Image enhancement

Mårten Björkman

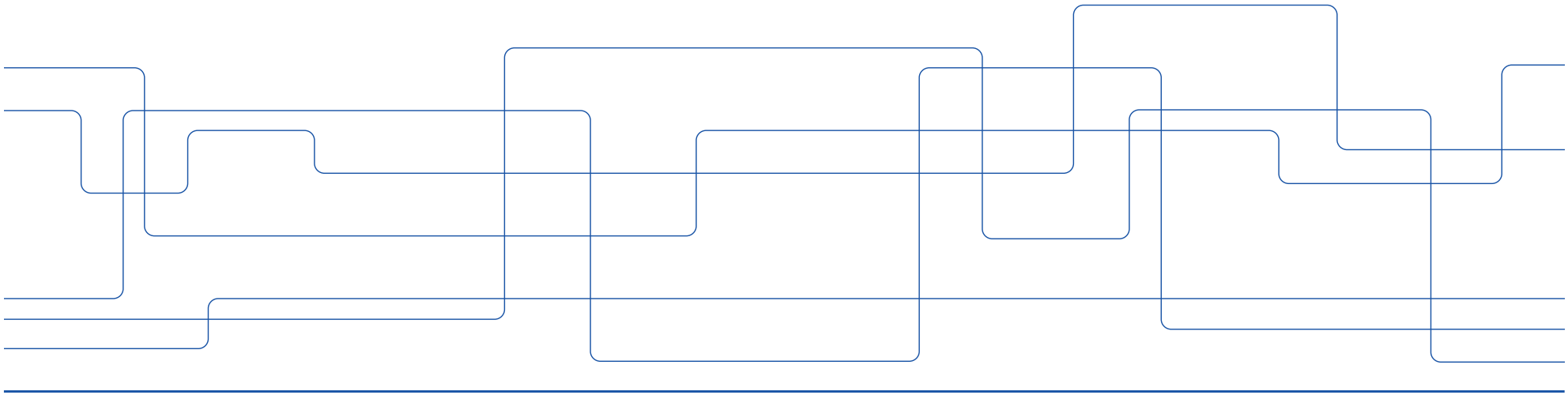
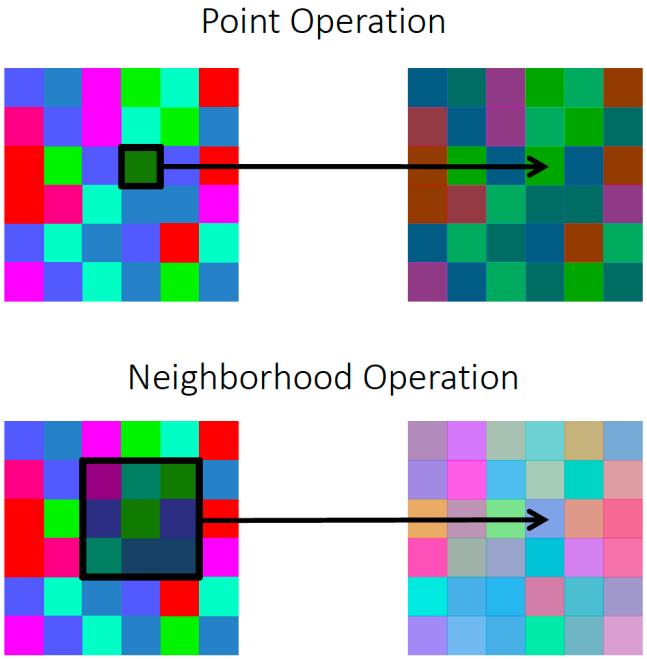


Image Filtering



point processing

filtering

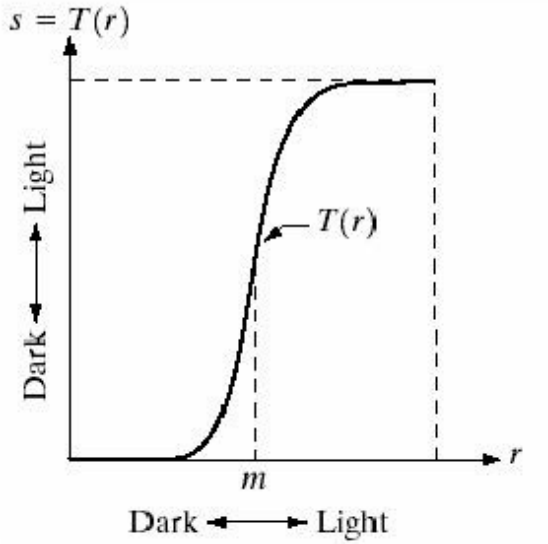
- Most spatial image filters work in local neighbourhoods. You combine a group of pixels in a neighbourhood to compute a new value for the pixel in the centre.
- However, you can do a lot by just updating the values for each pixel individually.

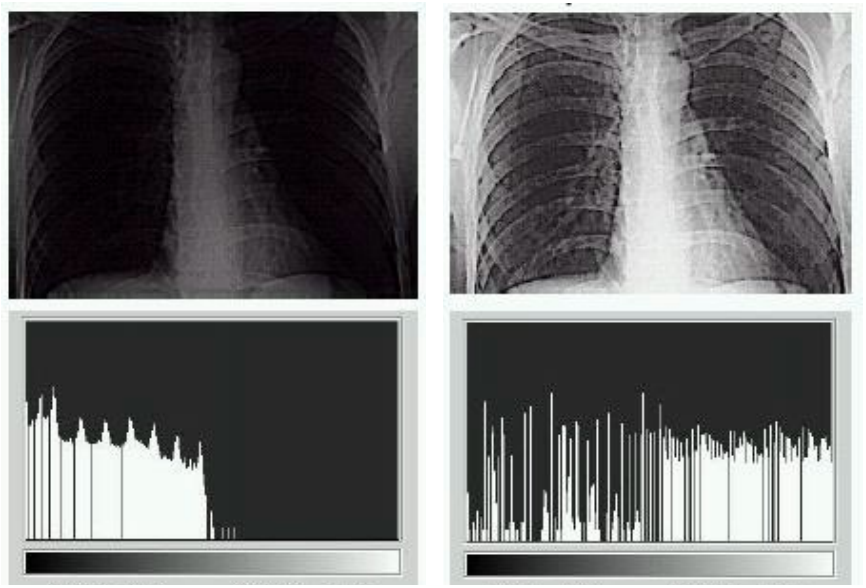
Image enhancement by grey-level transformation

- Grey-level transformations

$$s = T(r)$$

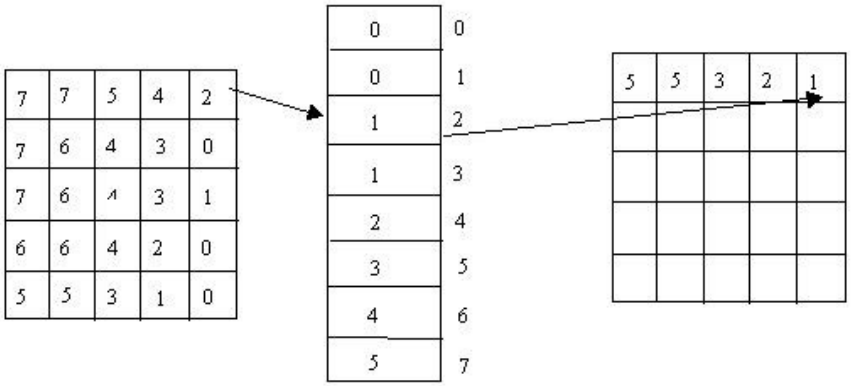
where s and r are intensities after and before. $T(r)$ may e.g. be a piecewise linear, negative, logarithm transformation.

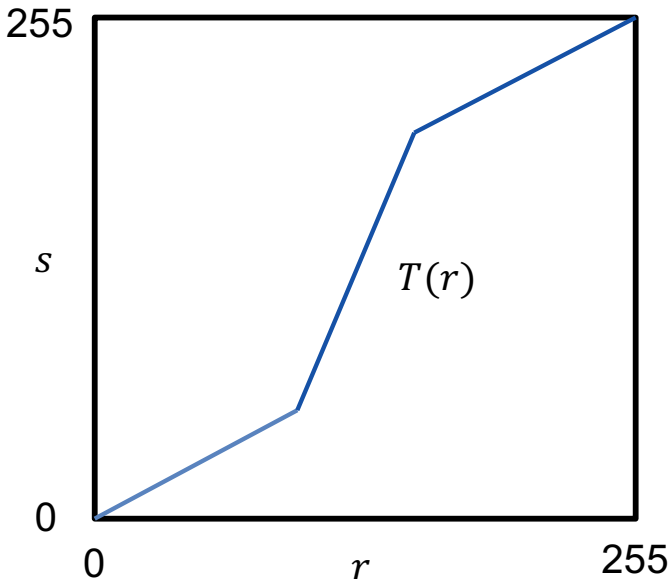




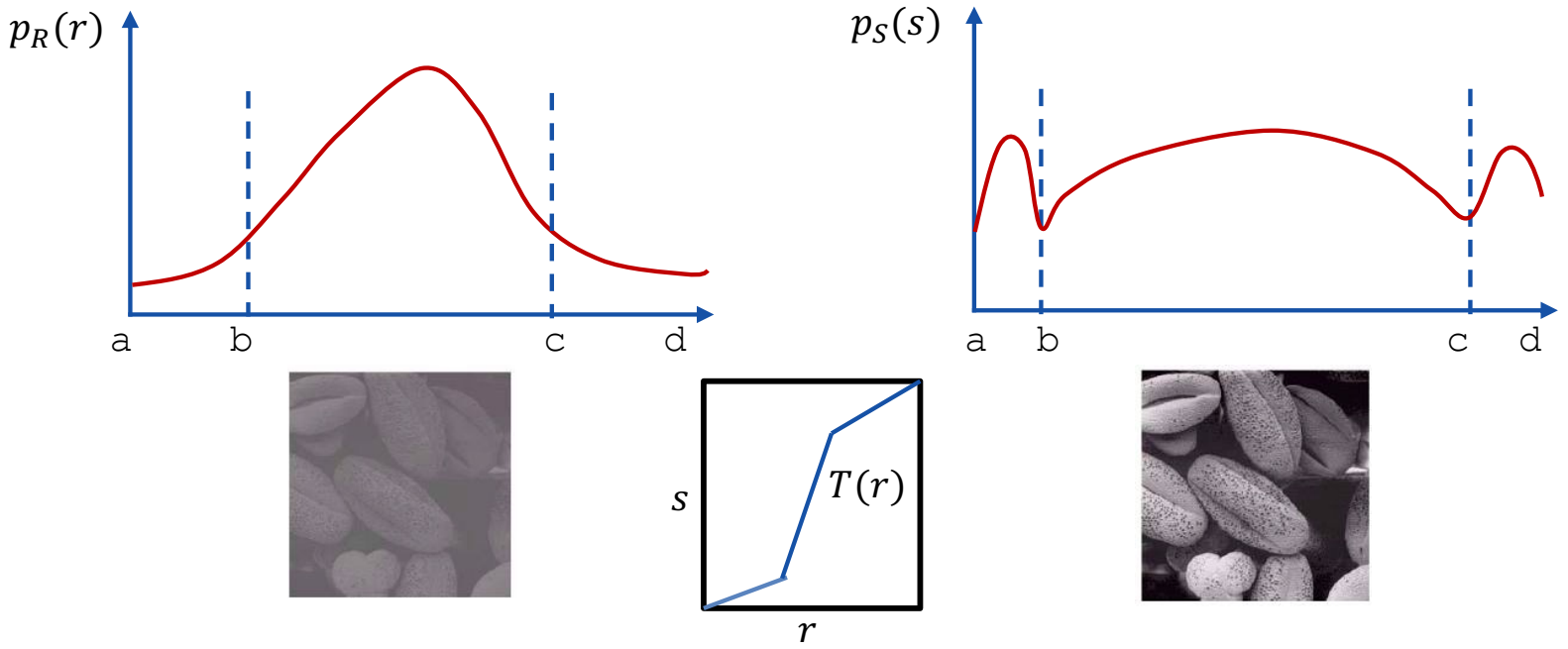
Look-Up Tables (LUT)

- Often implemented with LUTs (256 entries), at least for complex functions. Sometimes included directly in the camera. Different setting, different LUT.



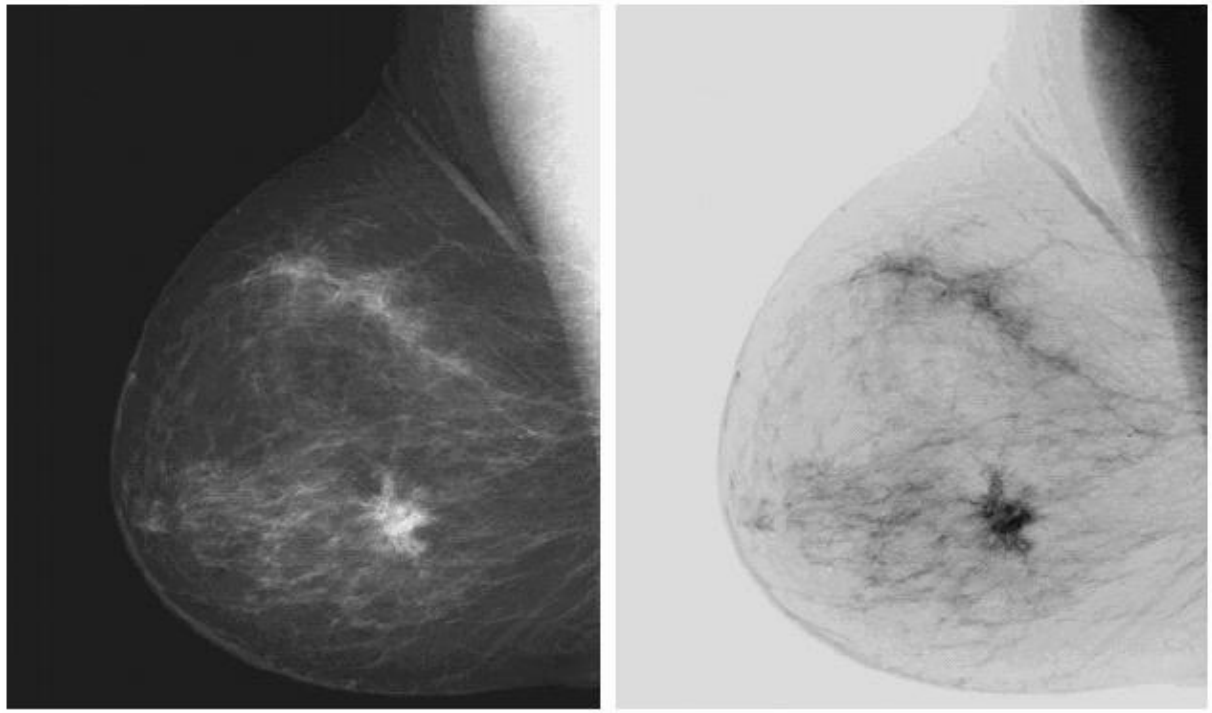


Histogram stretching



- Increase contrast by letting the interval `[b,c]` cover a larger range of grey-levels. Note: Information loss in `[a,b]` and `[c,d]`.
- Note: If you stretch one interval, you need to compress another.

Image negative



Just computing an image negative can be surprisingly effective.

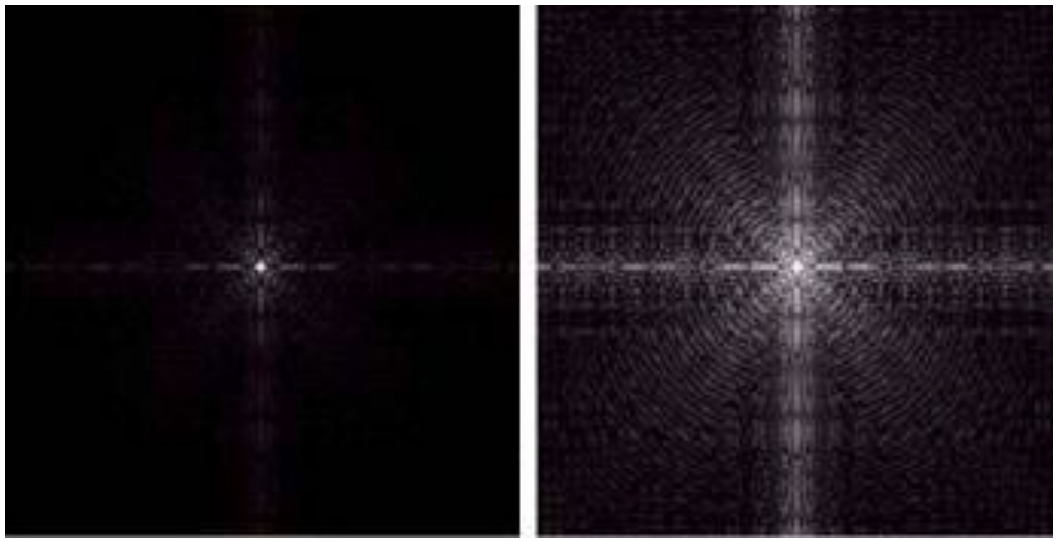
Log transformations

- Useful for compressing large dynamic range and make details visible.

$$s = c \log(1 + r)$$

Example: Fourier spectrum

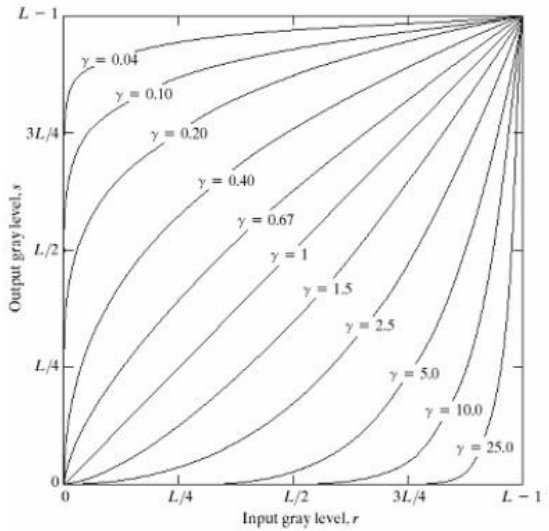
$$0 \rightarrow 1.5x10^6 \quad to \quad 0 \rightarrow 6.2$$

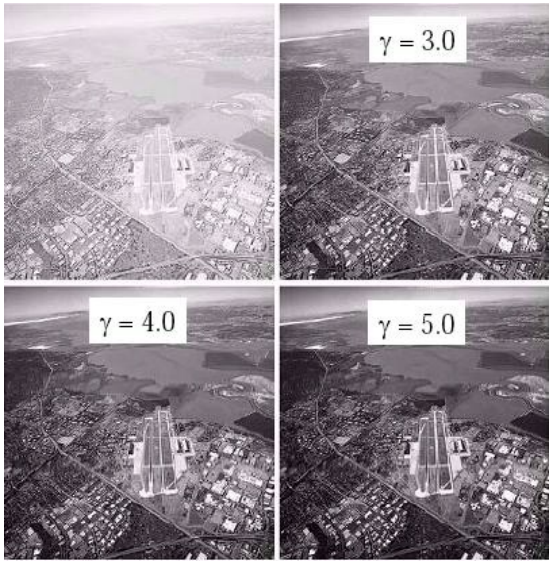


Power-law transformations

- A variety of devices used for image capture, printing, and display respond according to a power law (gamma correction).

$$s = c\, r^{\gamma} \quad or \quad s = c\,(r + \varepsilon)^{\gamma}$$





Histogram equalization (continuous case)

Idea: Find transformation $s = T(r)$ such that the distribution $p_S(s)$ of pixel values is uniform, $p_S(s) = 1$, given a distribution from an image $p_R(r), r \in [0,1]$.

A small range in R of width Δr is mapped to a small range in S of width Δs. The number of pixels in these ranges must be the same.

$$N = p_R(r)\Delta r = p_S(s)\Delta s$$

For really small intervals we have

$$\frac{p_R(r)}{p_S(s)} = p_R(r) = \frac{\Delta s}{\Delta r} \rightarrow \frac{ds}{dr} = T'(r), \text{ as } \Delta r \rightarrow 0$$

Thus the transformation is given by

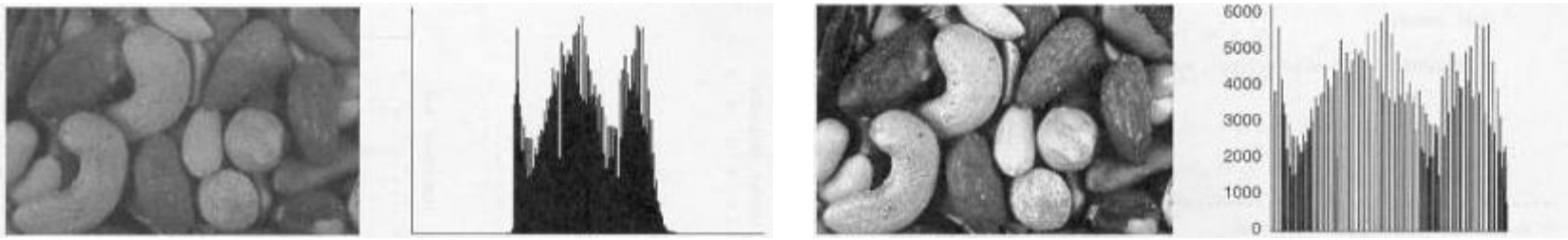
$$s = T(r) = \int_0^r p_R(r')\, dr'$$

Histogram equalization (discrete case)

1. Compute histogram: count each distinct pixel value in the image.
2. Store cumulative sum of all the histogram values and normalize them by dividing each element by the number of pixels.

$$s_k = T(r_k) = \sum_{i=0}^{k} p_r(r_i) = \sum_{i=0}^{k} \frac{n_i}{N}, \qquad 0 \leq r_k, s_k \leq 1, \quad k = 0, 1, \ldots, 255$$

3. Use LUT from step 2 to transform the input image.



Note! Typically values of s_k are scaled up by 255 and rounded to the nearest integer so that the output values are between 0 and 255. Due to discretization the transformed image will not have a perfectly uniform histogram.

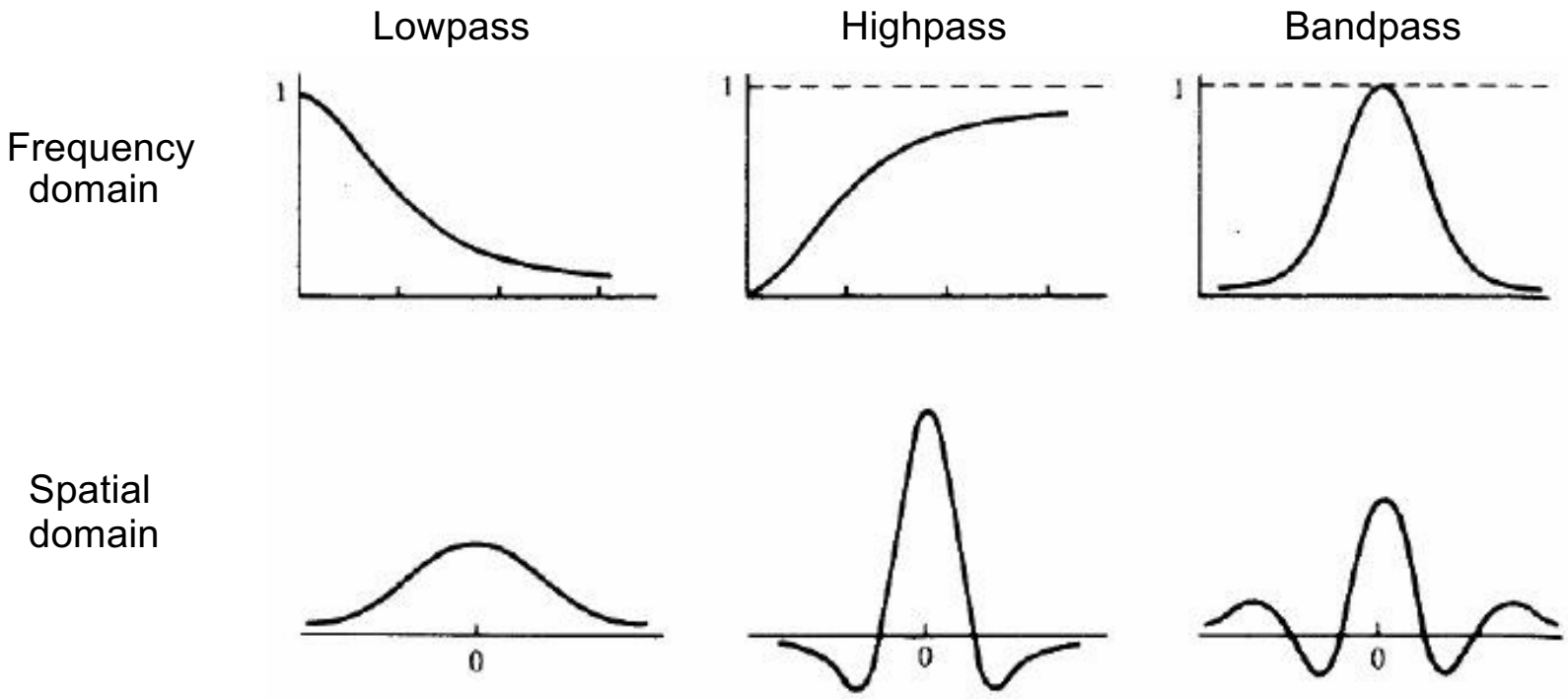
Image enhancement by spatial filtering

- Spatial filters use spatial masks and can be either linear or non-linear.

$$g(x,y) = \sum_{i,j} h(i,j) f(x-i, y-j)$$

- A linear filters can be either:
  - Lowpass: eliminate high frequency components such as characterized by edges and sharp details in an image. ⇒ Net effect is image blurring.
  - Highpass: eliminate low frequency components such as slowly varying characteristics (shadings). ⇒ Removes everything but sharp changes, e.g. noise.
  - Bandpass: eliminate outside a given frequency range. ⇒ Combination of the above. Common in practice.
- A non-linear filter cannot be written as a weighted sum of input pixels.

Linear filter examples



Exercise

- Assume you have a filter kernel $[1,0,-1]$. How does this look like in the Fourier domain? Is it a lowpass, highpass or bandpass filter?

Answer: To see this we have to express the filter in continuous domain, which we can do with Dirac functions.

$$h(x) = \delta(x+1) - \delta(x-1)$$

To get the Fourier Transform we exploit the sifting property of Dirac functions.

$$\hat{h}(\omega) = \int h(x) e^{-i\omega x} dx = e^{i\omega} - e^{-i\omega} = 2i \sin(\omega)$$

$$\left|\hat{h}(\omega)\right| = 2|\sin(\omega)|$$

Since $\left|\hat{h}(0)\right| = \left|\hat{h}(\pi)\right| = 0$ and $\left|\hat{h}(\pi/2)\right| = 2$ it must be a bandpass filter.

Different kinds of noise

Noise is the result of errors in the image acquisition that lead to pixel values that do not reflect the true intensities of the real scene.

- Signal independent additive noise (sampling noise)

$$g = f + n$$

- Signal dependent multiplicative noise (illumination variations)

$$g = f + nf = (1 + n)f$$

- Measurement noise (salt & pepper) – individual pixels being completely wrong.

Local spatial averaging / Mean filtering

Assume we average pixels in local neighbourhoods: $g(x) = \sum_{k \in N} h(k)\, f(x-k)$

Usually $\sum h(k) = 1$, Example: $N = N_8,\ h(k) = \frac{1}{9}$ gives the box filter $\frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$

- Two main problems with mean filtering:
  - A single pixel can significantly affect the mean value of all the pixels in its neighbourhood.
  - It blurs edges - a problem if we require sharp edges in the output.

Local spatial averaging (continue)

- Common requirements:
  - Coefficients should sum up to 1.
  - It should be symmetric up/down and left/right.
  - Centre pixel should have most influence on output.
  - Filter should be separable.
- This leads to:

$$h = \begin{pmatrix} \frac{\Delta t}{2} \\ 1-\Delta t \\ \frac{\Delta t}{2} \end{pmatrix} \begin{pmatrix} \frac{\Delta t}{2} & 1-\Delta t & \frac{\Delta t}{2} \end{pmatrix} = \begin{pmatrix} \frac{\Delta t^2}{4} & \frac{\Delta t}{2}(1-\Delta t) & \frac{\Delta t^2}{4} \\ \frac{\Delta t}{2}(1-\Delta t) & (1-\Delta t)^2 & \frac{\Delta t}{2}(1-\Delta t) \\ \frac{\Delta t^2}{4} & \frac{\Delta t}{2}(1-\Delta t) & \frac{\Delta t^2}{4} \end{pmatrix}$$

- Special case: $\Delta t = \frac{1}{2}$ gives

$$h = \begin{pmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{pmatrix}$$

Lowpass filters

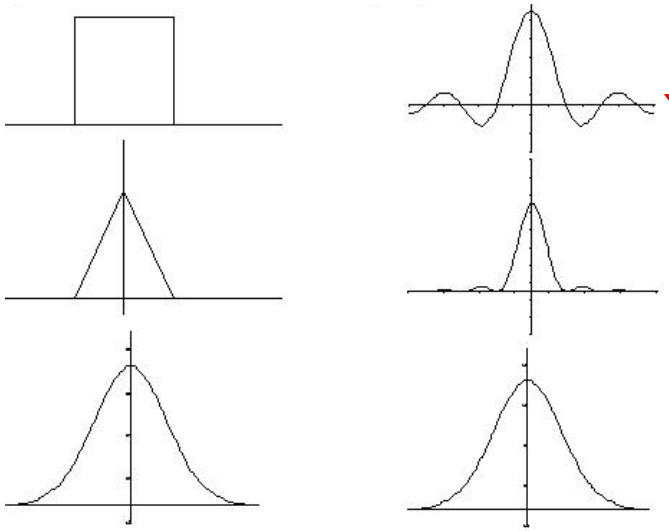
- Most information in images is concentrated at low frequencies.
- Noise is uniformly distributed over all frequencies (white noise).

  $\Longrightarrow$ Suppress high frequency.
- Different filters have different qualities in Fourier space.

Box

Triangle

Gaussian



Negative values are unreasonable.

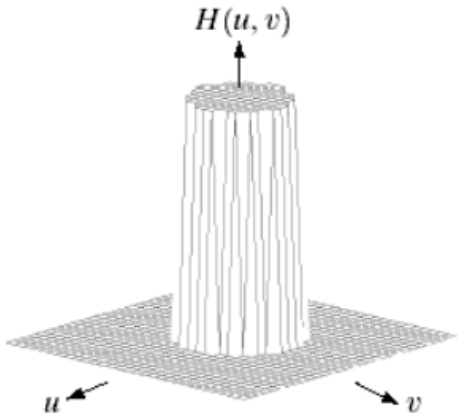
Much more reasonable

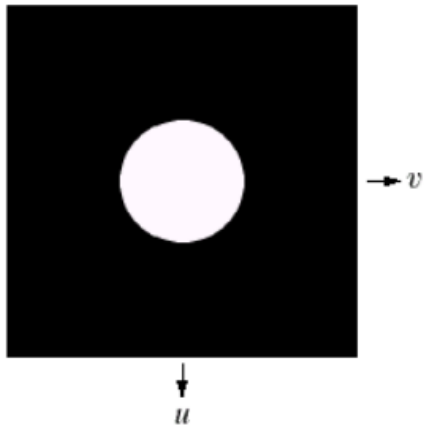
Ideal lowpass filter

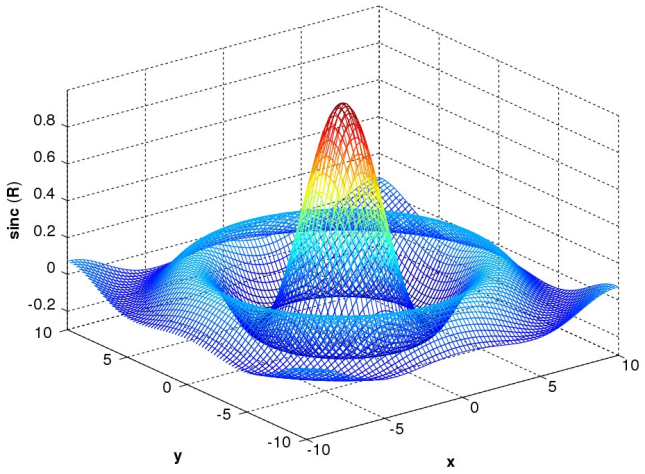
A transfer function $H(u,v)$ of an ideal low-pass filter is given as

$$H(u,v) = \begin{cases} 1 & \text{if } D(u,v) \leq D_0 \\ 0 & \text{if } D(u,v) > D_0 \end{cases}$$

where D_0 is a nonnegative cutoff frequency and $\mathrm{D}(\mathrm{u},\mathrm{v}) = \sqrt{u^2 + v^2}$ is the distance of (u,v) to the centre of the frequency plane.

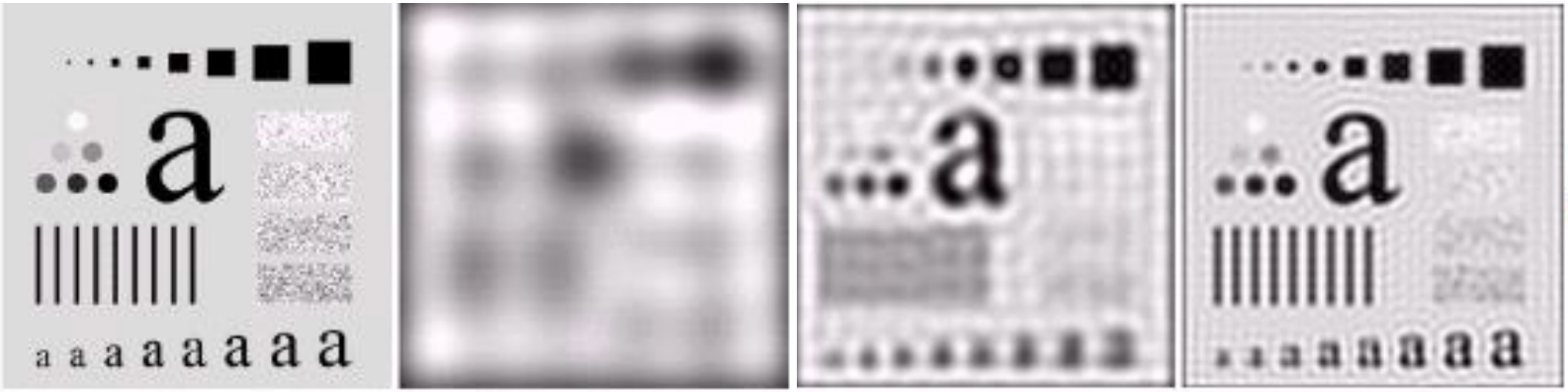






Ideal lowpass filter

- Results after applying ideal lowpass filters with different cutoff frequencies.
- Depending on the cutoff frequency, severe ringing effects can be seen.



Gaussian lowpass filter

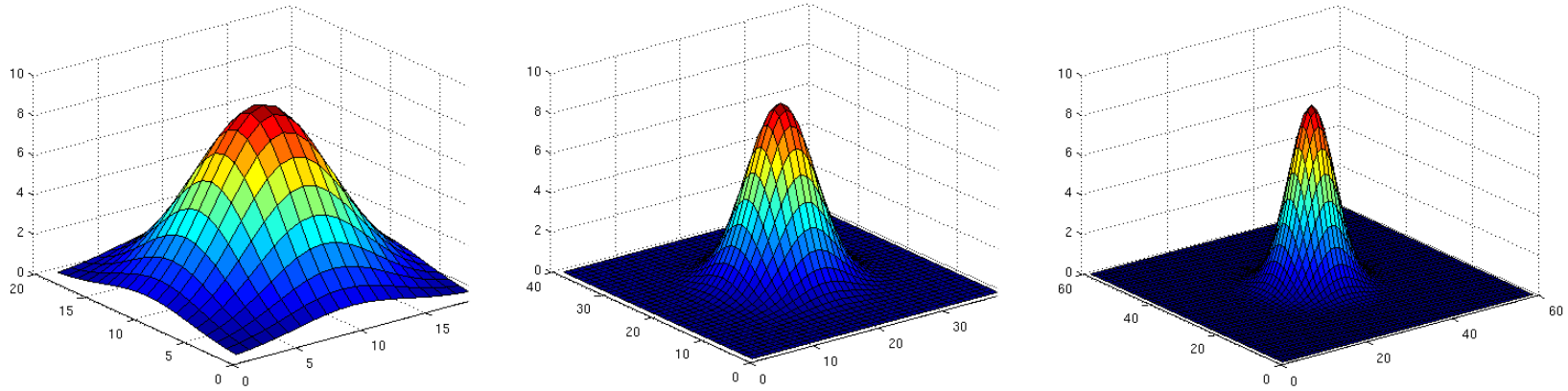
$$g(x, y; \sigma^2) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

$$\hat{g}(u, v; \sigma^2) = e^{-\sigma^2(u^2+v^2)/2}$$

Both are Gaussians!!

where $(x^2 + y^2)$ = squared distance from the origin.

- The parameter σ measures spread of the Gaussian curve. When $(x^2 + y^2) = \sigma^2$, the filter is at 0.607 of its maximum value.

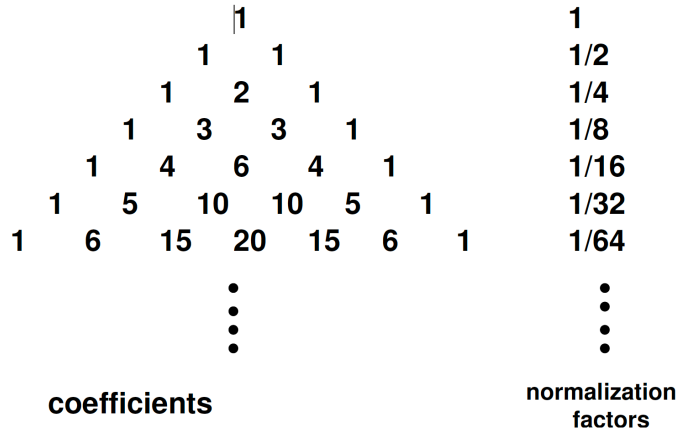


Binomial kernels

- The filter $(\frac{\Delta t}{2}, \Delta t, \frac{\Delta t}{2})$ can for $\Delta t = \frac{1}{2}$ be written

$$\left(\frac{1}{2}, \frac{1}{4}, \frac{1}{2}\right) = \frac{1}{4}(1,2,1) = \frac{1}{2}(1,1) * \frac{1}{2}(1,1)$$

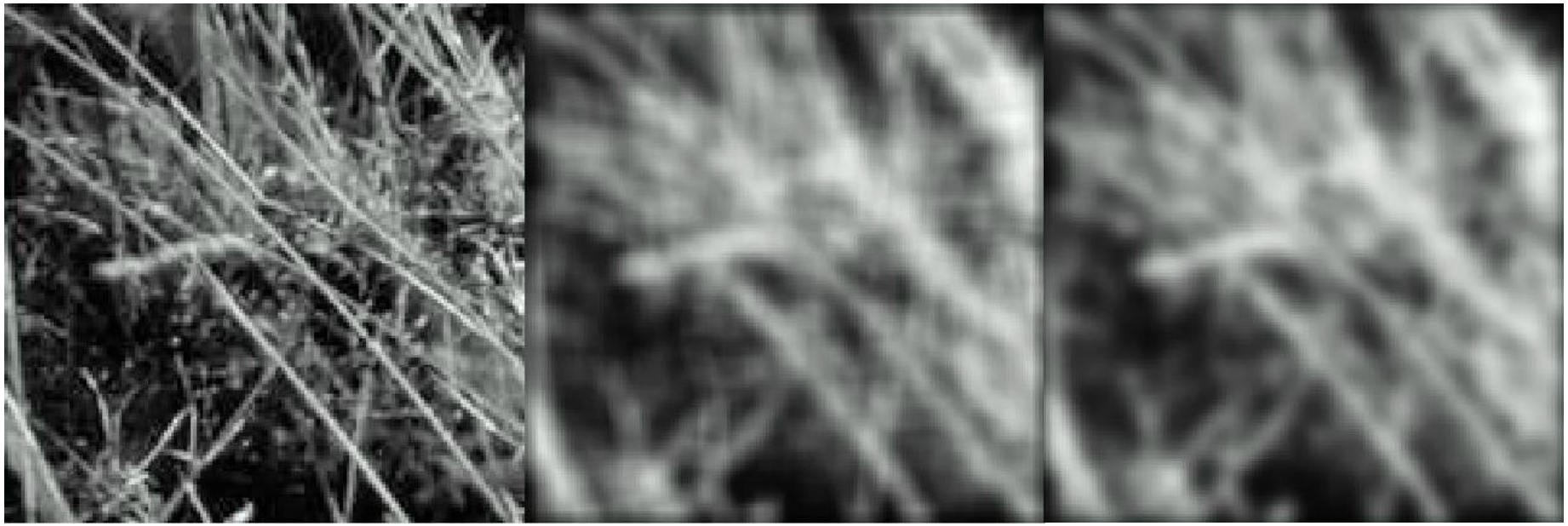
Repeated use of $(1,1)$ kernels gives rise to Pascal's triangle.



Central limit theorem $\Longrightarrow$ kernels approach Gaussian kernels.

Blurring: Mean vs. Gaussian

A Gaussian is isotropic and does not lead to the same blocky patterns.



Original Mean Gaussian

Non-linear filtering

- Nonlinear spatial filters usually also operate on local neighbourhoods.
- Operations are based directly on pixel values in neighbourhoods.

  They do not explicitly use coefficient values as in filter masks.
- Typical purpose: Incorporate prior knowledge to avoid destructive behaviour, typically at edges and corners.
- Basic methods:
  - median filtering
  - selective averaging
  - weighted averaging

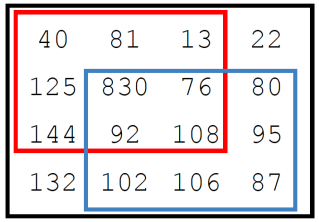
Median filtering

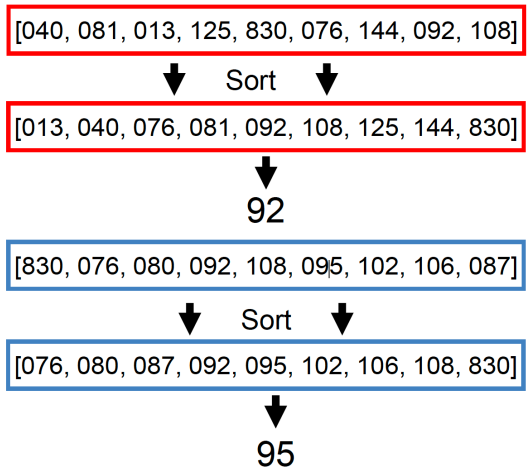
Compute the median value of pixels in local neighbourhoods.

Properties:

+ Preserves the value in 1D monotonic structures (shading).
+ Preserves the position of 1D step edges.
+ Eliminates local extreme values (e.g. salt-and-pepper).
- Creates painting-like images.

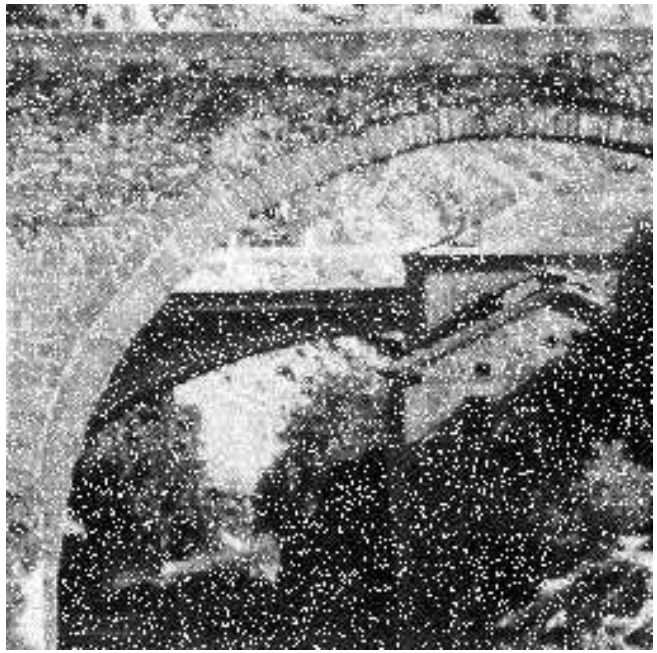
$$g(x) = median_{n \in N(x)} \; f(n)$$





Median filtering (example)

Original



Median 5x5

Mean 5x5

Median filtering can be very good at ignoring occasional really wrong pixels.

Anisotropic smoothing with bilateral filtering

- Anisotropic smoothing: smooth differently in different directions, usually in order to preserve edges.
- Idea: smooth pixels based on the similarity $s(x,n)$ between pixel at position x and neighbouring pixel at n.

$$g(x) = \frac{\sum_{n \in N(x)} f(n)\ s(x,n)}{\sum s(x,n)}$$

- Similarity $s(x,n)$ can be measured in colour, position, etc.
- Examples:

$$s(x,n) = e^{-(\frac{f(x)-f(n)}{K})^2}, \quad s(x,n) = \frac{1}{1+(\frac{f(x)-f(n)}{K})^2}$$

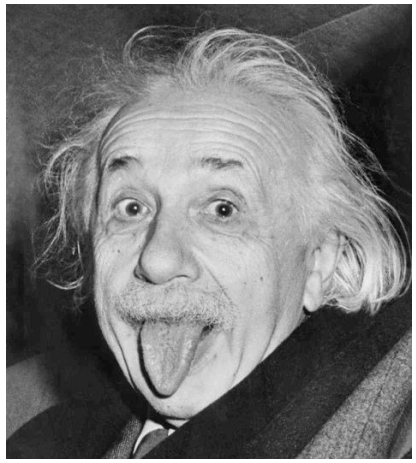
- Problem: Not shift-invariant, but different kernels at different positions $\Longrightarrow$ Impossible to analyse in frequency domain.

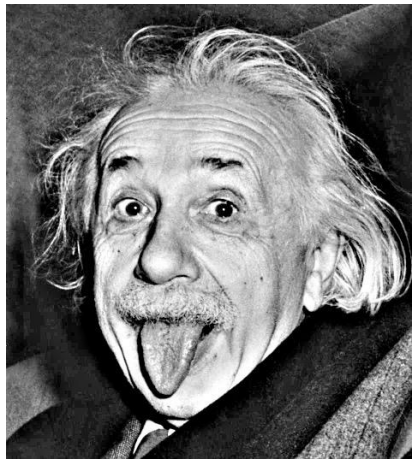
Anisotropic smoothing with bilateral filtering

Note: the image is smoother, but individual hairs are not blurred out.

Sharpening

- Purpose: Enhance local contrast, highlight fine details.
- Methods:
  - Unsharp masking (spatial)
  - Highpass filtering (spectral)
  - Differentiation (image derivatives)
- Common desirable property:
  - Isotropy (rotational invariance)
- Common problems:
  - Highpass filtering enhance noise.





- Difference compared to grey-level transformations:
  - Spatial variations are taken into account.
  - Changes the image only around edges, the rest is the same.

Unsharp masking

- Idea: “subtract out the blur”
- Blur image → subtract from original → weight the difference → add to original

$$g(x,y) = f(x,y) + \alpha(f(x,y) - \bar{f}(x,y))$$

$$g = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} + 4\left(\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} - \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}\right) = \frac{1}{4}\begin{pmatrix} -1 & -2 & -1 \\ -2 & 16 & -2 \\ -1 & -2 & -1 \end{pmatrix}$$

Sharpening with highpass filters

- Sharpening with a high-pass filter:

$$G(u,v) = F(u,v) + \alpha(H_{hp}(u,v)F(u,v))$$

- Quite similar to unsharp masking, but in Fourier domain.

$$H_{hp}(u,v) = 1 - H_{lp}(u,v)$$

- Ideal:

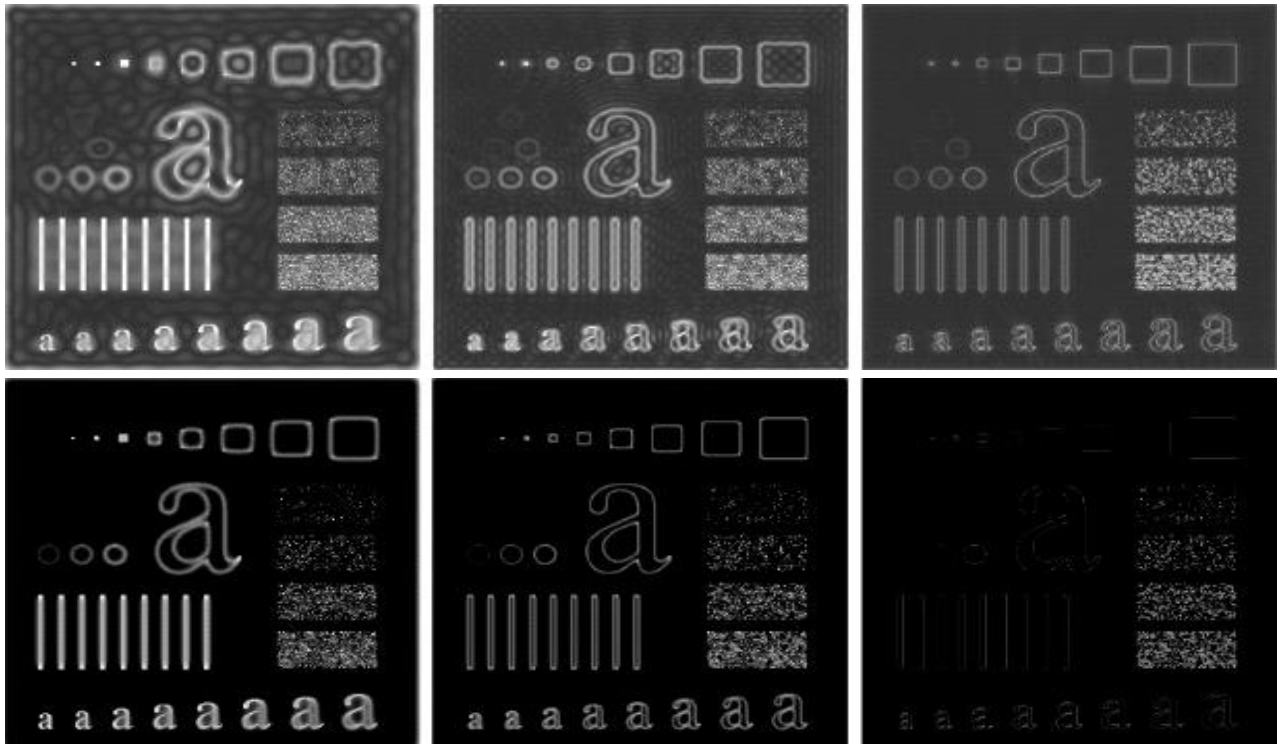
$$H_{hp}(u,v) = \begin{cases} 1 & \text{if } (u^2+v^2) > D_0^2 \\ 0 & \text{if } (u^2+v^2) \leq D_0^2 \end{cases}$$

- Gaussian:

$$H_{hp}(u,v) = 1 - e^{-\sigma^2(u^2+v^2)/2}$$

Highpass filters

Ringing



Results with ideal (top) and Gaussian (bottom) filters.

Image derivatives (first order)

- How do you normally detect changes in a function? Compute derivatives!

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h}$$

- But for discrete images h has to be an integer. The best we can do is

$$f'(x) \approx \frac{f(x+1) - f(x-1)}{2}$$

- This corresponds to a filter kernel

δ_x $\frac{1}{2}$ | 1 | 0 | -1 |

Why does the 1 come first?

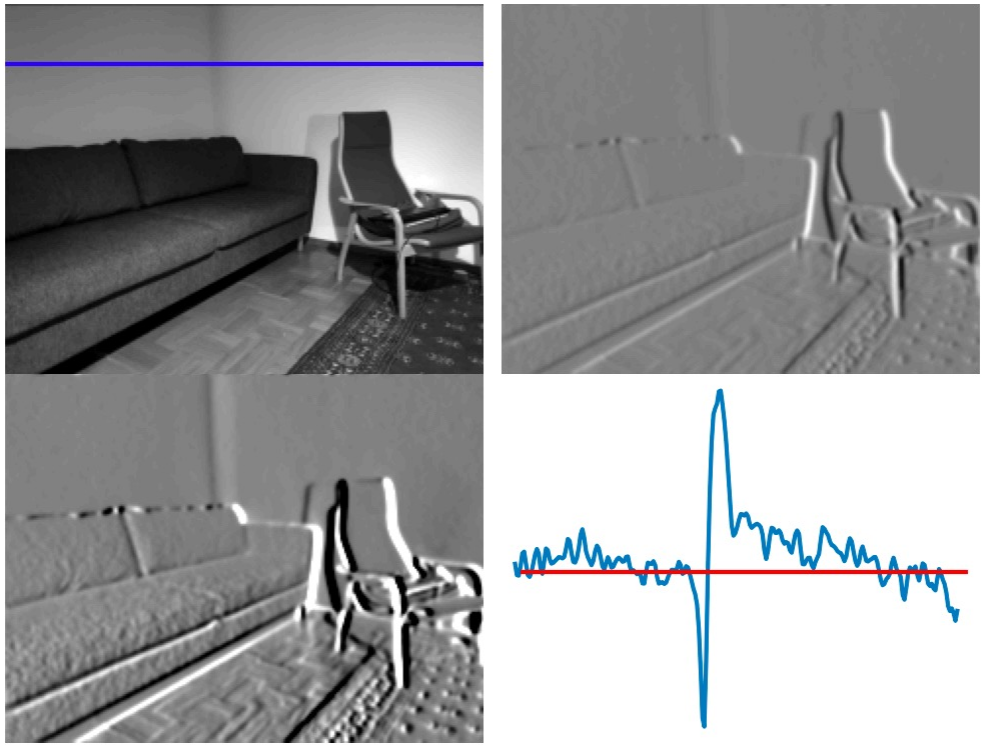
The Sobel filter

- Problem: A derivative is a highpass filter and will enhance noise.
- Solution: Add some smoothing, but do it in the opposite direction.

$$\frac{1}{8}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \frac{1}{2}\begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

- A derivative in the y-direction can be implemented correspondingly.

First order derivative (example)



- Along the line, the first order derivative varies due to the position of the light source. It is also quite noisy.

Image derivatives (second order)

- In images there are a lot of gradual changes in intensity, often due to shading. If you apply a second order derivatives, these changes should be close to zero.
- The best approximation of a second order derivative is

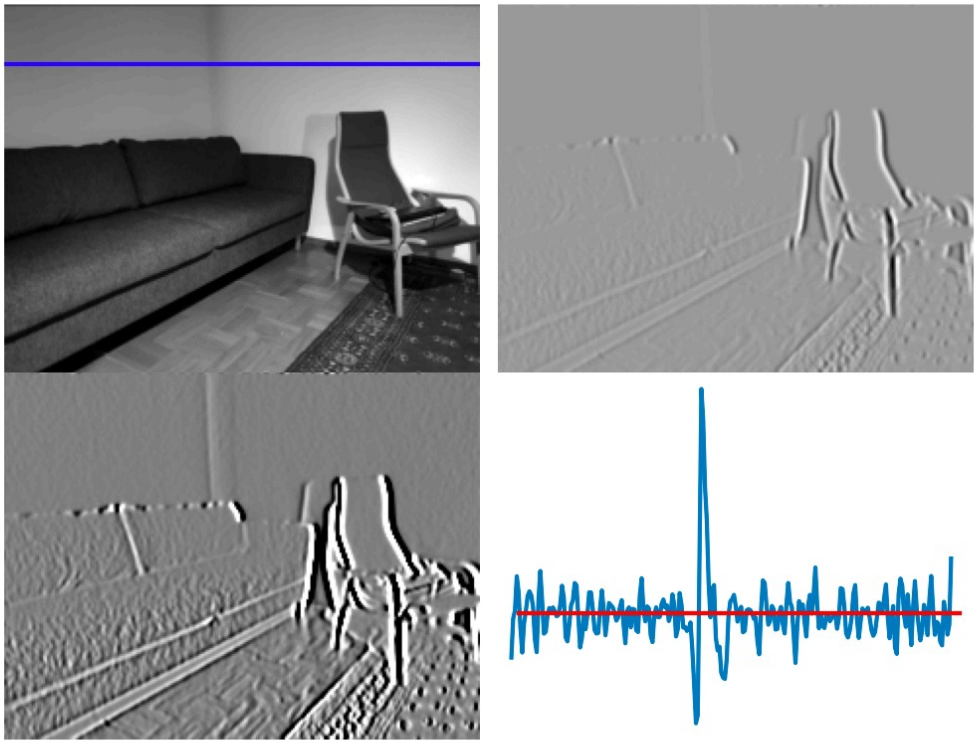
$$f''(x) \approx f(x+1) - 2f(x) + f(x-1)$$

- This corresponds to a filter kernel

δ_{xx}

1	-2	1

Second order derivative (example)



- Along the line, the second order derivative is close to zero everywhere except in the corner. However, it is even noisier.

Commonly used differential operators

- We apply a differential operator δ_x to an image f and get

$$f_x = \delta_x * f$$

- Gradient: $\nabla f = (f_x, f_y)$. often use to compute edge direction
  - First order, linear, non-isotropic
- Gradient magnitude: $|\nabla f| = \sqrt{f_x^2 + f_y^2}$ often used to detect edges
  - First order, non-linear, isotropic
- Laplacian: $\Delta f = f_{xx} + f_{yy}$ often used to detect blobs
  - 2nd order, linear, isotropic
- Isotropic means that the result does not depend on the orientation. It is good if we want to detect an edge and want it to behave similarly for all edge directions.

Fourier transform of a derivative

$$\mathcal{F}[f(x)] = \hat{f}(\omega)$$

$$\mathcal{F}[f_x(x)] = \lim_{h\to 0} \frac{\mathcal{F}[f(x+h)] - \mathcal{F}[f(x-h)]}{2h} = \lim_{h\to 0} \frac{\hat{f}(\omega)e^{i\omega h} - \hat{f}(\omega)e^{-i\omega h}}{2h} =$$

$$= \lim_{h\to 0} \frac{\hat{f}(\omega)\cdot 2i\ sin(\omega h)}{2h} = i\omega\hat{f}(\omega)\ \lim_{h\to 0} \frac{sin(\omega h)}{\omega h} = i\omega\hat{f}(\omega)$$

- A perfect differential operator (approximation of a derivative) should thus have a Fourier transform equal to $i\omega$ (in the continuous domain).

Laplacian operator

- The simplest possible Laplacian operator:

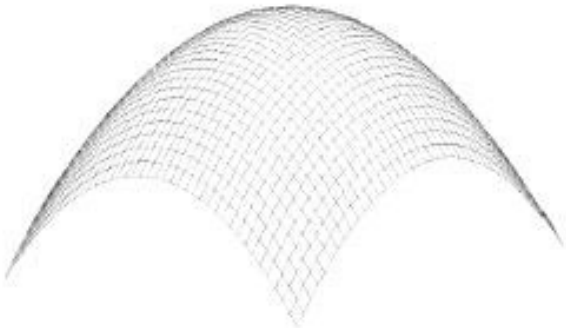
$$f(x,y) = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Isotropic?

$$\hat{f}(\omega_x, \omega_y) = e^{-i\omega_x} + e^{i\omega_x} + e^{-i\omega_y} + e^{i\omega_y} - 4 = 2\cos(\omega_x) + 2\cos(\omega_y) - 4 =$$
$$= 2(1 - \omega_x^2/2 + \mathcal{O}(\omega_x^4)) + 2(1 - \omega_y^2/2 + \mathcal{O}(\omega_y^4)) - 4 = -(\omega_x^2 + \omega_y^2) + \mathcal{O}(\omega_x^4, \omega_y^4)$$

Isotropic!

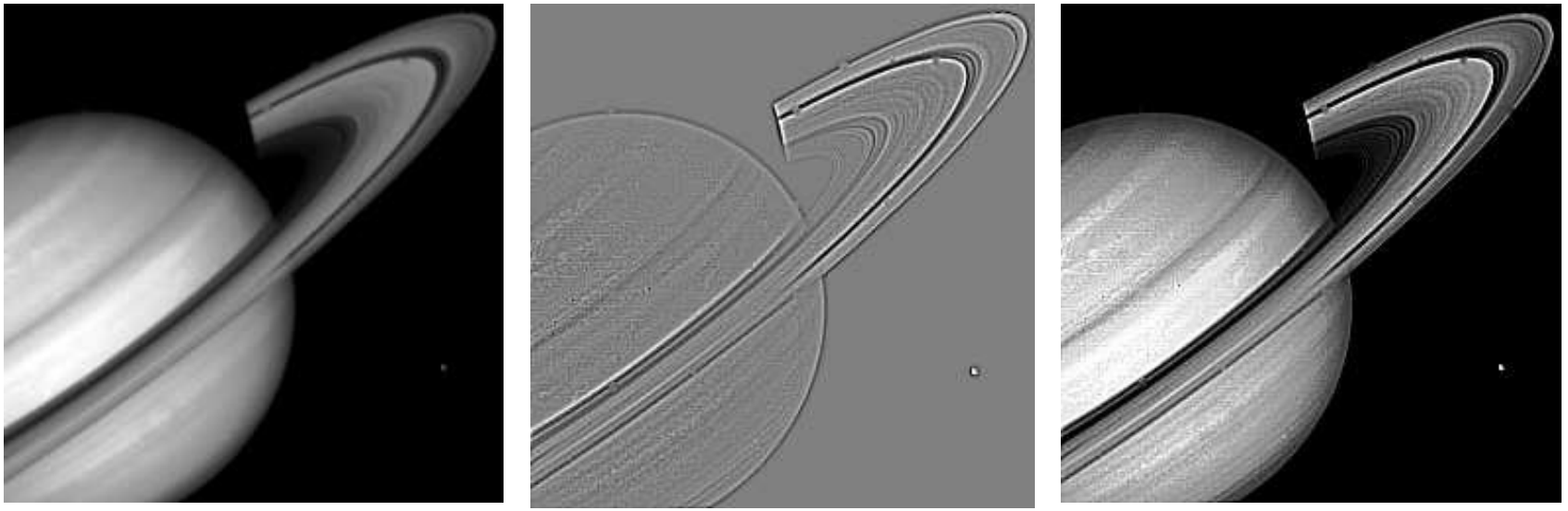
- With larger kernels, the approximation can be even better.



Sharpening using Laplacian operator

$$g(x,y) = f(x,y) - \nabla^2 f(x,y)$$

$$\hat{g}(\omega_x,\omega_y) = \hat{f}(\omega_x,\omega_y) + (\omega_x^2+\omega_y^2)\hat{f}(\omega_x,\omega_y) = (1+\omega_x^2+\omega_y^2)\hat{f}(\omega_x,\omega_y)$$



Original image (left), application of Laplacian operator (middle), and subtraction of the Laplacian from the original image (right).

Summary of good questions

- Mention a typical grey-level transformation. When to use it?
- What do histogram stretching and compression mean?
- What are the principles of histogram equalization?
- What are the differences between lowpass, bandpass and highpass filters?
- Why are ideal lowpass filter rarely used in practice?
- What characteristics does a Gaussian filter have?
- What is the difference between mean and median filters?
- How can you do sharpening?
- How can you approximate a first order derivative?
- What is a Laplacian?

Recommended reading

- Gonzalez & Woods: Chapters 3.2 - 3.3, 3.5 - 3.6
- Szeliski: Chapters 3.1, 3.3.1 - 3.3.2