

Visual Computing

Linear Filtering, Edge Detection, Wiener Filter

General Remarks

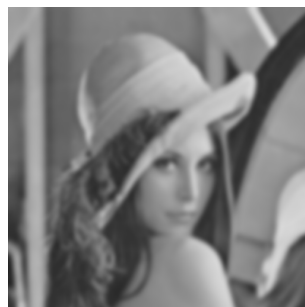
It is not necessary to hand in your results. You will find an exemplary solution on the lecture's web page.

1) Linear Filtering in Matlab

Using functions from the image processing toolbox, it is easy to study the effects of various linear filters.



Lenna image



Gaussian filter



Laplacian filter

- a)** Load the Lenna image 'lena.jpg' from the homepage (of course you can use any other image, too). To import the image into Matlab and convert the grayvalues to the range $[0, 1]$, use the following commands:

```
ima = imread('lena.jpg');  
ima = double(ima)/255;
```

The image can be displayed by:

```
figure; imshow(ima);
```

- b)** A two-dimensional linear filter can be defined in Matlab by using the command 'fspecial'. For example, smoothing an image with a box filter of size 3×3 can be achieved in the following way:

```
h = fspecial('average',3);  
imaf = imfilter(ima,h,'replicate');
```

Smooth the Lenna image with box filters of size 3, 5, 10, 20, 50, and display the results.

- c) Several computer vision applications require extracting different types of structure from an image. For this purpose, a *Gaussian pyramid* representation is very useful. The different layers of such a pyramid contain subsequently smoothed and resampled versions of the image. Starting with the original image as the finest layer, the next layer is computed by first applying a Gaussian filter with fixed size k and standard deviation σ and then sampling every second pixel in each direction to obtain an image of half the size (this is similar to the Mipmaps discussed in the last exercise).

Compute a Gaussian pyramid with 8 layers for the Lenna image (using e.g. $\sigma = 0.5$) and store the result in a cell array:

```
gp = cell(8,1);
gp{1} = ima;
...
```

- d) A simple idea to detect edges in an image is based on the corresponding *Laplacian of Gaussian (LoG)*: As large brightness changes are characterized by extremal magnitudes of the derivative, zero crossings of the second derivative of an image (the Laplacian in 2D) are good indicators for edges. In order to obtain more robust estimates, the image is smoothed first (to reduce noise) with a Gaussian filter before the second derivative is computed.

The 'LoG'-filter in Matlab directly computes the Laplacian after Gaussian smoothing. Apply this filter for standard deviations $\sigma = 2, 4, 8$ and display the filtered image as well as the corresponding zero crossings. How do the edges change with increasing σ ?

Hints: For a given σ , an appropriate size n of the filter can be computed as

```
n = ceil(sigma(i)*3)*2+1;
```

As the 'LoG'-filter returns positive and negative values, the result should be displayed by:

```
imagesc(imaf); colormap(gray); axis image; axis off;
```

Instead of computing the zero-crossings explicitly, you could also use the 'edge'-function of Matlab.

- e) Image smoothing and edge detection can also be performed by applying lowpass and highpass filters in the frequency domain. To this end, first compute the Fourier transform of the image and shift the zero-frequency component to the middle of the spectrum:

```
fima = fftshift(fft2(ima));
```

The ideal lowpass filter then directly erases all frequencies above a given threshold by setting the corresponding entries in `fima` to zero. This can be achieved by multiplying the image in the frequency domain with an appropriate 0-1 image of the same size. For example, the following code erases all frequencies that are above the threshold in either of the two dimensions:

```
thresh = 30;
cx = size(ima,1)/2; cy = size(ima,2)/2; % find the center of the image
h = zeros(size(ima));
h(cx-thresh:cx+thresh, cy-thresh:cy+thresh) = 1;
fima2 = fima.*h;
```

To display the result, the image has to be transformed back to the spatial domain:

```
ima2 = abs(ifft2(fima2));
```

How do you obtain a highpass filter? In general, it is more convenient to threshold according to the distance $(u^2 + v^2)$ from the origin in the frequency domain to obtain a circular filter instead of a box — how can this be achieved?

Filter the Lenna image with lowpass and highpass filters for thresholds 10, 30, 50, and display the results. Which artefacts do you observe? How can they be prevented?

- f) The Wiener filter is in some sense ‘optimal’ to restore a noisy image. To see it working, add Gaussian white noise with variance $\sigma = 0.005$ to the Lenna image, and restore the image by using the Matlab function `wiener2`:

```
s = 3;
ima_noise = imnoise(ima, 'gaussian', 0, 0.005);
ima_wiener = wiener2(ima_noise, [s s]);
```

Here, s gives the size of the window that is used to estimate the local mean and standard deviation at every pixel for adapting the Wiener filter. Varying s , how does the restoration of the image change? How do you explain the effects?

2) Wiener Filter for Blurred Images

For simplicity, let us consider a 1-dimensional ‘image’. Assume that the original image $f(x)$ has been blurred due to a convolution with the kernel (box filter)

$$h(x) = \begin{cases} 1 & \text{if } |x| \leq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

After that, the noise $\eta(x)$ has been added. We assume that the noise and the original image are uncorrelated, i.e., the cross-correlation $\Phi_{f\eta}(x)$ vanishes for all x . The resulting image $b(x) = (f * h)(x) + \eta(x)$ is thus blurred and noisy.

In order to improve the quality of the image $b(x)$, let us apply the Wiener Filter, determined by its kernel $\tilde{h}(x)$. The goal is to obtain the improved image $o(x) = (b * \tilde{h})(x)$ that is as similar as possible to the original image $f(x)$.

- What measure can we use to quantify the ‘similarity’ of the images $o(x)$ and $f(x)$?
- Given the auto-correlation Φ_{bb} of the blurred and noisy image b and the cross-correlation Φ_{bf} between the degraded image b and the original image f , which equation has to hold for the kernel function $\tilde{h}(x)$ of the Wiener filter? Write down the equation that explicitly determines its Fourier transform $\mathcal{F}[\tilde{h}](u)$ in terms of the signal-to-noise ratio $\text{SNR}(u) = \frac{\hat{\Phi}_{ff}(u)}{\hat{\Phi}_{\eta\eta}(u)}$ given by the power spectra $\hat{\Phi}_{ff}$ and $\hat{\Phi}_{\eta\eta}$ of the image and the noise.
- Now consider the case that there is no noise, i.e., we have for the signal-to-noise ratio (SNR): $1/\text{SNR} = 0$. Sketch the qualitative behavior of the Fourier transform $\mathcal{F}[\tilde{h}](u)$ of the Wiener filter as a function of the frequency u .
Hint: Use the fact that the Fourier transform of h is $\hat{h}(u) = \text{sinc}(\pi u)$.
- Now consider the case that the signal-to-noise ratio (SNR) is constant over all frequencies: $\text{SNR}(u) = c$. Sketch the qualitative behavior of the Fourier transform $\mathcal{F}[\tilde{h}](u)$ of the Wiener filter for $c = 1000$ and $c = 300$ as a function of the frequency u . What is the difference between the Wiener filter in this question compared to the one in the previous question?
- Now consider the case that the signal-to-noise ratio (SNR) decreases with growing frequencies like $\text{SNR}(u) = a/u^2$ for some constant $a > 0$. Sketch the qualitative behavior of the Fourier transform $\mathcal{F}[\tilde{h}](u)$ of the Wiener filter for $a = 1000$ and $a = 300$ as a function of the frequency u .