

This document has been published under a Creative Commons - Attribution-NonCommercial-ShareAlike (CC by-nc-sa). The conditions of the licence can be found [here](#).



⊕_⊕ Find any errors? Please send them back, I want to keep them!}

- Image Analysis: “low to mid level”, “early” vision
 - image to image (image processing/enhacement)
 - image labeling (segmentation, depth, motion, features)
- Computer Vision: “high level”, scene interpretation
 - image to model (3D, texture, lighting)
 - recognition (objects, activities, ...)

1 Image Types and Discretization

- mapping f from a rectangular domain $\Omega = (0, w) \times (0, h)$ to a co-domain \mathbb{R}
$$f : \mathbb{R}^2 \supset \Omega \rightarrow \mathbb{R}$$
 - Ω is called *image domain* or *image plane*
- image data only given on rectangular point grid within image domain Ω
- grid point (i, j) is called *pixel* (picture element)
- **Aliasing:** Signals become indistinguishable when sampled
 - Sampling rate must be above *Nyquist rate* (two times the bandwidth of the signal) => aliasing-free result

1.1 Image Noise

1.1.1 Additive Noise

- most important noise
- Gray values and Noise are *independent*

$$f_{i,j} = g_{i,j} + n_{i,j}$$

where g is the original image, n is the noise and f is the noisy image

- noise n may have different distributions

- Uniform Distribution (very simple)

$$p(x) = \begin{cases} b - a & \text{for } x \in [a, b] \\ 0 & \text{else} \end{cases}$$

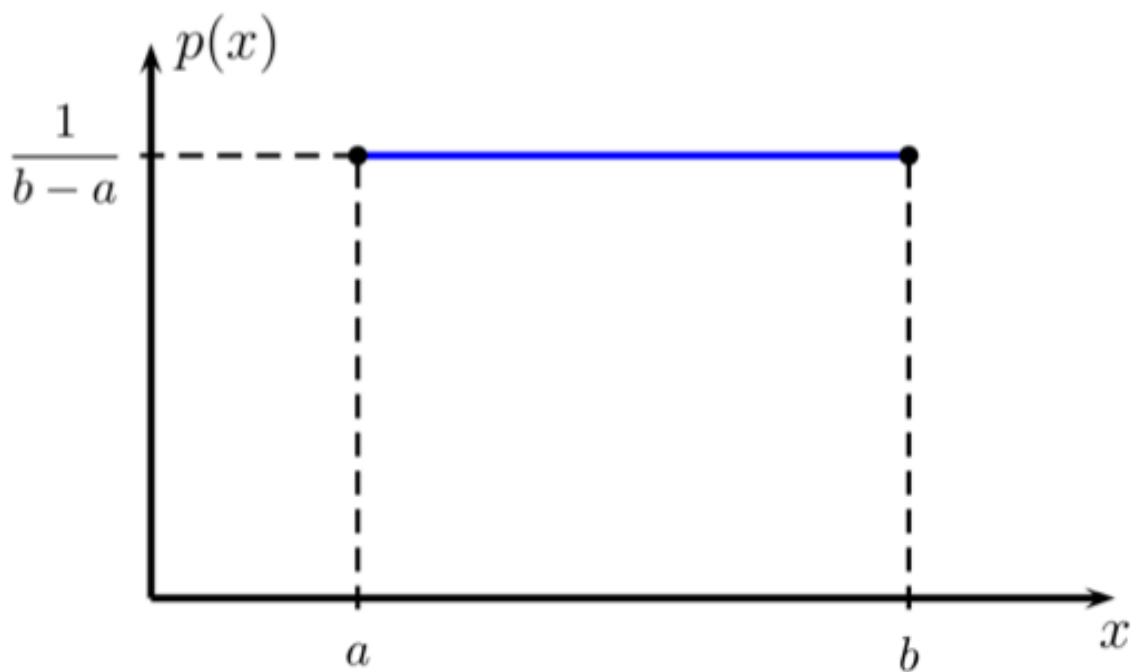


Abbildung 1: Uniformly Distributed Noise

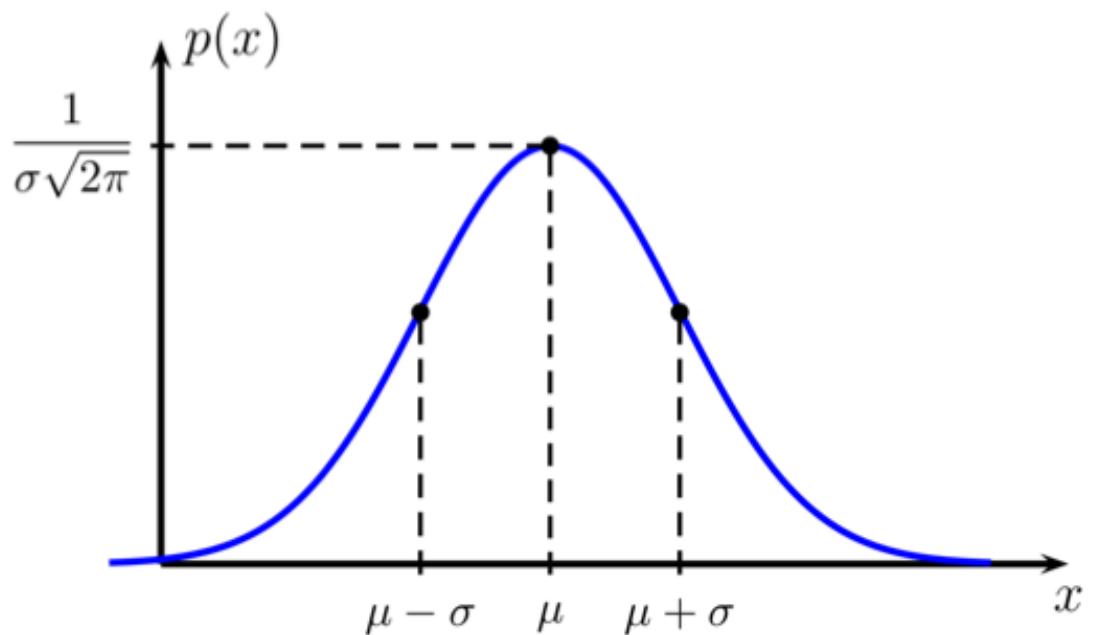


Abbildung 2: Gaussian Noise Distribution

- Gaussian Distribution (very common)
 - * most important noise model
 - * good approximation for many practical situations
 - * density function:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where μ is the *mean* and σ is the standard deviation (σ^2 is called the *variance*)

- * $\mu - \sigma \Rightarrow 68\%$
- * $\mu - 2\sigma \Rightarrow 95.5\%$
- * $\mu - 3\sigma \Rightarrow 99.7\%$

1.1.2 Multiplicative Noise

- signal dependent
- often proportional to the gray value:

$$f_{i,j} = g_{i,j} + n_{i,j} \cdot g_{i,j} = (1 + n_{i,j}) \cdot g_{i,j}$$

- often present in radar, ultrasound and tomographic images

1.1.3 Impulse Noise

- degrade image at some pixels only (erroneous grey values are created)
- *unipolar impulse noise* \Rightarrow only one grey value
- *salt-and-pepper noise* \Rightarrow bipolar noise, highest and lowest value inserted

1.2 Correlation and Convolution

- goal is to reduce noise
- take multiple images with same camera (and setup) and average them
- **Image Filtering:** modify pixels based on some function of local neighborhood



Abbildung 3: Image neighborhood

- replace pixel by average of neighborhood

1.2.1 Correlation

- weighted combination of pixels in small neighborhood, e.g. weighted sum:

$$g_{i,j} = \sum_{k,l \in \mathbb{Z}} f(i+k, j+l) h(k, l)$$

where $h(k, l)$ is the **mask** or **weighted kernel** (*filter coefficients*)

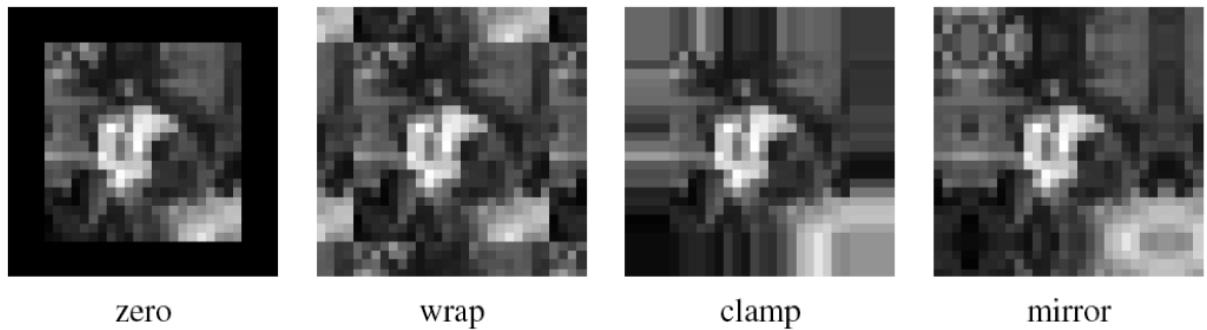


Abbildung 4: Boundary Conditions

- **correlation operator:** $g = f \otimes h$
- *boundary conditions:* What happens on image border.
 - *zero*: everything outside of images is zero
 - *wrap*: move out of image? move back in on the other side!
 - *clamp*: last pixel gets replicated
 - *mirror*: move out of image? move back on the same side, reverse direction
- **Gaussian Correlation:** Removes high-frequency components from the image (low-pass filter)

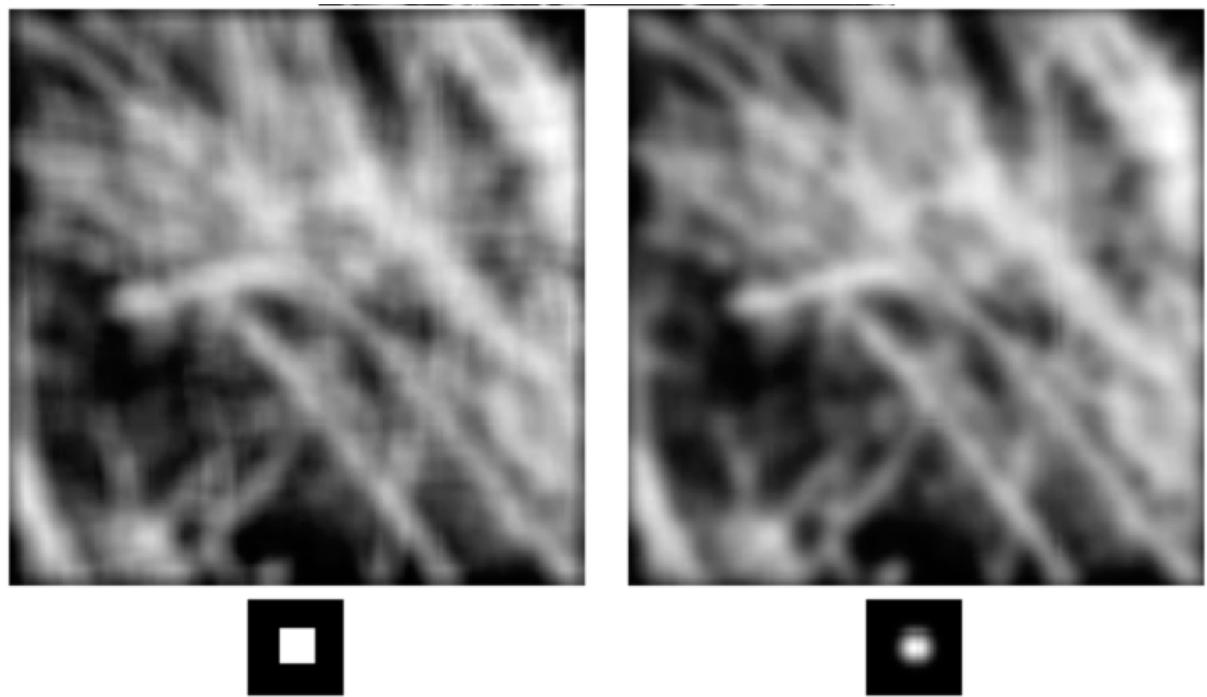


Abbildung 5: Filtering: mean (left), gaussian (right)

- approximates a 2D Gaussian function
- Gaussian Function has infinite support, discrete filters use finite kernels
- *Standard Deviation* determines extent of smoothing

1.2.2 Convolution

- Correlation filters will be flipped by mathematical operation

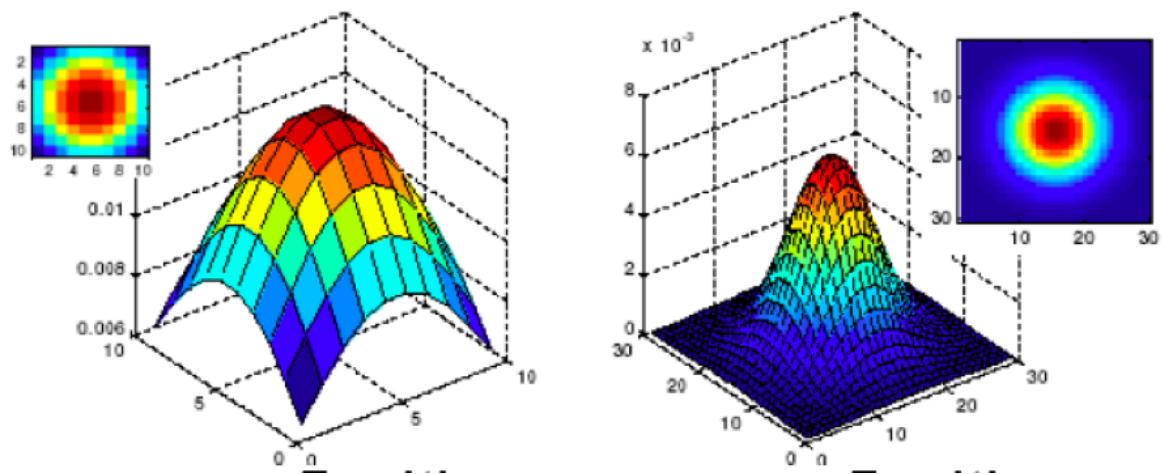


Abbildung 6: Gaussian Kernel $\sigma = 5$: 10×10 kernel (left), 30×30 kernel (right)

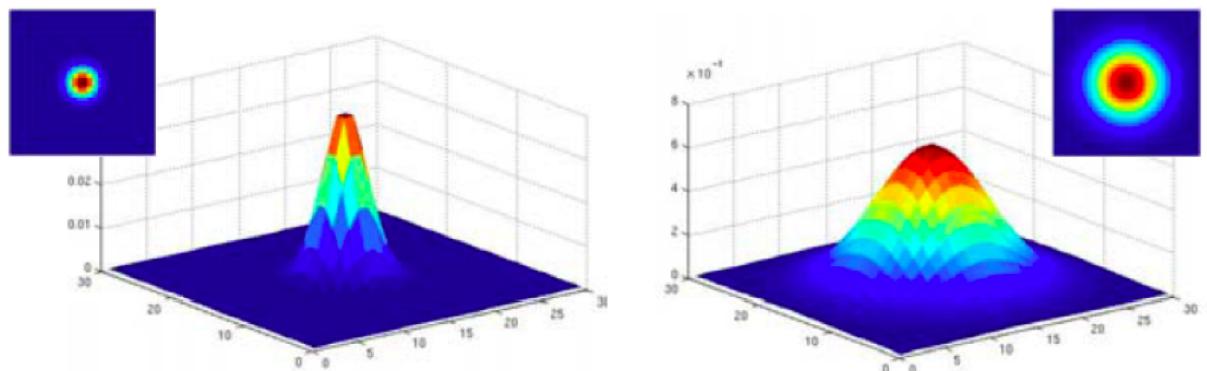


Abbildung 7: Gaussian Kernel: $\sigma = 2$ (left), $\sigma = 5$ (right), 30×30 kernel

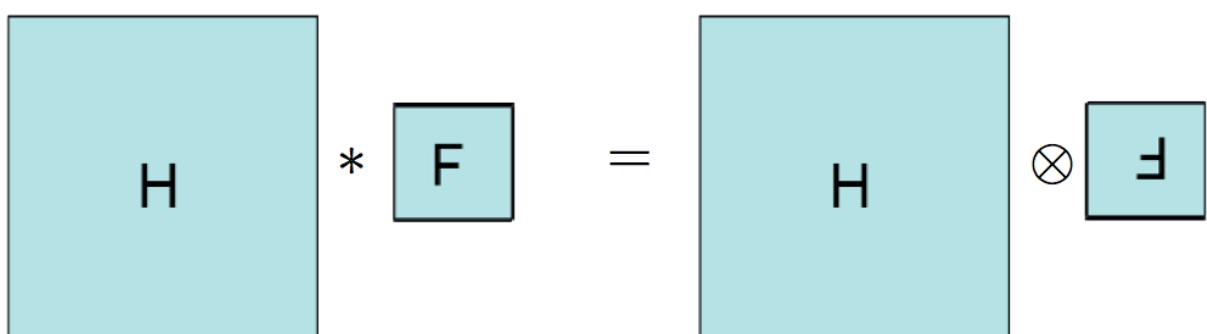


Abbildung 8: Correlation and Convolution

- Convolutions of two images: $g = f * h$ defined as:

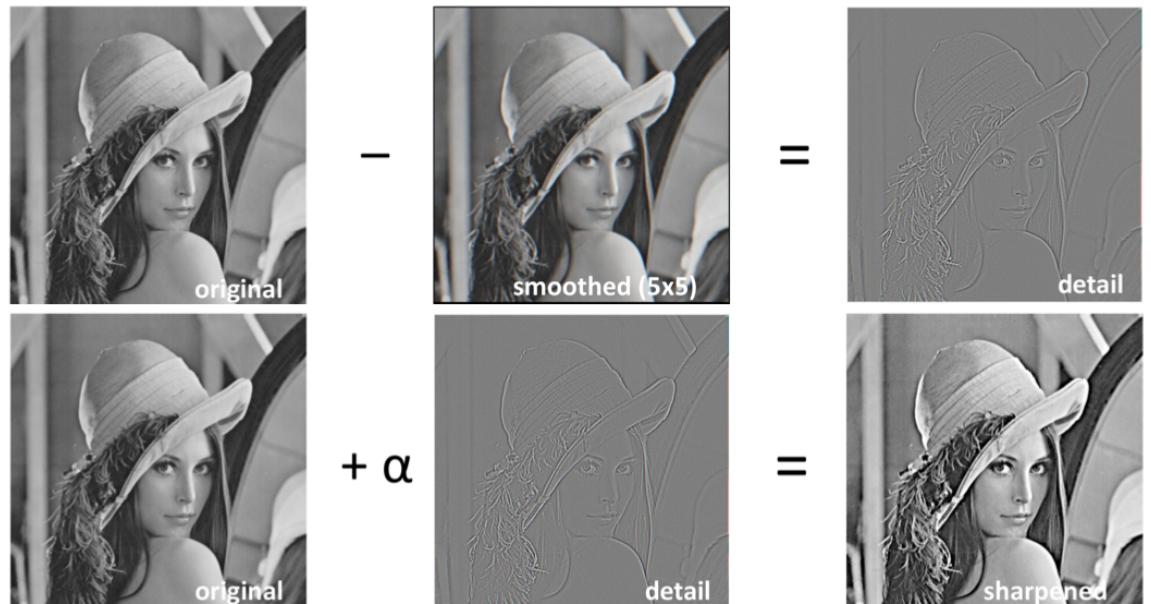
$$g(i, j) = \sum_{k, l \in \mathbb{Z}} f(i - k, j - l)h(k, l)$$

- for symmetrical filters (gaussian, box) result is the same
- Visualization of Correlation filters much more intuitive.
- mathematical properties:

- **commutative**: $f * h = h * f$
- **associative**: $(f * h) * g = f * (h * g)$
- **identity**: $\delta * h = h * \delta = h$
- **linear** (superposition):

$$\begin{aligned} f * (h_1 + h_2) &= f * h_1 + f * h_2 \\ (f + g) * h &= f * h + g * h \\ (\alpha f) * h &= \alpha(f * h) \end{aligned}$$

- **shift-invariant**: $(shifted\ f) * h = (f * h)shifted$ (shifted image => convolution with shifted impulse)



- **Sharpening**
- Different phenomena in Optics are actually convolutions, e.g. motion blur

1.2.3 Non-Linear filters + denoising

- Measures for denoising quality:
- Mean Square error (MSE):

$$MSE(f, g) = \frac{1}{N} \sum_{n=1}^N (f_n - g_n)^2$$

- Peak signal-to-noise ratio (PSNR)

$$PSNR(f, g) = 10 \log_{10} \left(\frac{V^2}{MSE(f, g)} \right)$$

- **bilateral filter**:

- idea: preserve edges, average only over pixels which are nearby *and* have similar color

- two weights: one for distance, one for intensity, both are gaussian filters
- parameters $\sigma, \tau > 0$, greyscale image f , *bilateral filter*:

$$g(n) = \frac{1}{K_n} \sum_{m=1}^N f_m N_\sigma(\|x_n - x_m\|) \cdot N_\tau(|f_n - f_m|)$$

$$K_n = \sum_{m=1}^N N_\sigma(\|x_n - x_m\|) \cdot N_\tau(|f_n - f_m|)$$

- **non-local means**: images have similar patches, **idea**: when averaging, put most weight on similar patches
 - around each pixel n , define a $k \times k$ window of pixels \mathcal{W}_n , which is written as vector of pixel indices
 - define distance
- $d_{n,m}^2 := \sum_{s=1}^{k^2} (f(\mathcal{W}_n(s)) - f(\mathcal{W}_m(s)))^2$
- kernel weights: $k_{n,m} := e^{-\frac{d_{n,m}^2}{2\sigma^2}}$, where $\sigma > 0$ is a parameter.
 - sums are usually restricted to window around n th pixel
 - very slow filter
- **median filter**: very good for removing outliers => *salt-and-pepper noise*
 - for each pixel n , compute vector f consisting of k^2 grayscale values: $f = (f_{n,1}, \dots, f_{n,k^2})$ in local $k \times k$ window around n .
 - filtered result is the *median value*
 - usually $k = 3$, better to apply it multiple times (twice)

2 Features and Patterns

2.1 Image Edges and Image Derivatives

- Identify sudden changes (discontinuities) in an image
- naive: changes in intensity
 - often does not work: texture, shadows, back- and foreground of similar color, ...
 - => focus on gradient-based edge detection ok for now
- **approximation of the gradient**

- *interpolate image as function, differentiate, resample*:

$$\delta_x f(x, y) = \lim_{\epsilon \rightarrow 0} \frac{f(x + \epsilon, y) - f(x, y)}{\epsilon}$$

- *difference approximation*: h_x is pixel size in x -direction (usually normalized to 1)

- * *forward differences*: $(\partial_x f)_{i,j} \approx \frac{f(i+1,j) - f(i,j)}{h_x}$

- * *backward differences*: $(\partial_x f)_{i,j} \approx \frac{f(i,j) - f(i-1,j)}{h_x}$

- * *central differences*: $(\partial_x f)_{i,j} \approx \frac{f(i+1,j) - f(i-1,j)}{2h_x}$

- * can be written as *convolutions*:

- forward difference kernel:

1	-1	0
---	----	---

- backward difference kernel:

0	1	-1
---	---	----

- forward difference kernel:

$\frac{1}{2}$	0	$-\frac{1}{2}$
---------------	---	----------------

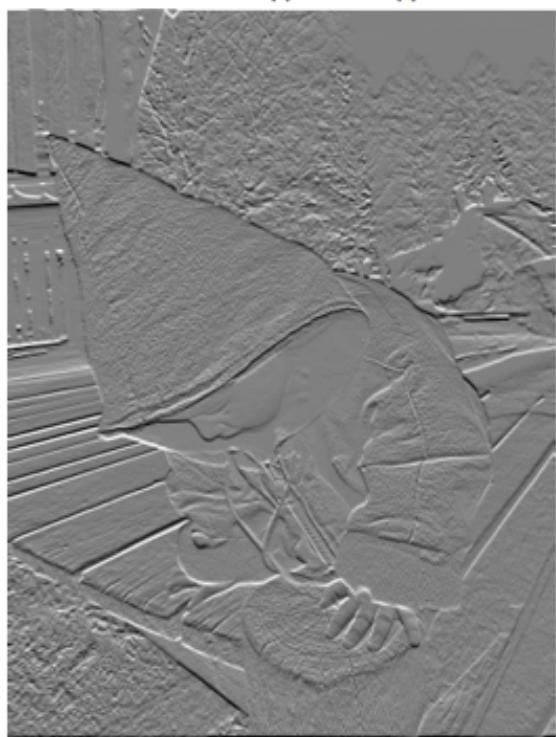
 f  $1 - \|\nabla f\|$  $\partial_x f$  $\partial_y f$

Abbildung 9: Central Difference Edge Detection

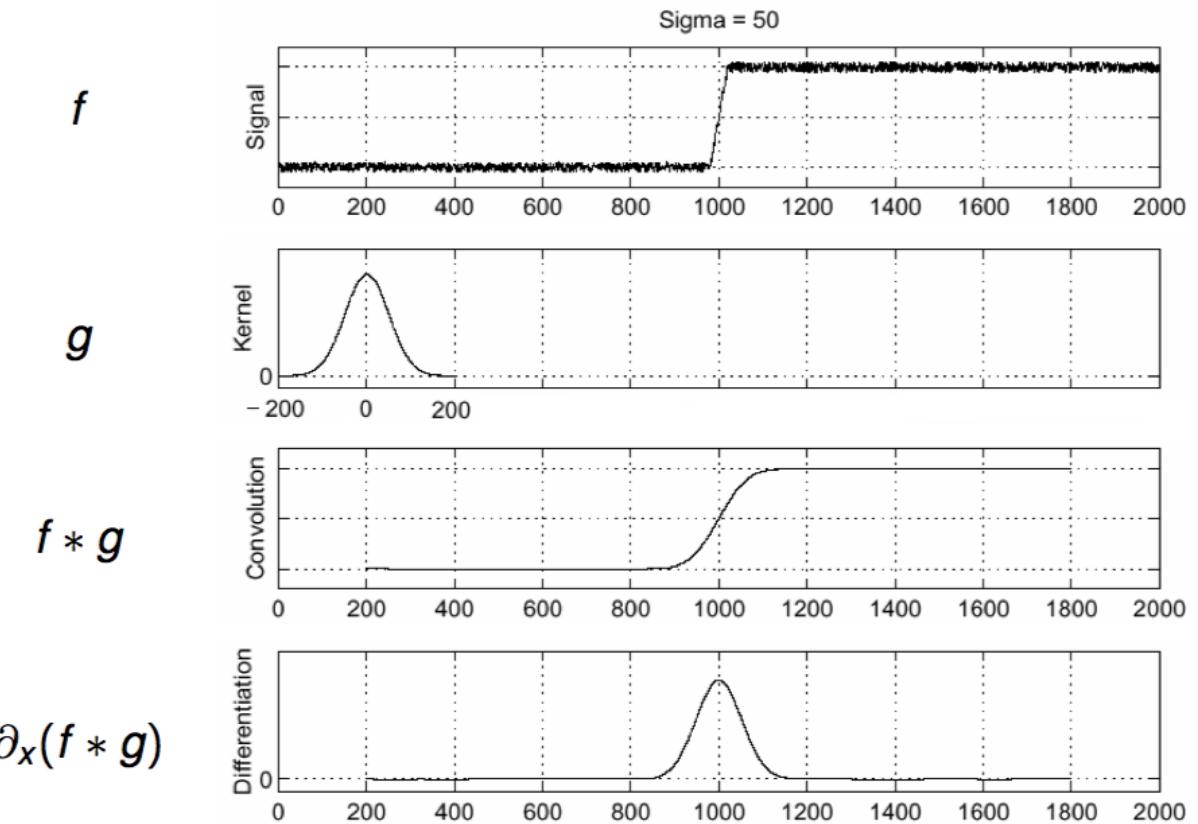


Abbildung 10: Smoothing, then building the derivative

- for greyscaleimage, gradient ∇f is a vector field consisting of partial derivatives: $\nabla f = \begin{bmatrix} \partial_x f \\ \partial_y f \end{bmatrix}$
- the magnitude of the gradient is the length of the vector in every point: $\|\nabla f\| = \sqrt{(\partial_x f)^2 + (\partial_y f)^2}$
- Noise makes this nearly impossible => smooth first, apply gradient later
- Under some technical conditions on f and g , partial derivatives wrt i th variable satisfy

$$\partial_i(f * g) = (\partial_i f) * g = f * (\partial_i g)$$

- * more *efficient*: can be pre-computed, no second convolution necessary
- * more *accurate*: if filter function is known, derivates can be computed analytically
- * normalize to zero after building filter => constant signal

2.2 Canny Edge Detection

- maxima of gradient
- not important for exam

2.3 Feature Detection

- *Detection*: identify interesting points
- *Description*: Extract Feature descriptor for each point
- *Matching*: Determine correspondences between views
- Goals for detectors:

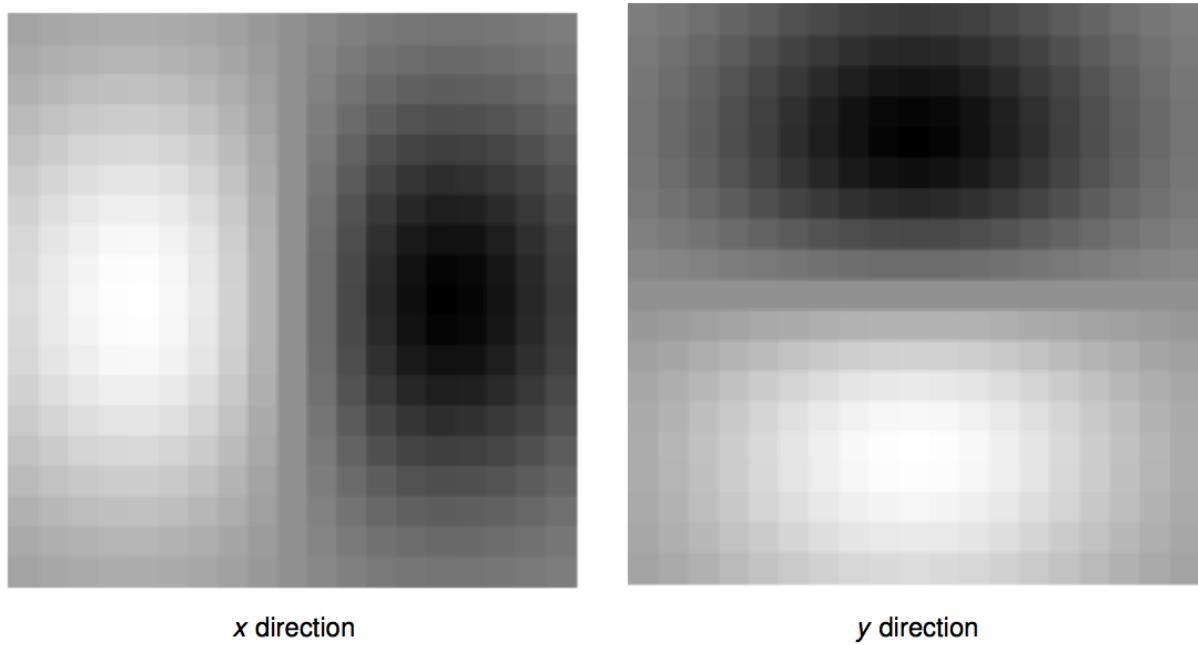


Abbildung 11: Derivatives Of Gaussian

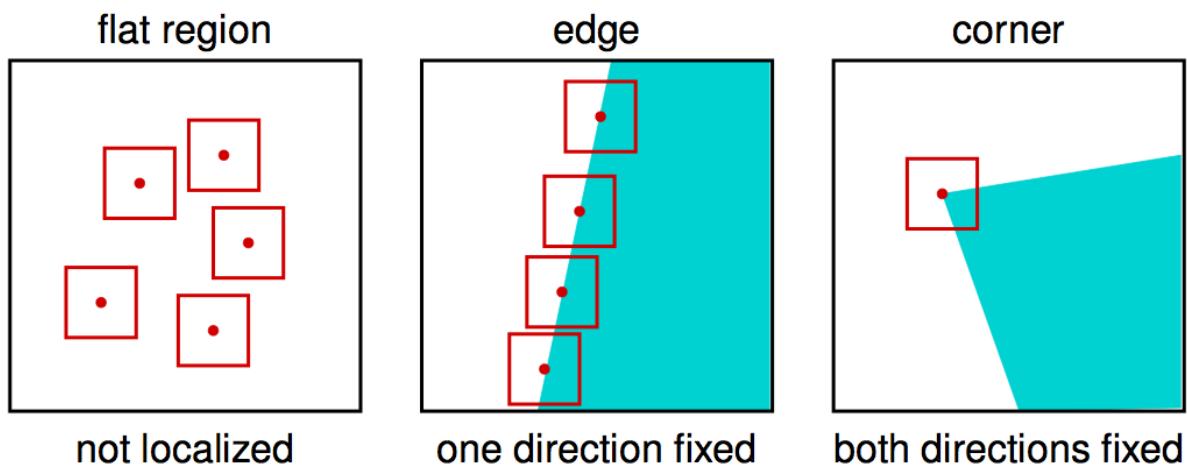


Abbildung 12: Localization

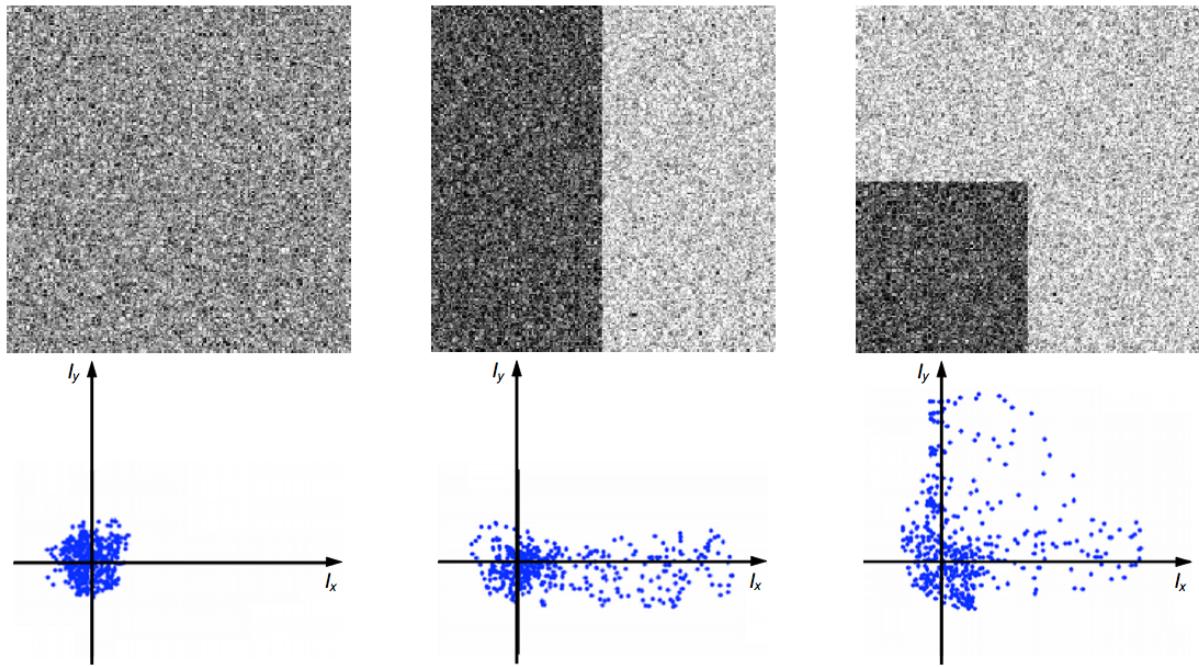


Abbildung 13: Mathematical Localization

- *locality*: robust to clutter and occlusion
- *quantity*: quantity: hundreds or thousands in single image
- *distinctiveness*: differentiate large database of objects
- *efficiency*: real-time performance?
- *geometric invariance*: translation, rotation, scale, ...
- *photometric invariance*: brightness, exposure, ...
- Look for **unusual** regions => unambiguous matches

2.3.1 Singular Value Decomposition (SVD)

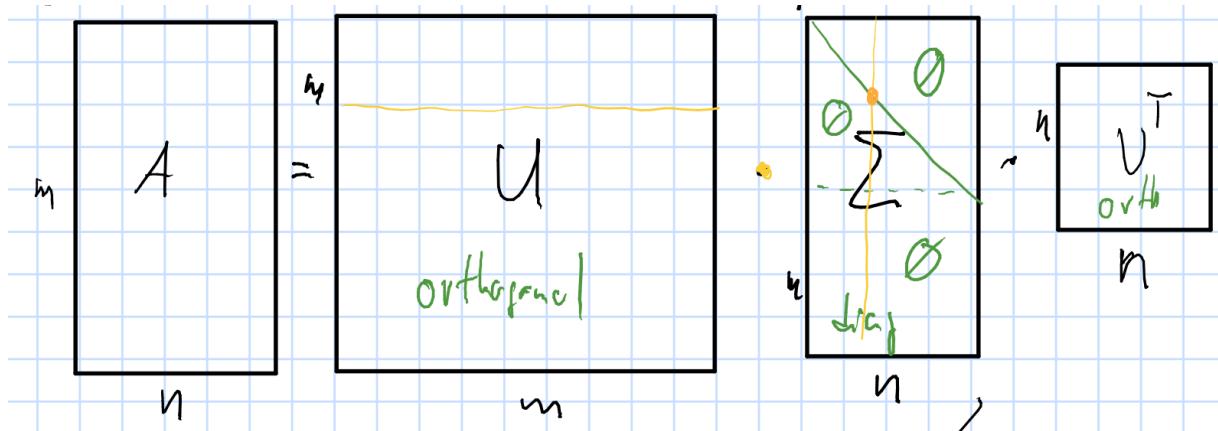


Abbildung 14: Singular Value Decomposition

- If $A \in \mathbb{R}^{m \times n}$ is a $m \times n$ matrix => there exists a factorization of the form $A = U\Sigma V^T$ where
 - $U \in \mathbb{R}^{m \times m}$ is an orthogonal matrix: $UU^T = U^T U = I_m$
 - $V \in \mathbb{R}^{n \times n}$ is also orthogonal

- $\Sigma \in \mathbb{R}^{m \times n}$ is a diagonal matrix, whose diagonal consists of sortet, non-negative real numbers
 $\sigma_1 \geq \sigma_2 \geq \dots \geq 0 \Rightarrow$ singular values of A
- $A = U\Sigma V^T \Leftrightarrow AV = \Sigma U$
 - A maps the columns of V to the columns of U scaled by the singular values
- **properties of SVD:**
 - $\text{rank}(A) \Rightarrow$ number of non-zero singular values
 - columns of V are eigenvectors of $A^T A$
 - columns of U are eigenvectors of AA^T
- **PCA:** Characterize joint probability distribution of several variables to find directions with greatest variance
 - sample expectation value: $E(Z) = \sum_{i=1}^m p_i z_i$ (\approx weighted average)
 - sample variance: $\text{Var}(Z) = \sum_{i=1}^m p_i (z_i - E(Z))^2$
 - sample standard deviation: $\sigma(Z) = \sqrt{\text{Var}(Z)}$
 - covariance: $\text{cov}(X, Y) := \sum_{i=m}^m p_i (x_i - E(X))(y_i - E(Y))$
 - * “centered measurements”: $E(X_j) = 0 \Rightarrow \text{cov}(X_j, Y_k) := \sum_{i=m}^m p_i \cdot x_{ji} \cdot y_{ki}$
 - * intuition: generally, whenever two values increase together, the covariance is positive
 - covariance matrix: $C(X) := (\text{cov}(X_{\underline{j}}, X_{\underline{k}}))_{\{j,k=1,\dots,n\}}$
 - * properties:
 - symmetric
 - $\text{cov}(X_j, X_j) = \text{var}(X_j) \Rightarrow$ diagonal consists of variances
 - semi-definite
- $M = U\Sigma V^T$ be SVD of measurement matrix \Rightarrow transform all measurements by $V \Rightarrow M' = MV = U\Sigma \Rightarrow$ covariance matrix: $C' = \Sigma^T U^T U \Sigma = \text{Sigma}^2$

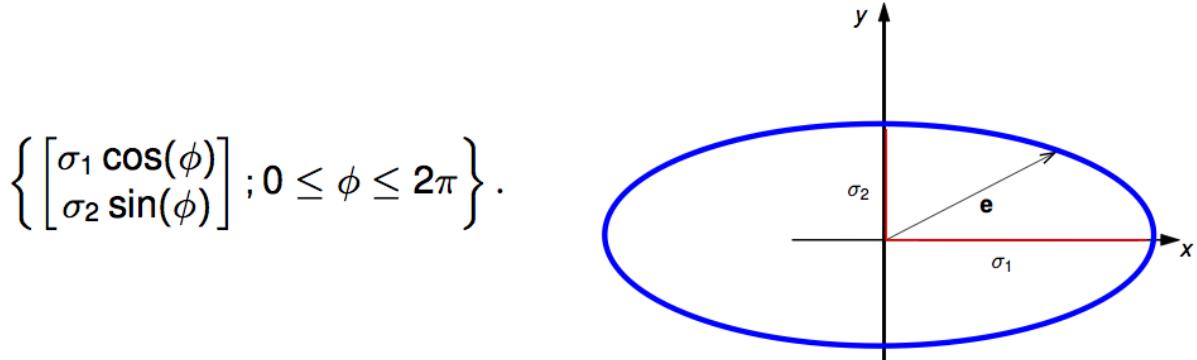


Abbildung 15: Standard Deviation Ellipse

- The radius of the ellipse yields the standard deviation of the measurements projected onto the corresponding direction.
- PCA can be used to *decorrelate data*
- dimensions corresponding to larger singular values have higher variance \Rightarrow more important

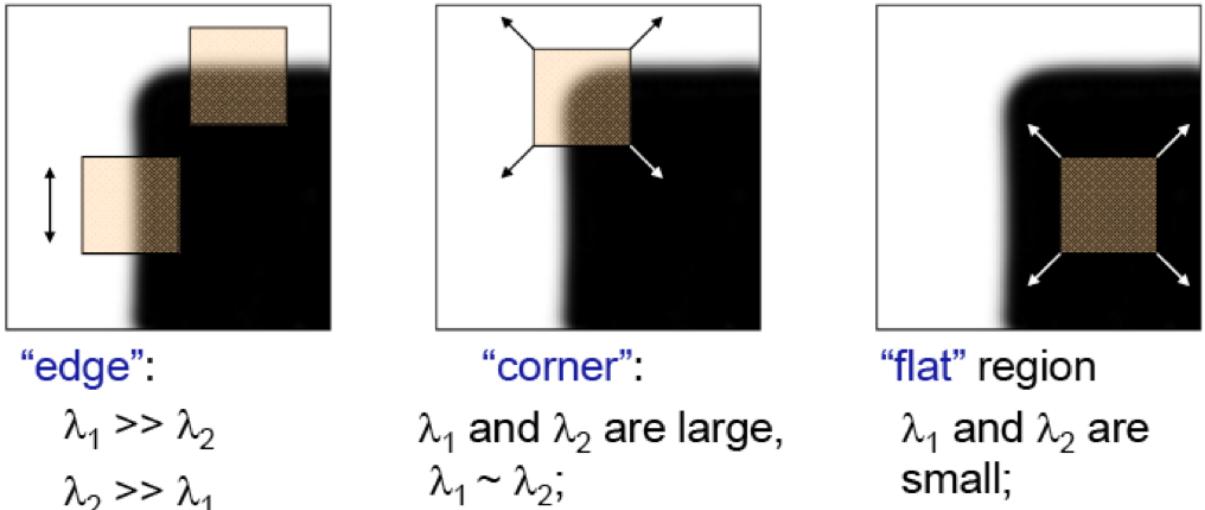


Abbildung 16: Pixel Characterization based on Eigenvalues

2.3.2 Structure Tensor

$$\tau := \begin{bmatrix} G_\tau * (f_x^\sigma)^2 & G_\tau * (f_x^\sigma f_y^\sigma) \\ G_\tau * (f_x^\sigma f_y^\sigma) & G_\tau * (f_y^\sigma)^2 \end{bmatrix} = G_\tau * \begin{bmatrix} (f_x^\sigma)^2 & f_x^\sigma f_y^\sigma \\ f_x^\sigma f_y^\sigma & (f_y^\sigma)^2 \end{bmatrix}$$

- *inner scale parameter* $\sigma > 0$: determines image scale at which derivates are taken
- *outer scale parameter* $\tau > 0$: determines size of the windows on which derivate information is sampled from

=> Harris Corner Detector

2.3.3 Compare Features in different images

- **Sum of Squared Differences (SSD)**: Given two rectangular image patches f, g on $\{1, \dots, W\} \times \{1, \dots, H\}$, the *sum of squared differences distance*:

$$E_{SSD}(f, g) = \sum_{i=1}^W \sum_{j=1}^H (f(i, j) - g(i, j))^2$$

- Smaller value => better fit
- normalization: divide by $W \cdot H$
- use error threshold θ to compare patches (for every possible matching patch)
- *properties*
 - fast
 - invariant to *translation*
 - *not* invariant to rotation
 - *not* invariant to scaling
 - *not* invariant to contrast/brightness changes
- **Normalized Cross-Correlation (NCC)** Given two rectangular image patches f, g on $\{1, \dots, W\} \times \{1, \dots, H\}$, the *normalized cross-correlation distance*:

$$E_{NCC}(f, g) = \frac{\text{cov}(f, g)}{\sigma(f) \cdot \sigma(g)}$$

- range is within $[-1, 1]$, larger value is better, value below 0 is opposite of fit.
- *properties*

- *not* fast
- invariant to *translation*
- *not* invariant to rotation
- *not* invariant to scaling
- invariant to contrast/brightness changes

- **Autocorrelation** for patch f and shift u (sum runs over all pixels)

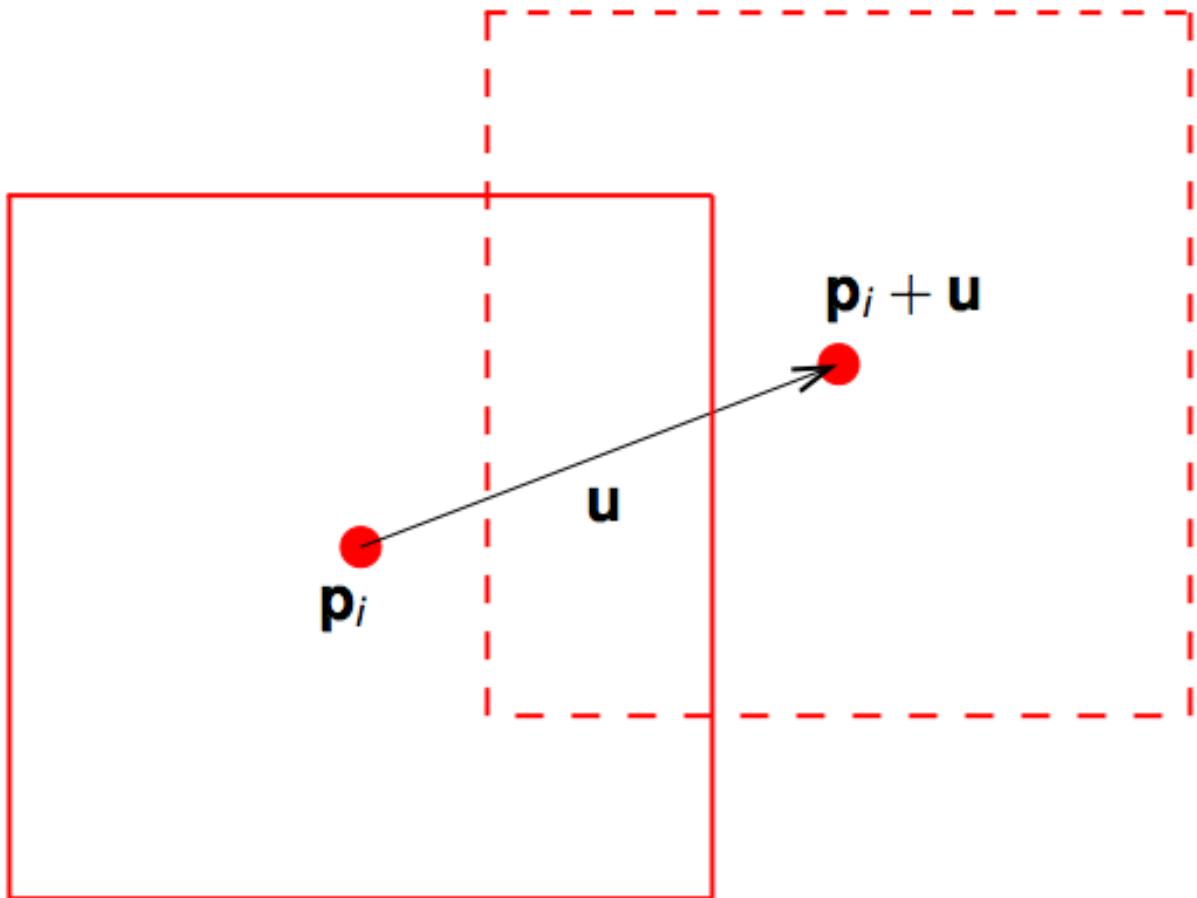
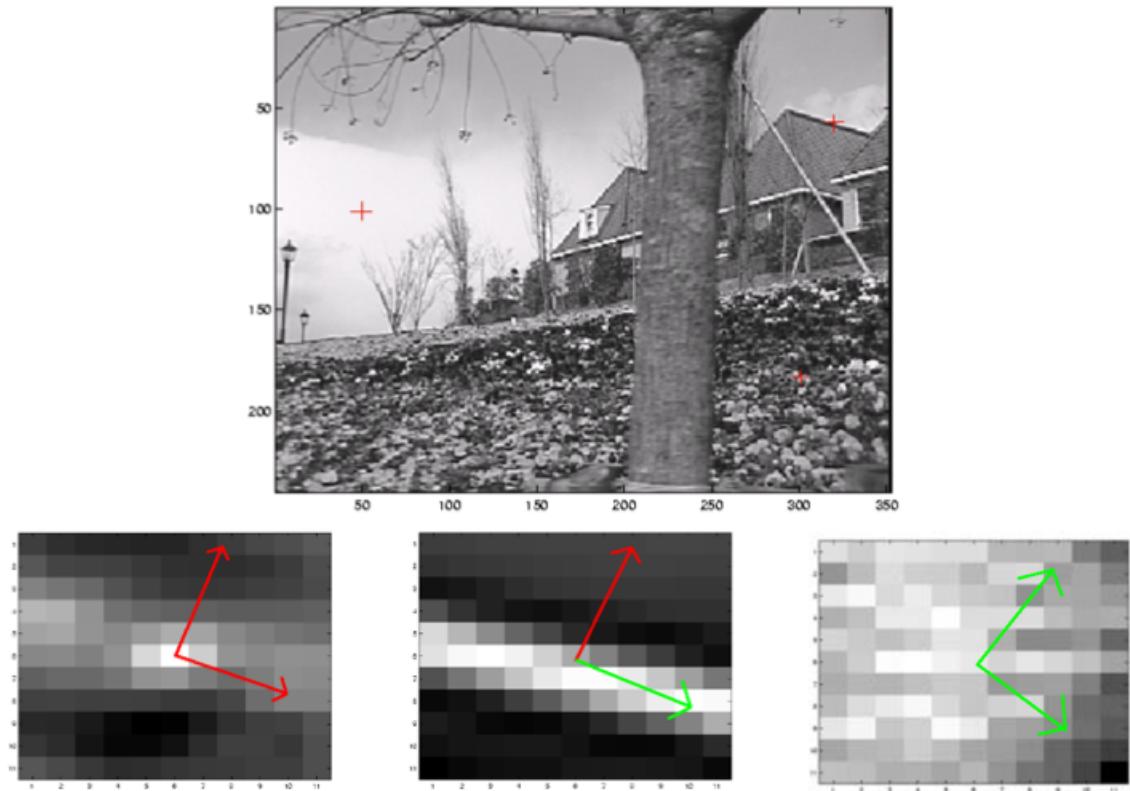


Abbildung 17: Autocorrelation: shifting

$$E_{AC}(f, u) := \sum_i w(p_i) (f(p_i + u) - f(p_i))^2$$



– algorithm:

1. taylor series approximation: $f(p_i + u) \approx f(p_i) + \nabla f(p_i)^T \cdot u$
2. compute E_{AC} using 1. $E_{AC}(f, u) \approx u^T (\Sigma_i w(p_i) \nabla f(p_i) \nabla f(p_i)^T) \cdot u$
 - * its actually the structure tensor:
 - * approximation to *autocorrelation* for small shifts $u \mapsto u^T \tau u$

3 Frequency

- Fourier's idea:

Any univariate function can be rewritten as a weighted sum of sines and cosines of different frequencies.

- Elementary building blocks: $f(x, y) = r \cdot \cos(2\pi(\omega_x x + \omega_y y) + \varphi)$
 - wave number: $\omega = [\omega_x \omega_y]$
 - frequency: $f = |\omega|_2 \Rightarrow$ number of peaks per unit length
 - direction of the wave: $\frac{\omega}{f}$
 - phase: $\varphi \in \mathbb{R} \Rightarrow$ gives distance of the first peak to the origin
 - amplitude: $r \geq 0 \Rightarrow$ gives maximum peak height
- write waves with complex numbers \Rightarrow formulas become easier to understand
 - $e^{it} = \cos(t) + i \cdot \sin(t)$
 - lies on the unit circle
 - angle between e^{it} and real axis is t
 - complex number $\Rightarrow z = r \cdot e^{i\varphi}$
 - * norm: $r = |z|$

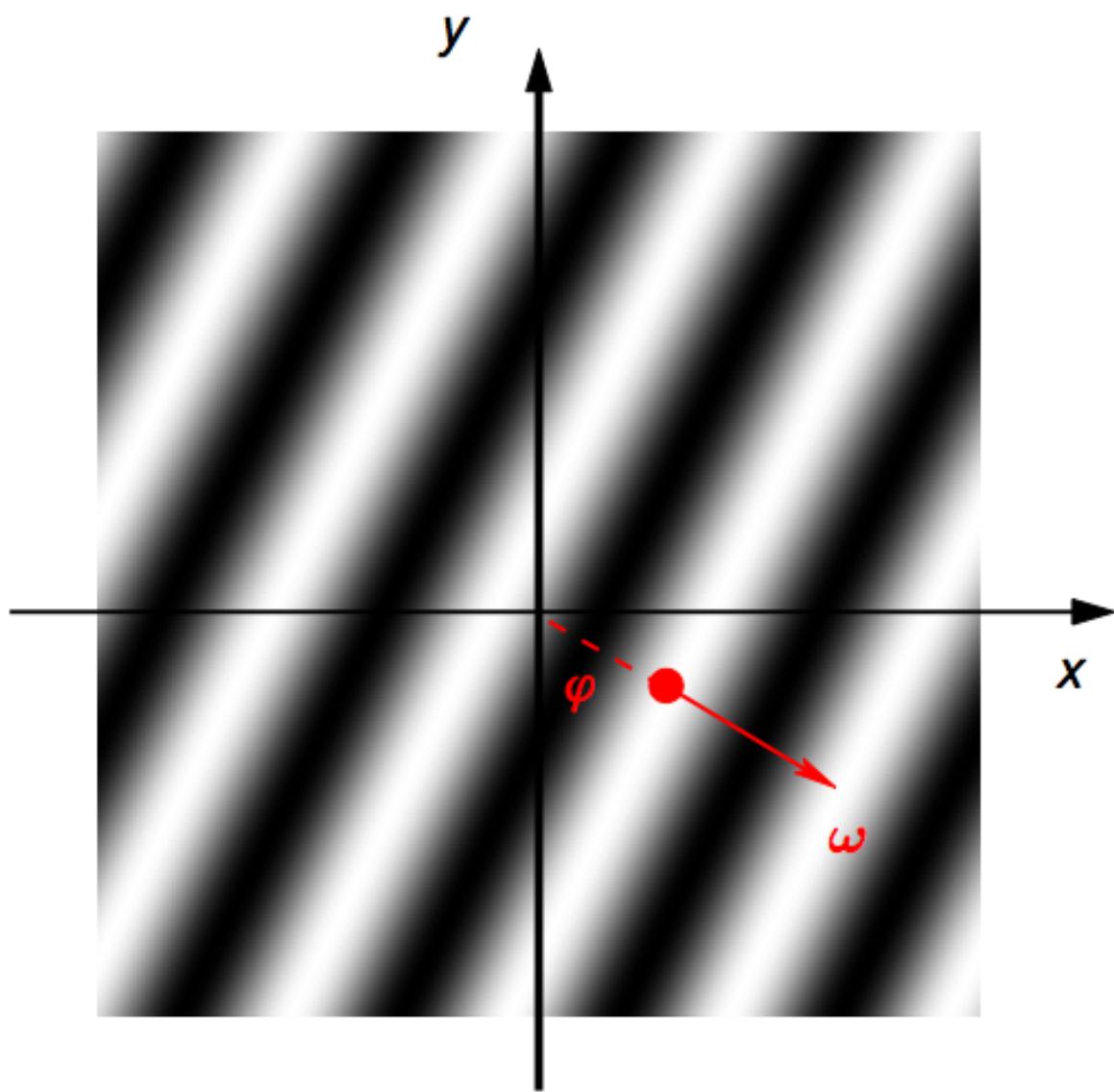


Abbildung 18: Waves in 2D

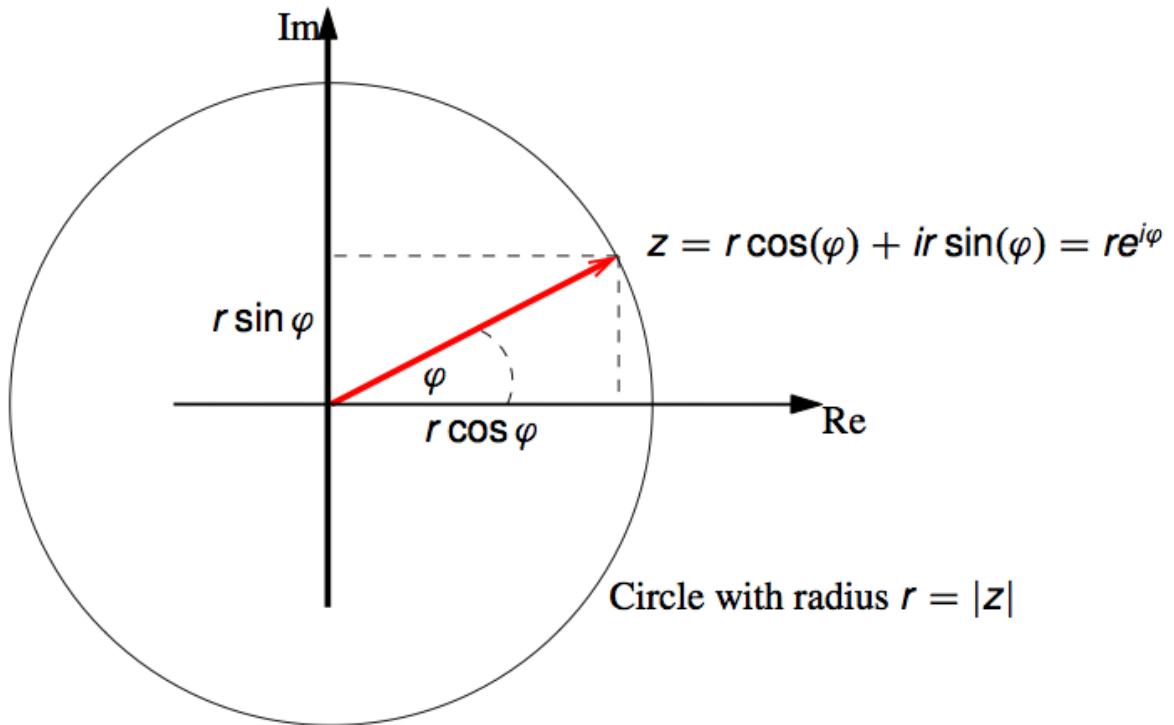


Abbildung 19: Polar representation of complex numbers

* argument φ

- complex elementary wave:

$$\begin{aligned} W_\omega(p) &= e^{2\pi i(\omega \cdot p)} \\ &= \cos(2\pi(\omega \cdot p)) + i \sin(2\pi(\omega \cdot p)) \end{aligned}$$

- multiplication with complex numbers:

$$\begin{aligned} W_\omega(p) &= r \cdot e^{i\varphi} \cdot e^{2\pi i(\omega \cdot p)} \\ &= e^{i(2\pi(\omega \cdot p) + \varphi)} \end{aligned}$$

amplitude r and phase φ

- Write function $f : \mathbb{R}^2 \rightarrow \mathbb{C}$ as *infinite linear combination* of elementary waves with complex coefficients $\hat{f}(\omega)$:

$$f(p) = \int_{\mathbb{R}^2} \hat{f}(\omega) W_\omega(p) d\omega$$

- Complex valued function $\hat{f} : \mathbb{R}^2 \rightarrow \mathbb{C}$ is called **Fourier Transform**

– For each point in frequency space, the value $\hat{f}(\omega) = re^{i\varphi}$ gives amplitude and phase of the elementary wave W_ω

- interpretation:

– Noise amplifies high frequencies:

- Phase seems more important for the actual image

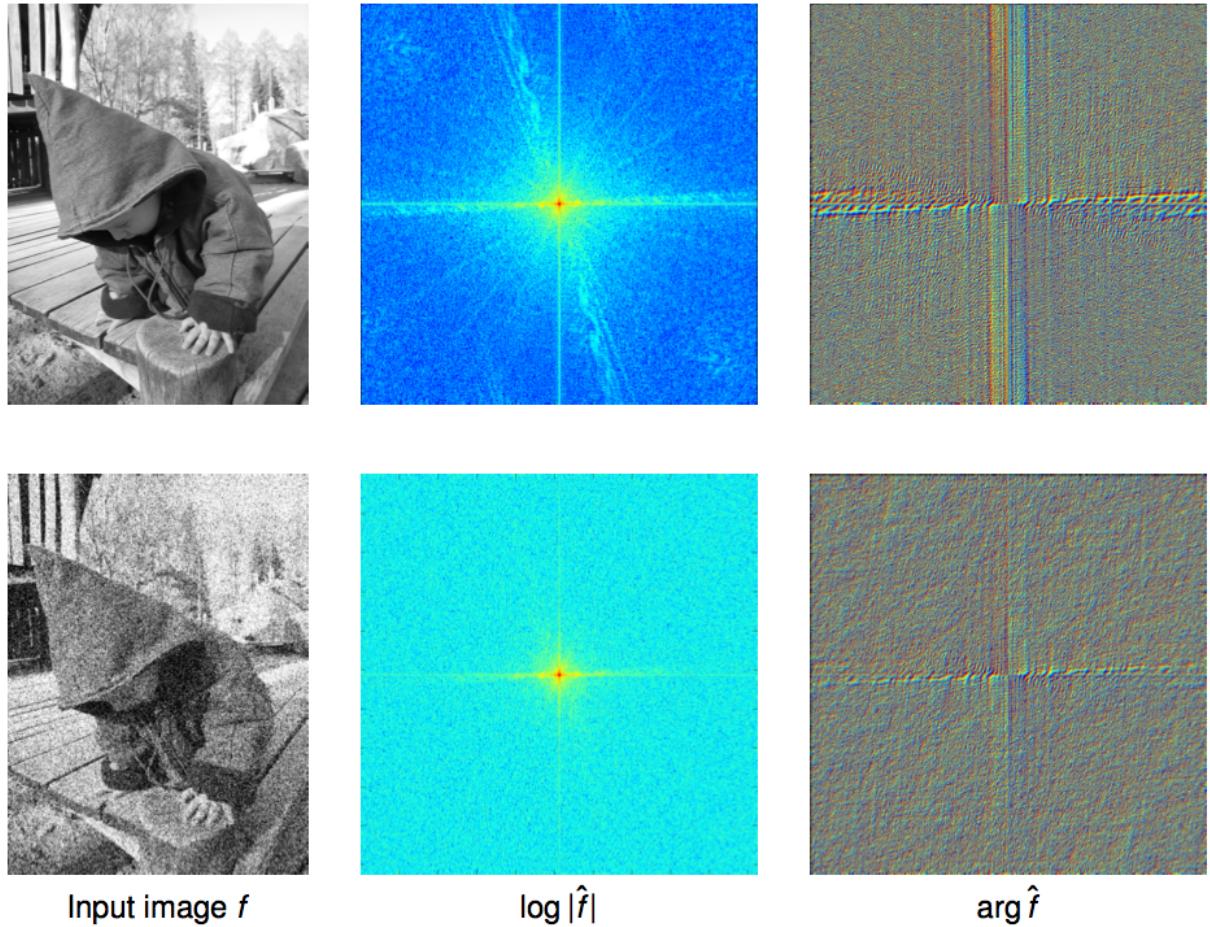


Abbildung 20: Noise and Fourier Transform

3.1 Filtering in Frequency Space

- convolution of elementary wave: $(g * W_\omega)(p) = W_\omega(p)\hat{g}(\omega)$
 - \Rightarrow convolution theorem: $\widehat{f * g} = \hat{f} \cdot \hat{g}$
 - much more efficient than direct convolution

3.1.1 Gaussian in Frequency Space

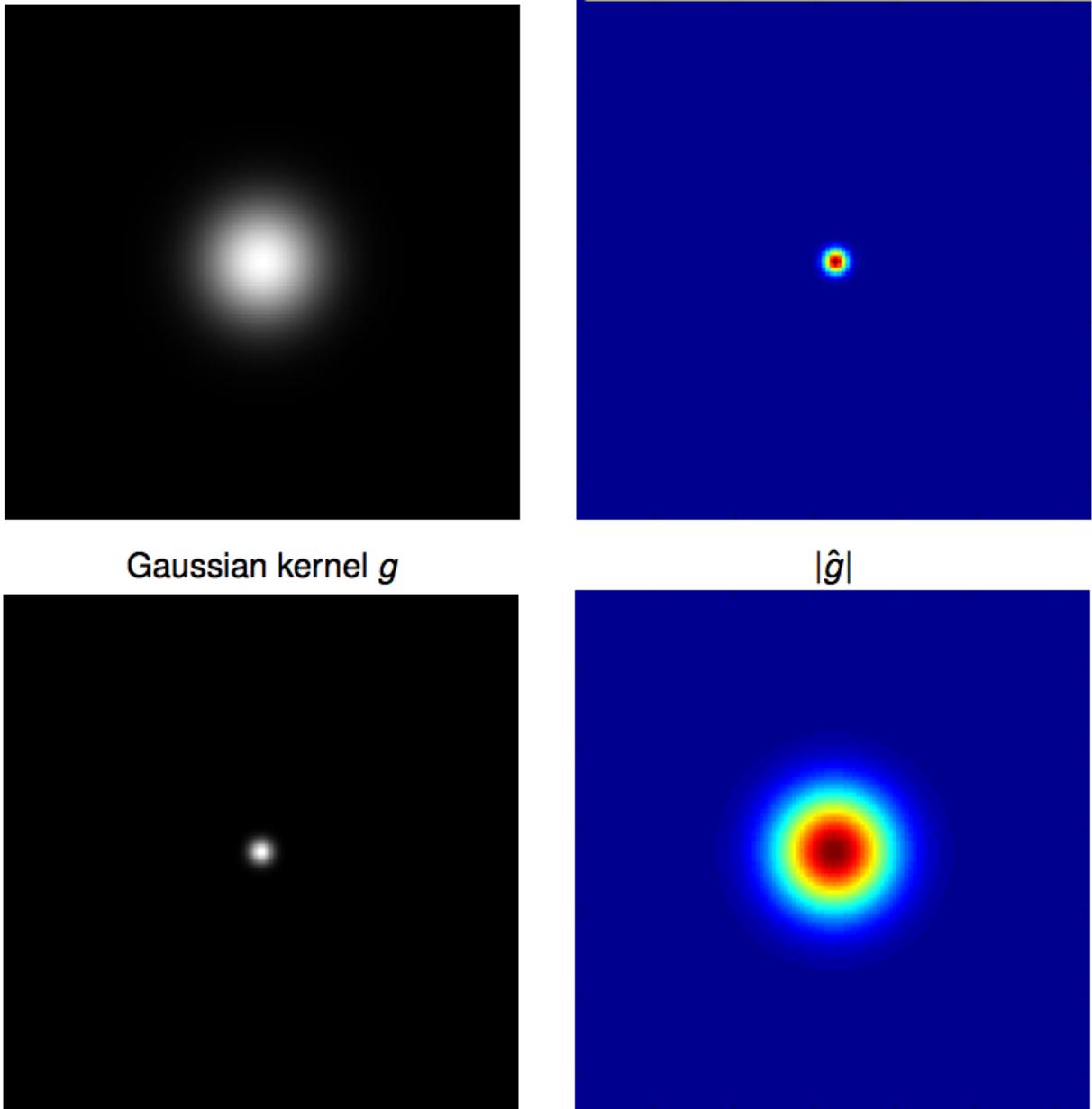


Abbildung 21: Gaussians in Fourier Space

- The Fourier Transform of a Gaussian is a Gaussian-like Function:

$$f(p) = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{p^2}{2\sigma^2}} \Rightarrow \hat{f}(\omega) = e^{-\frac{4\pi^2\omega^2}{2\sigma^2}}$$

3.1.2 Types of Filters:

- **low-pass filter** filter which leaves low frequencies intact (e.g. gaussian)

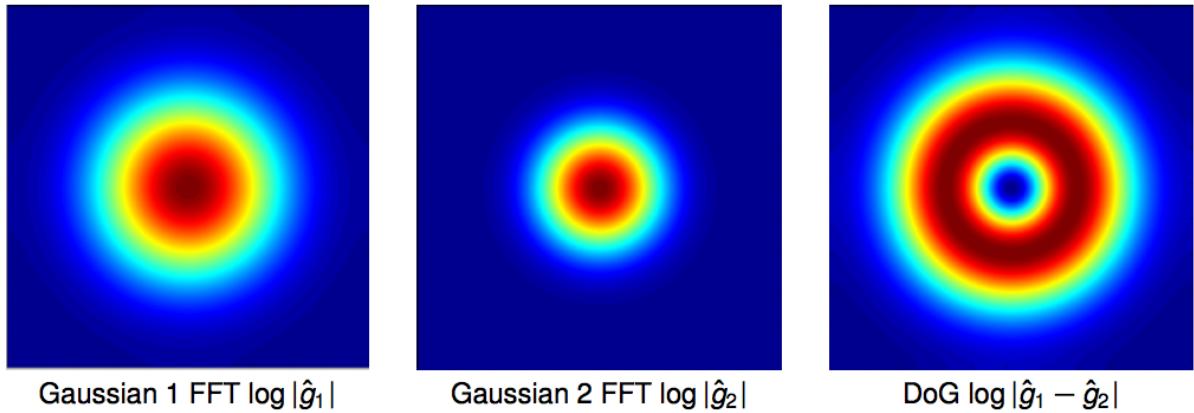


Abbildung 22: Difference of Gaussians

- **high-pass filter** filter which leaves high frequencies intact (e.g. impuls - Gaussian)
- **band-pass filter** filter which leaves a range of frequencies intact while suppressing high and low frequencies (Difference of Gaussians)

3.1.3 Properties of Fourier Transform

- *linearity*: $\widehat{\alpha f + \beta g} = \alpha \widehat{f} + \beta \widehat{g}$
- *rotation invariance*: when f is rotated, the Fourier transform \widehat{f} is rotated by the same angle.
- *shift theorem*: shifting p_0 leads to a phase change according to $\widehat{f(p-p_0)}(\omega) = e^{-2\pi i \omega \cdot p_0} \widehat{f}(\omega)$
- *convolution theorem*: $\widehat{f * g} = \widehat{f} \cdot \widehat{g}$
- *derivatives*: $\widehat{\partial_x^n \partial_y^m f}(\omega) = (2\pi i \omega_x)^n (2\pi i \omega_y)^m \widehat{f}(\omega)$

4 Scale

- If the image has too many pixels compared to the patch => compute difference “scales” of the image and test for each image.

4.1 Sampling & Antialiasing

- not important for exam
- blur image with a corresponding gaussian before taking only every n th pixel => Gaussian Pyramid

5 Scale Invariant Feature Transform (SIFT)

- **Goals:**
 - detect characteristic feature points in images
 - create detailed descriptors that represent each feature as uniquely as possible
 - ensure invariance with respect to changes in location, scale and orientation
- **Applications:**
 - Finding sparse correspondences between images
 - useful in computer vision: global relations between images, tracking, structure-from-motion
 - object detection and recognition
- **idea**

1. detection of characteristic feature points
 - based on Gaussian scale-space using difference of Gaussians
 - extrema provide location and scale
2. accurate localization of key points
 - performs sub-pixel refinement by fitting quadratic functions
 - additionally discards points with high ratio between principal curvatures
3. assignment of the dominant orientation(s)
 - based on a histogram of gradients within the local neighbourhood
 - refines orientation by fitting quadratic functions
4. computation of suitable key point descriptor
 - unit vector based on accumulated histograms of gradients
 - compensated by location, scale and dominant orientation

5.1 SIFT Feature Detection

- Starting Point: *Gaussian Scale-Space of input image f*
 - discrete levels of smoothing: $\sigma_0, \sigma_1, \dots, \sigma_t, \dots, \sigma_{max}$
 - scale-space given by convolution of f with increasing Gaussian G_σ :
- $$f_t = G_{\sigma_t} * f$$
- Difference-of-Gaussian (DoG)

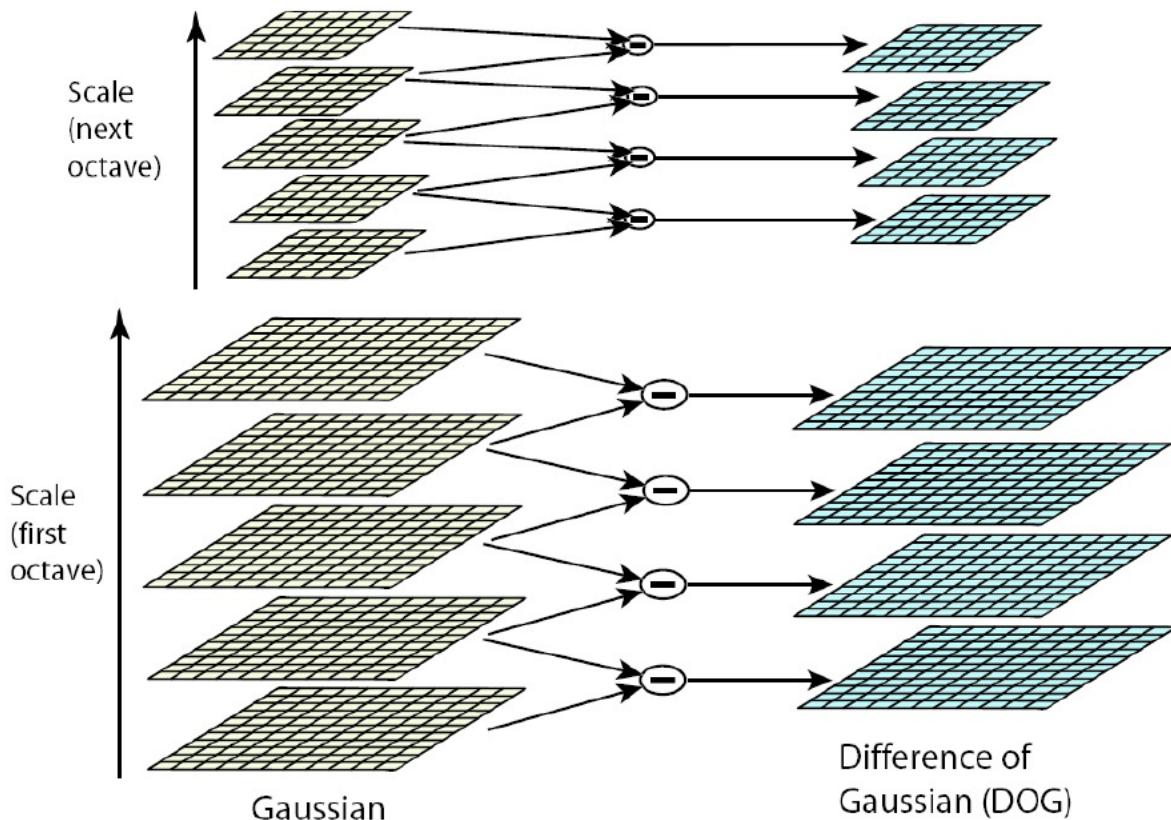


Abbildung 23: SIFT Scale Space Pyramid

- Difference of two consecutive scales of the Gaussian scale-space:

$$\begin{aligned}
D_t &:= f_{t+1} - f_t \\
&= G_{k\sigma_t} * f - G_{\sigma_t} * f \\
&= (G_{k\sigma_t} - G_{\sigma_t}) * f
\end{aligned}$$

- DoG is related to scale derivative via approximation:

$$\partial_\sigma \approx \frac{G_{k\sigma} - G_\sigma}{k_\sigma - \sigma}$$

- The analytical scale derivative is the scaled *Laplacian-of-Gaussian* (LoG):

$$\partial_\sigma G_\sigma = \sigma \delta G_\sigma = \sigma (\partial_x^2 G_\sigma + \partial_y^2 G_\sigma)$$

- DoG is an *approximation* of the LoG:

$$G_{k\sigma} - G_\sigma \approx (k - 1)\sigma^2 \delta G_\sigma$$

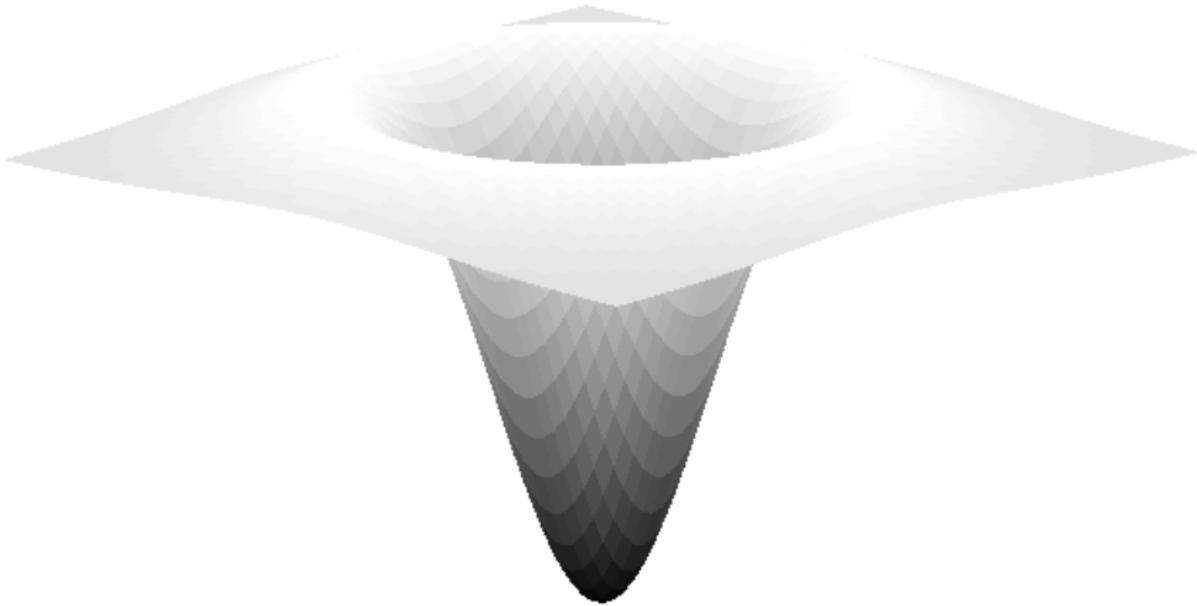


Abbildung 24: The normalized Laplacian of Gaussian

- * $(k - 1)$ can be neglected (independent of scale σ)
 - * $\sigma^2 \delta G_\sigma$ is the *normalized Laplacian-of-Gaussian*
- filters detect shapes that look like them => the LoG detects “blobs”
- Structures live only at certain scale => appear at certain scale, then vanish again (bandpass property of DoG)
 - *characteristic scale* => magnitude of normalized LoG => extremal value (min or max)
 - provides *scale* and *location*

5.2 Accurate Localization of Key Points

- not important for exam

5.3 Assignment of Dominant Orientations

- histograms of gradients (HoG) (computed from gradients of corresponding scale $\hat{\sigma}_i$)
- not important for exam

5.4 SIFT Descriptor

- histograms
- not important for exam

5.5 SIFT properties

- **invariant** to:
 - rotation
 - translation
 - scaling
- **robust** wrt “general changes” (brightness, contrast, …)
- downside: computation intensive

6 Coordinate Transformation

- Image warping (not important for exam)

6.1 Linear Transformations

- linear transformation $T : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ is a map of the form: $T(p) = Ap$ with $A \in \mathbb{R}^{2 \times 2}$
- **scaling:** $A = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix}$ with $s_x, s_y \neq 0$
- **rotation:** $A = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$ with $\cos \theta \in \mathbb{R}$
- $A = U\Sigma V^T \Leftrightarrow AV = U\Sigma$, A maps the columns of V onto scaled columns of U (SVD)
 - any linear transformation can be written as rotation V^T followed by scaling Σ followed by another rotation U
- **properties:**
 - origin maps to origin
 - lines map to lines
 - parallel lines remain parallel
 - ratios are preserved
 - closed under composition

6.2 Affine Transformations

- extend linear transformations with **translations**:

$$T(x) = A \cdot x + t, \quad A \in \mathbb{R}^{2 \times 2}, t \in \mathbb{R}^2$$

- **properties:**

- origin does *not necessarily* map to origin (only when $t = 0$)
- lines map to lines
- parallel lines remain parallel
- ratios are preserved
- closed under composition

- Special cases of affine transformations:
 - *pure translation*: $A = I_2$
 - *Euclidean*: (rigid motion): A is pure rotation (no scaling)
 - *Similarity*: $A = sR$, where R is rotation and $s > 0$, rotation and scaling
 - *Area preserving*: $\det(A) = 1$, five degrees of freedom

6.3 Homogenous Transformations => Homographies

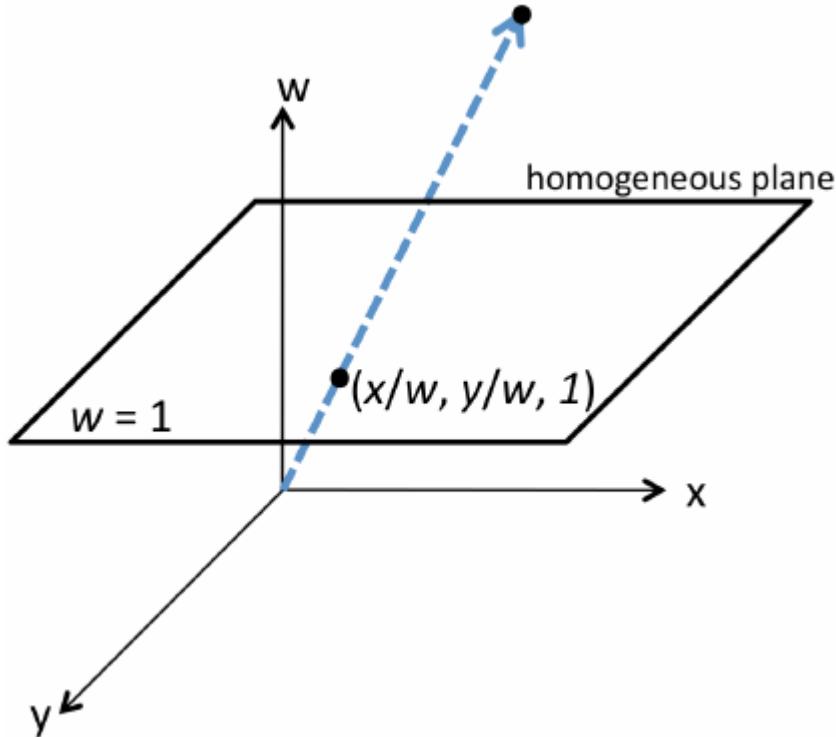


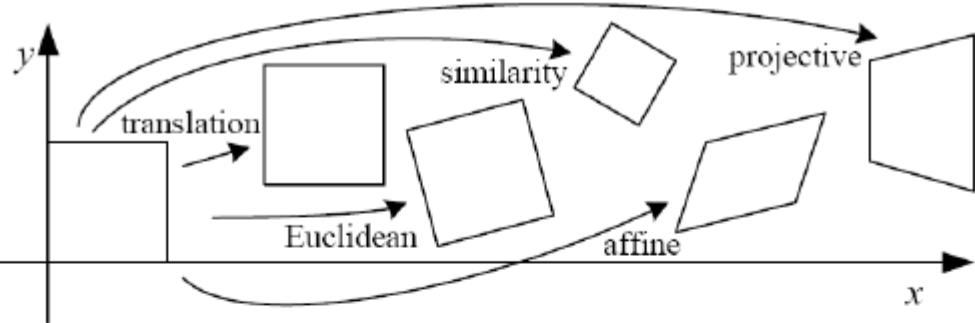
Abbildung 25: Homogenous Coordinates

- moving to *homogenous coordinates*: $p \in \mathbb{R}^2 \implies \hat{p} = \widehat{\begin{bmatrix} x \\ y \end{bmatrix}} := \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \in \mathbb{P}^2$
- \mathbb{P}^2 is called *projective space*, coordinates are *homogenous coordinates*
- Reverse projection to image coordinates is given by:

$$\pi \left(\begin{bmatrix} x \\ y \\ w \end{bmatrix} \right) := \begin{bmatrix} x/w \\ y/w \end{bmatrix}$$

- *affine transformations* can now be written as matrices: $\widehat{T(p)} = \begin{bmatrix} A & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p \\ 1 \end{bmatrix}$
- **Homography**: $H = \begin{bmatrix} A & t \\ b^T & 1 \end{bmatrix}, A \in \mathbb{R}^{2 \times 2}, \det(A) \neq 0, t, b \in \mathbb{R}^2$
- Special case: $H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{1}{f} & 0 & 1 \end{bmatrix}$ with $f \neq 0$

- image point of $\begin{bmatrix} x \\ y \end{bmatrix}$ is $\begin{bmatrix} x \\ y \\ -\frac{x}{f} + 1 \end{bmatrix} \sim \begin{bmatrix} fx \\ f-x \\ fy \\ f-x \end{bmatrix}$
- at $x = f$ the denominator becomes 0 => points go to infinity
- horizontal lines are mapped onto lines which “vanish” in $\begin{bmatrix} -f \\ 0 \end{bmatrix}$ => vanishing point
- two vanishing points: $H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{1}{f_x} & -\frac{1}{f_y} & 1 \end{bmatrix}$ with $f \neq 0$



Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} I & t \end{bmatrix}_{2 \times 3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} R & t \end{bmatrix}_{2 \times 3}$	3	lengths	
similarity	$\begin{bmatrix} sR & t \end{bmatrix}_{2 \times 3}$	4	angles	
affine	$\begin{bmatrix} A \end{bmatrix}_{2 \times 3}$	6	parallelism	
projective	$\begin{bmatrix} \tilde{H} \end{bmatrix}_{3 \times 3}$	8	straight lines	

Abbildung 26: Coordinate Transformations

6.4 Model Fitting 1: Least squares

- estimating Homographies can be done from *correspondence information* => feature detection + matching
- want to find: $H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix}$ => eight unknowns
 - projective coordinates are only correct up to a scale factor:

$$\begin{aligned}x'_i &= h_{11}x_i + h_{12}y_i + h_{13} \\y'_i &= h_{21}x_i + h_{22}y_i + h_{23} \\w'_i &= h_{31}x_i + h_{32}y_i + 1\end{aligned}$$

- normalizing with w and rearranging this yields linear system.

$$\left[\begin{array}{ccccccccc} & & & & & & \vdots & & \\ -x_i & -y_i & -1 & 0 & 0 & 0 & \frac{x'_i}{w'_i}x_i & \frac{x'_i}{w'_i}y_i & \\ 0 & 0 & 0 & -x_i & -y_i & -1 & \frac{y'_i}{w'_i}x_i & \frac{y'_i}{w'_i}y_i & \\ & & & & & & \vdots & & \end{array} \right] = \left[\begin{array}{c} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{array} \right]$$

Abbildung 27: Linear System for m correspondences

- System is *over-determined* with linearly dependent equations => very improbable
- naive way: If $A^T A$ has an inverse, normal equations can be solved as a normal linear system => inefficient and not very robust.
- better way => SVD: $A = U\Sigma V^T$ => normal equations reduce to: $\Sigma^2 V^T \hat{x} = \Sigma U^T b$
 - * Use *pseudo-inverse* σ^+ to solve: $\hat{x} = V\Sigma^+U^T b$

6.4.1 Weighted Least Squares

- Often: knowledge about which features are better (e.g. from SIFT) => assign weight $\sqrt{w_i} > 0$ to each row of $Ax = b$ => adjust minimization problem:

$$\operatorname{argmin}_{x \in \mathbb{R}^n} 1/2 \sum_{i=1}^m w_i (Ax - b)_i^2$$

- Solve as before: $A' = W^{1/2}A$ and $b' = W^{1/2}b$
- very sensible to outliers

6.5 Model Fitting 2: RANdom SAmple Consensus (RANSAC)

1. Repeat N times:
 1. select random subset of samples to fit the model
 2. compute model from selected subset of samples
 3. compute error between samples and estimated model
 4. count *inliers*: points with error below a threshold
2. Refit model using largest inlier set found during any of the above N iterations
- parameters:
 - *number of s points selected to fit the model*: always choose minimum number => fast, allows more trials in given time, might not get good initialization from sample size
 - *threshold for inliers*: choose in a way, that inliers have high probability to stay below threshold (e.g. 95%)

- number N of trials: choose N s.t. with high probability p at least one random sample is free of outliers:

$$N = \frac{\log(1 - p)}{\log(1 - (1 - e)^s)}$$

- In practice, RANSAC works well for outlier ratios below 50%
- e is usually unknown in advance
 - compute e from data during RANSAC:

$$e \leftarrow \min \left(e, 1 - \frac{\text{number of inliers}}{\text{total number of points}} \right) N \quad \leftarrow \frac{\log(1 - p)}{\log(1 - (1 - e)^s)}$$

- this might run for a *long time* => use maximum number of iterations

- **Pro**

- simple and general
- applicability to different problems
- works well in practice

- **Con**

- several tunable parameters
- not deterministic
- iterative with unknown number of iterations
- does not work well for low inlier ratios
- can always get good initialization based on minimum number of samples

6.6 Model Fitting 3: Hough transform

- another voting algorithm for fitting low-parameter models
- simple objects represented by *very few parameters*
- not important for exam