

# Image filtering in the frequency domain

Václav Hlaváč

Czech Technical University in Prague

Czech Institute of Informatics, Robotics and Cybernetics

160 00 Prague 6, Jugoslávských partyzánů 1580/3, Czech Republic

<http://people.ciirc.cvut.cz/hlavac>, [vaclav.hlavac@cvut.cz](mailto:vaclav.hlavac@cvut.cz)

also Center for Machine Perception, <http://cmp.felk.cvut.cz>

## Outline of the talk:

- ◆ Convolution as filtration in frequency domain.
- ◆ Low pass filtering examples, sharp cut off, smooth Gaussian.
- ◆ High pass filtering examples, sharp cut off, smooth Gaussian.
- ◆ Butterworth filter.
- ◆ Homomorphic filter separating illumination and reflectance.
- ◆ Systematic design of 2D FIR filters.

## Filtration in the frequency domain

1.  $F(u, v) = \mathcal{F}\{f(x, y)\}$
2.  $G(u, v) = H(u, v) .* F(u, v)$ ,  
where  $.*$  denotes element-wise multiplication of matrices.
3.  $g(x, y) = \mathcal{F}^{-1}\{G(u, v)\}$

---

*Note for lab exercises: We usually use  $\ln \|F(u, v)\|$  for visualization purposes. The original spectrum  $F(u, v)$  has to be used in the actual filtration.*

## Convolution as Fourier spectrum frequency filtration

- ◆ Matrix element by element multiplication.
- ◆ The speed of operations is determined by the (high) speed of FFT.

---

Consider a matrix  $a$  with dimensions  $M \times N$  and a matrix  $b$  with dimensions  $P \times Q$ .

The convolution  $c = a * b$  can be calculated as follows:

1. Fill in matrices  $a$ ,  $b$  by zeroes to have dimensions  $M + P - 1$ ,  $N + Q - 1$  (usually up to the order of 2 to ease FFT).
2. Calculate 2D FFT matrix of matrices  $a$ ,  $b$  (in MATLAB, using `fft2`). The outcome are matrices  $A$ ,  $B$ .
3. Multiply complex Fourier spectra element-wise,  $C = A .* B$ .
4. The result of the convolution  $c$  is obtained by the inverse Fourier transformation (in MATLAB using `ifft2`).

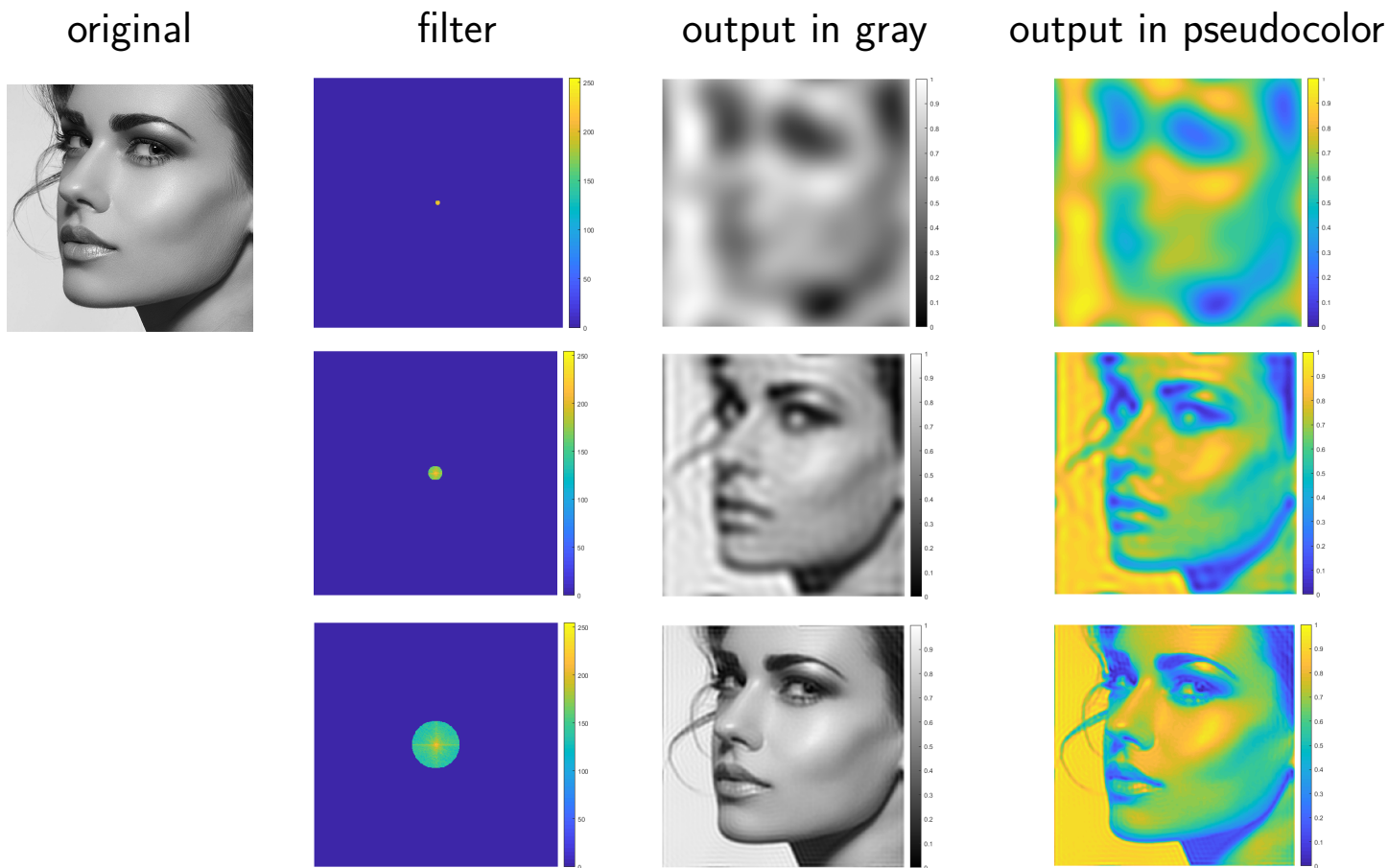
## 2D convolution in frequency domain in MATLAB

```
A = magic(3);
B = ones(3);
A(8,8) = 0; % zero-pad A to be 8-by-8
B(8,8) = 0; % zero-pad B to be 8-by-8
C = ifft2(fft2(A).*fft2(B));
C = C(1:5,1:5); % extract the nonzero portion
C = real(C) % remove imaginary part caused by roundoff error
```

$$C = \begin{bmatrix} 8.0000 & 9.0000 & 15.0000 & 7.0000 & 6.0000 \\ 11.0000 & 17.0000 & 30.0000 & 19.0000 & 13.0000 \\ 15.0000 & 30.0000 & 45.0000 & 30.0000 & 15.0000 \\ 7.0000 & 21.0000 & 30.0000 & 23.0000 & 9.0000 \\ 4.0000 & 13.0000 & 15.0000 & 11.0000 & 2.0000 \end{bmatrix}$$

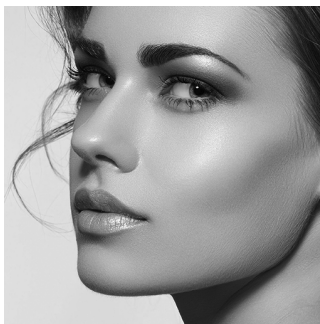
Note: the convolution calculated via spectra is faster in MATLAB for large matrices. The calculation via `conv2`, `filt2` is faster for small matrices.

# Low pass filter, circle sharp cut-off, $r=5, 15, 50$

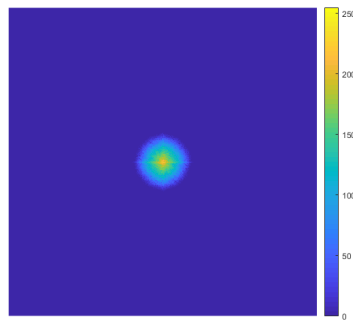


# Low pass Gaussian filter, $\sigma = 10, 30$

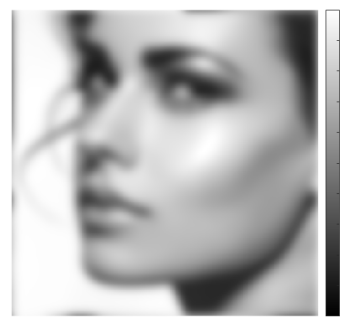
original



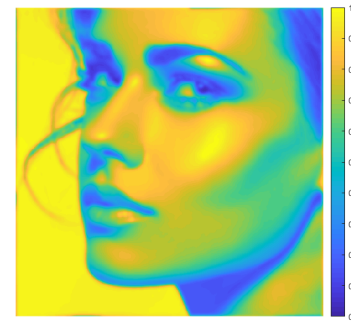
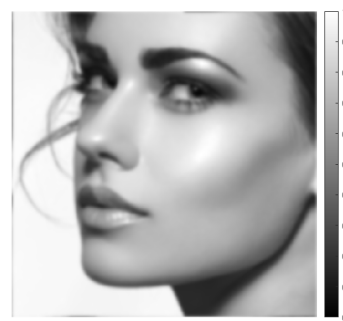
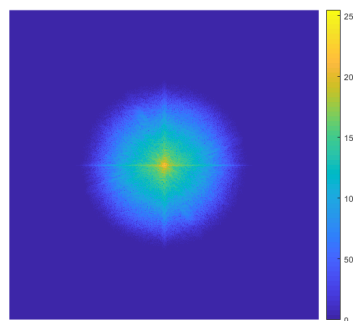
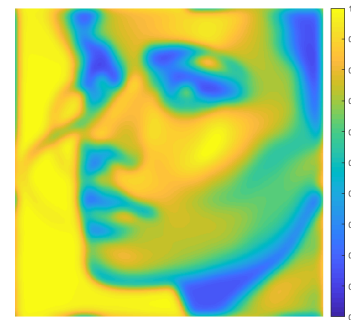
filter



output in gray

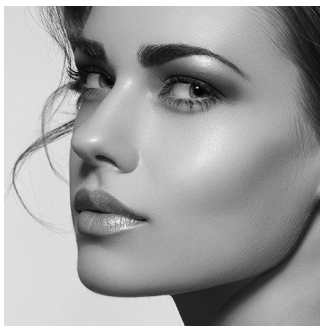


output in pseudocolor

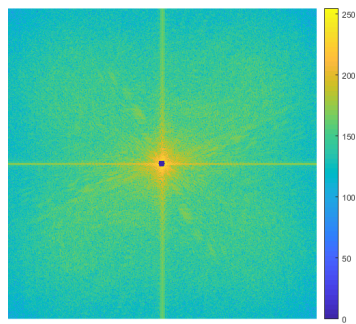


# High pass circle sharp cut-off, $r=5, 15$

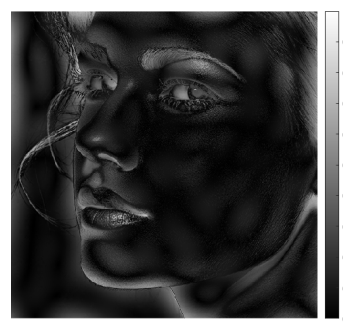
original



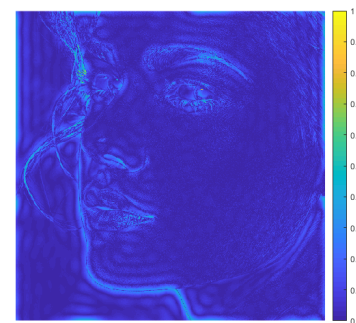
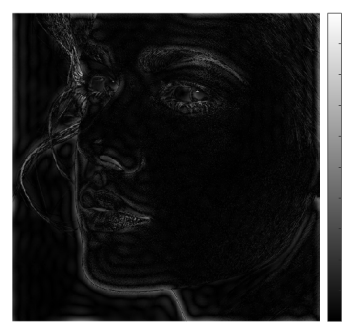
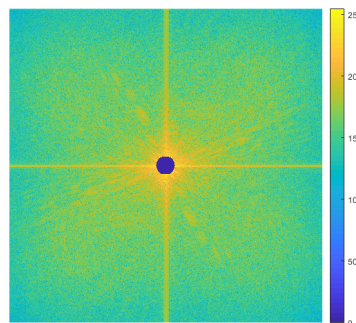
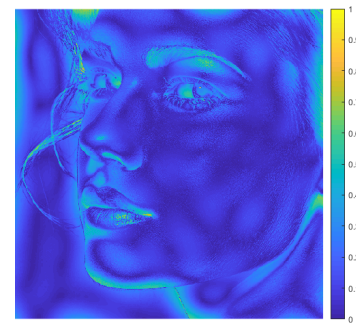
filter



output in gray



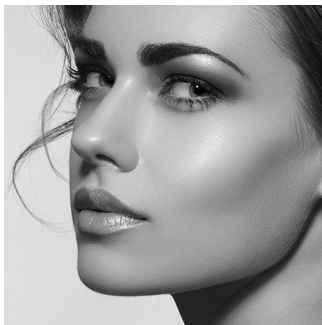
output in pseudocolor



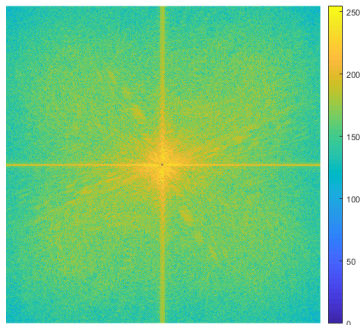


# High pass Gaussian filter, $\sigma=10, 30$

original



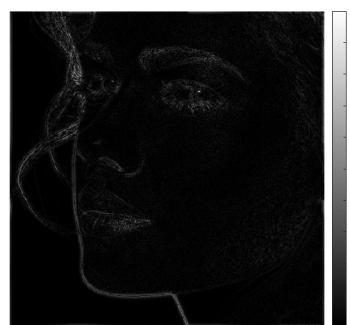
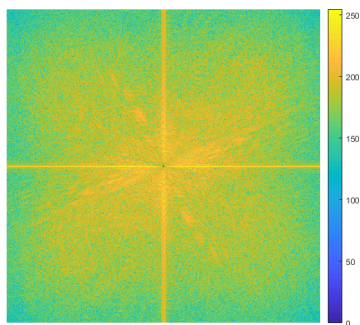
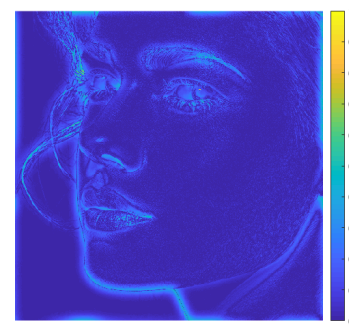
filter



output in gray



output in pseudocolor



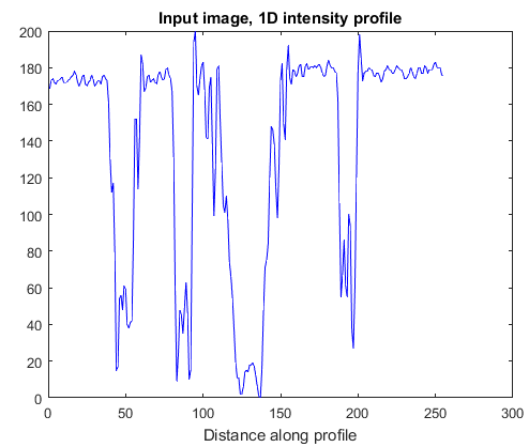
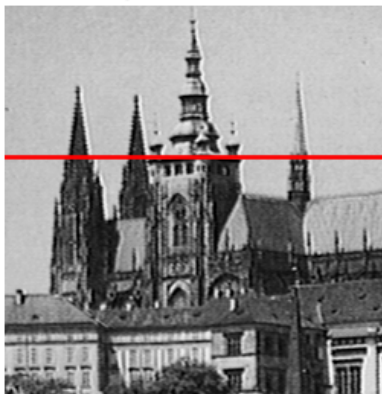


# Example, Fourier filtration, Gaussian filter

Input image



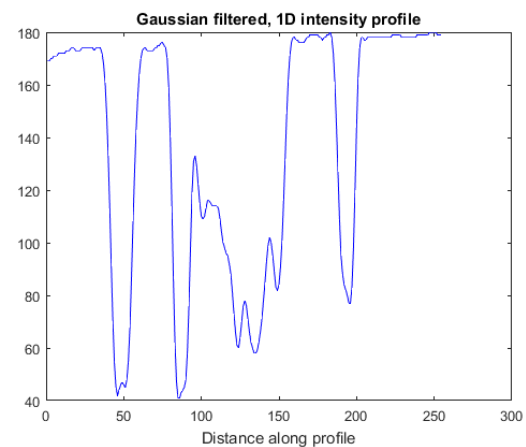
Input image, line for profile in red



Gaussian filtered



Gaussian filtered, line for profile in red

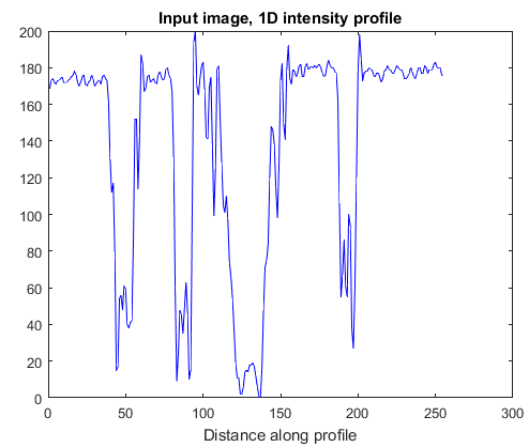
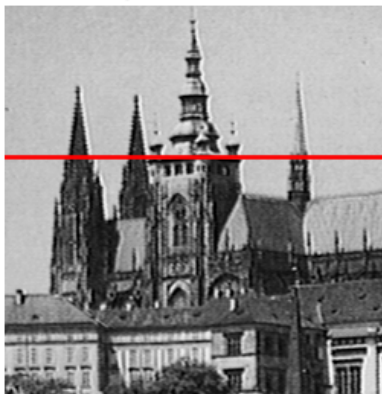


# Example, Fourier filtration, Sobel filter

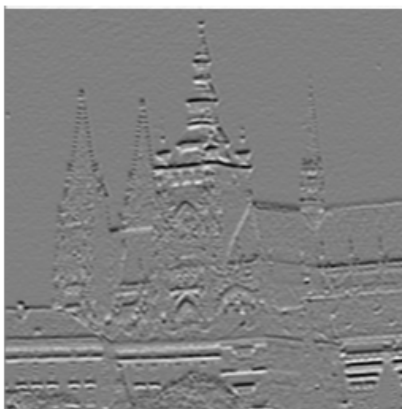
Input image



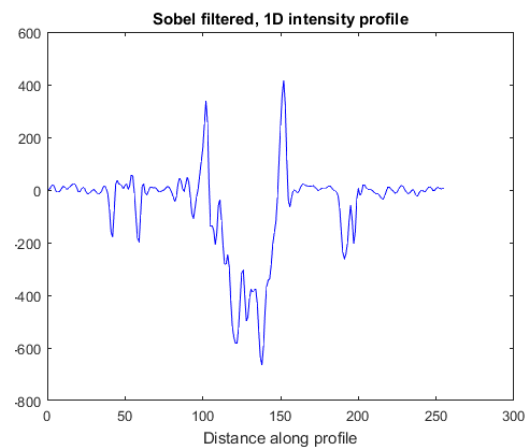
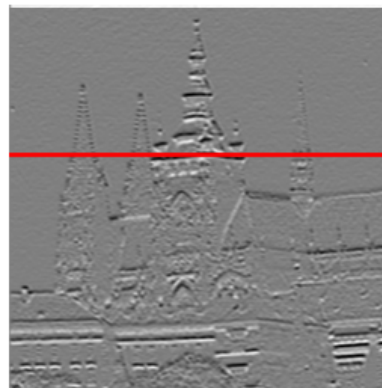
Input image, line for profile in red



Sobel filtered

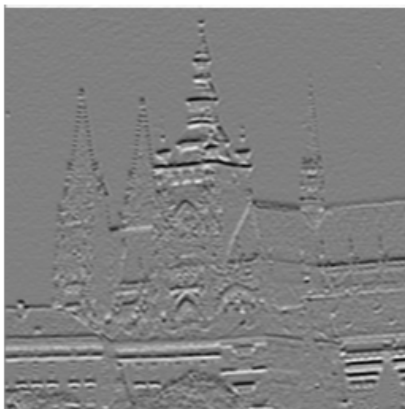


Sobel filtered, line for profile in red

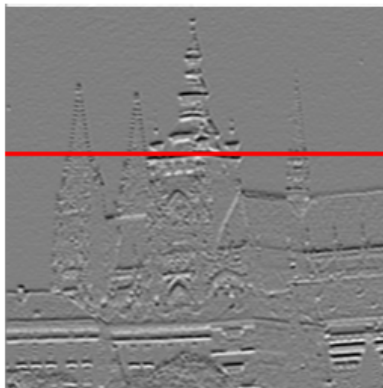


# Example, Fourier filtration, Abs(Sobel filter)

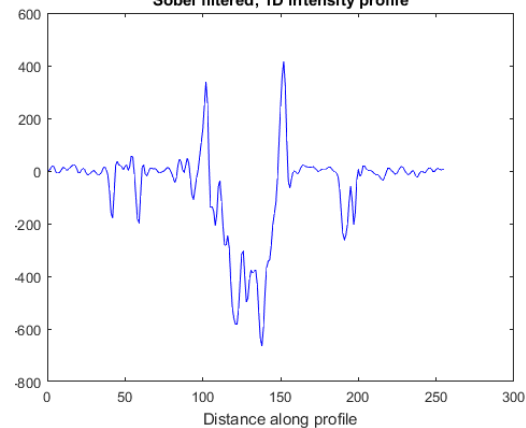
Sobel filtered



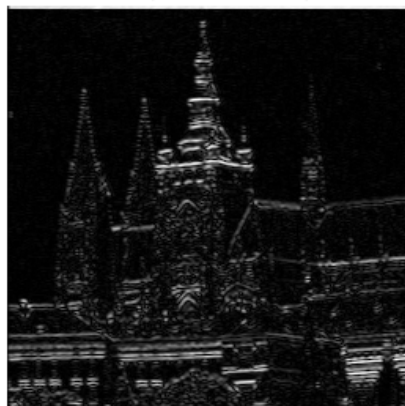
Sobel filtered, line for profile in red



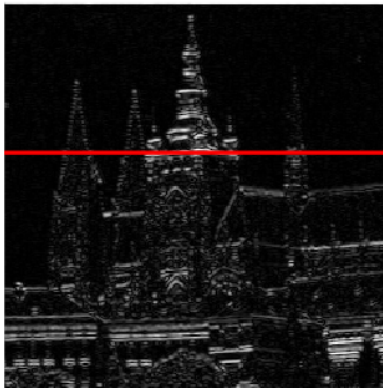
Sobel filtered, 1D intensity profile



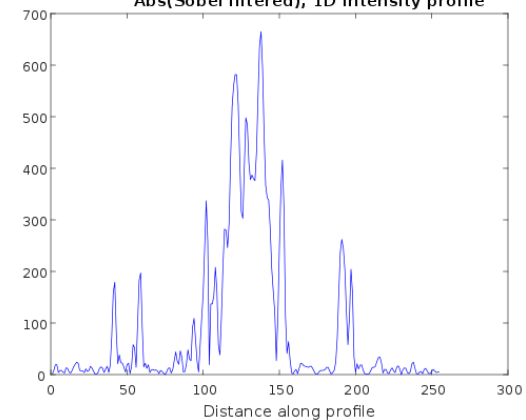
Abs(Sobel filtered)



Abs(Sobel filtered)



Abs(Sobel filtered), 1D intensity profile

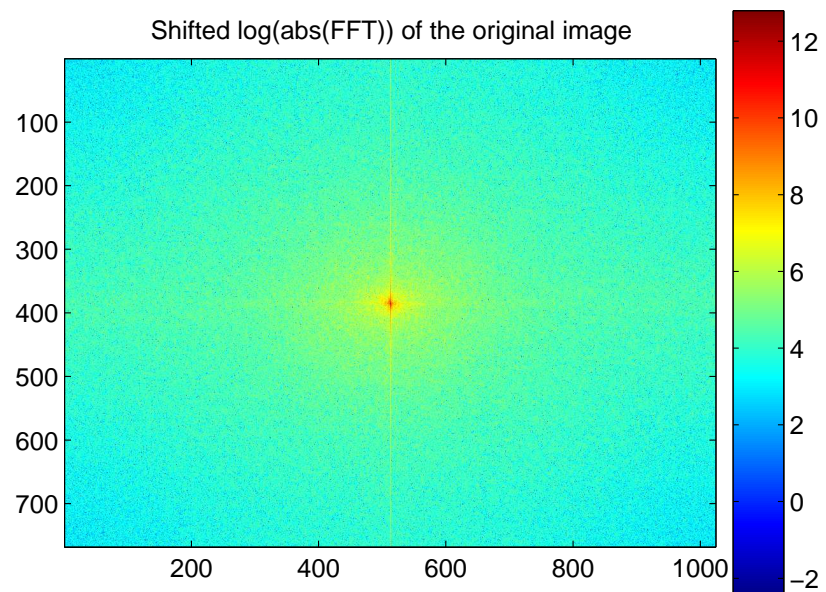


## Low pass filter, Butterworth (1)

Butterworth filter has the frequency spectrum with the smallest rippling, which converges to zero for maximal frequencies (S. Butterworth, 1930).

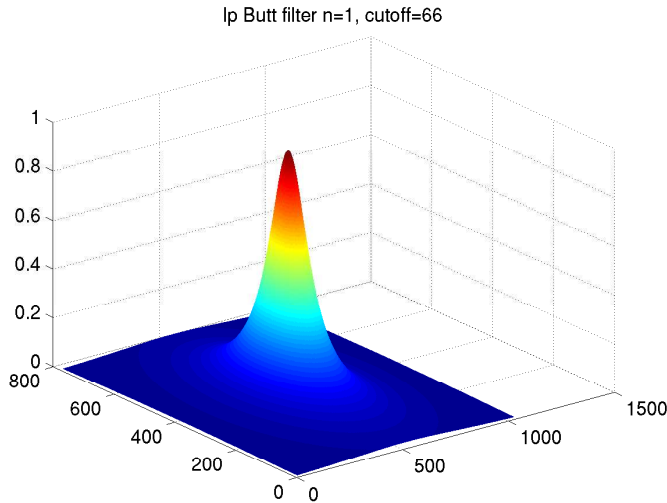


Input image

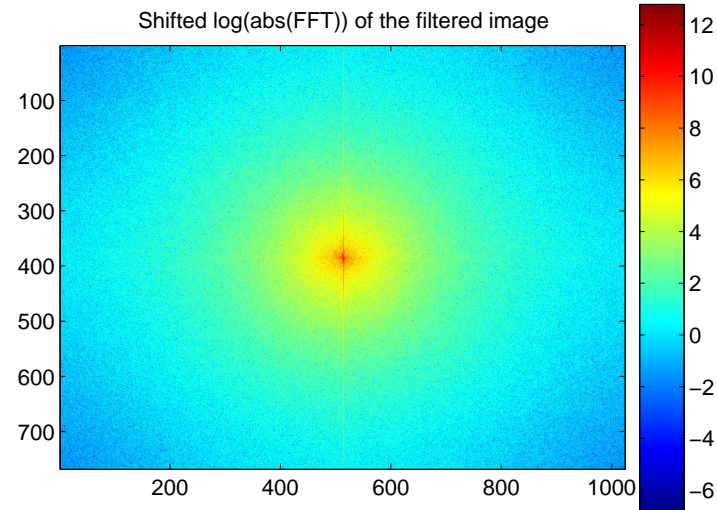




## Low pass filter, Butterworth (2)



Butterworth low pass filter



FFT of the filtered image

$$H(u, v) = \frac{1}{1 + \left(\frac{D(u, v)}{D_0}\right)^{\frac{2}{n}}}, \text{ where } D(u, v) = \sqrt{u^2 + v^2}. \text{ } n \text{ is the filter degree.}$$

$D_0$  is the frequency corresponding to the decrease of intensity by 3dB.

## Low pass filter, Butterworth (3)



Input image



Its frequency spectrum

## Homomorphic filter (1)

**The aim:** to normalize the intensity across the entire image and to increase contrast.

The method is based on the following assumptions:

- ◆ **Illumination**  $i$  changes in the image very slowly (low frequencies),
- ◆ **Reflectance**  $r$  changes in a more fast fashion, because the scene is usually rather diverse.
- ◆ The image can be **decomposed (factorized) in each pixel into a product of two components** – illumination  $i$  and reflectance  $r$ :  $f(x, y) = i(x, y) r(x, y)$ .

**The key idea:** the logarithm function can be used to separate the illumination and the reflectance components.



## Homomorphic filter (2)

$$z(x, y) = \ln f(x, y) = \ln i(x, y) + \ln r(x, y)$$

$$\text{Fourier spectrum } Z(u, v) = I(u, v) + R(u, v)$$

Filtering in the frequency domain

$$S(u, v) = H(u, v)Z(u, v) = H(u, v)I(u, v) + H(u, v)R(u, v)$$

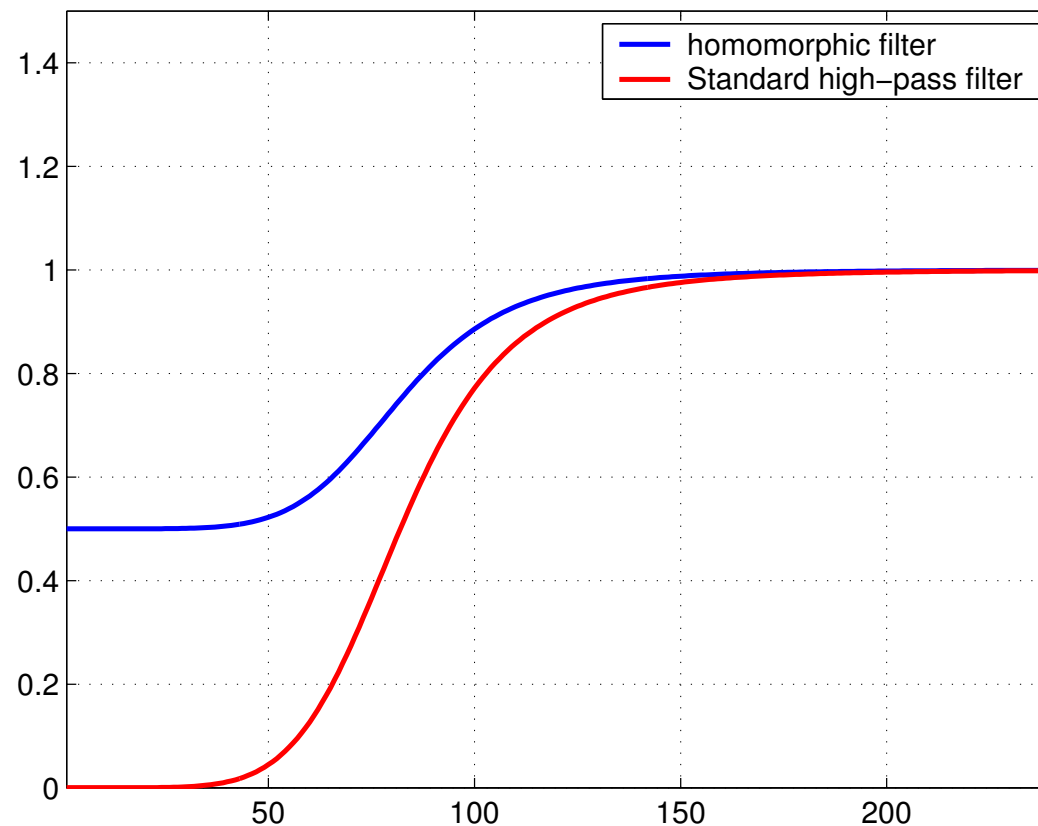
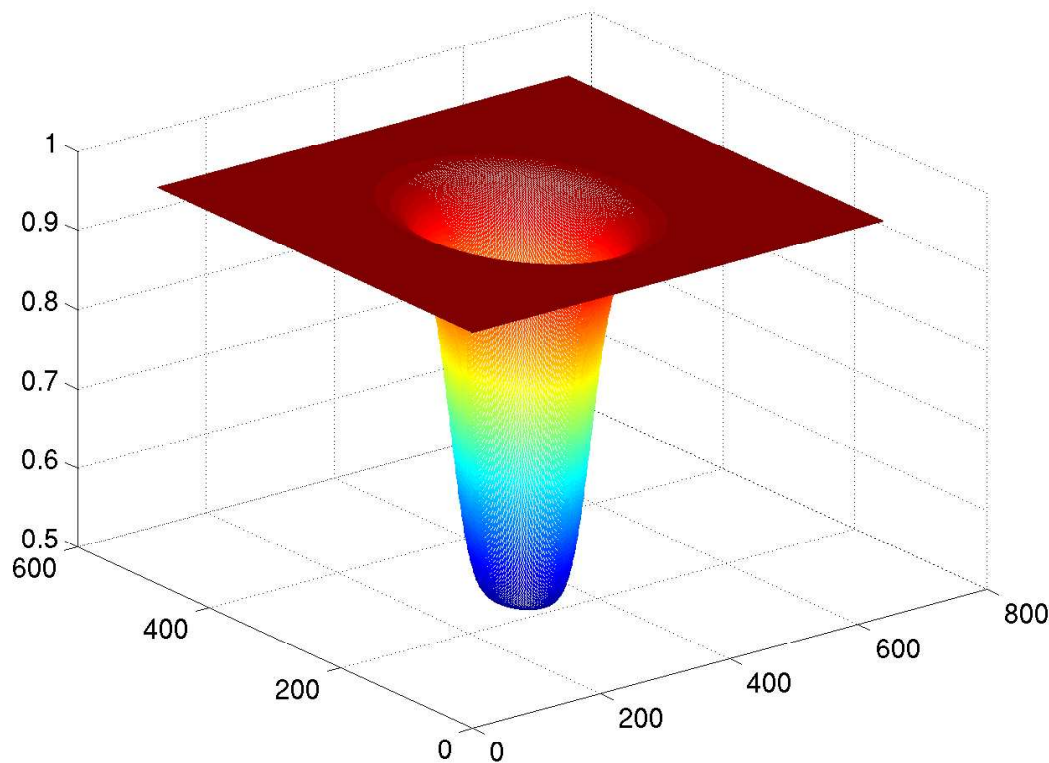
Inverse transformation back into spatial coordinates  $s(x, y) = \mathcal{F}^{-1}\{S(u, v)\}$  and return to original gray scale from the logarithmic one

$$g(x, y) = \exp(s(x, y))$$

The outcome is the suppression in the illumination changes in the scene and the improvement of the reflectance component.

# Homomorphic filters

Homomorphic filter made by adaptation of Butterworth highpass



Note: The filter is used to modify  $Z(u, v)$ , not the original spectrum  $F(u, v)$ !

## Outcome of homomorphic filtration



Original image.



Filtered image.

## Design of 2D FIR filters

- ◆ The 2D Infinite Impulse Response (IIR) filters are not used because of their instability. (causality is not secured).
- ◆ Finite Impulse Response (FIR) filters can be easily represented as a matrix of coefficients. The implementation is easy.
- ◆ 2D FIR filters are natural generalization of 1D FIR filters.
- ◆ FIR filters can be designed to have linear phase, which reduces distortions.
- ◆ Three design methods are usually used:
  1. Frequency transformation method transforms a 1D filter into 2D.
  2. Frequency sampling method creates the filter according to the desired frequency response.
  3. Windowing method composes the filter from the ideal impulse response and the smoothing window.

## Frequency transformation method

The established methods for designing 1D filters can be used. The 1D filter is converted into 2D by making the filter center symmetric. A good method.

A MATLAB example (Parks-McClellan optimal design):

```
b = remez(10,[0 0.4 0.6 1],[1 1 0 0]);
```

```
h = ftrans2(b);
```

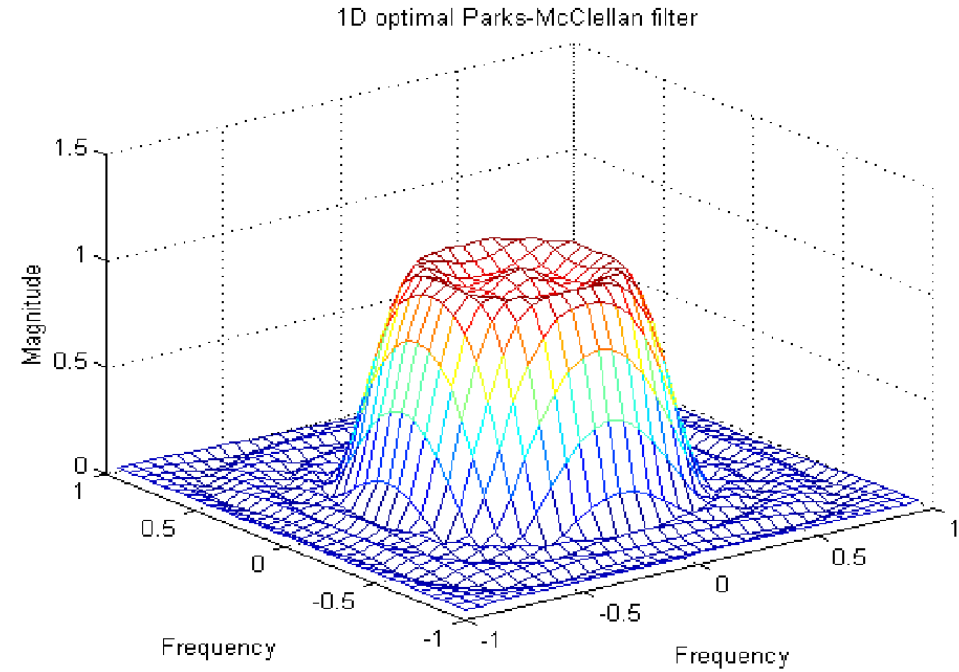
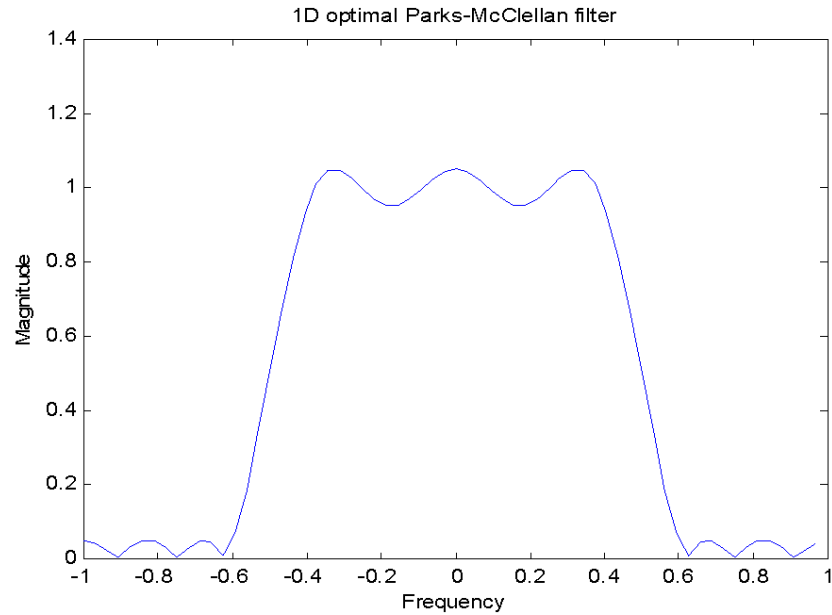
```
[H,w] = freqz(b,1,64,'whole');
```

```
colormap(jet(64))
```

```
plot(w/pi-1,fftshift(abs(H))) figure, freqz2(h,[32 32])
```

# 2D Parks-McClellan Filter

## Example continuation



## Frequency sampling method

The desired frequency response is given. The filter is created in the matrix form securing that the response passes given frequency response points. The behavior can be arbitrary outside the given points. Oscillations are common.

**MATLAB example** (design of the  $11 \times 11$  filter)

```
Hd = zeros(11,11); Hd(4:8,4:8) = 1;
```

```
[f1,f2] = freqspace(11,'meshgrid');
```

```
mesh(f1,f2,Hd), axis([-1 1 -1 1 0 1.2]), colormap(jet(64))
```

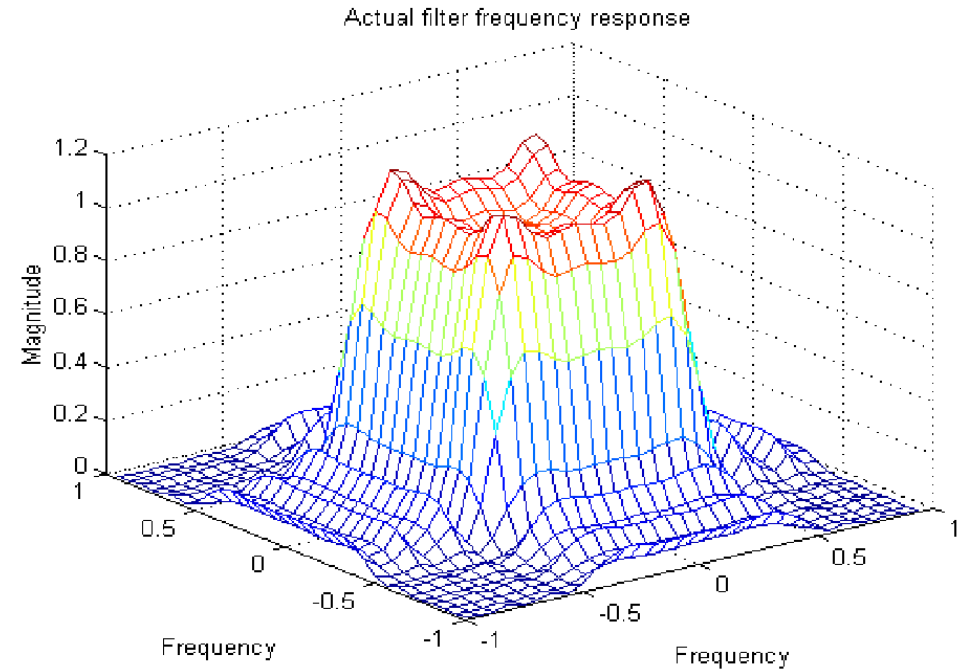
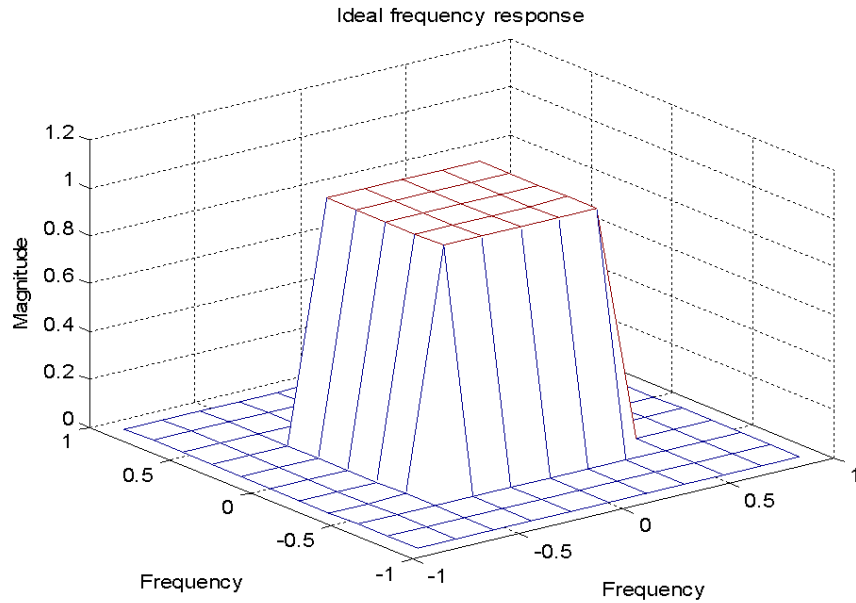
```
h = fsamp2(Hd);
```

```
figure, freqz2(h,[32 32]), axis([-1 1 -1 1 0 1.2])
```



# Frequency sampling method

## Example continuation



## Windowing method

The ideal response of the filter smoothes the coefficients in the windos. The ideal filter is approximated.

The results are usually better than the results of the Frequency Sampling Method.

```
Hd = zeros(11,11); Hd(4:8,4:8) = 1;
```

```
[f1,f2] = freqspace(11,'meshgrid');
```

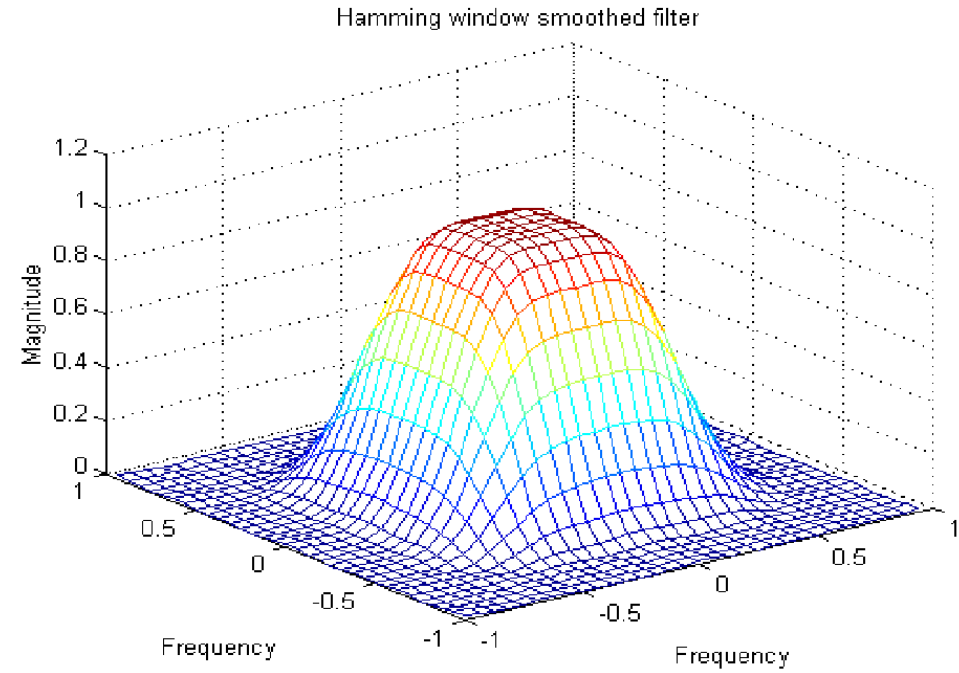
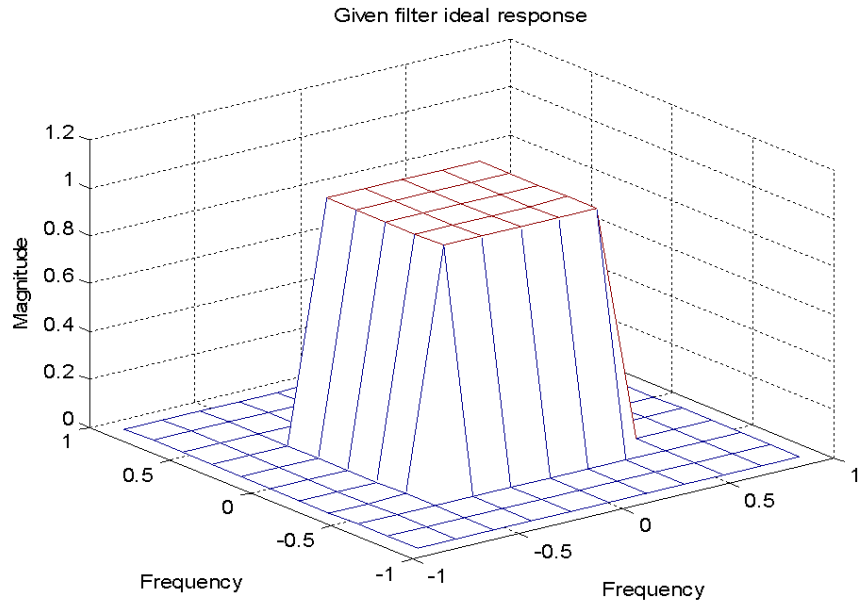
```
mesh(f1,f2,Hd), axis([-1 1 -1 1 0 1.2]), colormap(jet(64))
```

```
h = fwind1(Hd,hamming(11));
```

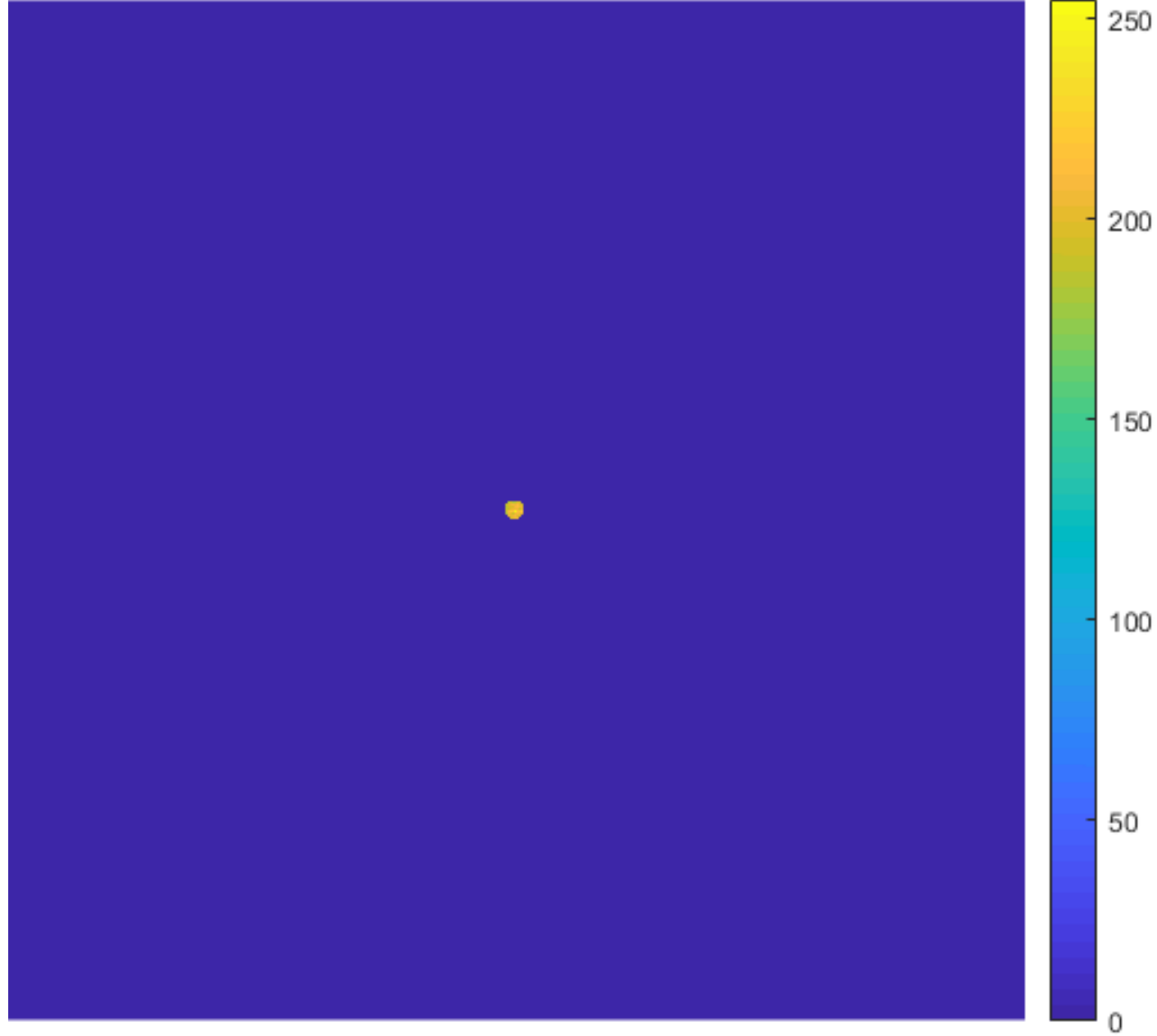
```
figure, freqz2(h,[32 32]), axis([-1 1 -1 1 0 1.2])
```

# Windowing method

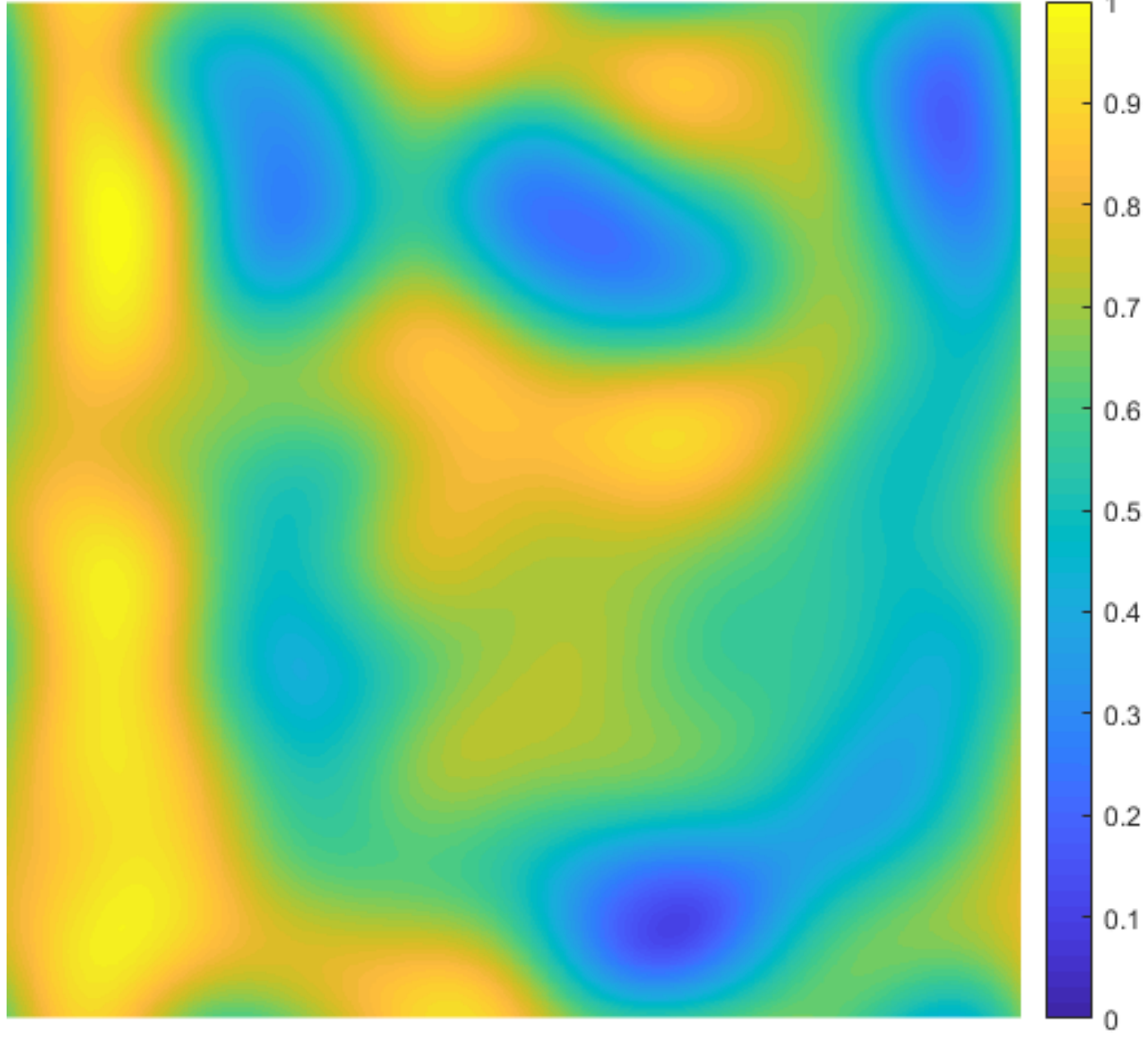
## Example continuation



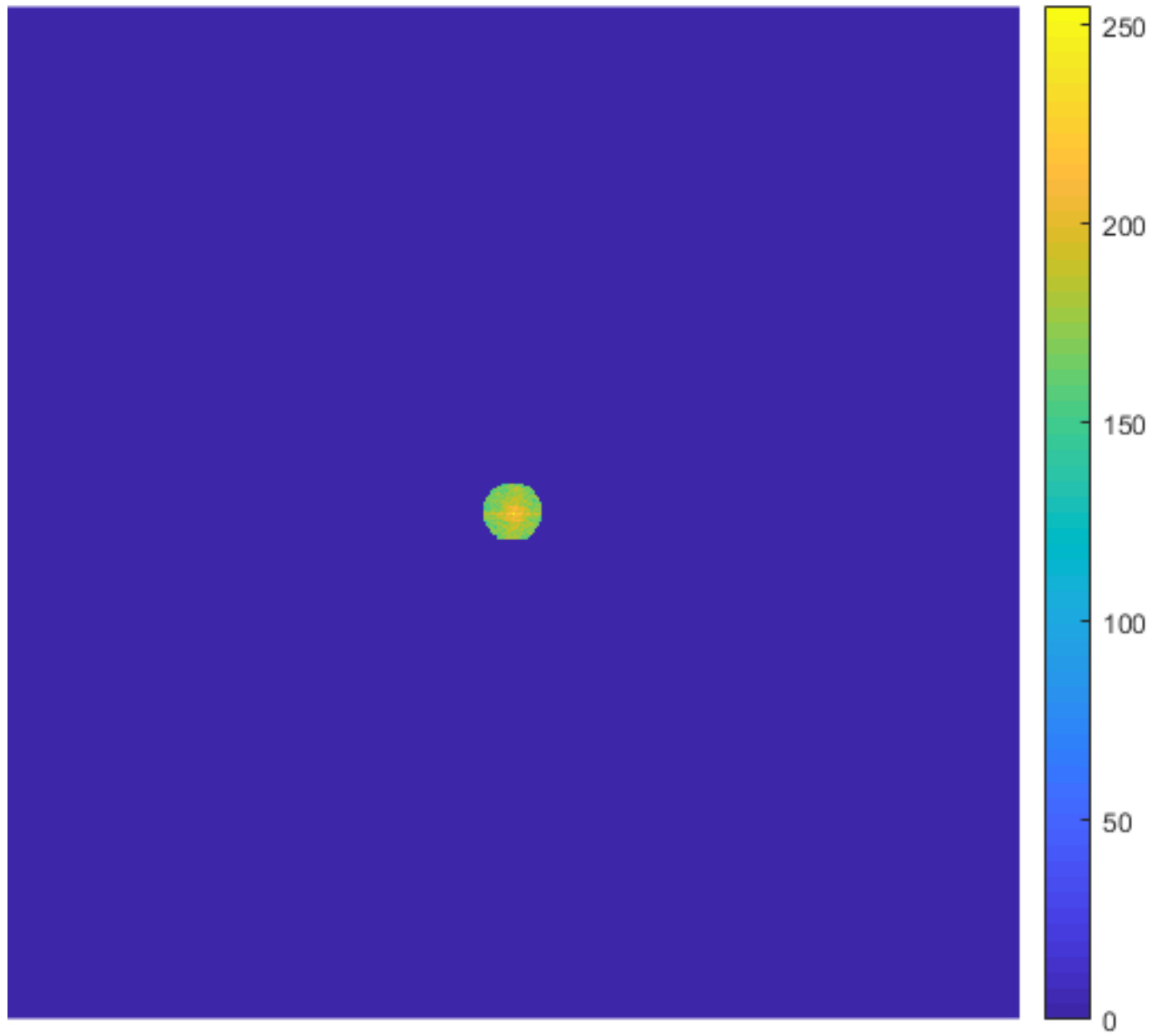




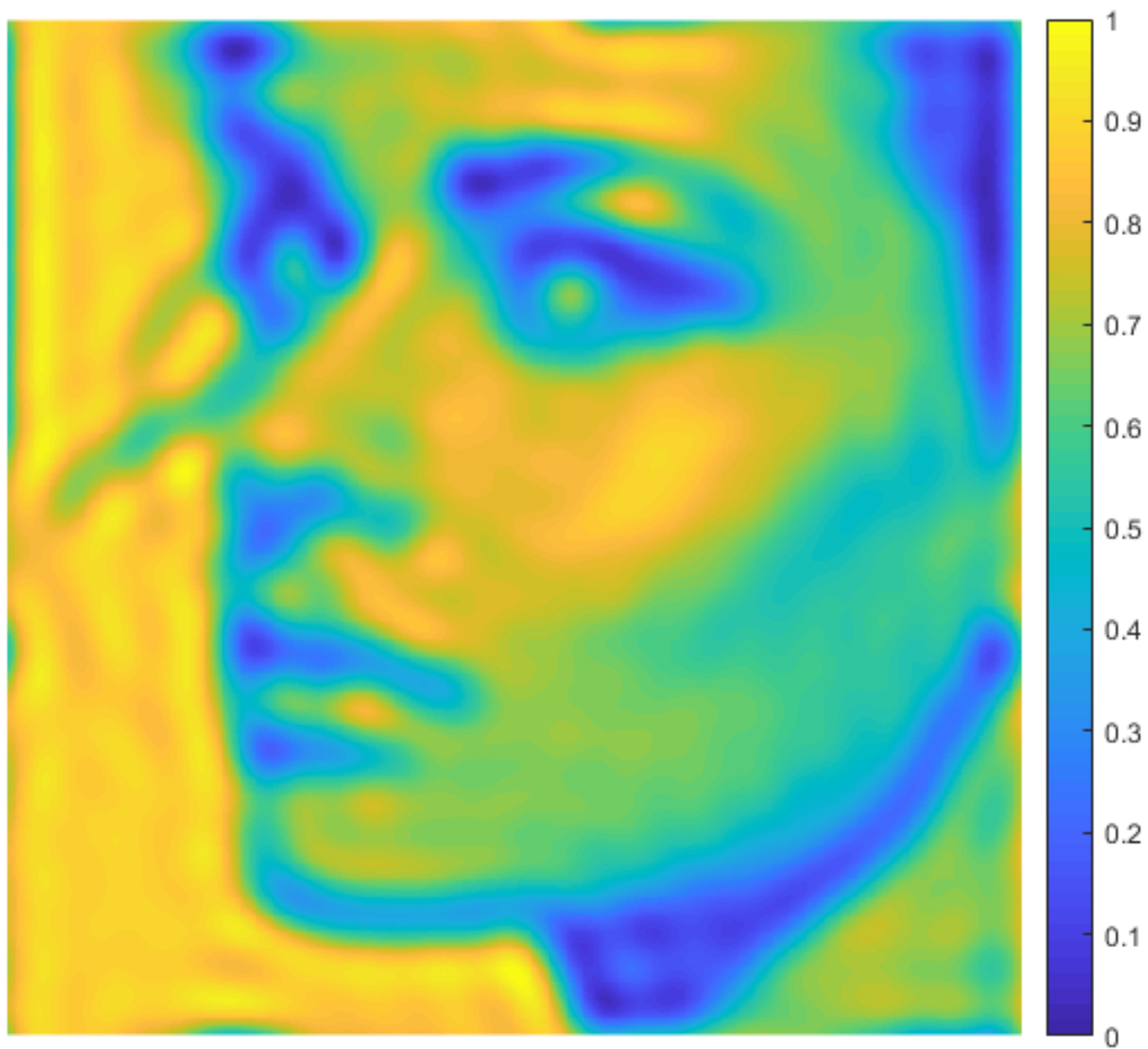


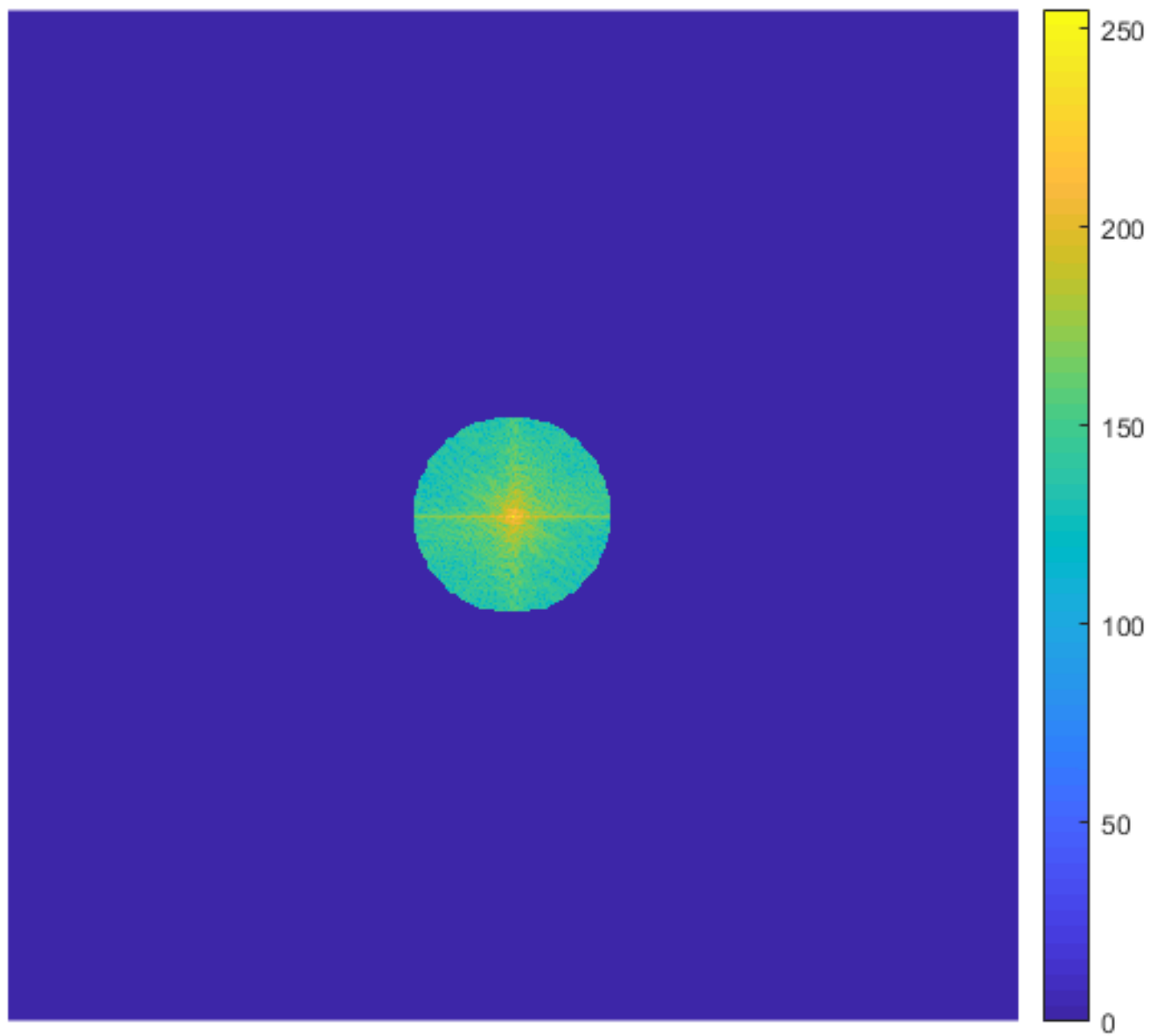




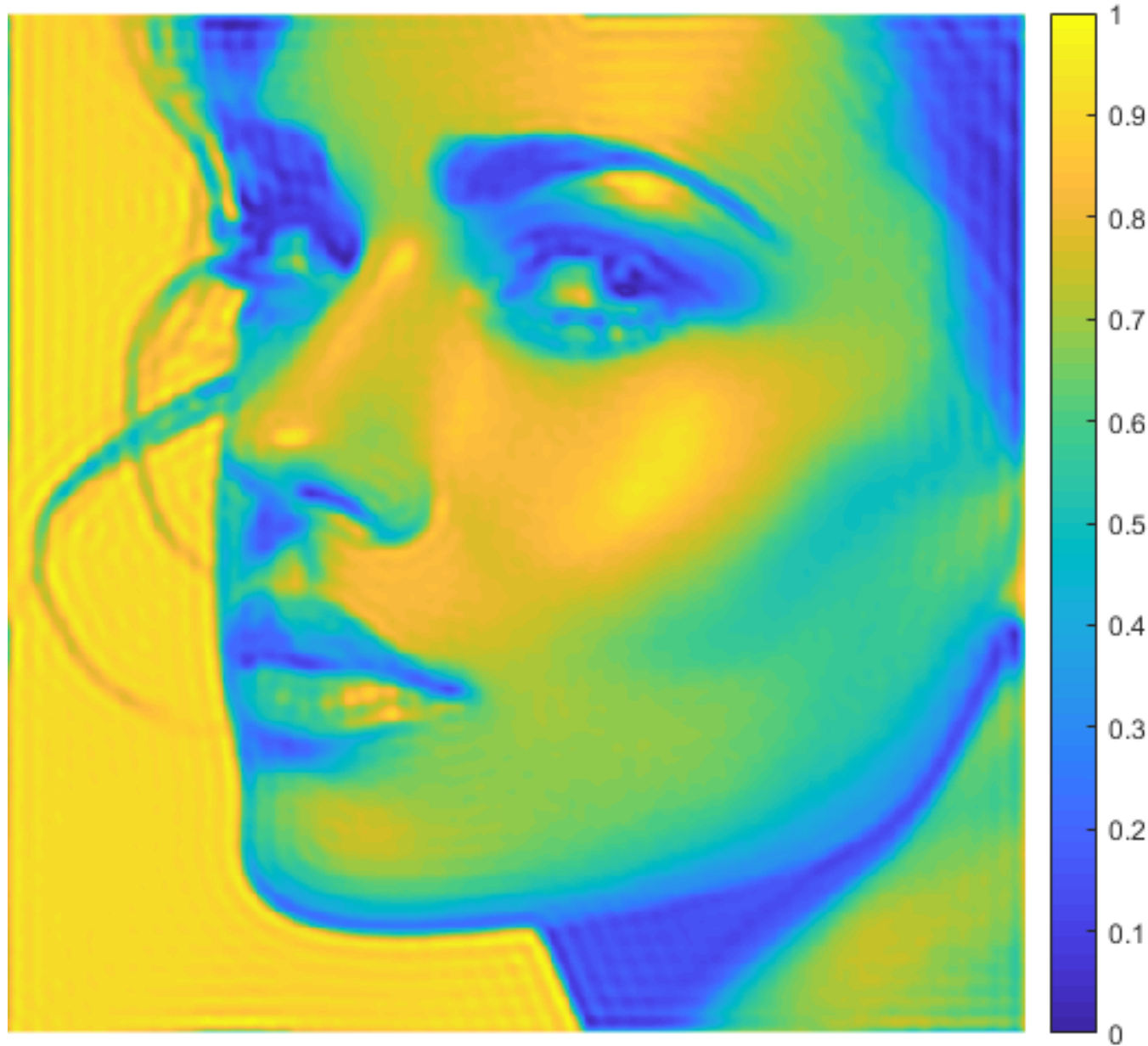






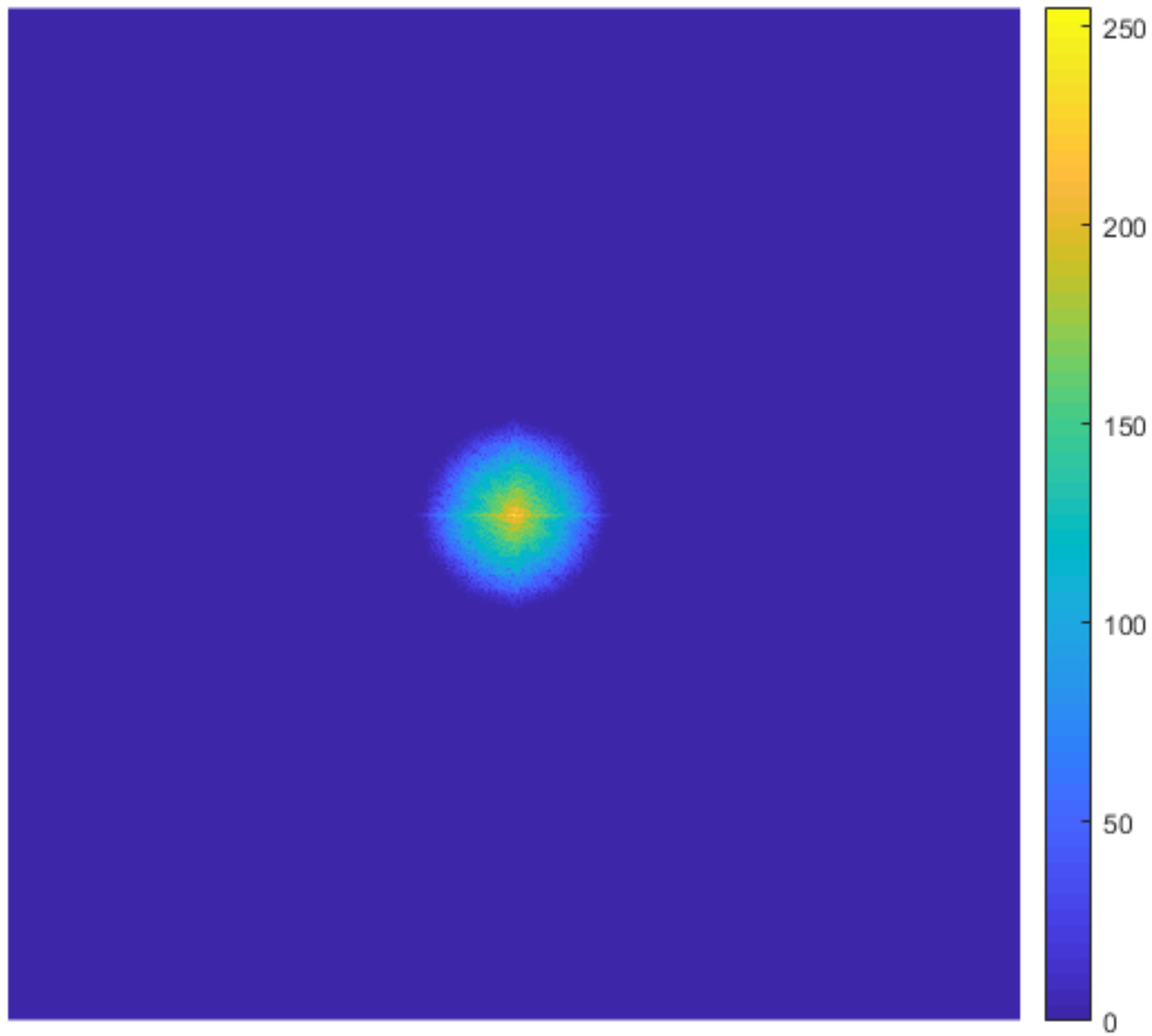


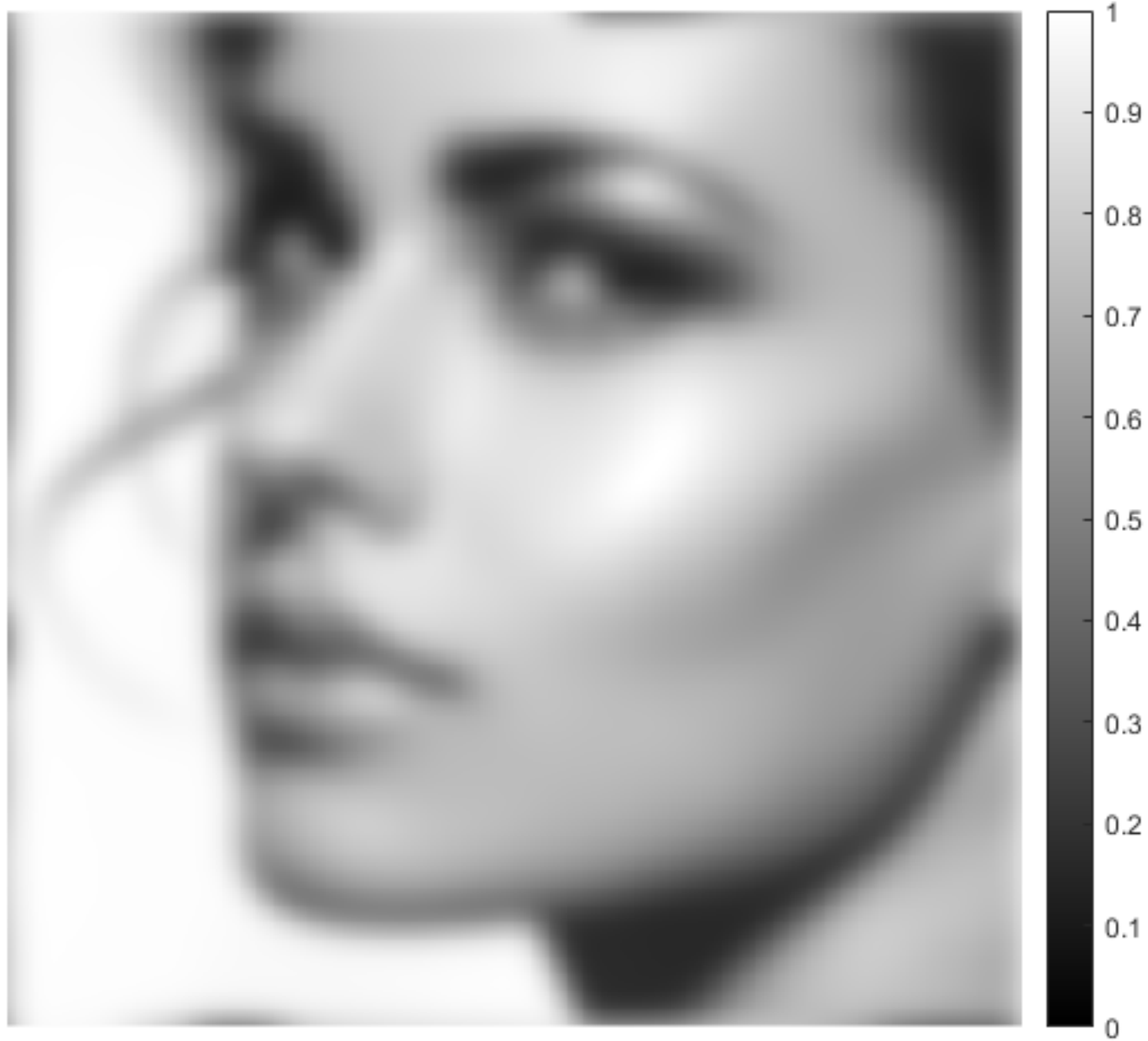


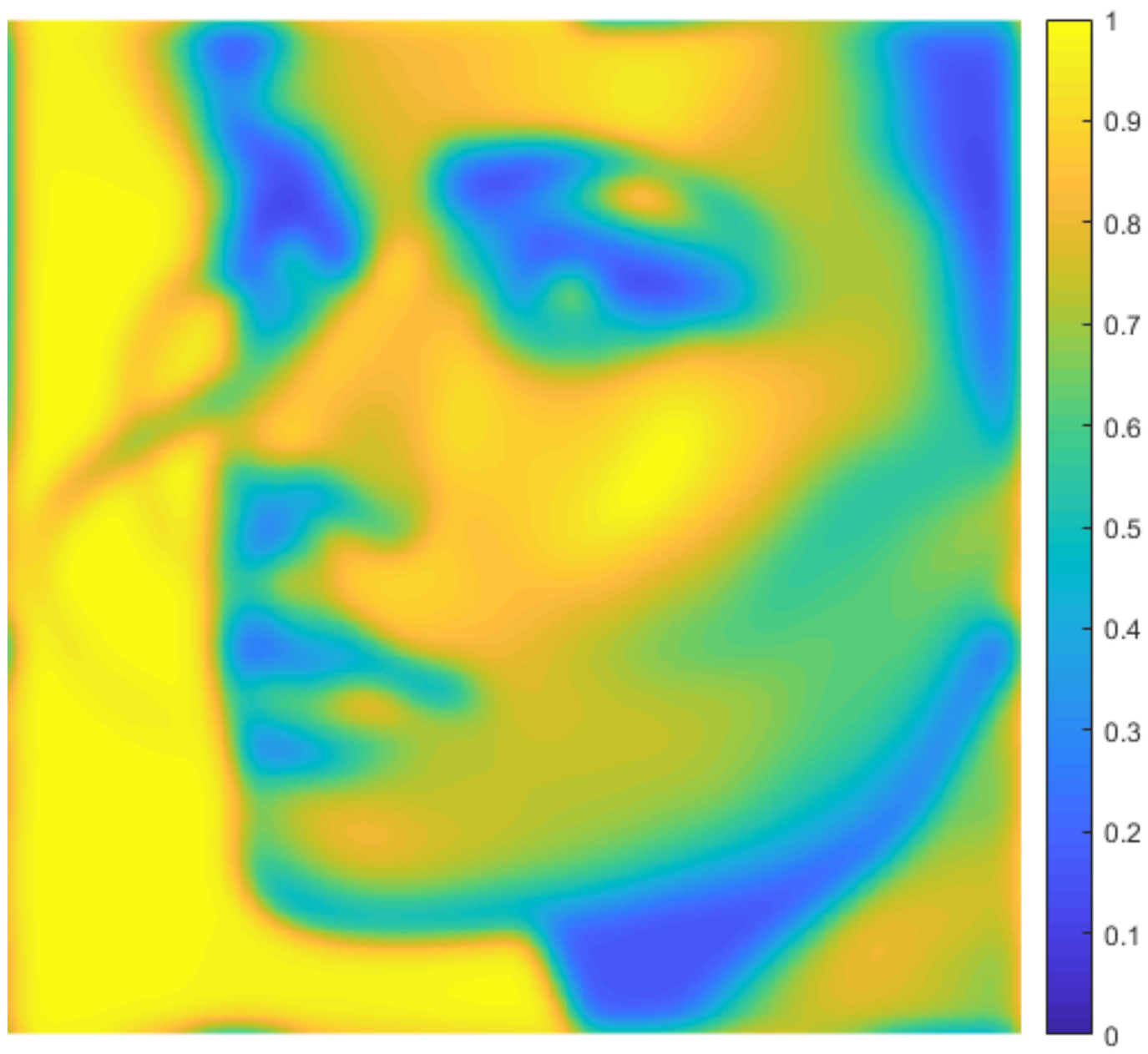


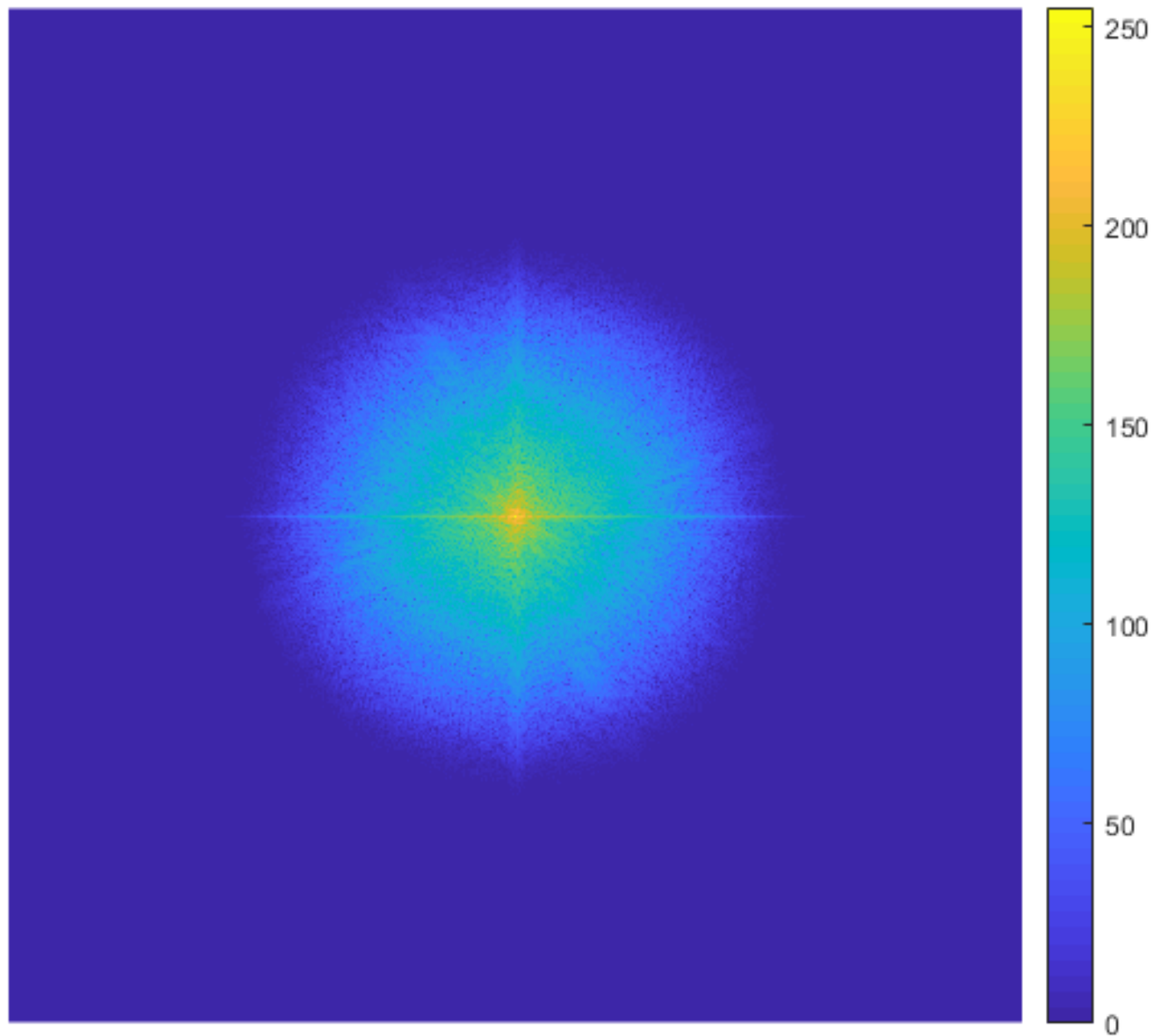


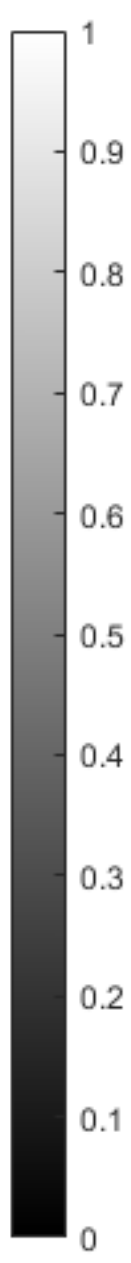
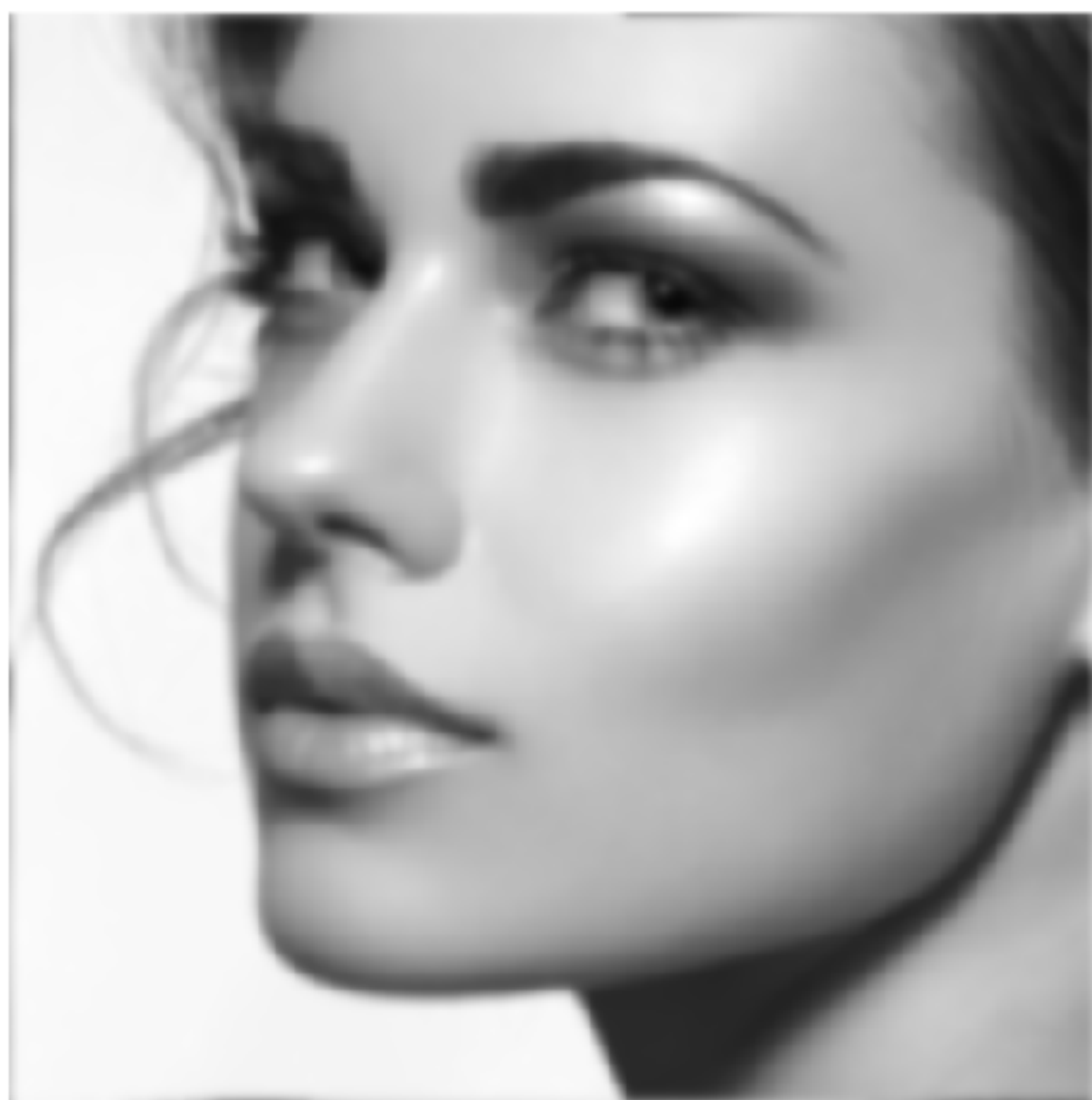


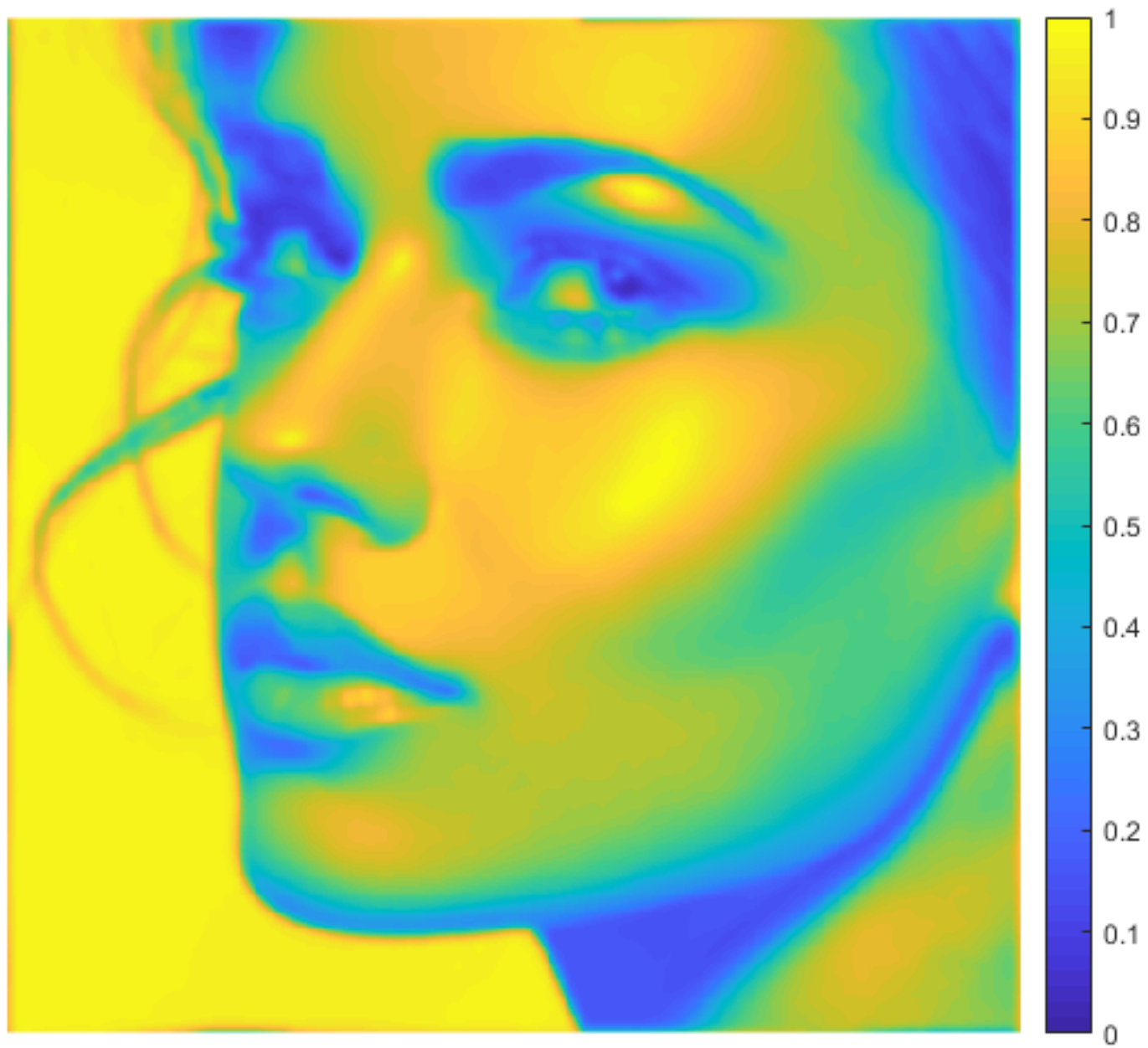






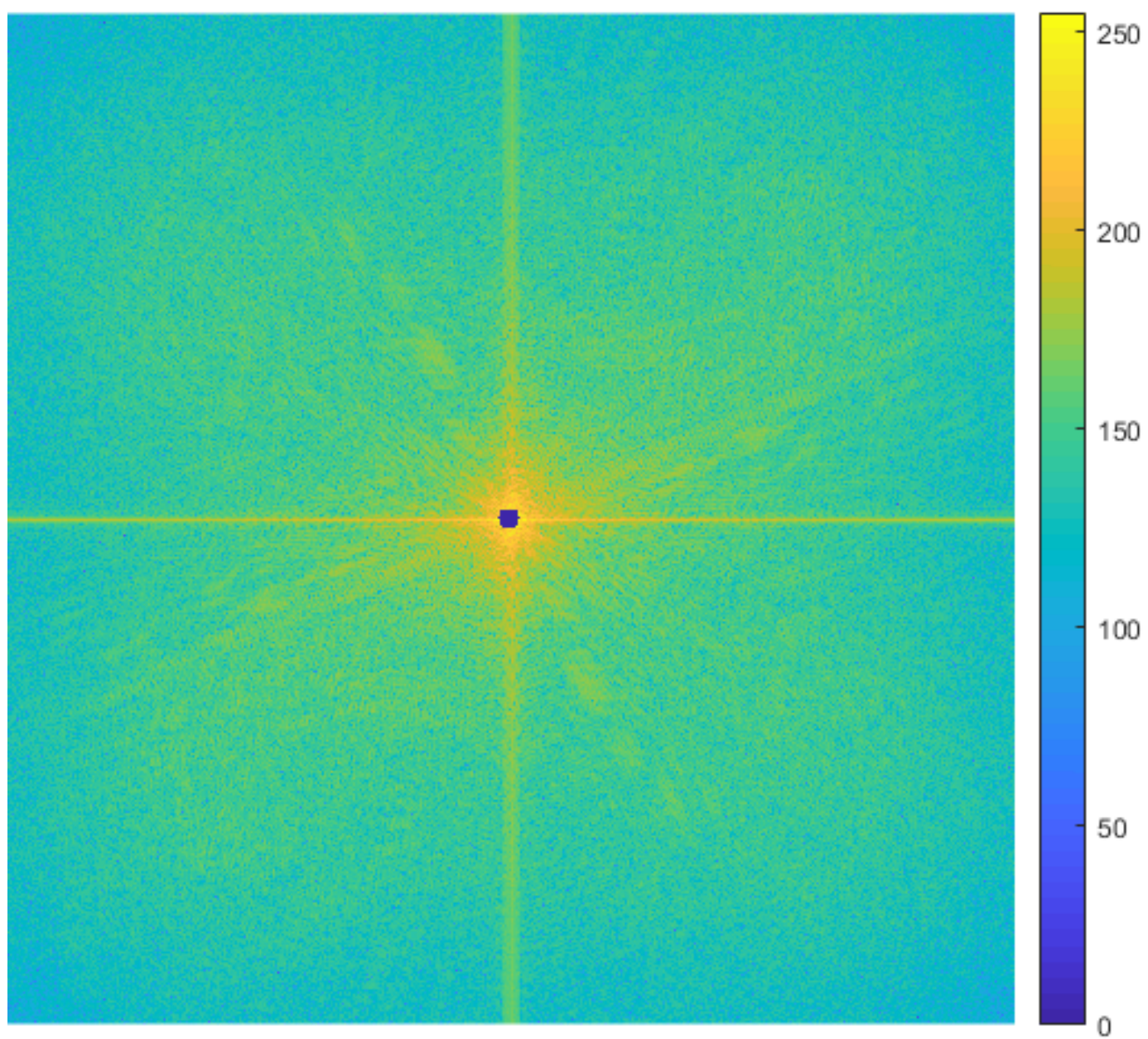


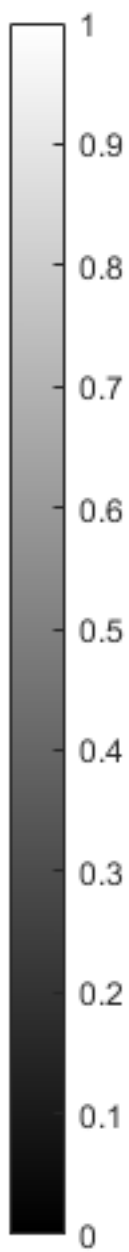
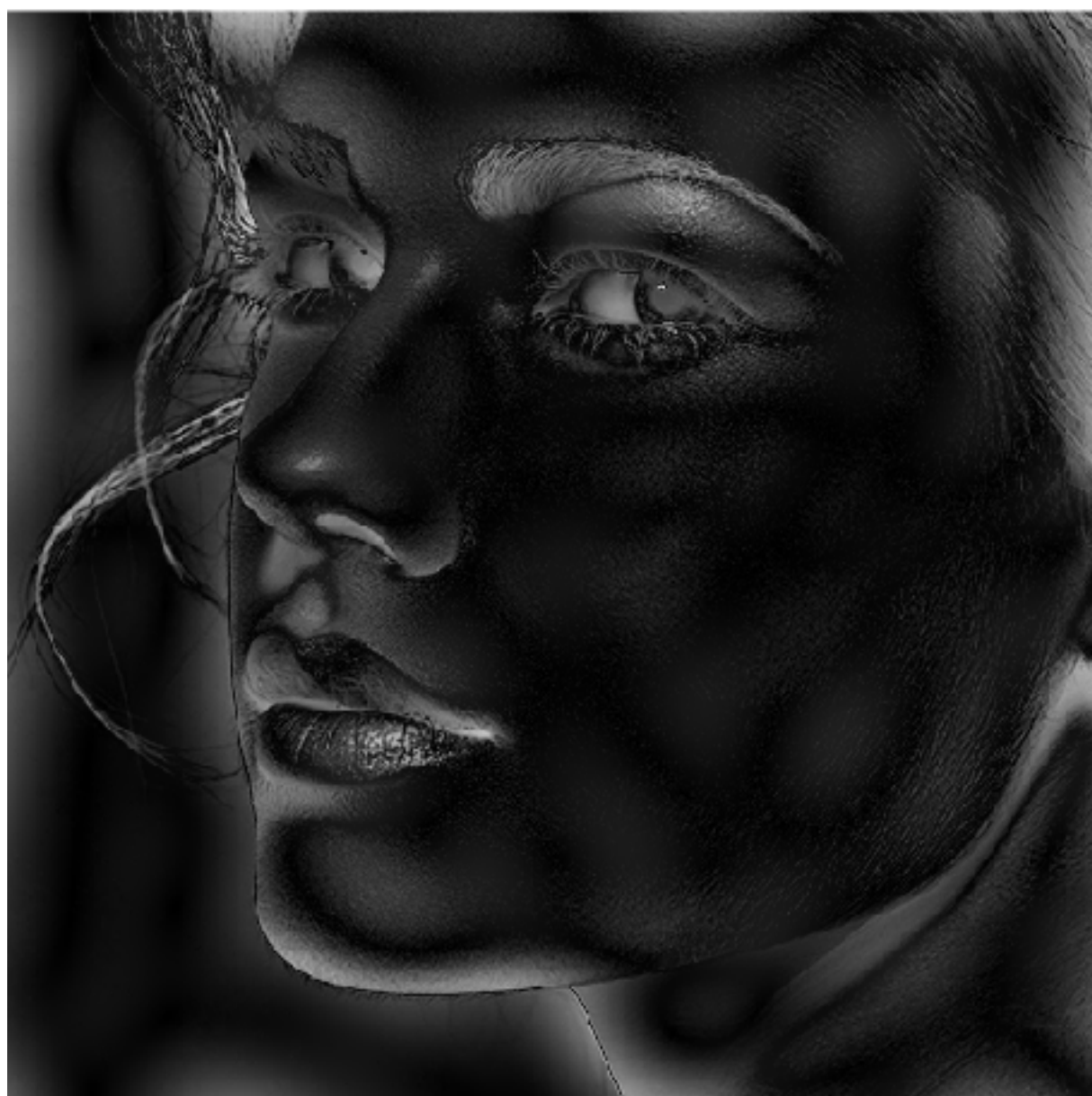




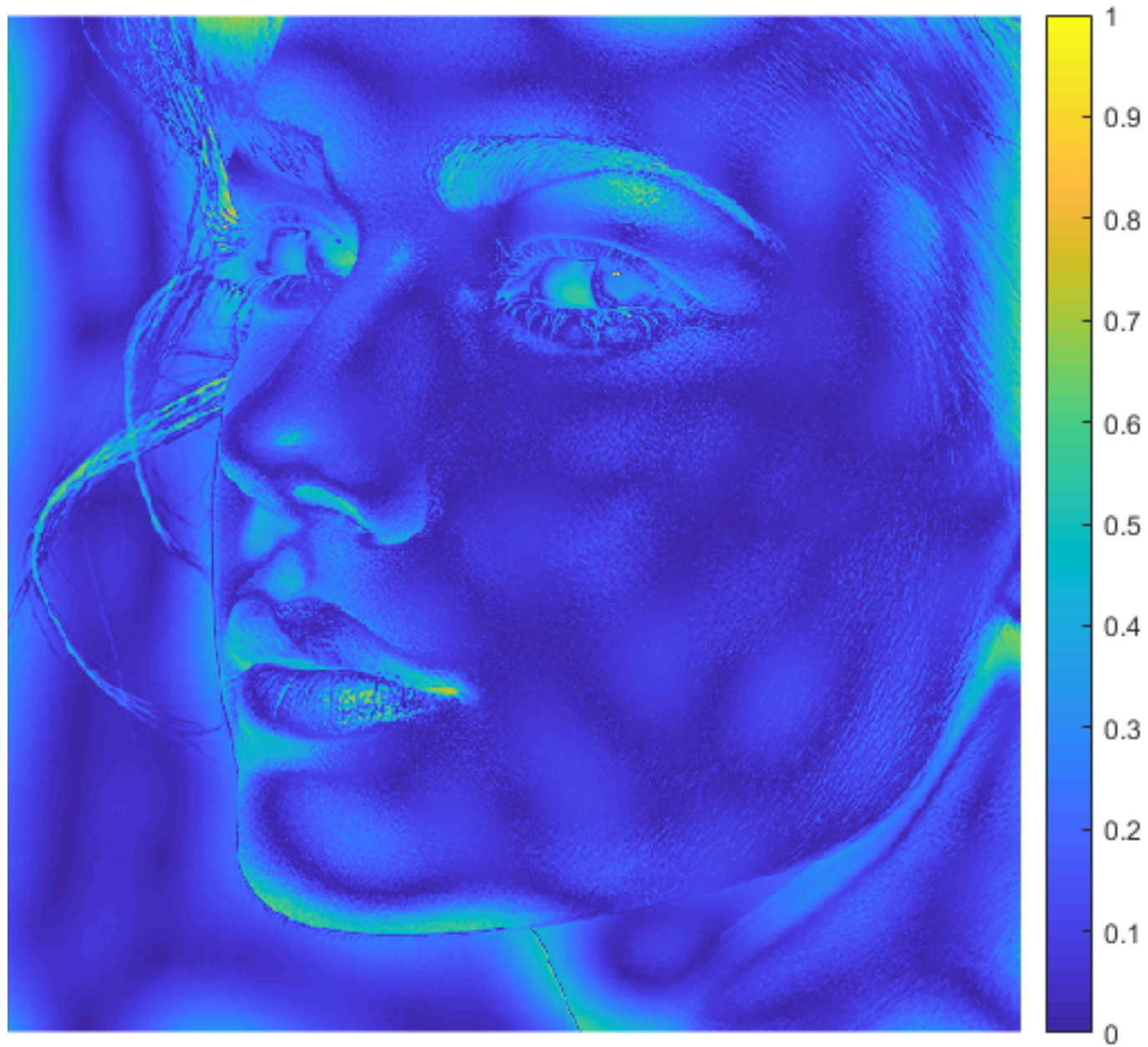


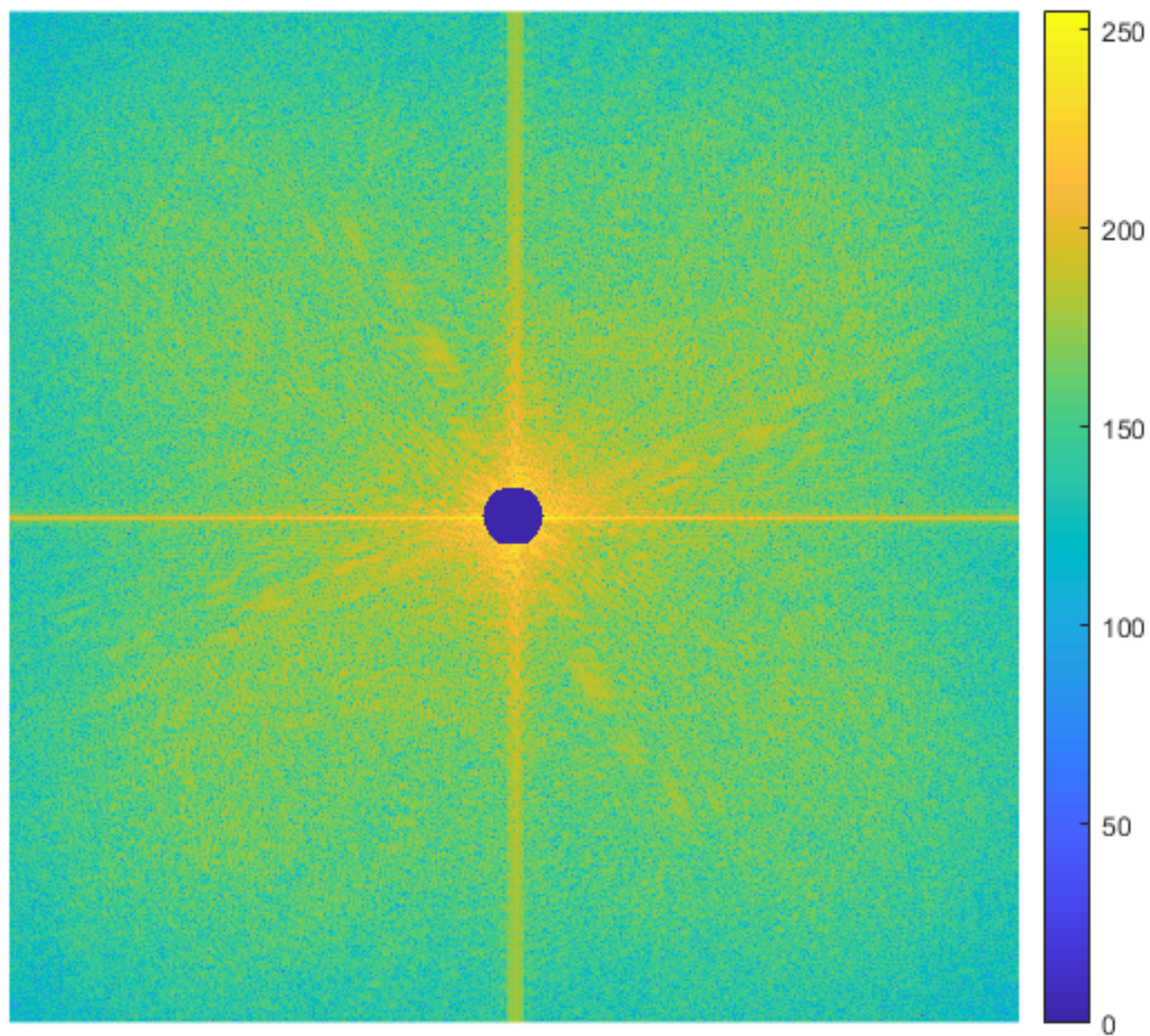




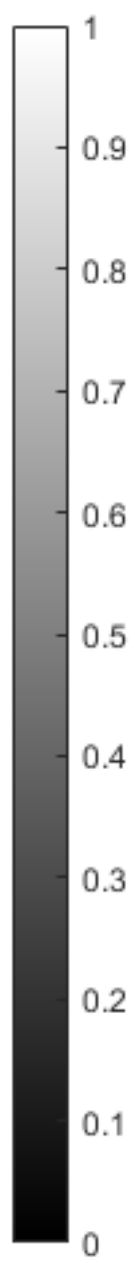
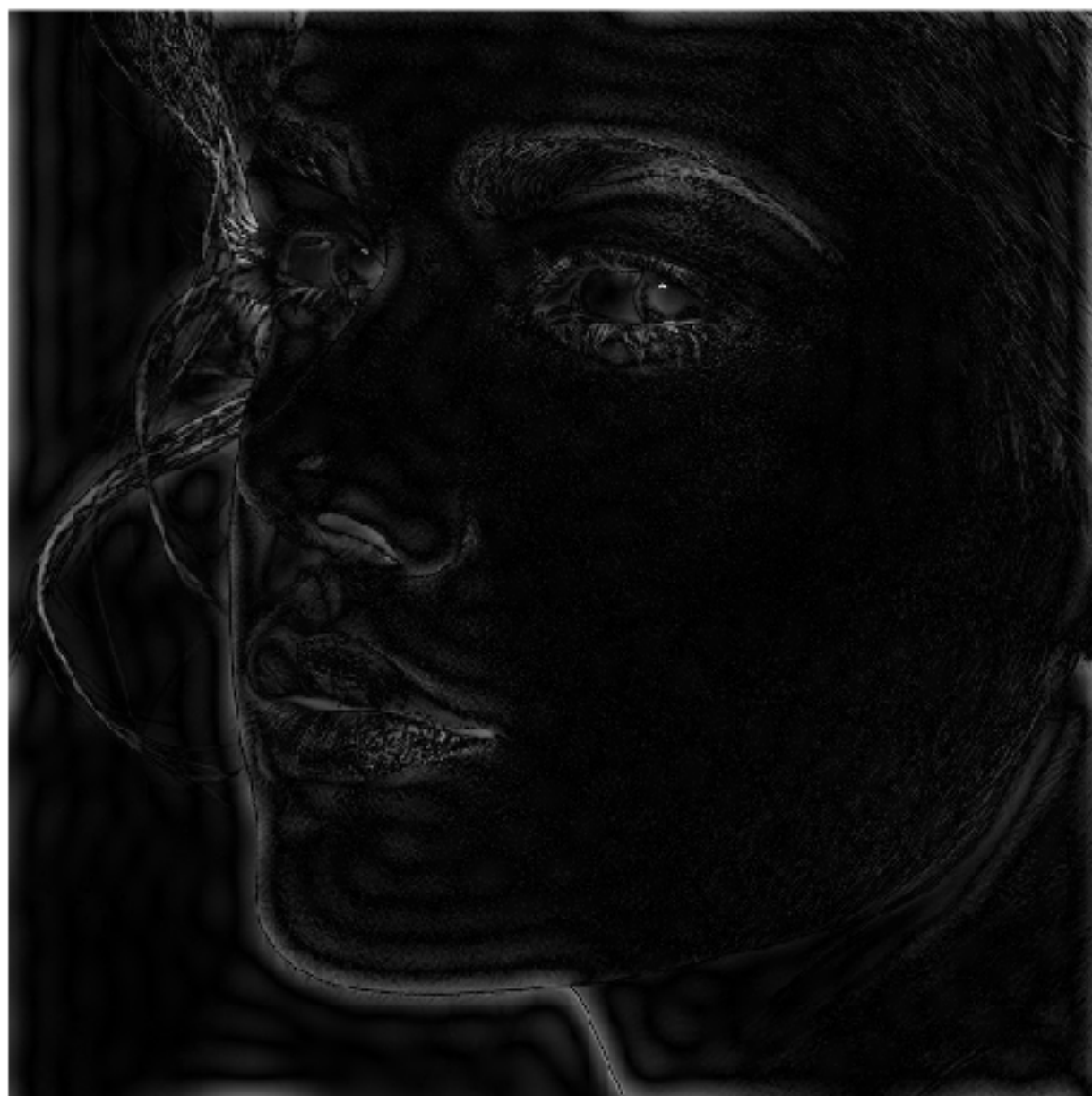


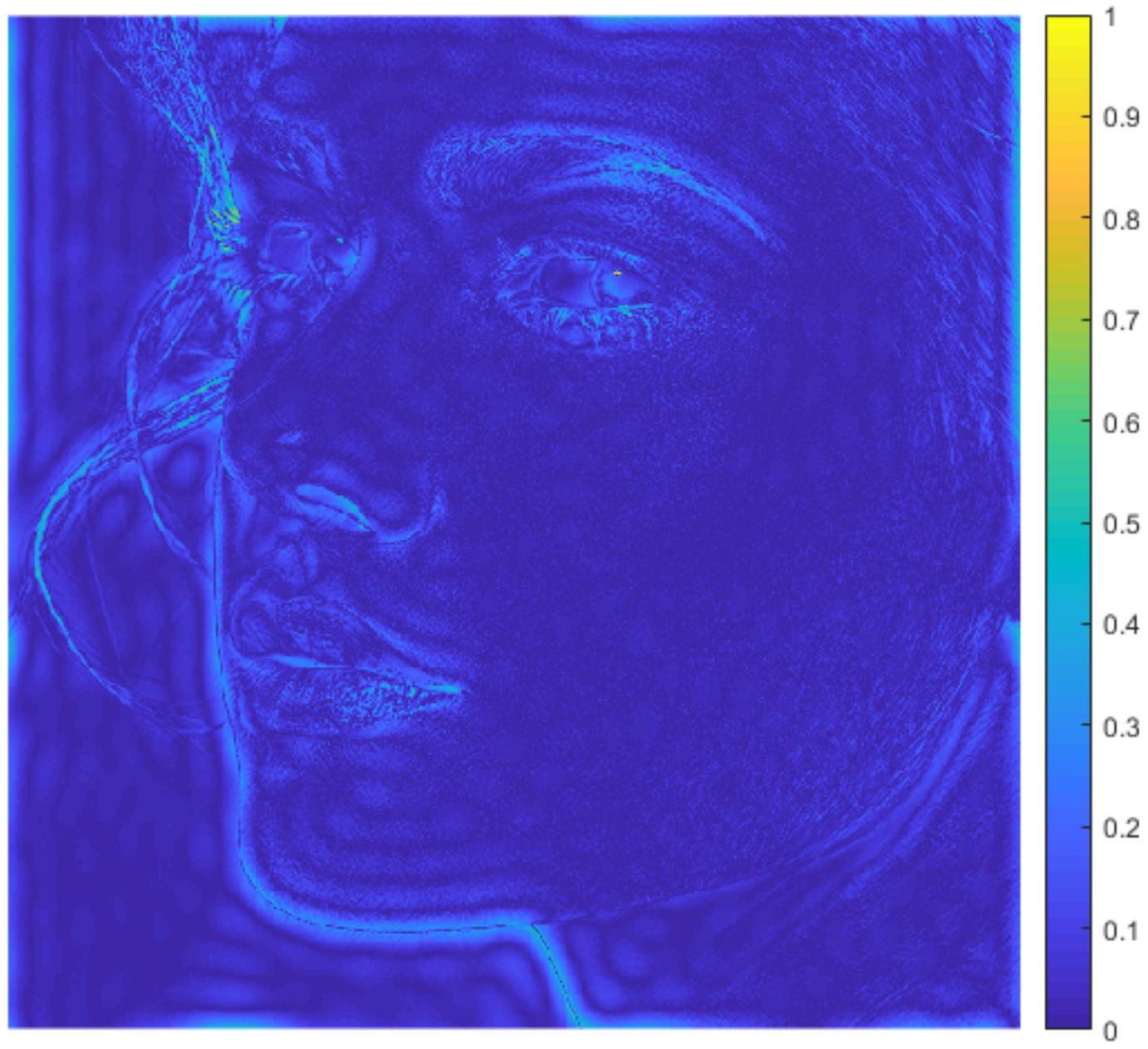






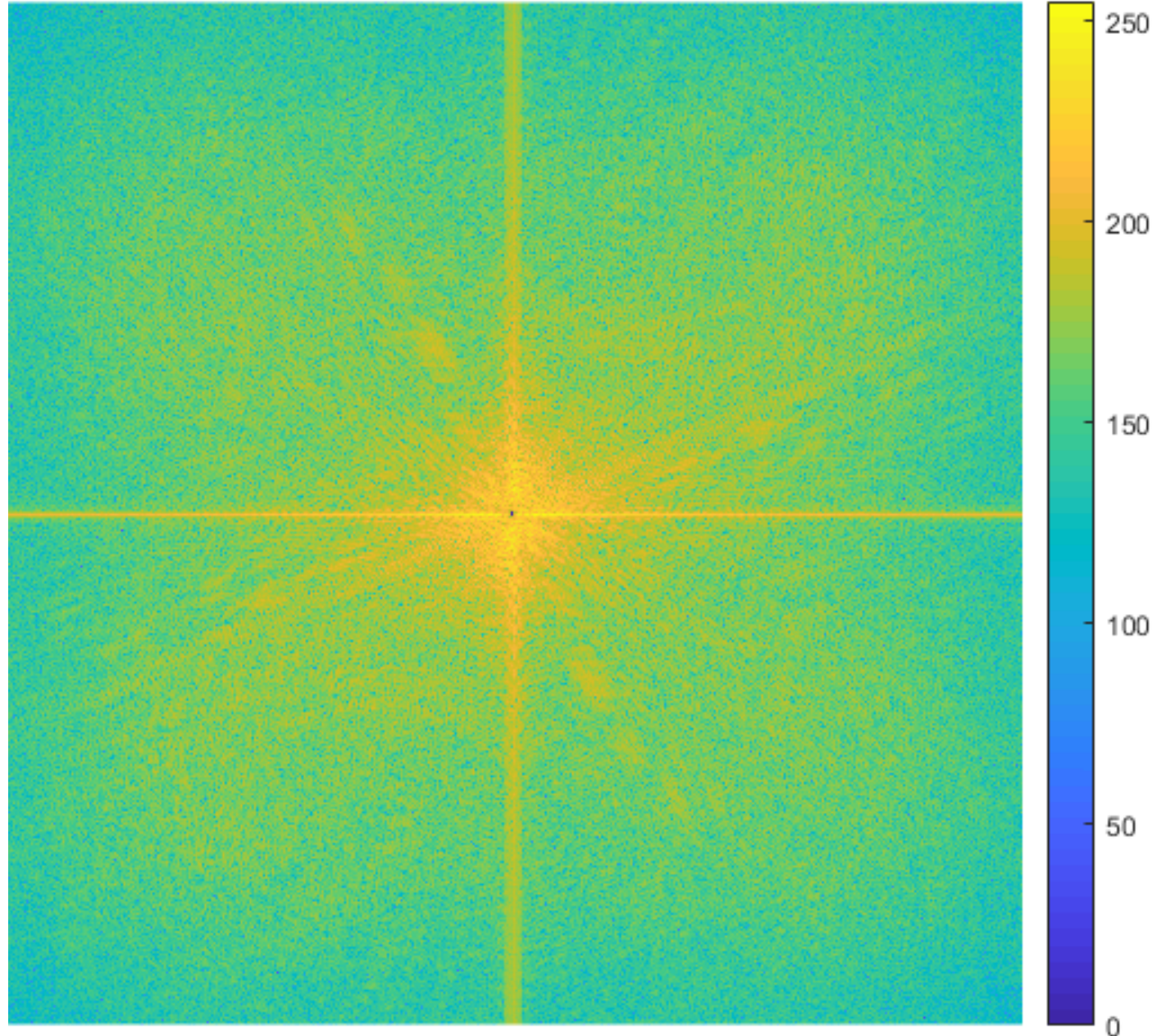






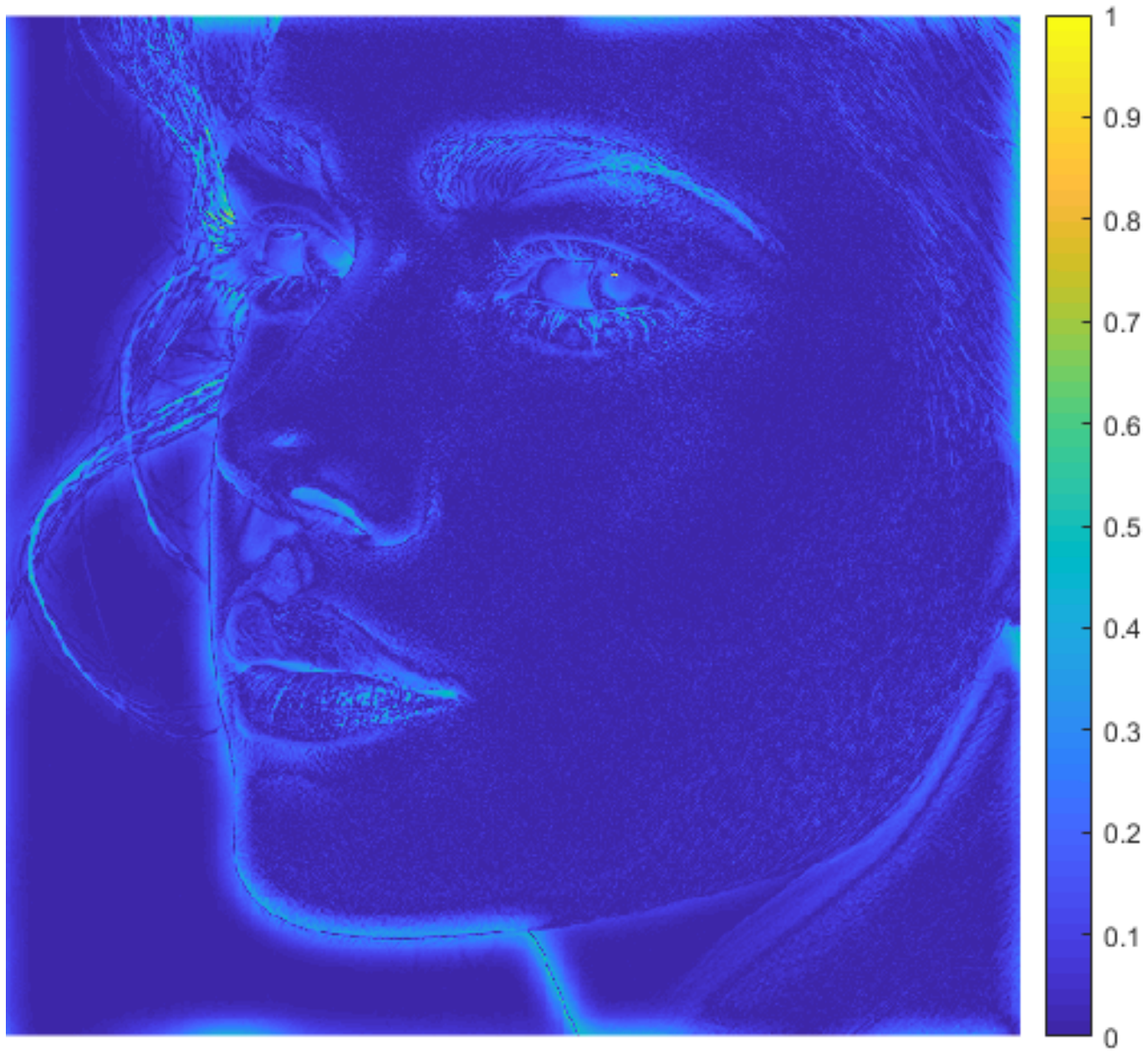


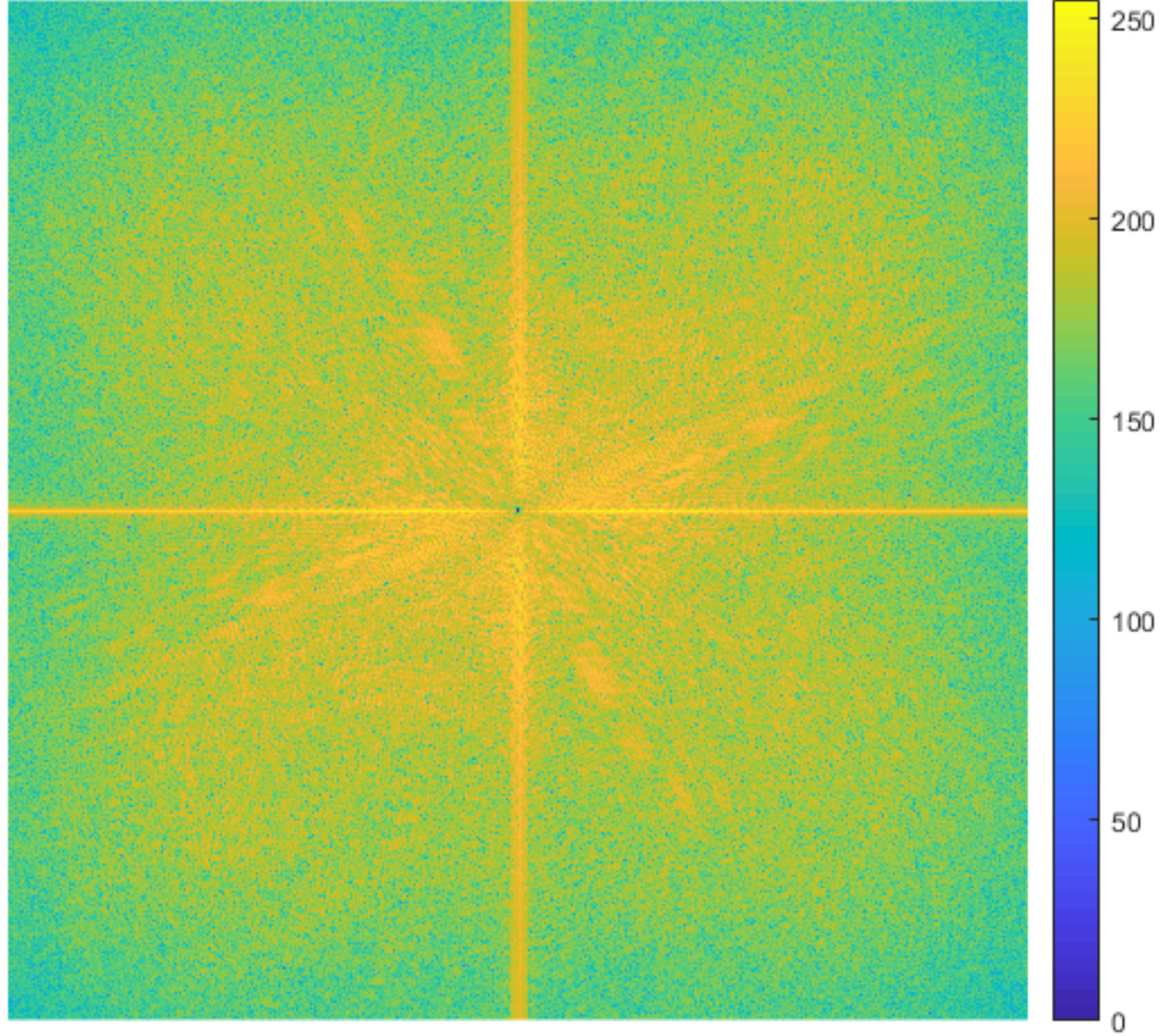




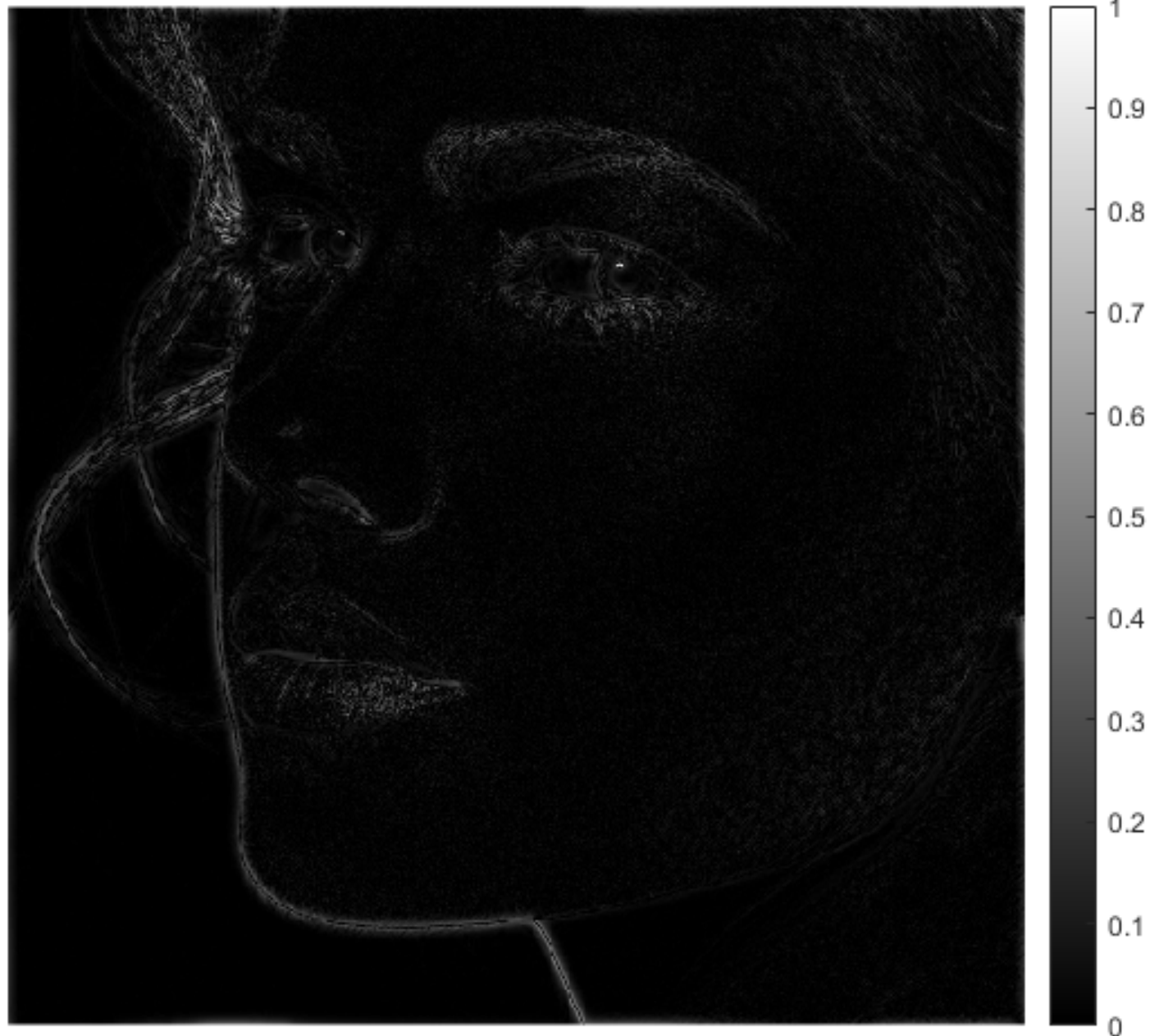


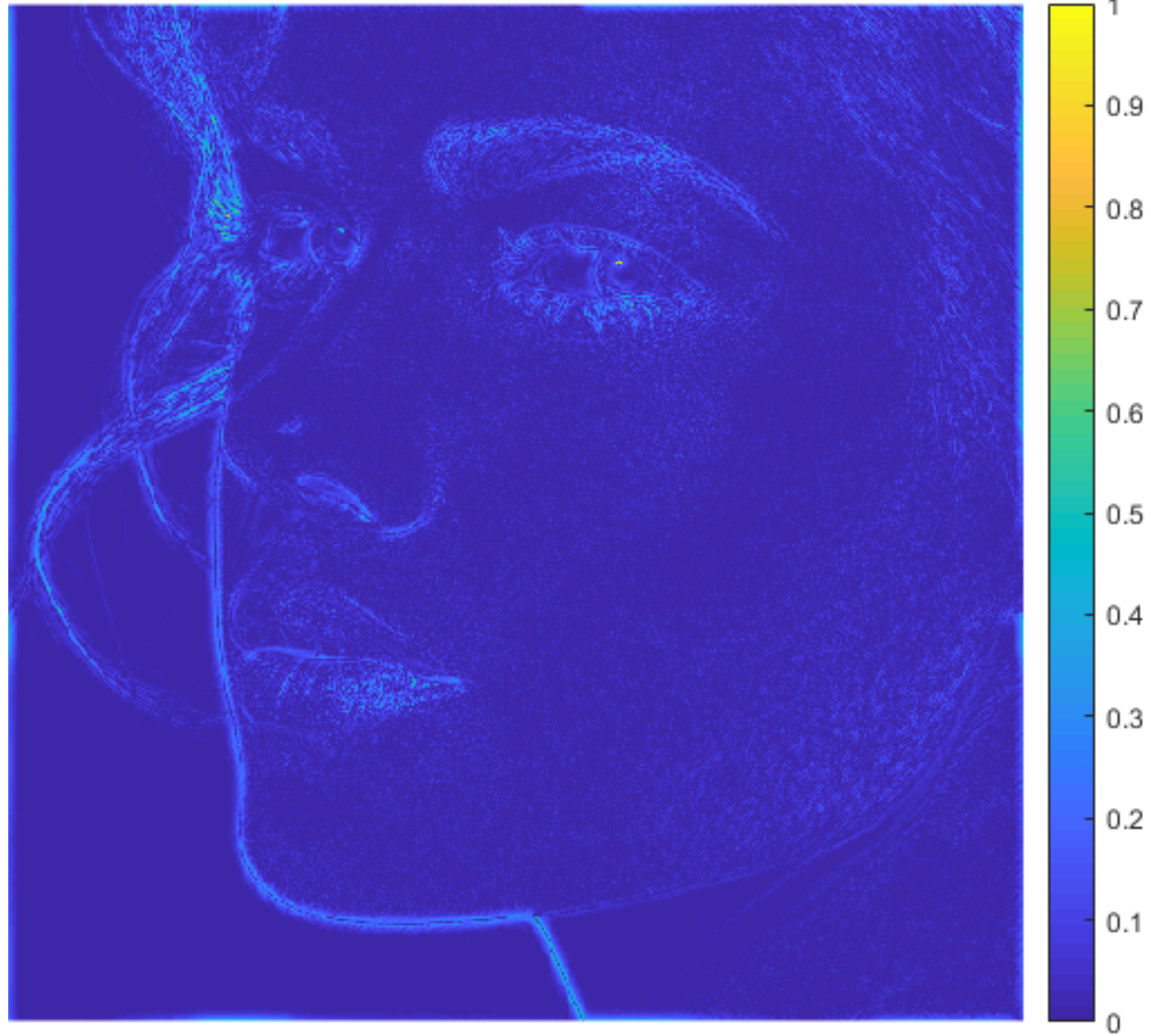










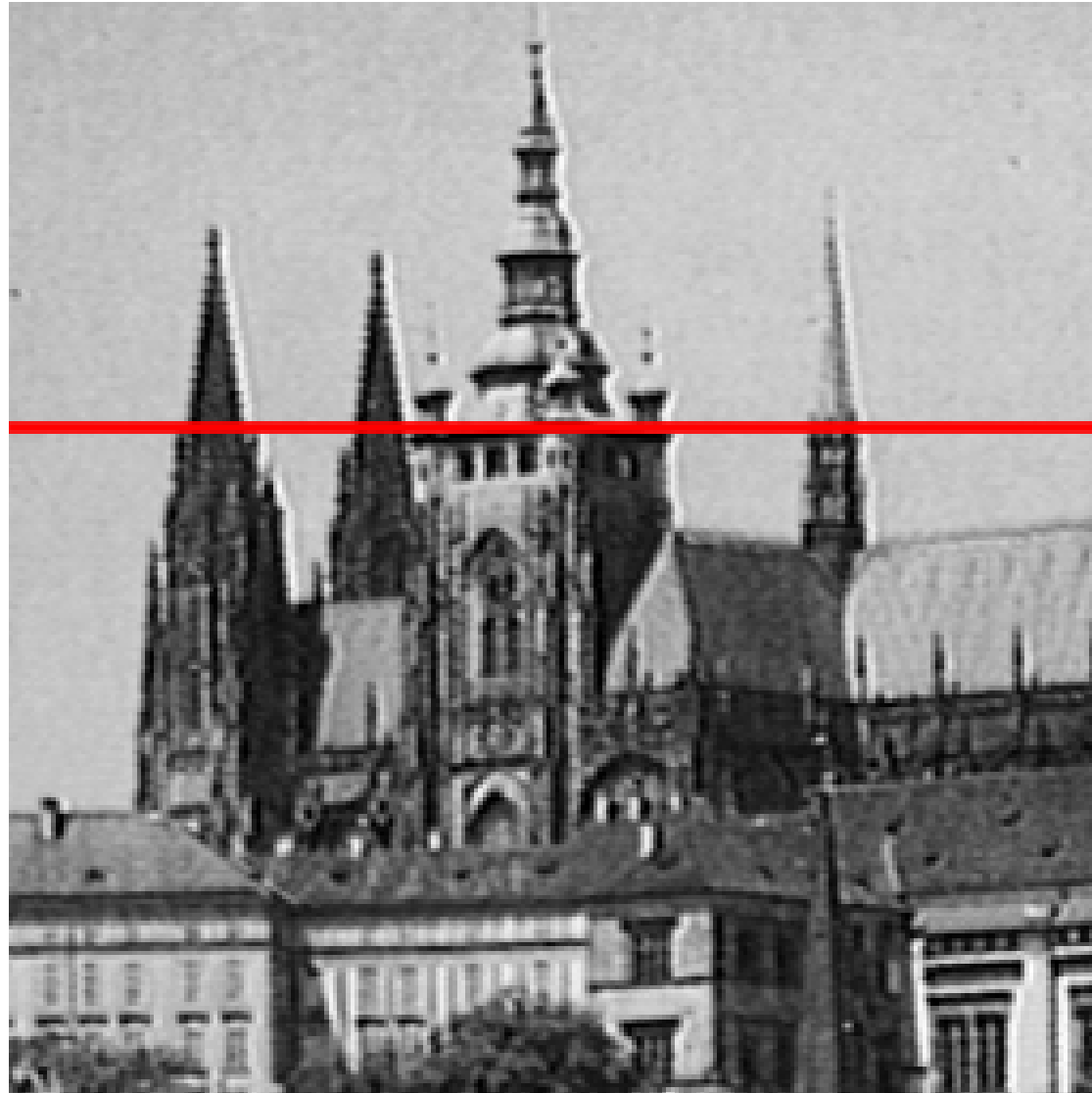


Input image

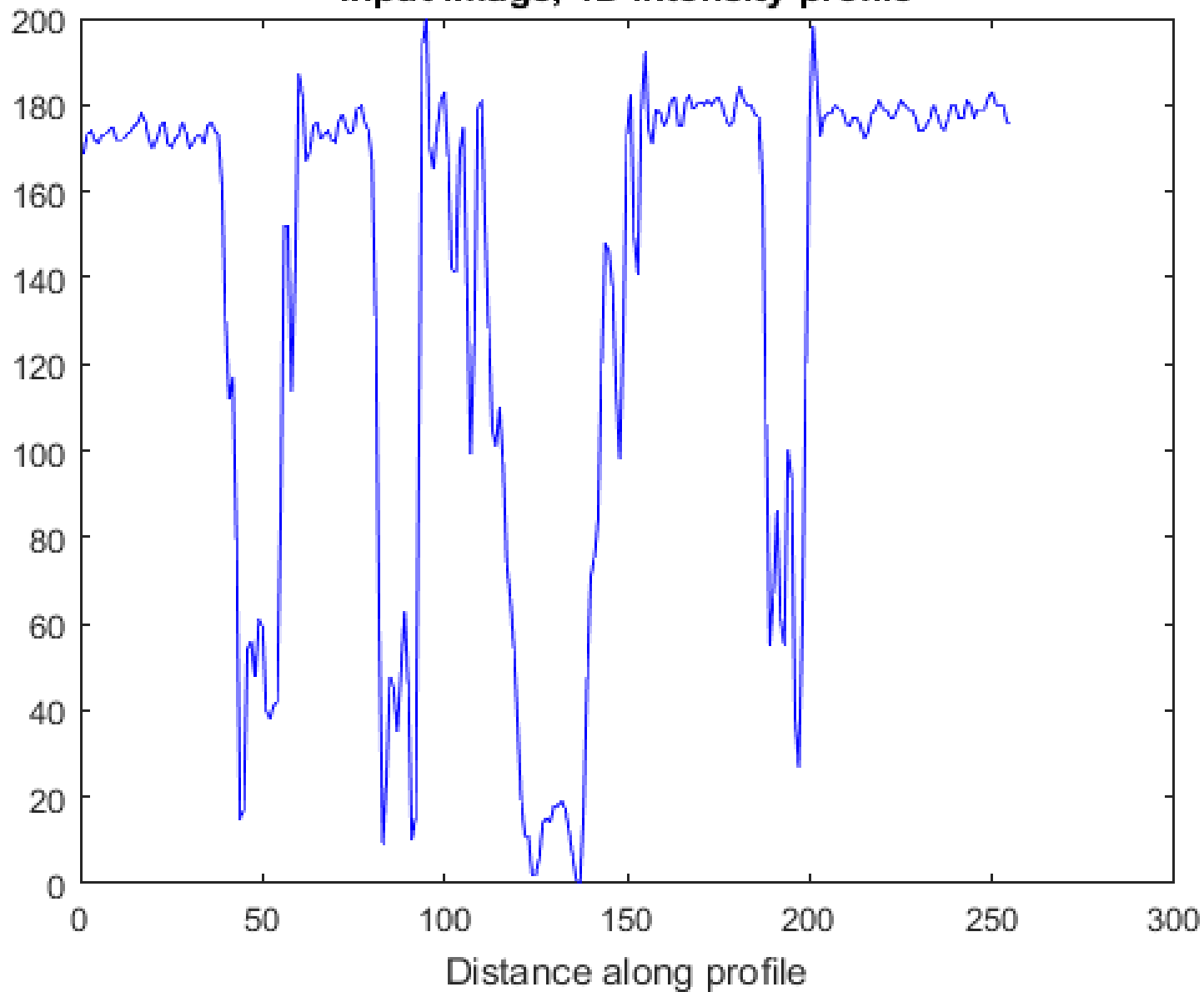




**Input image, line for profile in red**



Input image, 1D intensity profile



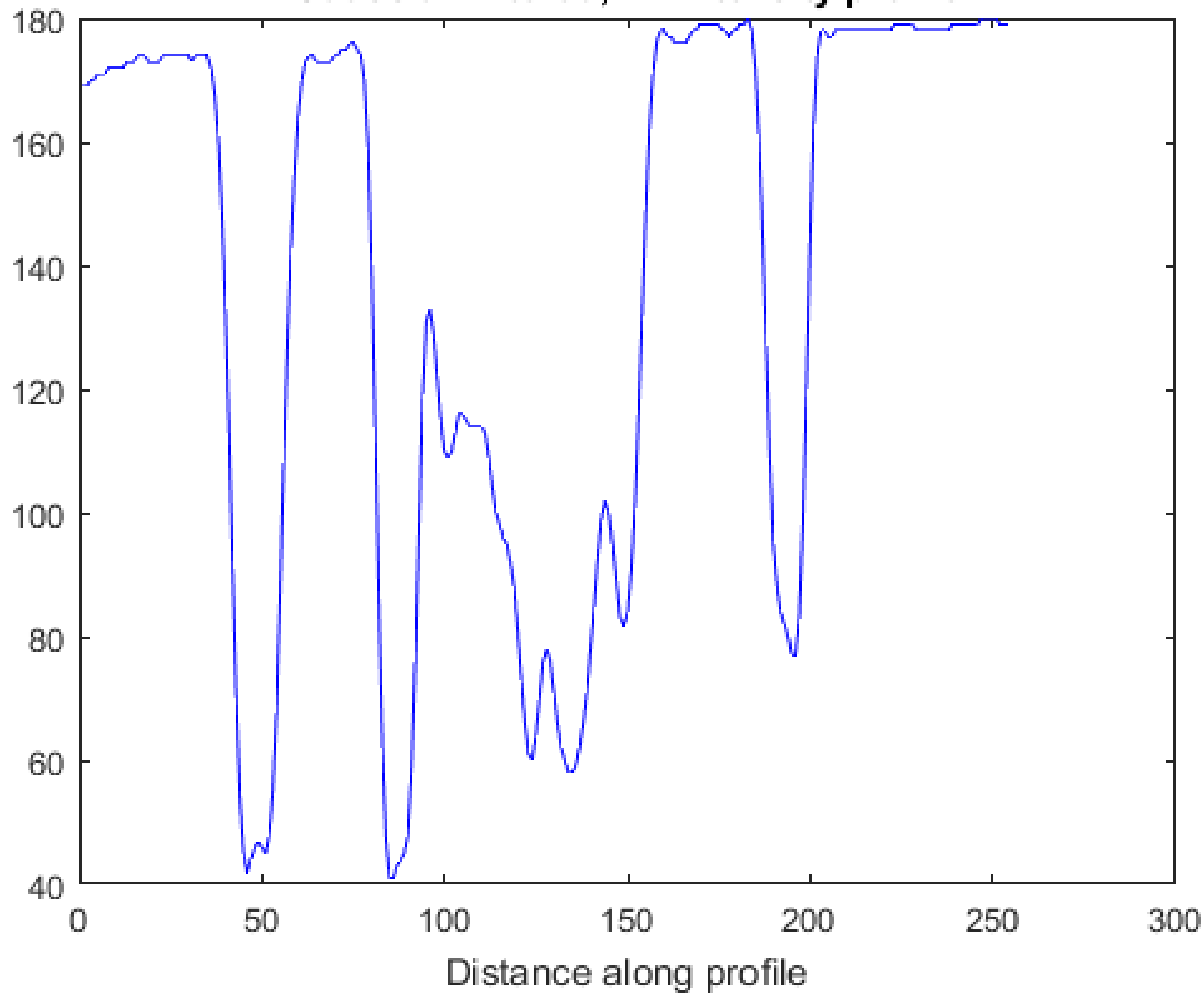
**Gaussian filtered**



**Gaussian filtered, line for profile in red**



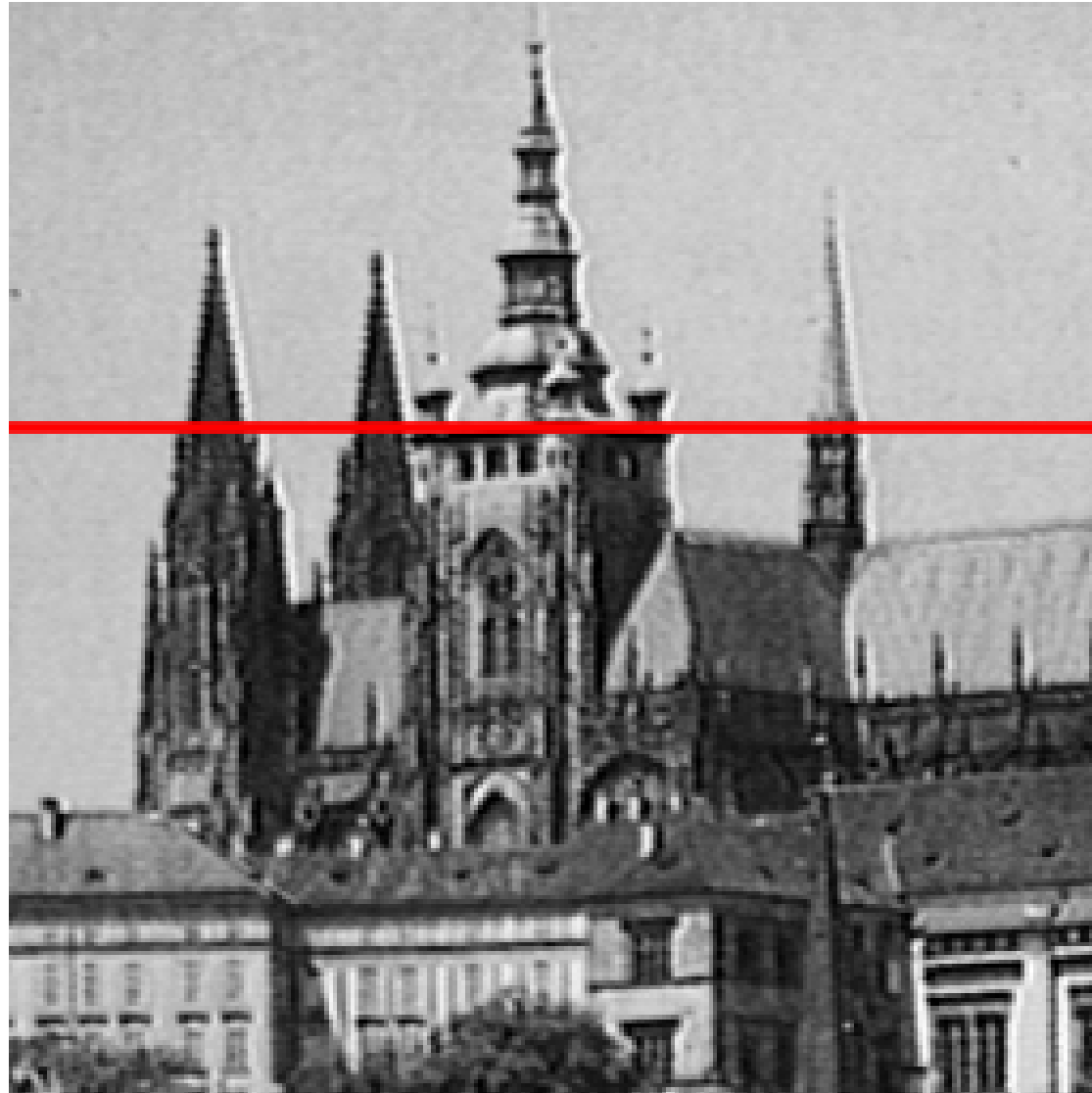
**Gaussian filtered, 1D intensity profile**



Input image

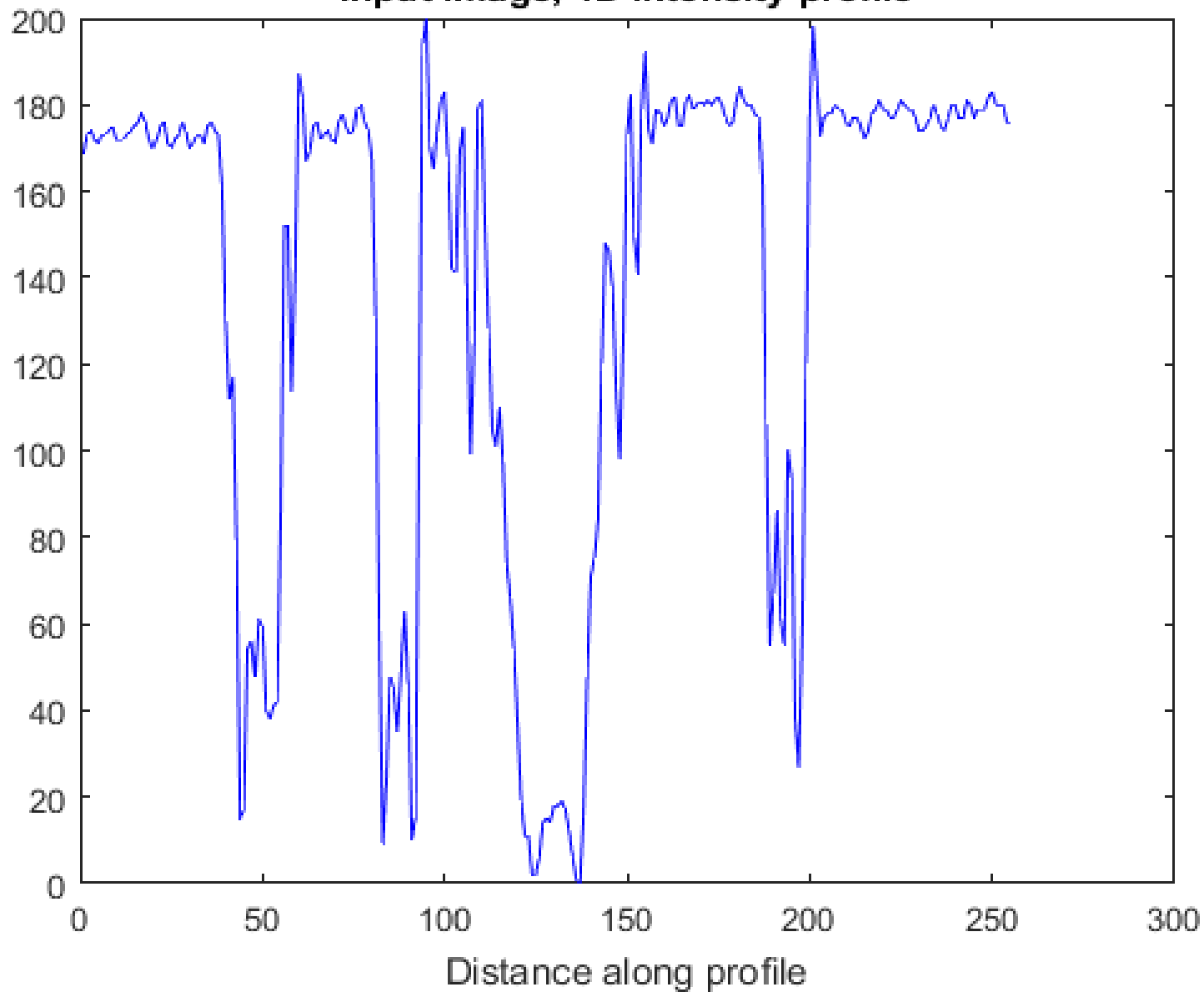


**Input image, line for profile in red**

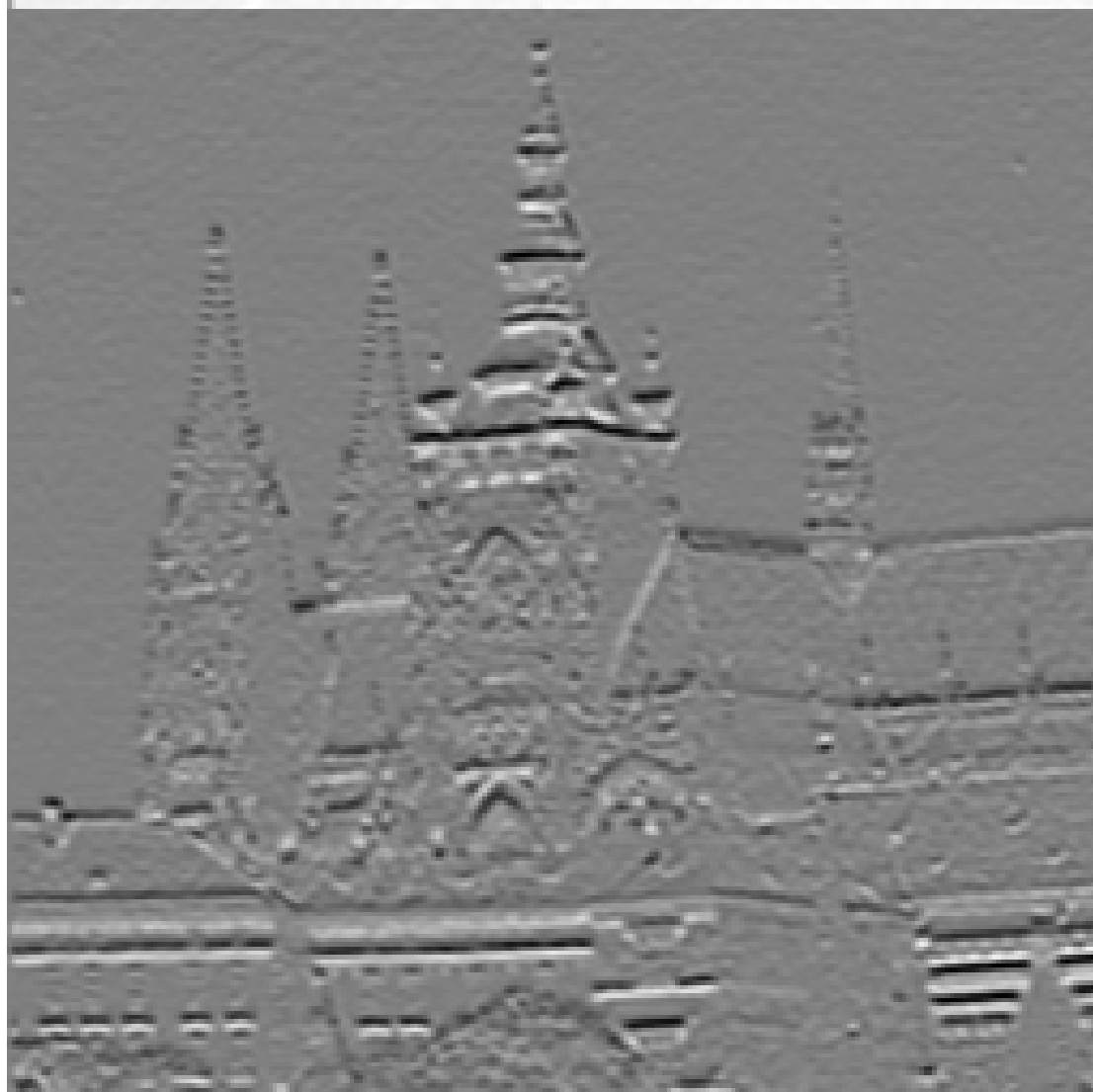




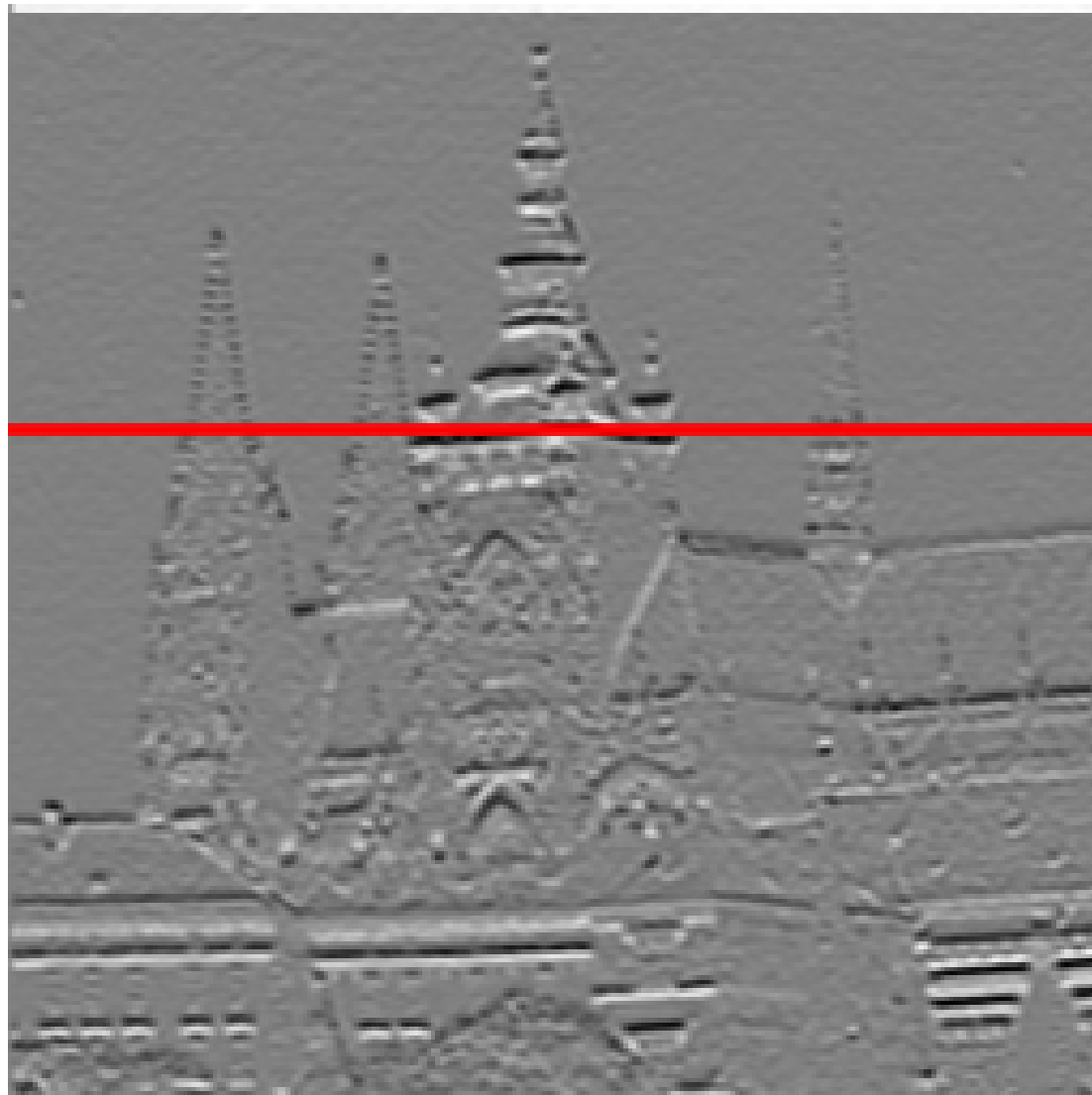
Input image, 1D intensity profile



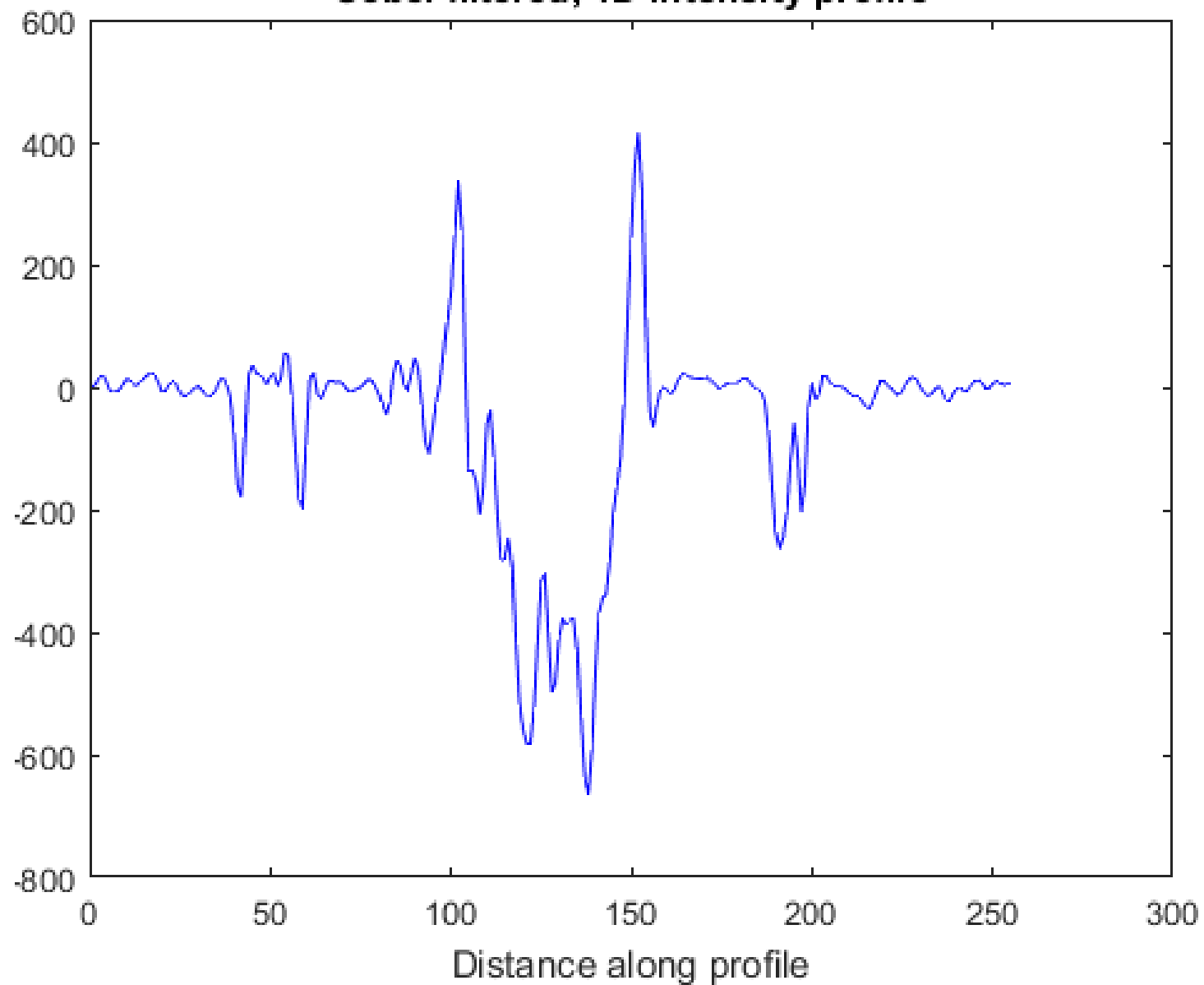
**Sobel filtered**



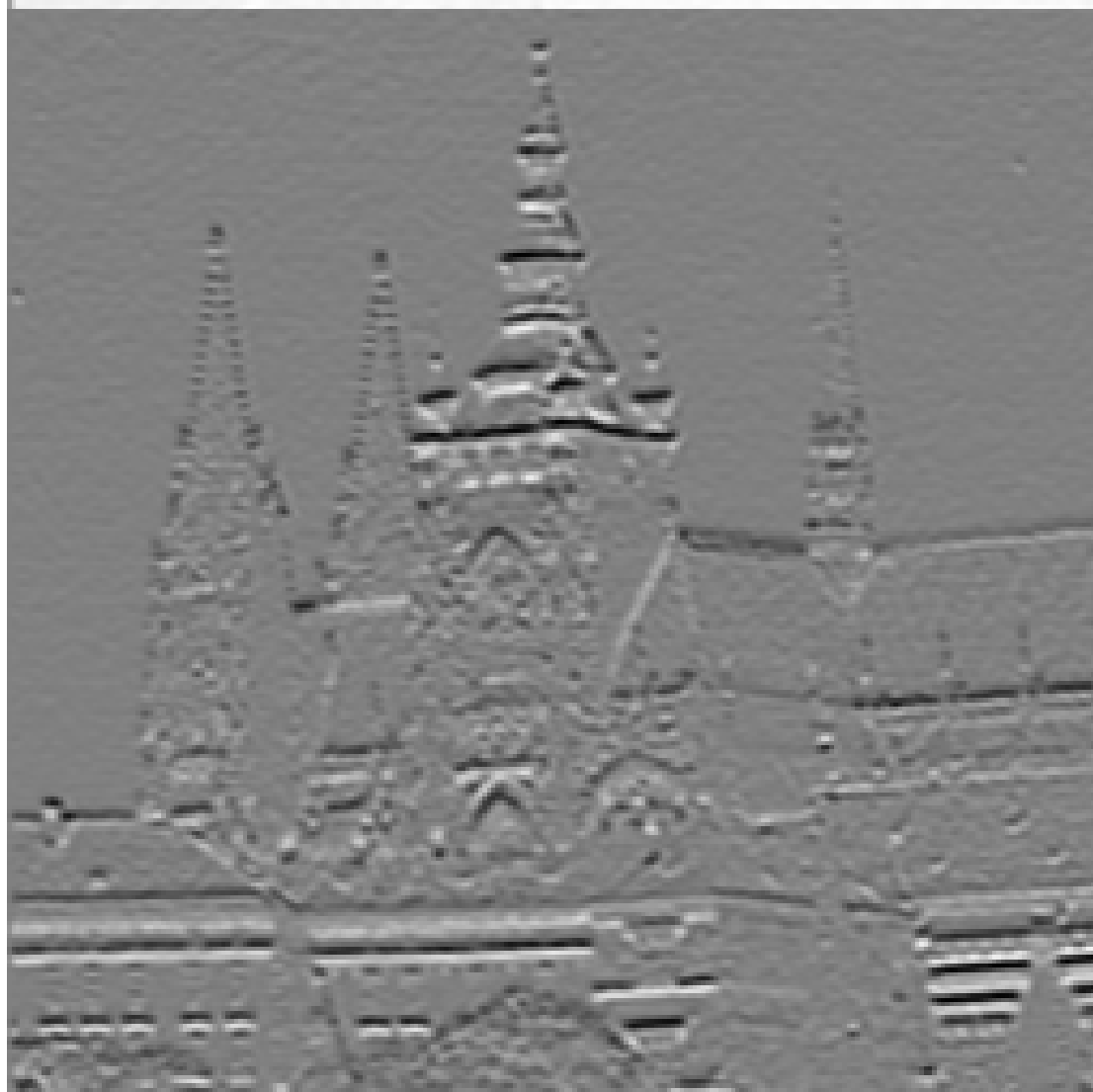
**Sobel filtered, line for profile in red**



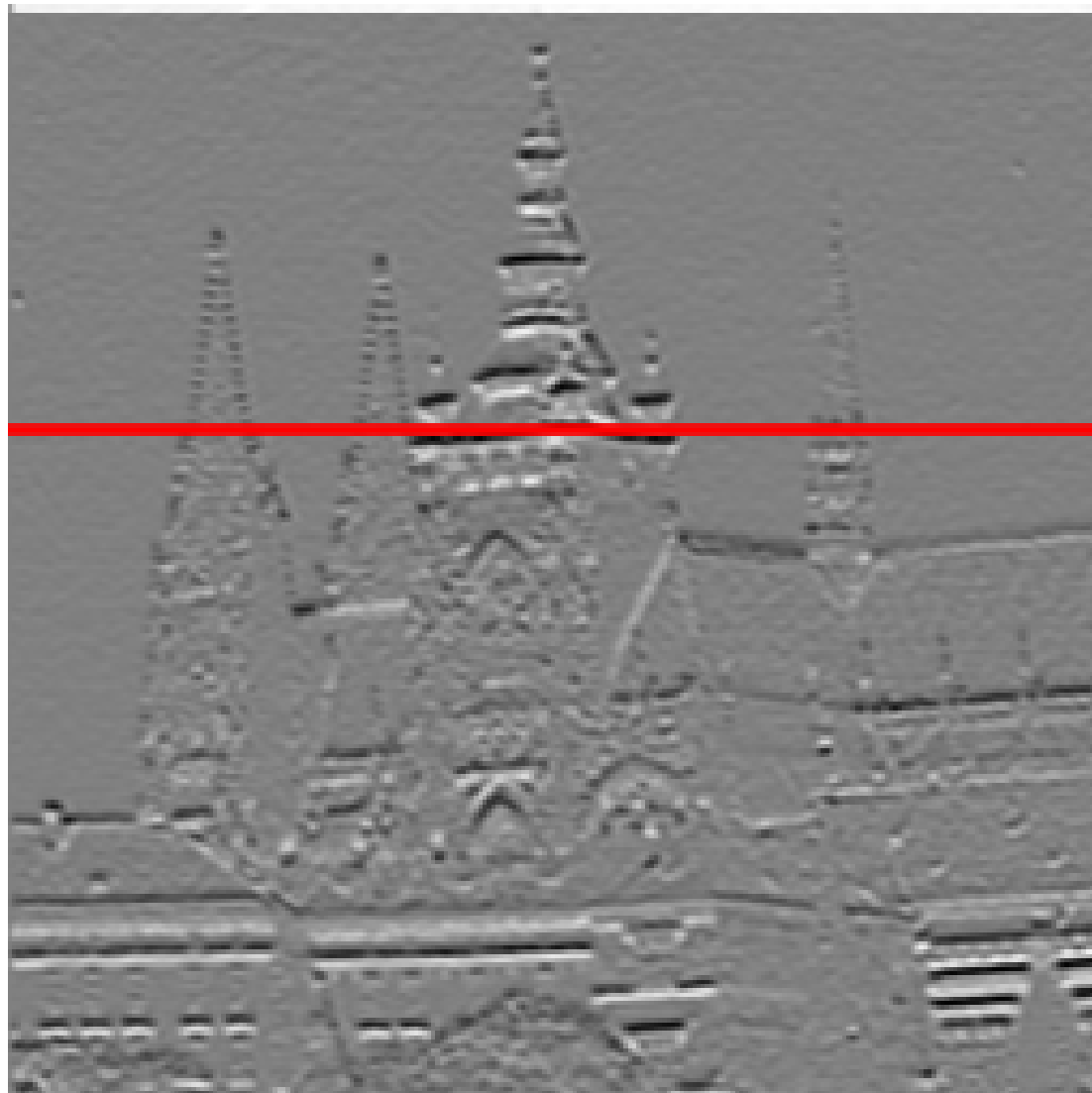
**Sobel filtered, 1D intensity profile**



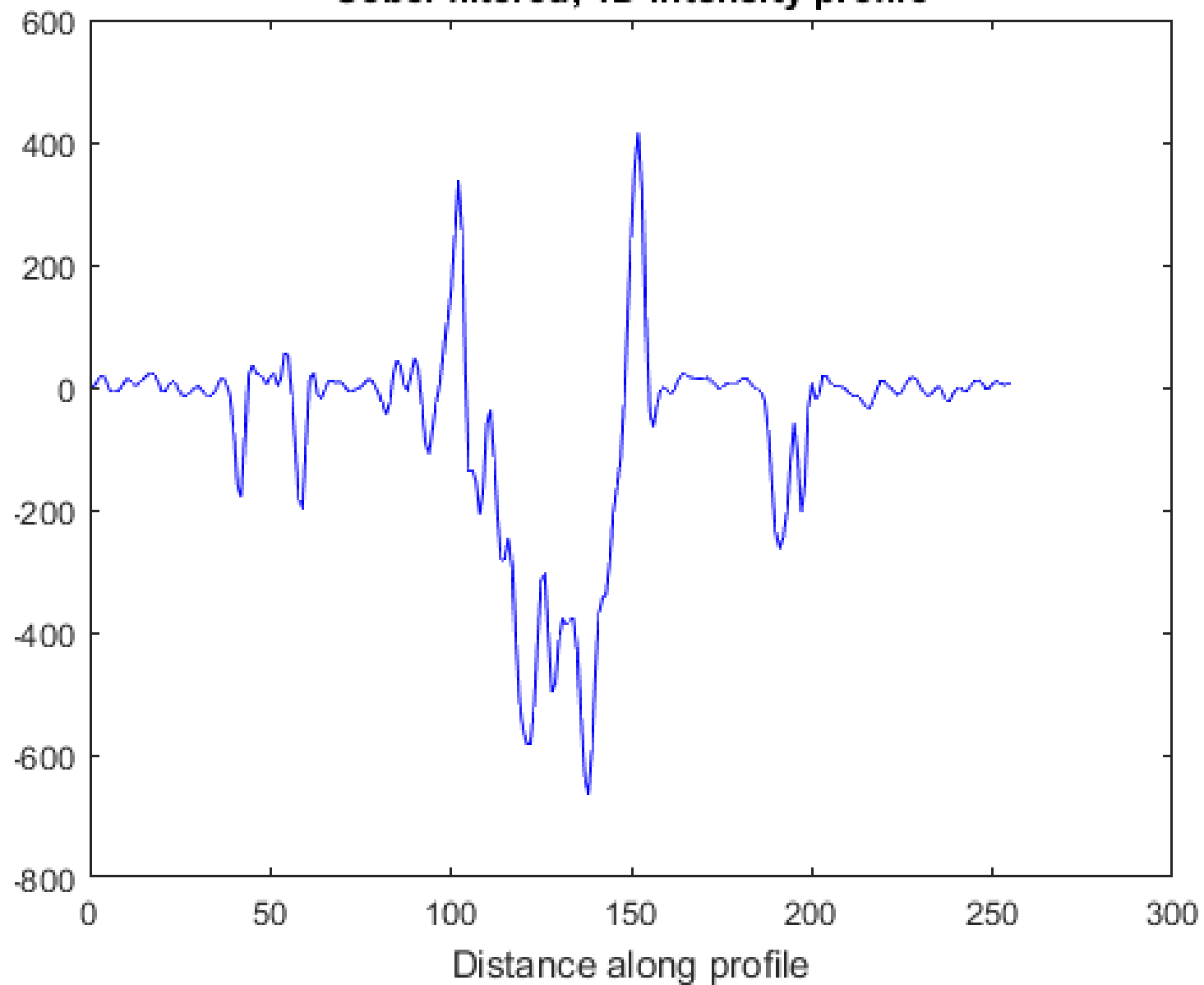
**Sobel filtered**



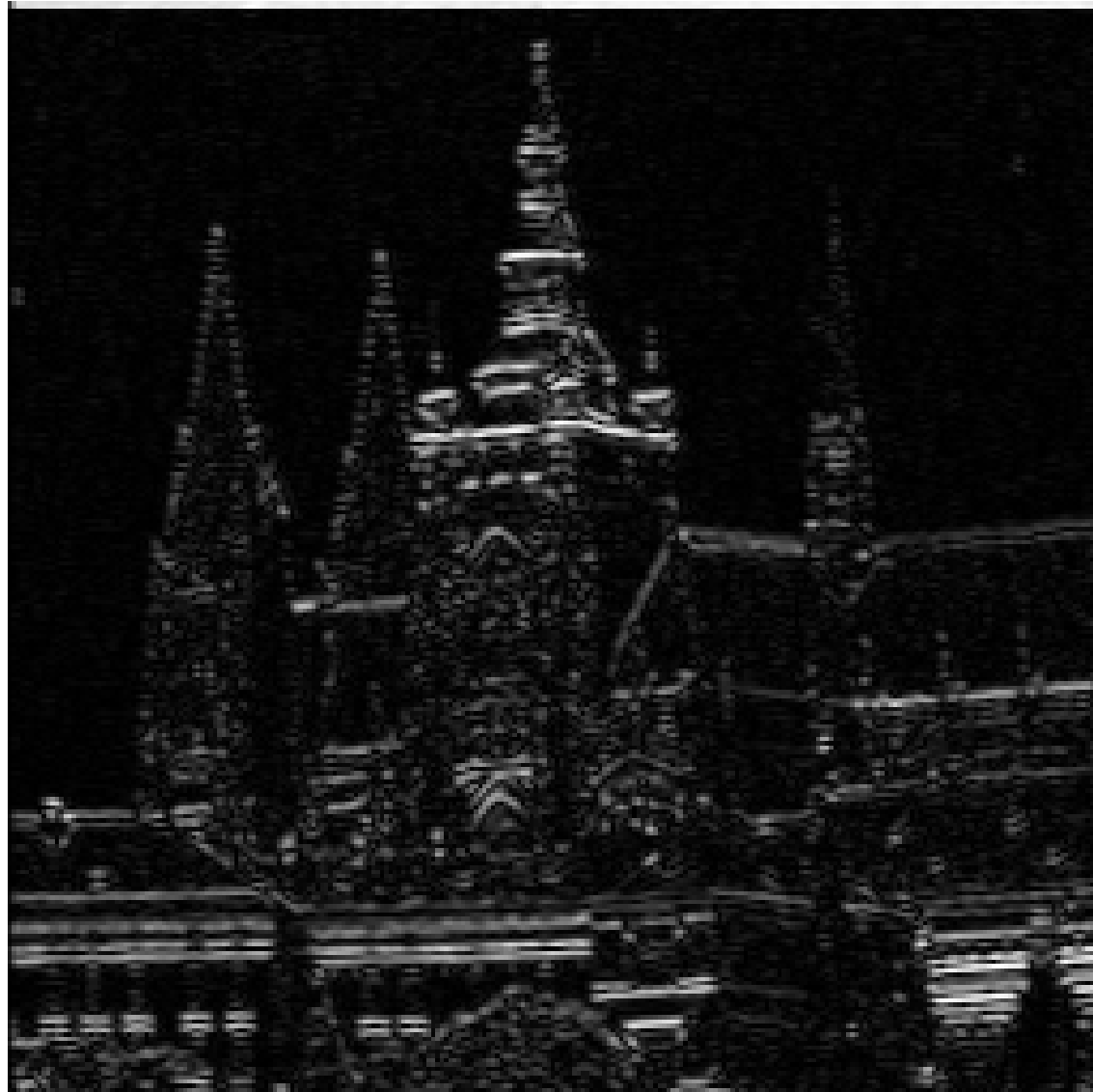
**Sobel filtered, line for profile in red**



**Sobel filtered, 1D intensity profile**

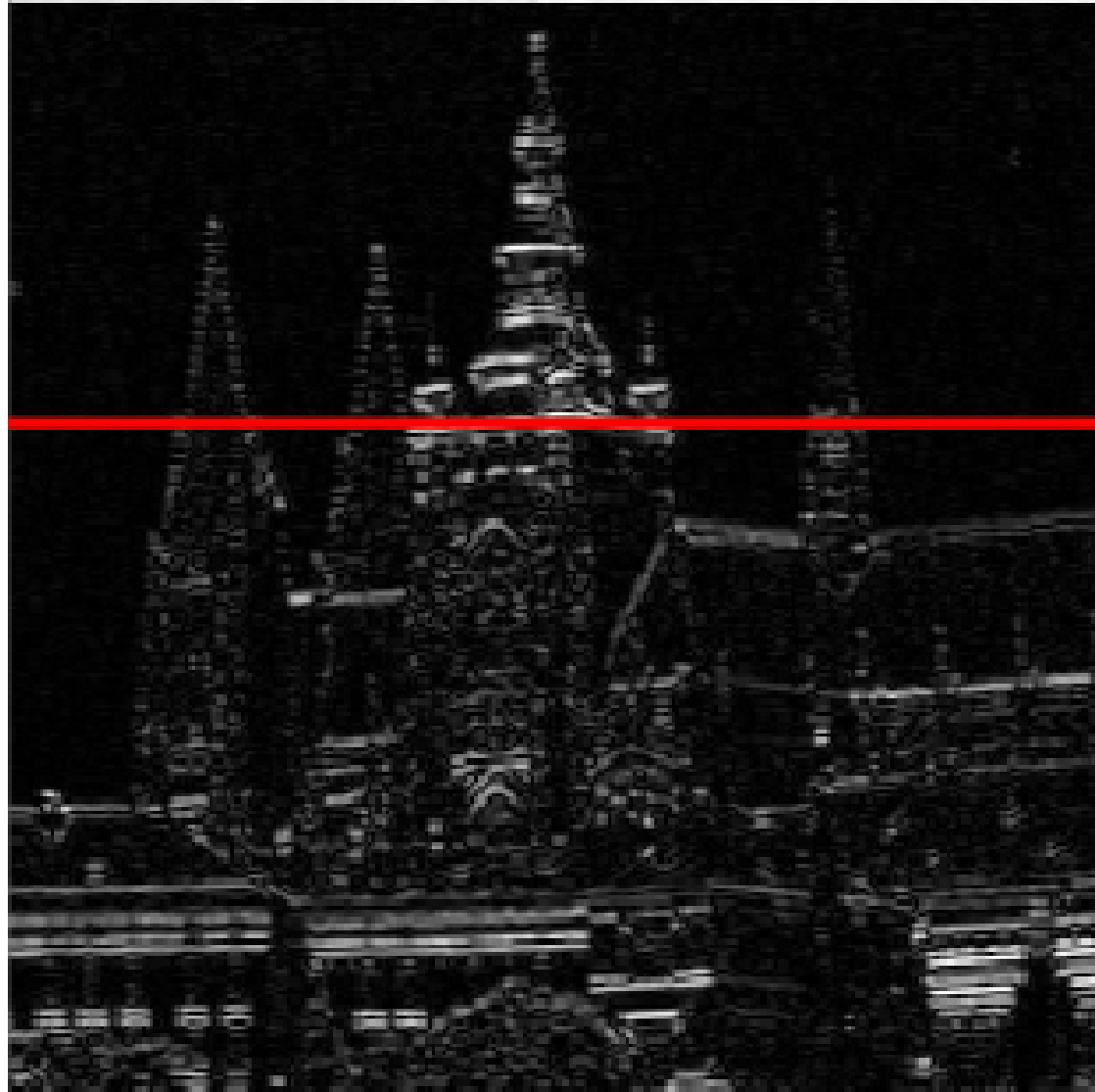


**Abs(Sobel filtered)**

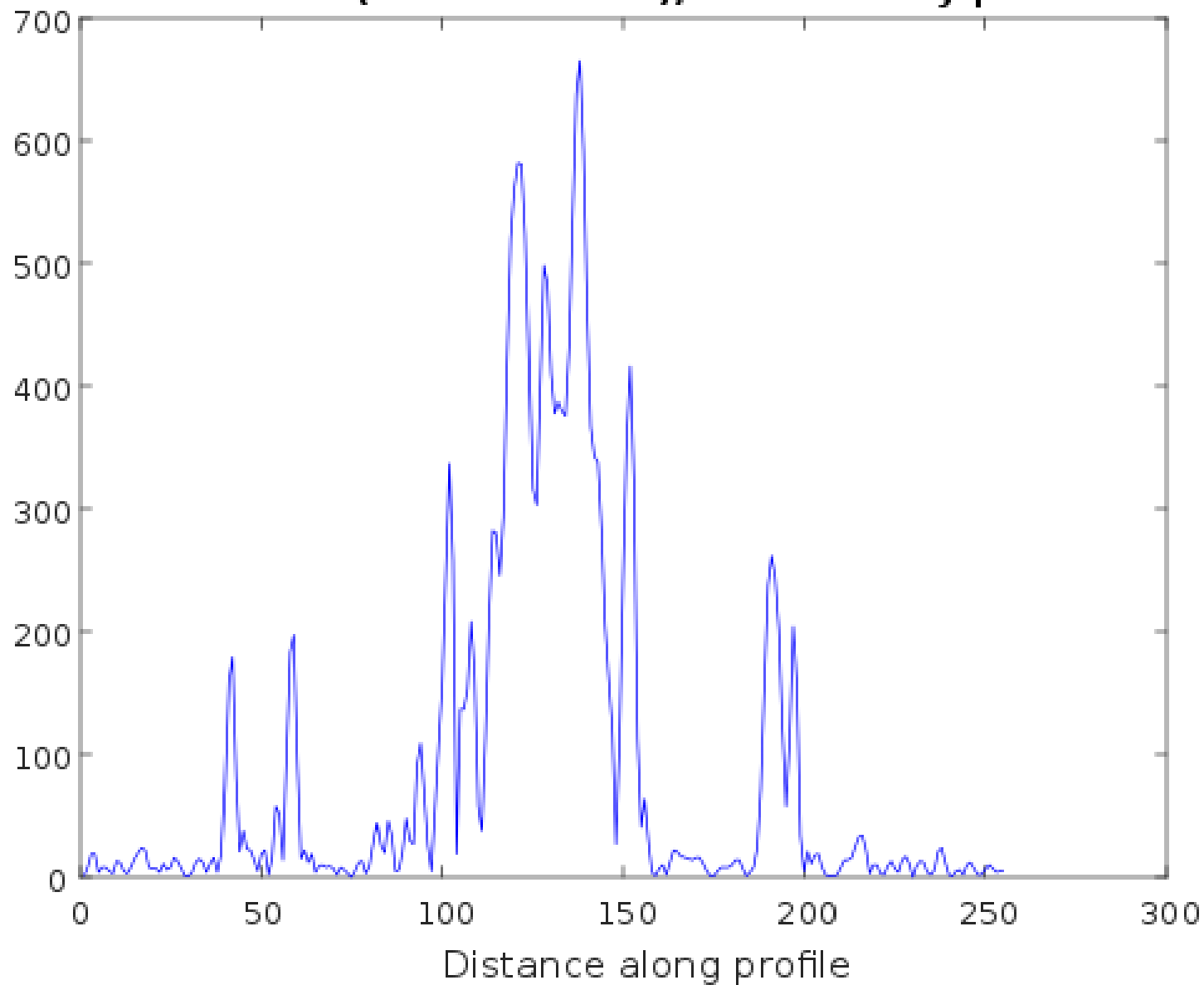




## Abs(Sobel filtered)

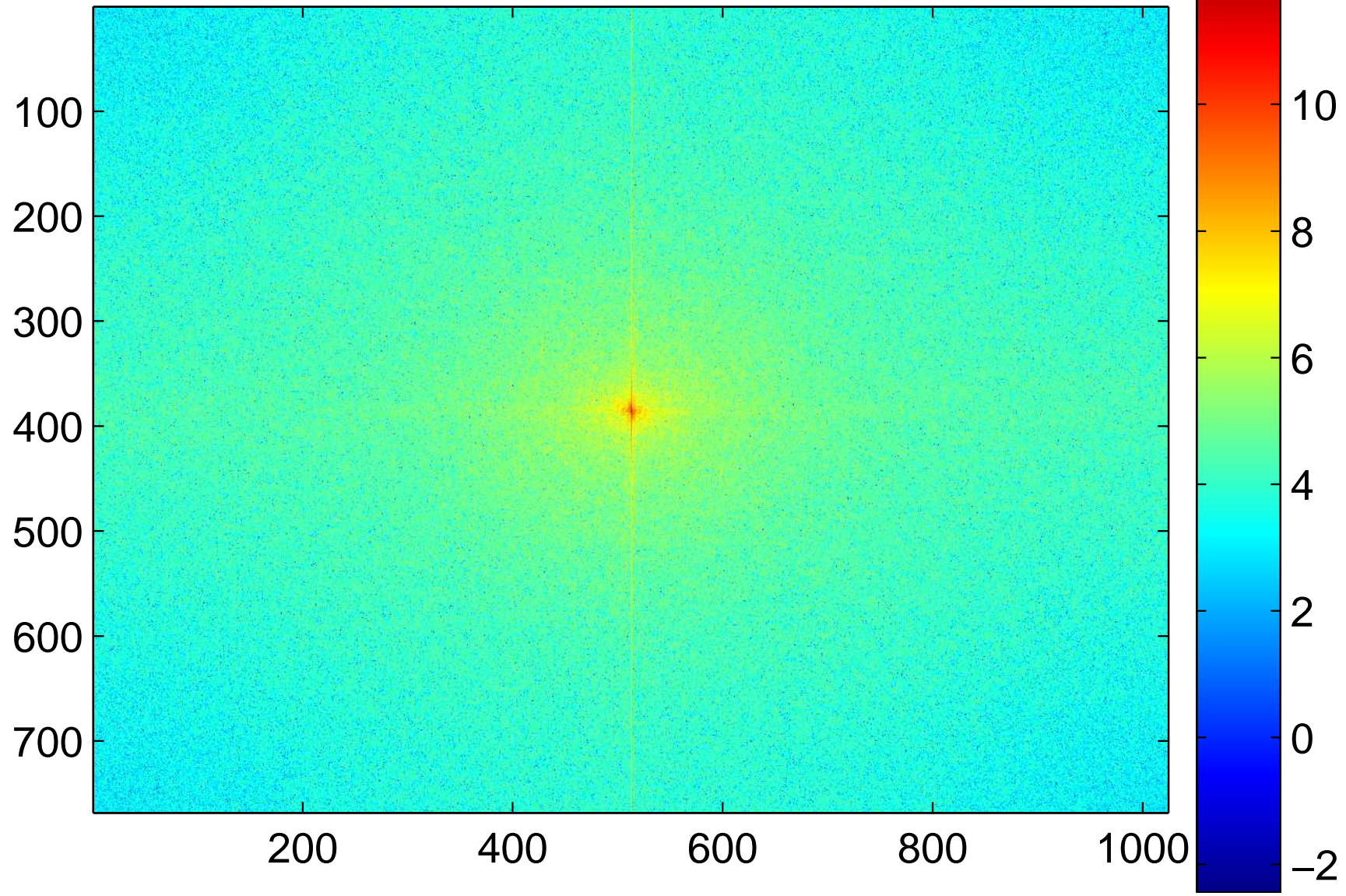


**Abs(Sobel filtered), 1D intensity profile**



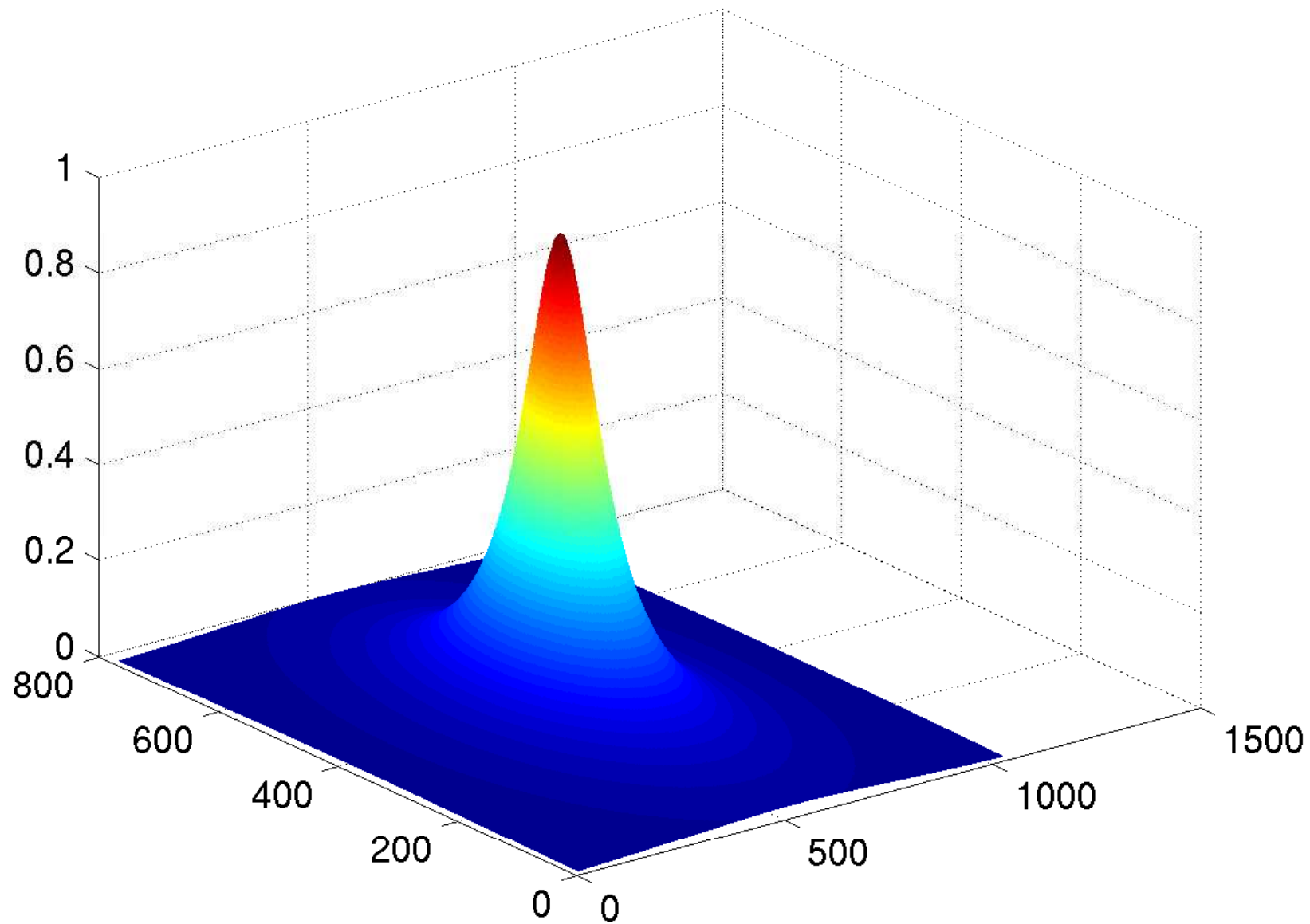


Shifted  $\log(\text{abs}(\text{FFT}))$  of the original image

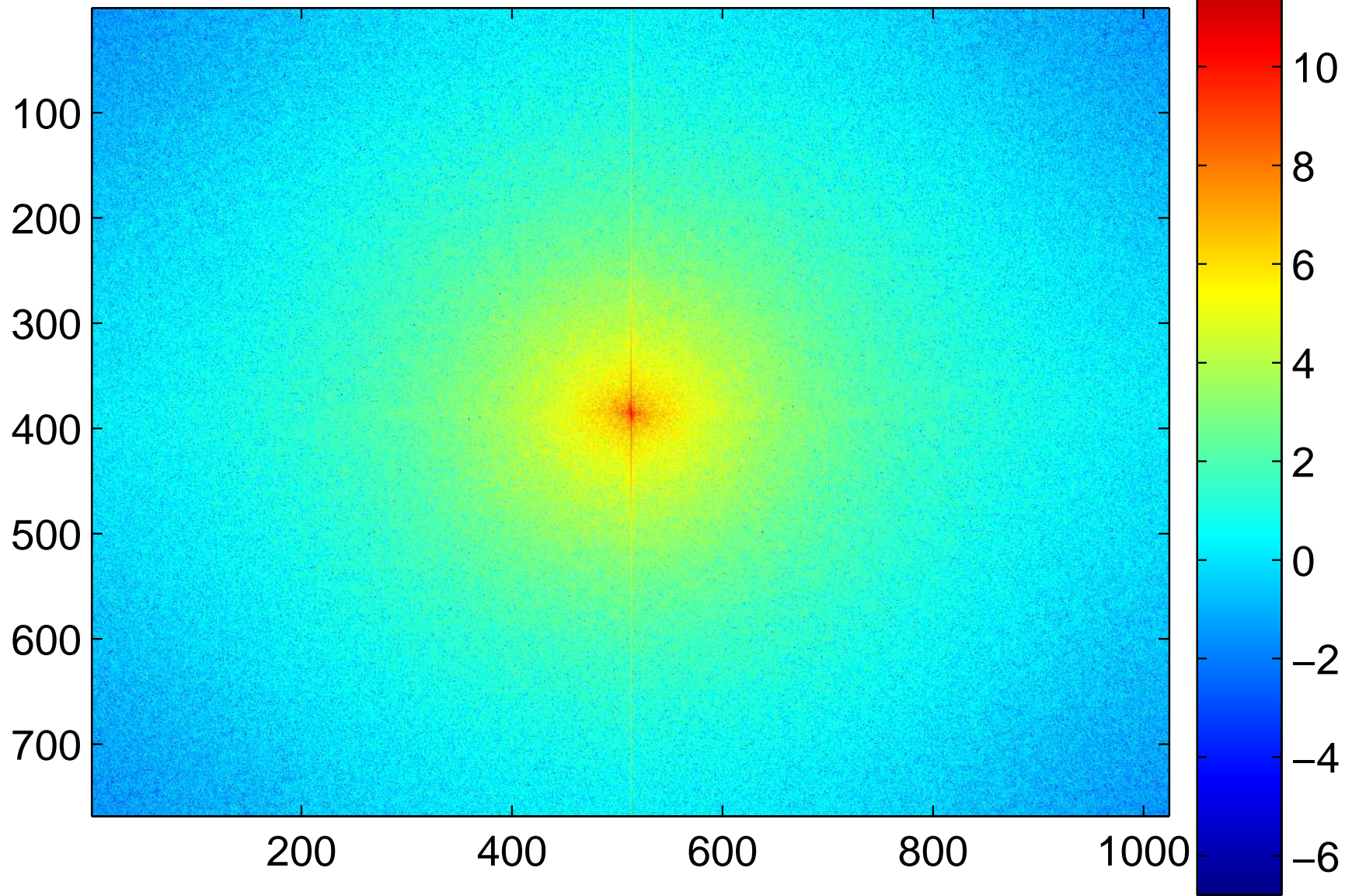




lp Butt filter n=1, cutoff=66



Shifted  $\log(\text{abs}(\text{FFT}))$  of the filtered image



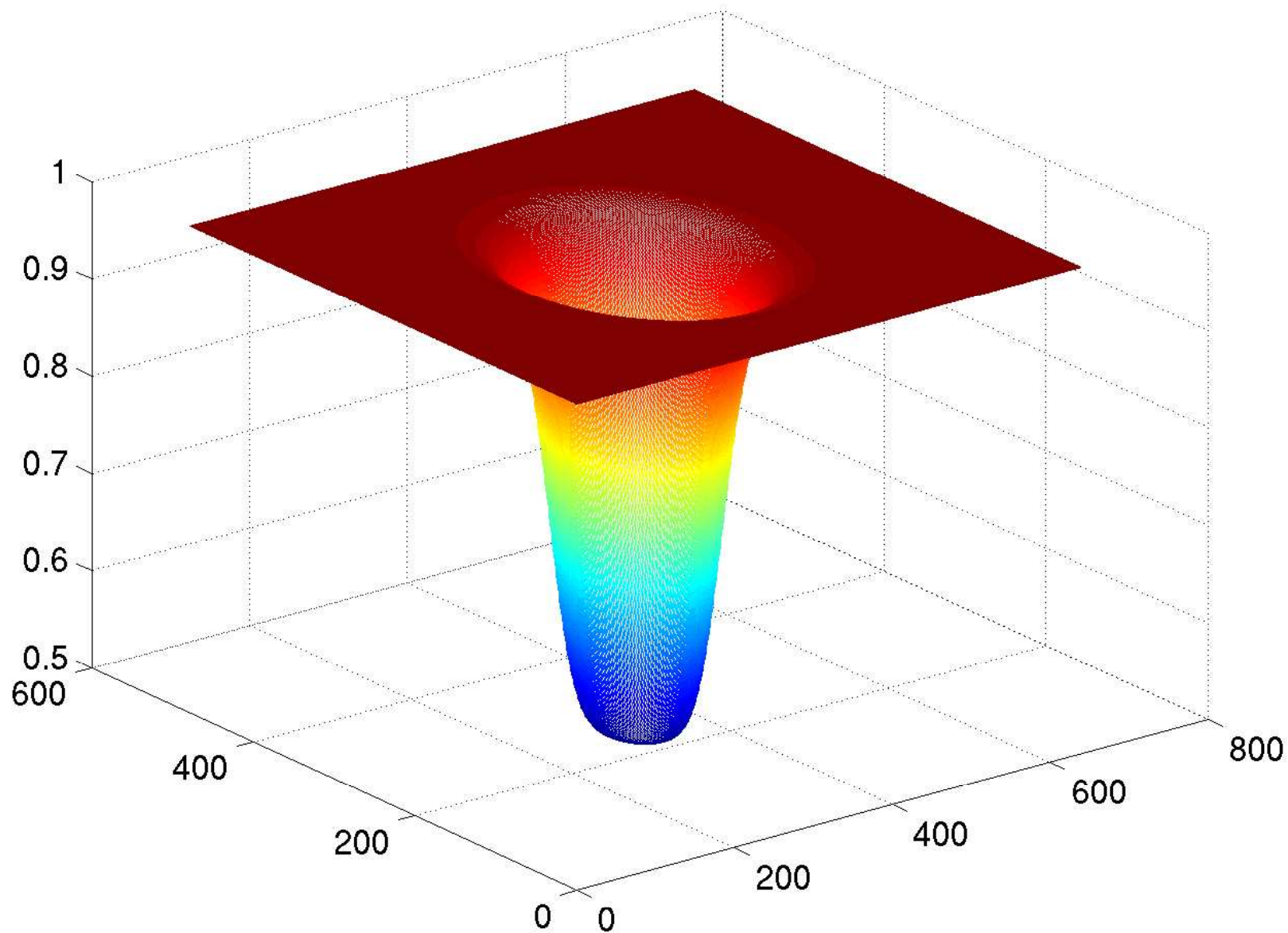


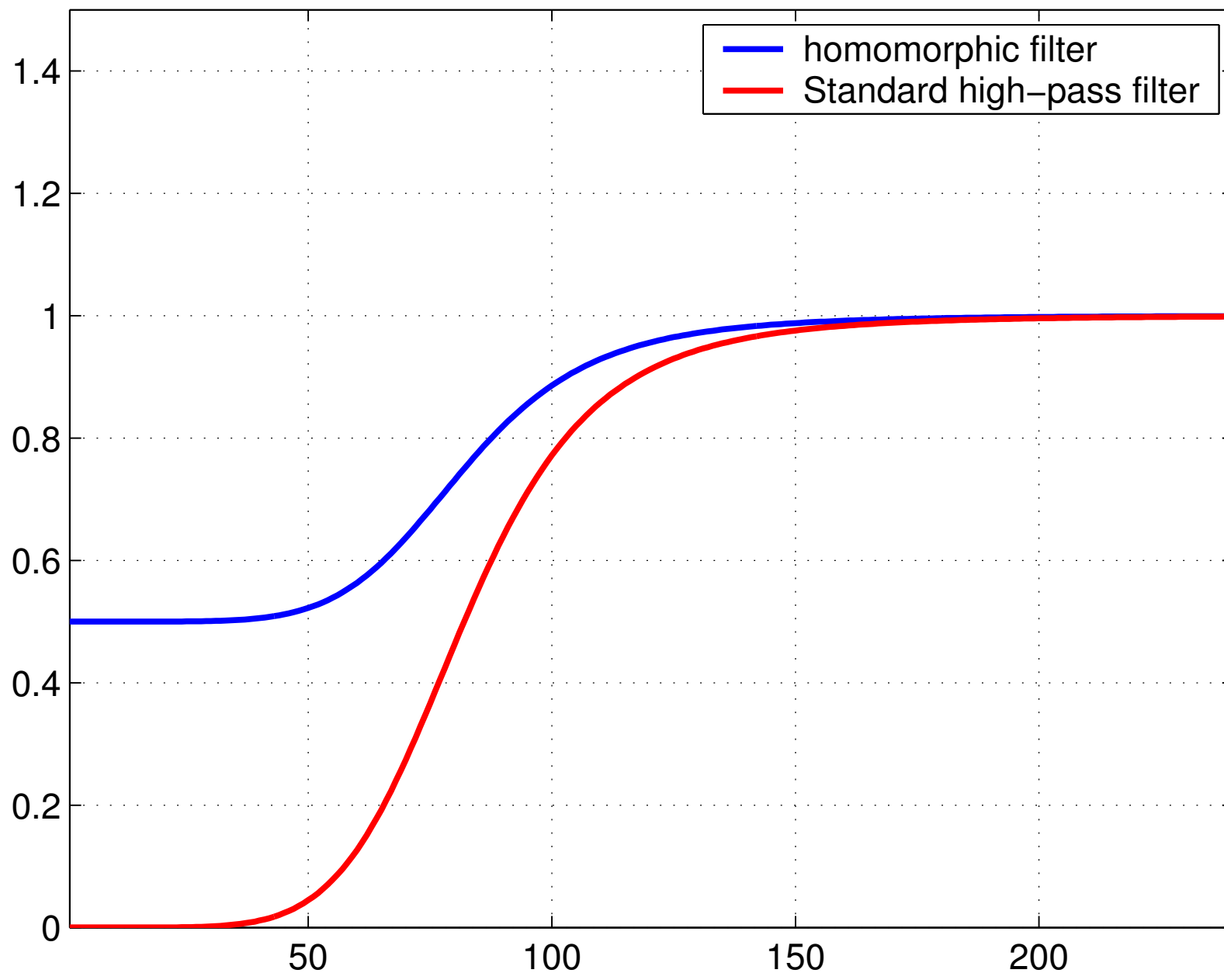






Homomorphic filter made by adaptation of Buttherworth highpass

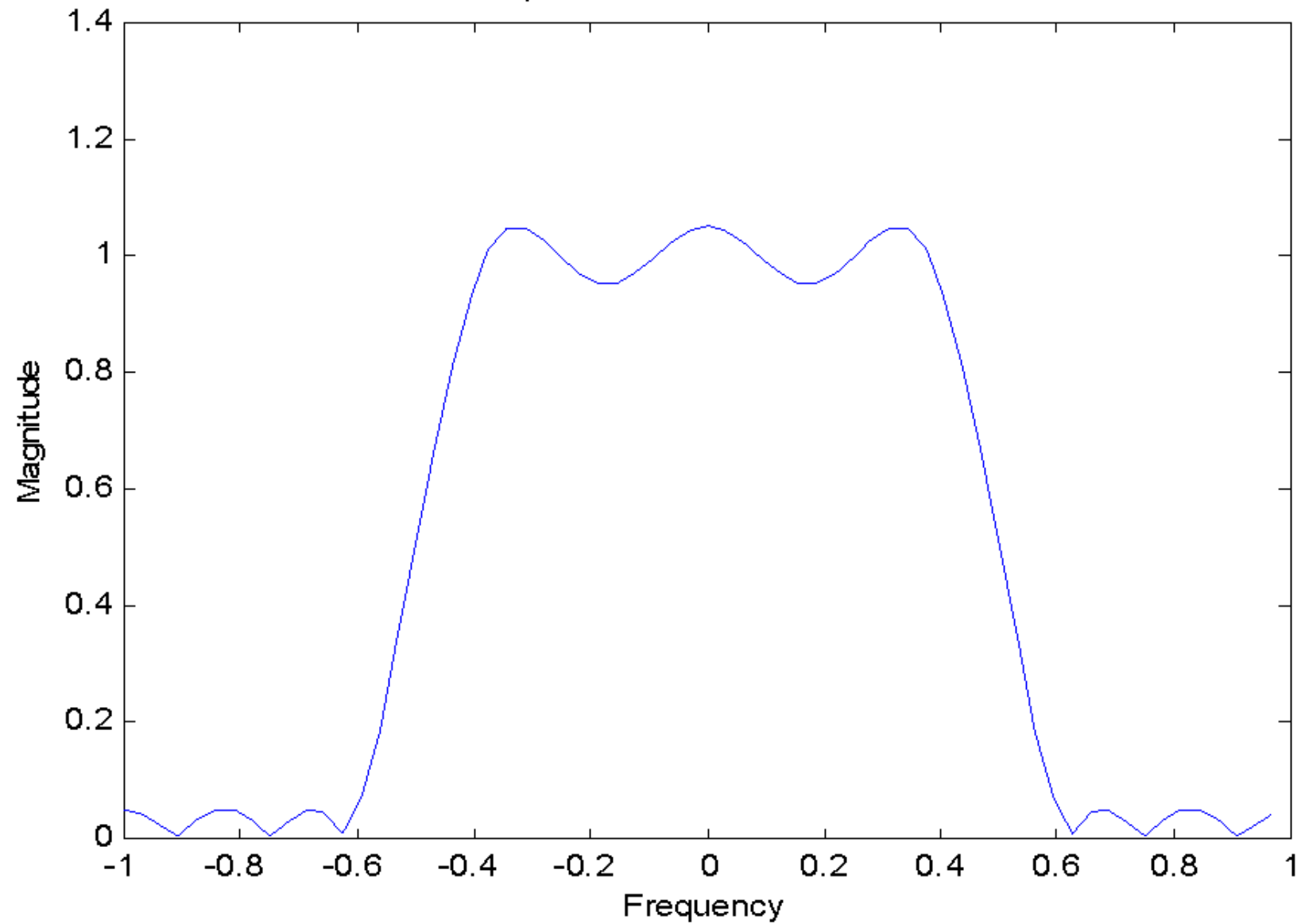




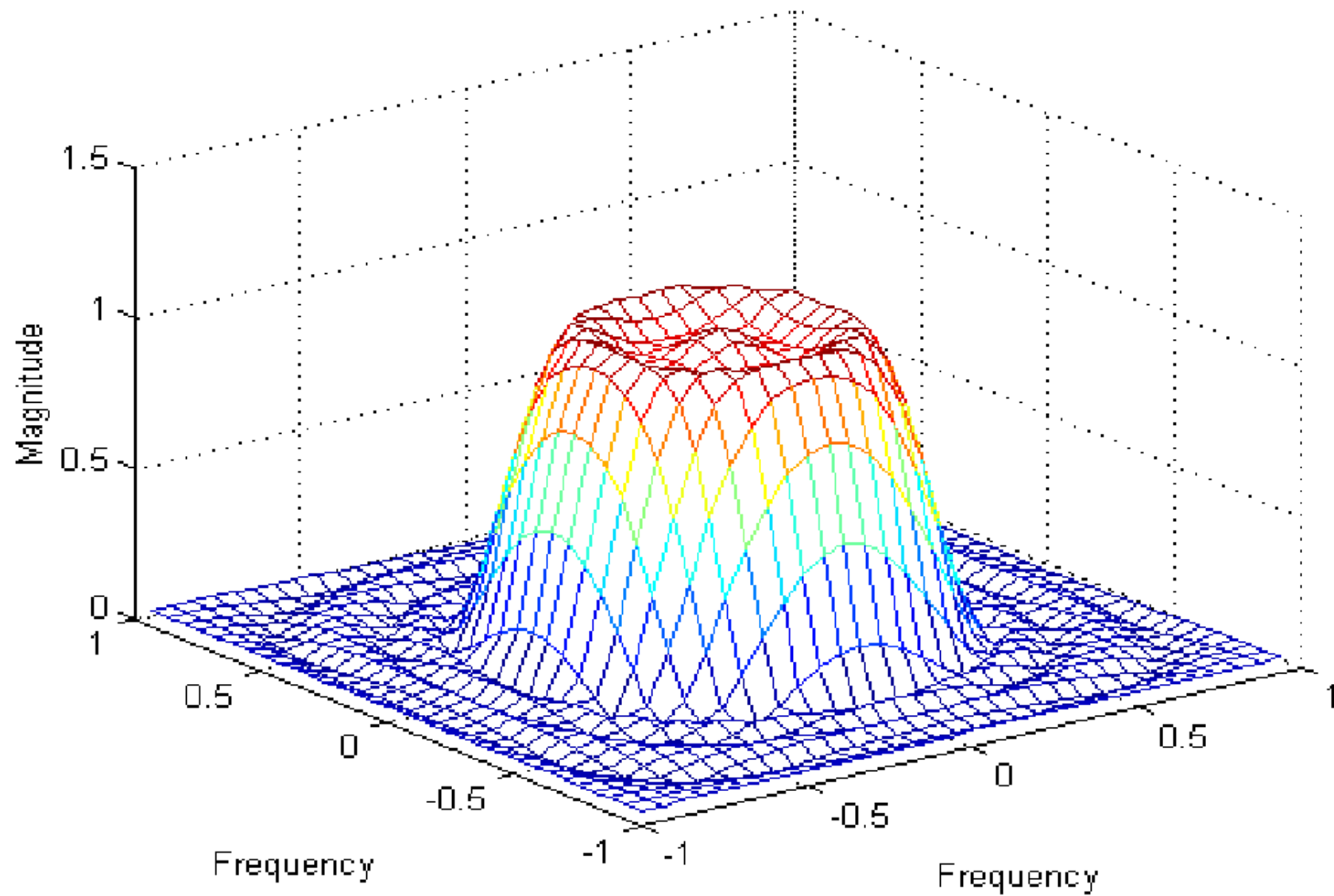




1D optimal Parks-McClellan filter

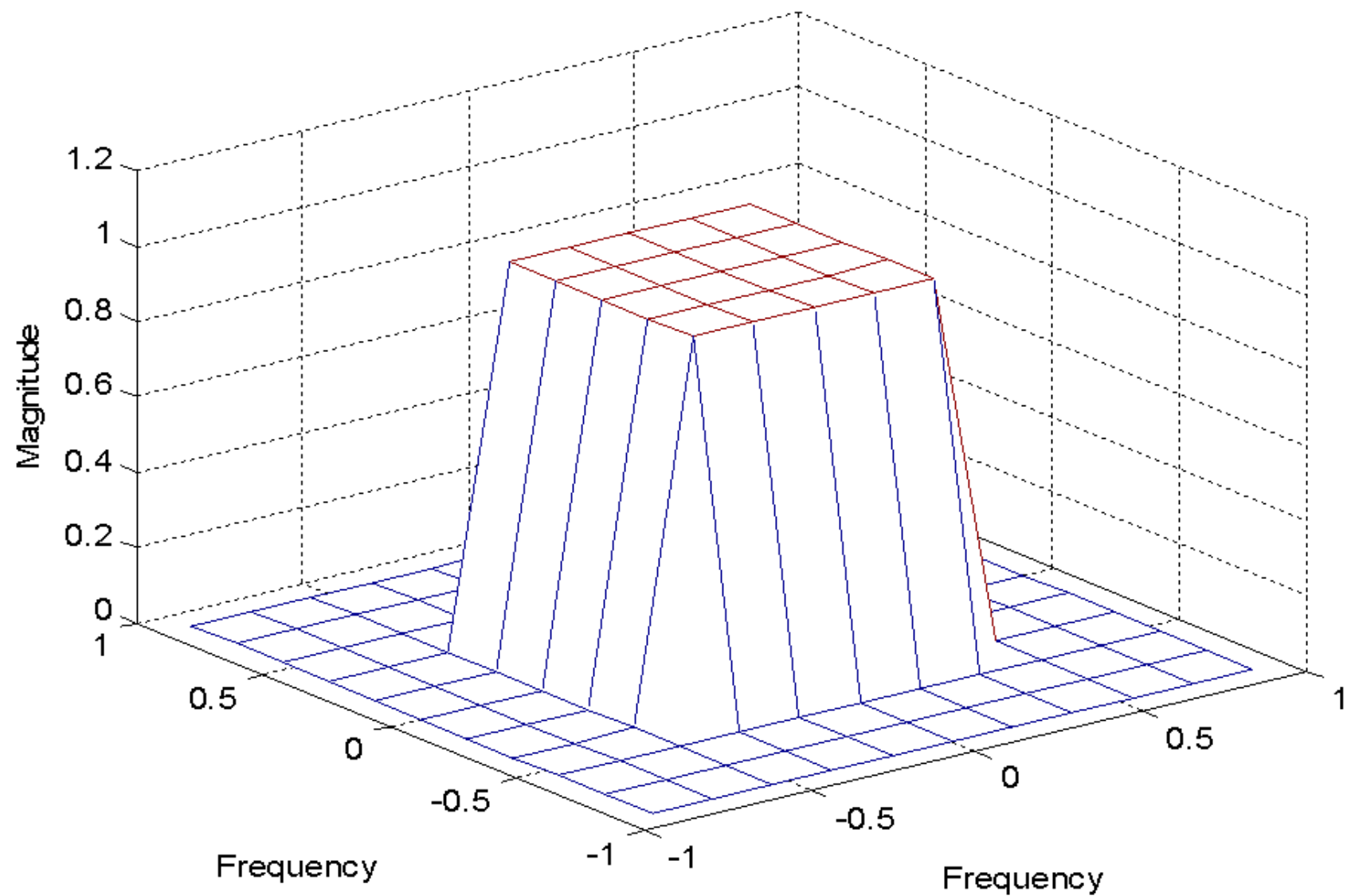


1D optimal Parks-McClellan filter

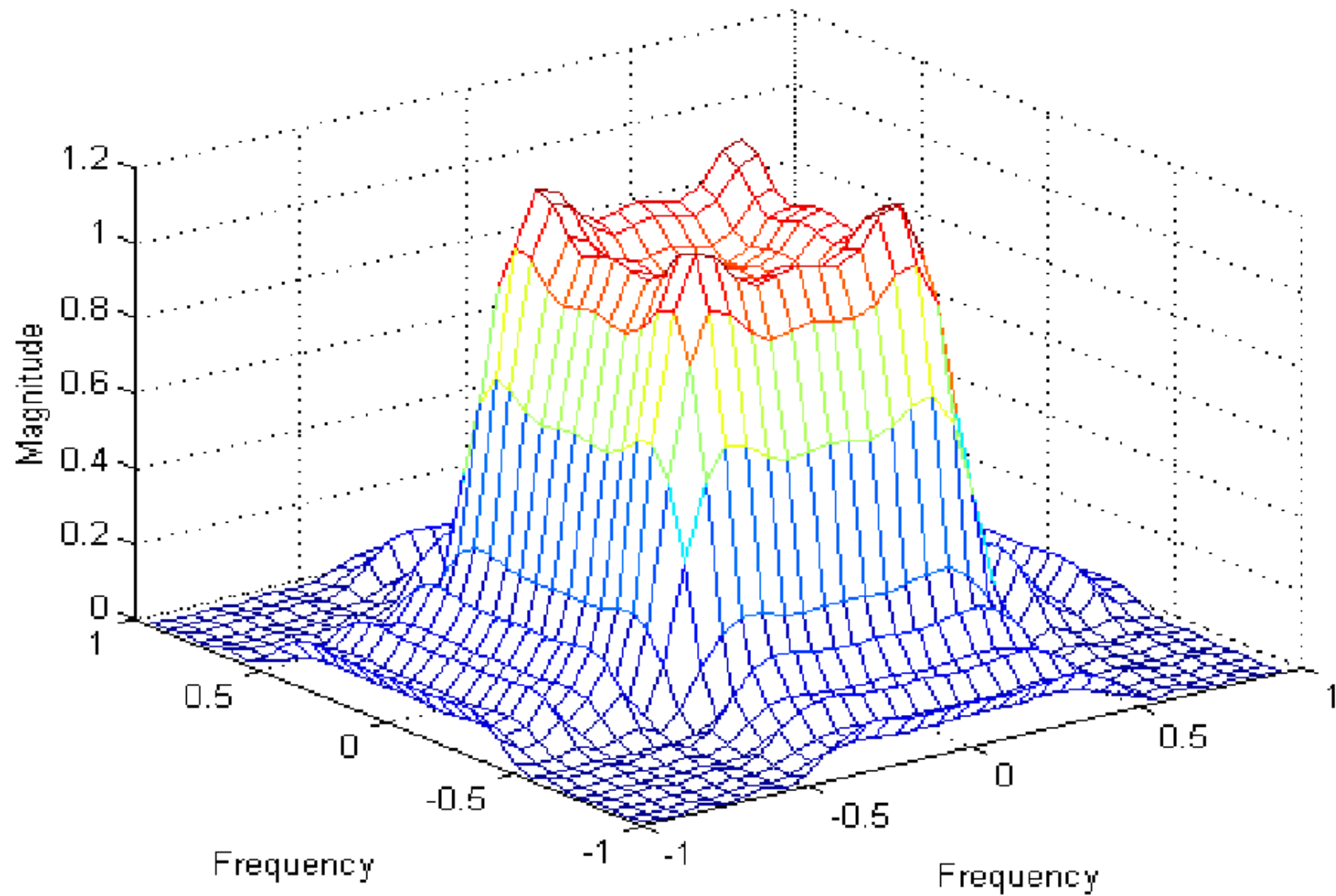




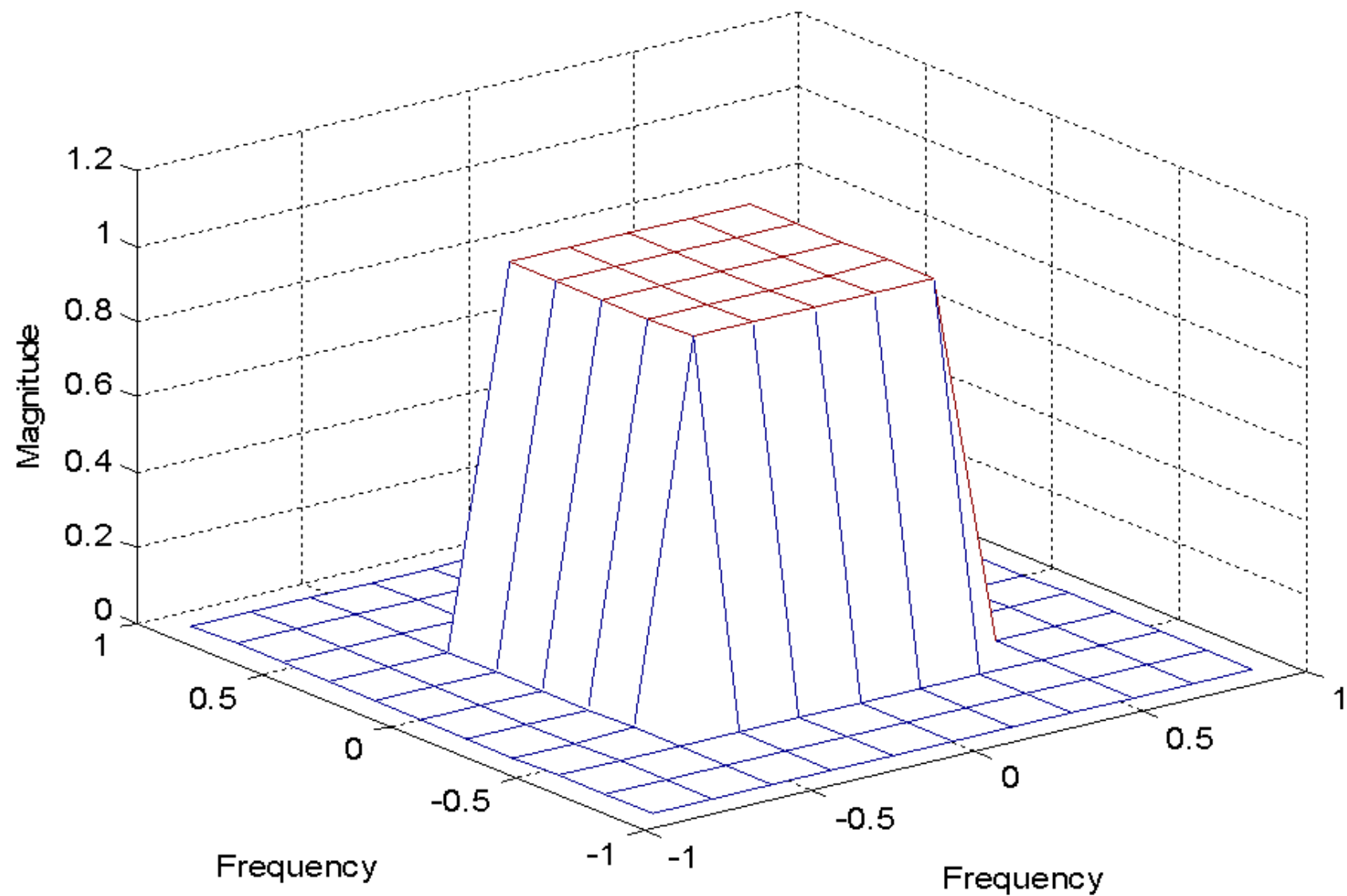
Ideal frequency response



Actual filter frequency response



Given filter ideal response



Hamming window smoothed filter

