ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

**Computer Graphics Lab**
**ML Group**

*Prof. M. Gross / Prof. J. Buhmann*

*Remo Ziegler, Christian Voegeli, Daniel Cotting, Christian Sigg, Jens Keuchel*

Solve before: June 6, 2006

# 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);
```

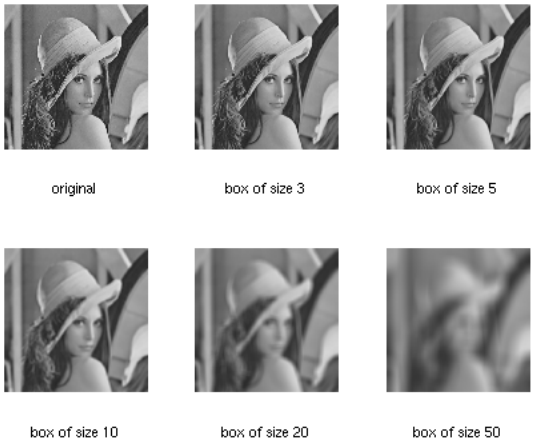
**b)** A two-dimensional linear filter can be defined in Matlab by using the command 'fspecial'. For example, smoothing an image with a box filter of size $3 \times 3$ can be achieved in the following way:

```
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.

*Solution:*

```
figure; subplot(2,3,1); imshow(ima); xlabel('original');
fsize=[3 5 10 20 50];
for i=1:length(fsize)
  h = fspecial('average',fsize(i));
  imaf = imfilter(ima,h,'replicate');
  subplot(2,3,i+1); imshow(imaf); xlabel(['box of size ' int2str(fsize(i))]);
end;
```




**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 $\sigma$ 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;
...
```

*Solution:*

```
gp = cell(8,1);
gp{1} = ima;
fsize = 3; sigma = 0.5;
h = fspecial('gaussian',fsize,sigma);
for i=2:8
  imaf = imfilter(gp{i-1},h,'replicate');
  gp{i} = imaf(1:2:size(gp{i-1},1),1:2:size(gp{i-1},2));
end;
figure;
for i=1:8
  subplot(2,4,i); imshow(gp{i}); xlabel(['layer ' int2str(i)]);
end;
```


**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 $\sigma$?

Hints: For a given $\sigma$, 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:

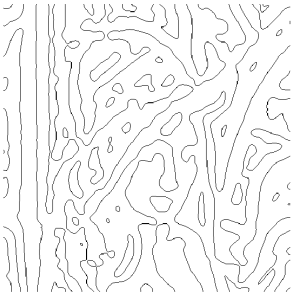
```
imagesc(imaf); colormap(gray); axis image; axis off;
```

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

*Solution:*

```
sigma=[2 4 8];
for i=1:3
  n = ceil(sigma(i)*3)*2+1;
  h = fspecial('log',n,sigma(i));
  imaf = imfilter(ima,h,'replicate');
  figure;
  subplot(1,2,1); imagesc(imaf); colormap(gray); axis image; axis off;
  imaf2 = ones(size(ima));
  for ii=1:size(ima,1)
    for jj=1:size(ima,2)
      if ( (ii > 1) && (imaf(ii,jj)<0) && (imaf(ii-1,jj)>0) ) || ...
         ( (jj > 1) && (imaf(ii,jj)<0) && (imaf(ii,jj-1)>0) ) || ...
         ( (jj < size(ima,2)) && (imaf(ii,jj)<0) && (imaf(ii,jj+1)>0) ) || ...
         ( (ii < size(ima,1)) && (imaf(ii,jj)<0) && (imaf(ii+1,jj)>0) )
         imaf2(ii,jj) = 0;
      end
    end
  end
```

```
  subplot(1,2,2); imshow(imaf2);
end
```



*With increasing $\sigma$, details are suppressed and fewer edges are obtained.*

**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));
```

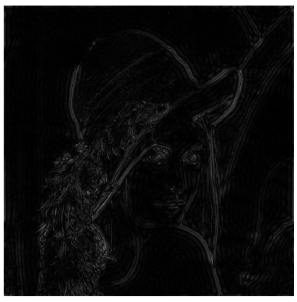
How do you obtain a highpass filter? In general, it is more convenient to threshold according to the distance ($u^2 + v^2$) from the origin in the frequency domain to obtain a circular filter instead of a box — how can this be achieved?

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?

*Solution:*

```
fima = fftshift(fft2(ima));
cx = size(ima,1)/2; cy = size(ima,2)/2;
thresh=[10 30 50];
for i=1:3
  h = zeros(size(ima));
  for x=-thresh(i):thresh(i)
    for y=-thresh(i):thresh(i)
      if (sqrt(x^2+y^2)<thresh(i))
        h(cx+x,cy+y) = 1;
      end
```

```
    end
  end
  fima_low = fima.*h;
  fima_high = fima.*(1-h);
  ima_low = abs(ifft2(fima_low));
  ima_high = abs(ifft2(fima_high));
  figure;
  subplot(1,2,1); imshow(ima_low);
  subplot(1,2,2); imshow(ima_high);
end
```



*The lowpass filter results in ringing artefacts (ripple like structures). Apply Gaussian lowpass filter to prevent this.*

**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?

*Solution:*

```
ima_noise = imnoise(ima,'gaussian',0,0.005);
figure;
subplot(2,2,1); imshow(ima_noise);
s = [3 5 10];
for i=1:3
  ima_wiener = wiener2(ima_noise,[s(i) s(i)]);
  subplot(2,2,i+1); imshow(ima_wiener);
end
```




*The Matlab implementation of the Wiener filter adapts locally to the variation estimated within a small window. This results in generally smoother images for larger window sizes $s$. However, at edges, where the signal to noise ratio is large, the image is less smoothed, which gives noisy boundaries for larger $s$.*

## 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 $f(x)$.

**a)** What measure can we use to quantify the 'similarity' of the images $o(x)$ and $f(x)$?
*Solution: A suitable measure is the average quadratic error:*

$$E = \int_{-\infty}^{+\infty} (o(x) - f(x))^2 \, dx$$

**b)** Given the auto-correlation $\Phi_{bb}$ of the blurred and noisy image $b$ and the cross-correlation $\Phi_{bf}$ 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.

*Solution: The kernel function $\tilde{h}(x)$ of the Wiener filter has to fulfill the Wiene r-Hopf equation:*

$$\Phi_{bf}(x) = (\Phi_{bb} * \tilde{h})(x).$$

*As this equation involves a convolution ($*$), i.e., an integration, it cannot be solved directly for $\tilde{h}$. Thus we transform this equation into Fourier space, $\hat{\Phi}_{bf}(u) = \hat{\Phi}_{bb}(u) \cdot \mathcal{F}[\tilde{h}](u)$, where it can be easily solved for the Fourier transform of the kernel function:*

$$\mathcal{F}[\tilde{h}](u) = \frac{\hat{\Phi}_{bf}(u)}{\hat{\Phi}_{bb}(u)} = \frac{\hat{h}(u) \cdot \hat{\Phi}_{ff}(u)}{\hat{h}^2(u) \cdot \hat{\Phi}_{ff}(u) + \hat{\Phi}_{\eta\eta}(u)} = \frac{\hat{h}(u)}{\hat{h}^2(u) + \underbrace{\frac{\hat{\Phi}_{\eta\eta}(u)}{\hat{\Phi}_{ff}(u)}}_{=1/SNR(u)}} \tag{1}$$

*The second equation holds as the cross-correlation in Fourier space becomes*

$$\hat{\Phi}_{bf}(u) = \hat{\Phi}_{f*h+\eta,f}(u) = \hat{\Phi}_{f*h,f}(u) + \underbrace{\hat{\Phi}_{\eta f}(u)}_{=0} = \hat{h}(u) \cdot \hat{\Phi}_{ff}(u).$$

*Equation* (1) *enables us to calculate the transformed kernel function $\mathcal{F}[\tilde{h}]$ of the Wiener filter given the Fourier transform of the 'degradation kernel' $h$ and the power spectra $\hat{\Phi}_{ff}$ and $\hat{\Phi}_{\eta\eta}$.*

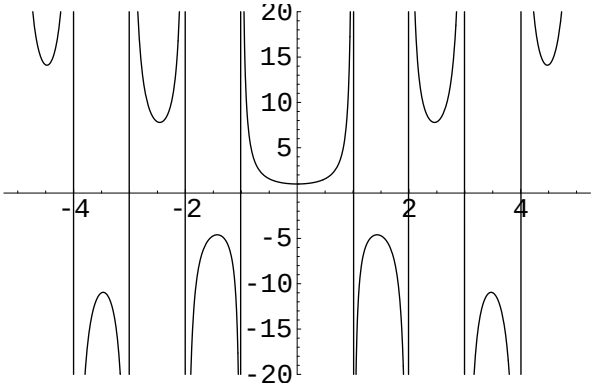
**c)** Now consider the case that there is no noise, i.e., we have for the signal-to-noise ratio (SNR): $1/\text{SNR} = 0$. Sketch the qualitative behavior of the Fourier transform $\mathcal{F}[\tilde{h}](u)$ of the Wiener filter as a function of the frequency $u$.

Hint: Use the fact that the Fourier transform of $h$ is $\hat{h}(u) = \text{sinc}(\pi u)$.

*Solution: When $1/SNR = 0$, then equation* (1) *simplifies considerably:*

$$\mathcal{F}[\tilde{h}](u) = \frac{1}{\hat{h}(u)}.$$

*For $\hat{h}(u) = \text{sinc}(\pi u)$, we obtain the following graph for $\mathcal{F}[\tilde{h}](u)$: as $\mathcal{F}[\tilde{h}](u)$ diverges at the frequencies $u = \pm 1, \pm 2, \ldots$, the frequencies in the vicinity of these frequencies get amplified extremely by the Wiener Filter.*

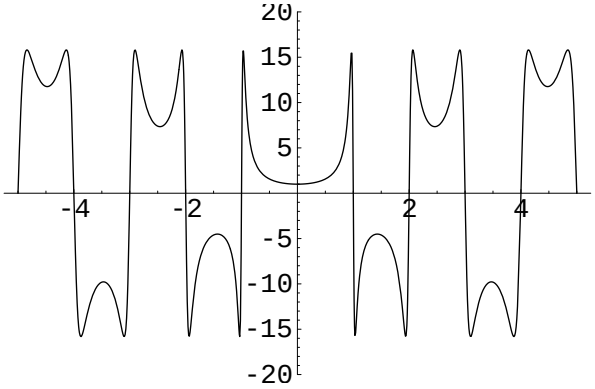



**d)** Now consider the case that the signal-to-noise ratio (SNR) is constant over all frequencies: $\text{SNR}(u) = c$. Sketch the qualitative behavior of the Fourier transform $\mathcal{F}[\tilde{h}](u)$ of the Wiener filter for $c = 1000$ and $c = 300$ as a function of the frequency $u$. What is the difference between the Wiener filter in this question compared to the one in the previous question?

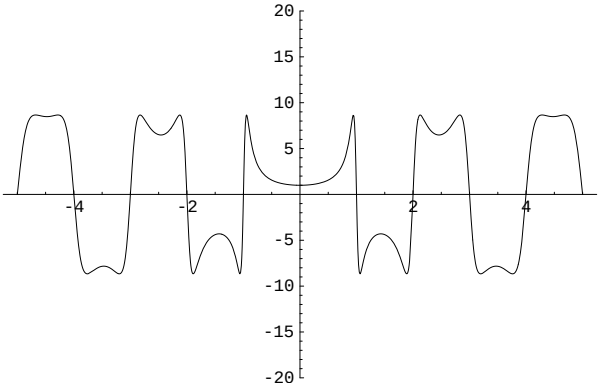
*Solution: When SNR$(u) = c$ for all frequencies, then equation* (1) *becomes:*

$$\mathcal{F}[\tilde{h}](u) = \frac{\hat{h}(u)}{\hat{h}^2(u) + \frac{1}{c}}$$

*The following graphs depict $\mathcal{F}[\tilde{h}](u)$ for SNR$(u) = 1000$ and SNR$(u) = 300$: as the noise level increases, the Wiener filter leads to an increased overall damping of the frequencies. As a result, this helps avoiding the amplification of noise.*




$c = 1000$



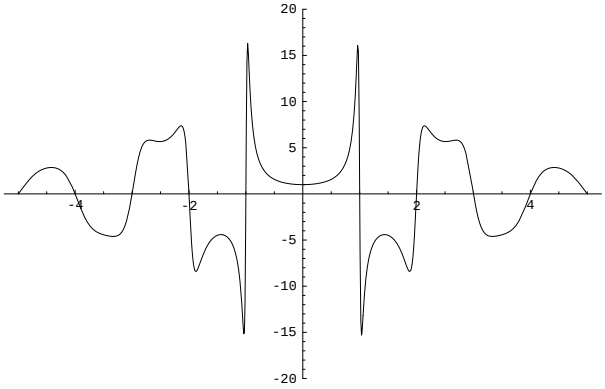

$c = 300$

**e)** Now consider the case that the signal-to-noise ratio (SNR) decreases with growing frequencies like 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 $u$.

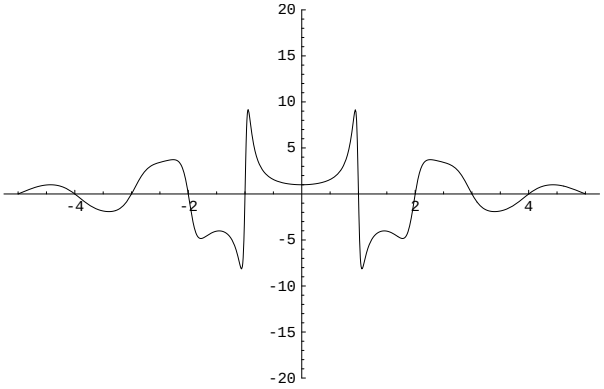
*Solution: When SNR$(u) = a/u^2$ for some constant $a > 0$, then equation* (1) *becomes:*

$$\mathcal{F}[\tilde{h}](u) = \frac{\hat{h}(u)}{\hat{h}^2(u) + u^2/a}$$

*The following graphs depict $\mathcal{F}[\tilde{h}](u)$ for $a = 1000$ and $a = 300$: as SNR decreases with growing frequency $u$, the Wiener filter leads to an increased damping of higher frequencies. Apart from that (and as in the previous case), if the overall noise level is larger (i.e., $a$ is smaller), the Wiener filter increases the overall damping of all frequencies.*




$a = 1000$




$a = 300$