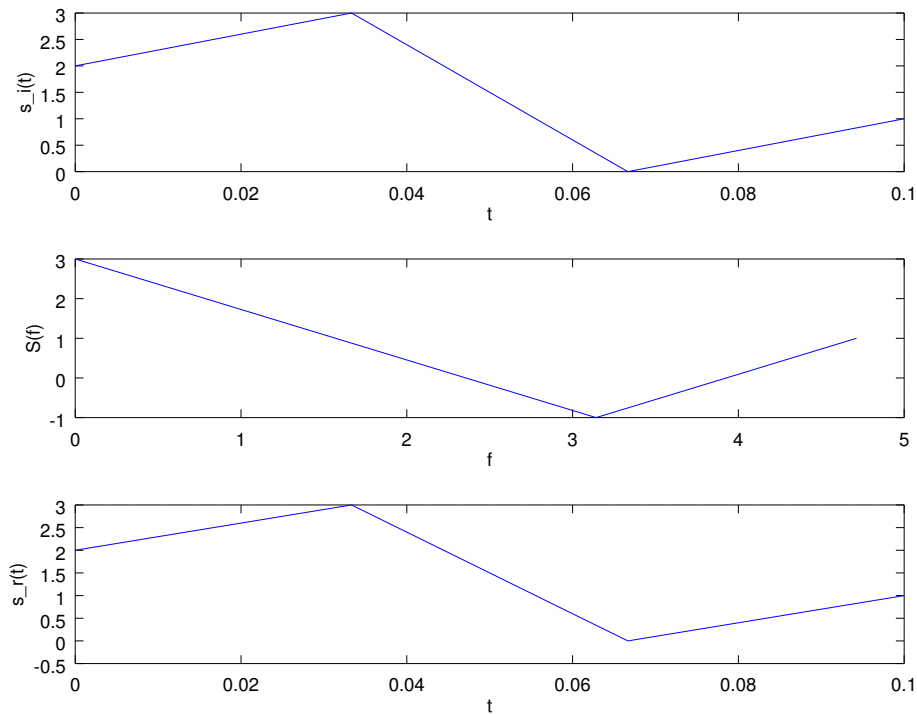


Abbildung 4: Output of the last part of the Matlab script

## 3 Fourier Transform for Image Quality Assessment

### 3.1

```

1 %% Part a
2 I1 = double(imread("flower01.png"))/255;
3 I2 = double(imread("flower02.png"))/255;
4
5 F1 = fftshift(fft2(I1));
6 F2 = fftshift(fft2(I2));
7
8 F1 = abs(F1);
9 F2 = abs(F2);

```

### 3.2

```

11 %% Part b

```

```

12 sigma = size(I1,1);
13 G = fspecial('gaussian', sigma, sigma/2);
14 G = padarray(G,(size(F1)-size(G))/2,0, 'replicate');
15 G /= max(max(G));
16 H = ones(size(G)) - G;
17
18 H1 = F1 .* H;
19 H2 = F2 .* H;

```

### 3.3

```

21 %% Part c
22 H1 .*= H1;
23 H2 .*= H2;
24
25 E1 = sum(sum(H1));
26 E2 = sum(sum(H2));

```

### 3.4

```

28 %% Part d
29 figure();
30 subplot(3,3,1);
31 imshow(I1);
32 title("flower01.png");
33 subplot(3,3,2);
34 imshow(F1);
35 title("Fourier transform of flower01.png");
36 subplot(3,3,3);
37 imshow(H1);
38 title("Filtered fourier transform of flower01.png");
39
40
41 subplot(3,3,4);
42 imshow(I2);
43 title("flower02.png");
44 subplot(3,3,5);
45 imshow(F2);
46 title("Fourier transform of flower02.png");
47 subplot(3,3,6);
48 imshow(H2);
49 title("Filtered fourier transform of flower02.png");
50
51 subplot(3,3,7);

```



```

52 imshow(G);
53 title("G");
54 subplot(3,3,8);
55 imshow(H);
56 title("H");
57 subplot(3,3,9);
58 bar([E1 E2]);
59 title("Energy of both images");
60
61 print("Quality.eps","-depsc");

```

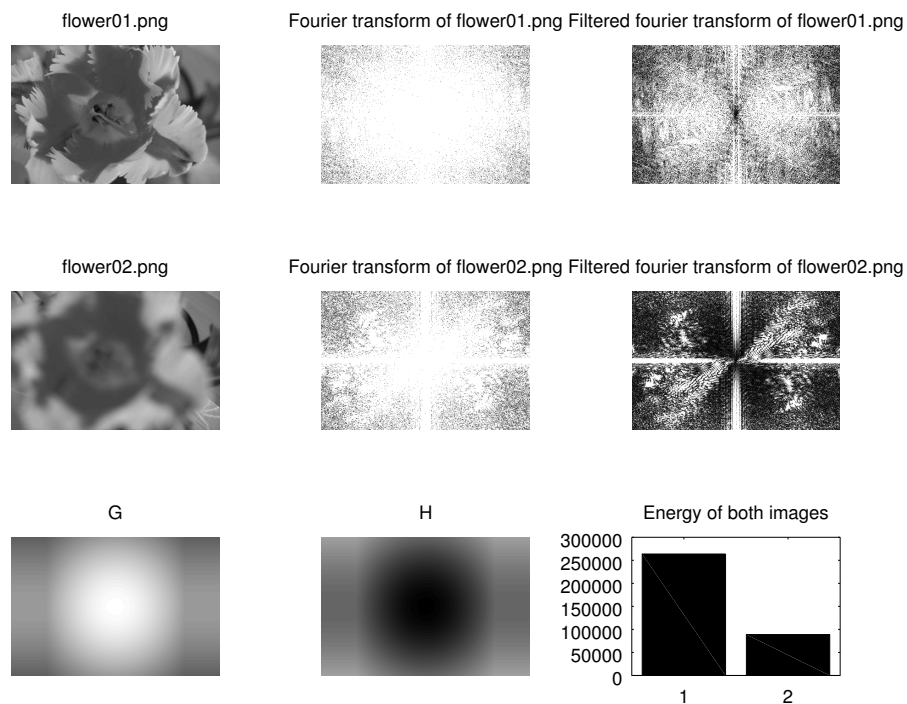


Abbildung 5: Output of the Matlab script