Summary Visual Computing

1 Introduction (19.09.2013)

- A pixel is not a little square, but a sample
- **Geometric resolution**: Number of pixels per area
- **Radiometric resolution**: Number of colours (or grayscales) per area

2 Image Segmentation (24.09.2013)

A **segmentation** is a partition of an image into *distinctive* (i.e. non-overlapping) parts.

2.1 Segmentation strategies

2.1.1 Machine learning

Berkeley Segmentation Database and Benchmark
http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html

2.1.2 Thresholding

Segments image in two categories of partitions: Inside (1) and outside (0).

$$I_\alpha(x,y) = \begin{cases} 1, & I(x,y) \geq T \\ 0, & I(x,y) < T \end{cases}$$

Where $I(x,y)$ denotes the image and T is the (greyscale) threshold. I_α denotes the alpha mask.

2.1.2.1 Gaussian threshold

Not a strict threshold, but with a certain probability distribution around the threshold.

2.1.2.2 Chromakeying

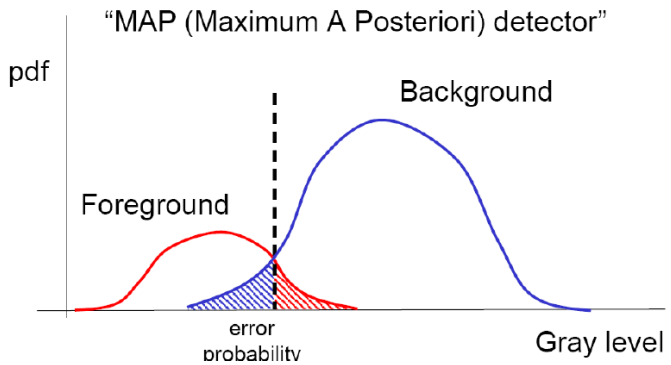
At shooting photographs, use a background with uniform color (e.g. bluebox).

$$I_\alpha = |I - g| > T$$

Where g is the uniform color.

2.1.2.3 Problems of thresholding

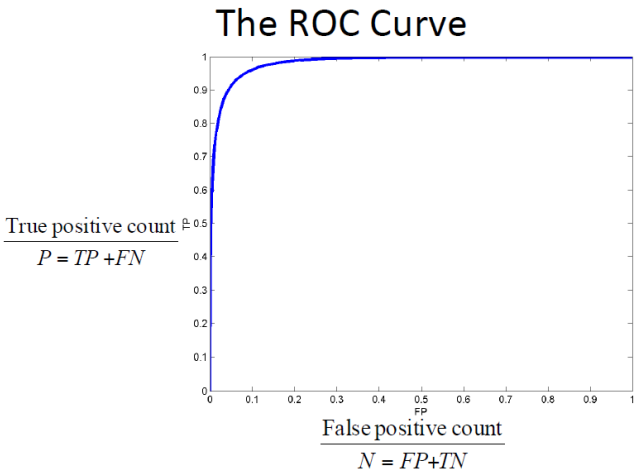
Overlapping regions in histogram: Thresholding only works well, if the positive and negative (RGB) values are clearly distinguishable. Mostly, background colours also belong to the foreground.



Mixed Pixels: Thresholding doesn't account for alpha (opacity) values. Some pixels (especially hair) are mixed by foreground and background colour, so a better "mask" would be: $I_{comp} = I_\alpha I_a + (1 - I_\alpha) I_b$ where I_α denotes the foreground, I_b denotes the background and I_α is the alpha value.

Binary classification	Classified as positive	Classified as negative
True classification	True positive	True negative
False classification	False positive (Truth: negative)	False negative (Truth: positive)

ROC curves



ROC curve shows trade-off in binary classification.

Point(0,0): If you don't accept anything, you're not doing true nor false decisions.

Point(1,1): You will have as many false positives as true positives.

The ideal point is at (0,1), since this means that there are no false positive counts, but only true positives.

Thresholding is limited, because it does not consider **context** (edges, other objects, etc.) at all.

2.1.3 Pixel connectivity

- Two neighbouring (side-by-side or diagonally) pixels connect, if both belong to the foreground (arbitrarily defined)
- A path of connected pixels is a connected region.
- Can be used for element counting (goose detection)

2.1.4 Region growing

- Start from a seed point or region
- Add neighbouring pixels that satisfy the criteria defining the region (inclusion criterion)
- Repeat until we can include no more pixels.
- Variations: seed selection, inclusion criterion, boundary constraints

2.1.5 Snakes

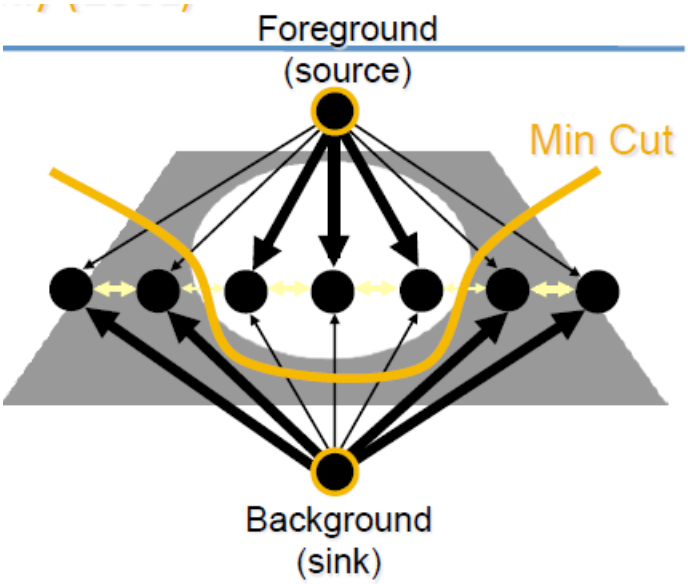
- Active contour: Each point on the contour moves away from the seed while its image neighbourhood satisfies an inclusion criterion.
- Iterative minimisation of energy function$E = E_{tension} + E_{stiffness} + E_{image}$

2.1.6 Distance measures

I_α: Image mask, T: Threshold, I_{bg}: Background image (with a Gaussian to account for small deviations like shadow or mixed pixels)

- **Background-subtraction**: $I_\alpha = \left| I - I_{bg} \right| > T$
- **Mahalanobis distance**: $I_\alpha = \left(I - I_{bg}\right)^T \Sigma^{-1} \left(I - I_{bg}\right) > T^2$
  Σ: Covariance matrix of background pixels.
- If Euclidean distance is a ball, Mahalanobis distance is an ellipsoid

2.1.7 GraphCut



Given an initial pixel-by-pixel segmentation, some individual foreground pixels might be surrounded by background pixels (or vice versa). GraphCut finds the segmentation which minimizes the energy due to such false positives.

If a single falsely labelled "background" dot would be in the foreground region, the min cut would still be at the same place. This method looks what the neighbours are doing.

3 Morphological operations

Morphological operations take two arguments

- A binary image I
- A structuring element S (a pattern)
  The centre (or any other specified point) of S is applied to each nonzero pixel $x \in I$. Only the ones (1s) in S are considered if the pattern matches or not.

3.1 Fitting, Hitting and Missing

- S fits I at $x \in I$, if S is **completely** contained in I
- S hits I at $x \in I$: Here, S is applied to all elements around x (Precisely: Those elements that are overlapped, if S was placed on x). If **any element** in this surround matches x, S hits I at $x \in I$.
- S misses I at $x \in I$, if it doesn't hit I.

Example

Sample of an image

1	1	1
1	0	1
1	1	1

Structuring element

0	0
0	1

Fitting result

0	1	N/A
1	1	N/A

Hitting result

N/A	N/A	N/A
N/A	1	1

N/A	N/A	N/A		N/A	1	0	

3.2 Dilation and Erosion

- Dilation (operator $\oplus$) is the application of a hitting structuring element to an image. Hitting pixels are retained, others discarded. (Bloats up the picture)
- Erosion (operator $\ominus$) is the application of a fitting structuring element to an image. Only fitting pixels are retained. (Image shrinks to a skeleton)

3.3 Opening and Closing

- **Opening**: First apply erosion, then dilation. $(I \ominus S) \oplus S$
- **Closing**: First apply dilation, then erosion. $(I \oplus S) \ominus S$
- Opening and closing can help to get rid of small-sized artifacts.

3.4 Hit-and-miss transform

- Exact pattern matching (considers zeroes as well).
- Operator $\otimes$

Example: Upper-Right-corner detector.

X	0	0
1	1	0
X	1	X

X is a “don’t care” variable

3.5 Thinning and Thickening

In the following, I only denotes the 1s of the image.

- **Thinning**: $I \setminus I \otimes S$

Abbildung 1 Sequential thinning

- **Thickening**: $I \cup I \otimes S$

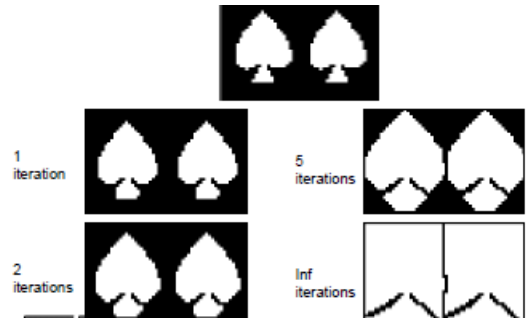
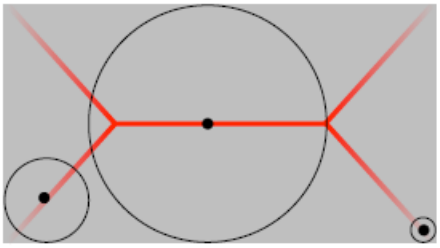


Abbildung 2 Sequential thickening

3.6 Medial axis transform (Skeletonization)



Start a grassfire at the boundary. The point where the grassfires meet, mark the skeleton.

The n-th skeleton subset is

$$S_n(x) = (X \ominus_n B) \setminus \lfloor (X \ominus_n B) \circ B \rfloor$$

The skeleton is the union of all skeleton subsets

$$S(X) = \bigcup_{n=1}^{\infty} S_n(X)$$

Reconstruction of X from skeleton subsets

$$X = \bigcup_{n=0}^{\infty} S_n(X) \oplus_n B$$

4 Convolution and Filtering

4.1 Linear filtering

$$I'(x,y) = \sum_{(i,j)\in N(x,y)} K(x,y;i,j)I(i,j)$$

- K is the kernel (i.e. the linear operator)
- If K is shift-invariant, $K(x,y) = K\ \forall x,y$
- X,y is the "current" pixel on which the kernel is applied.
- I,j are the neighbouring pixels of x,y which the kernel accesses to compute I'(x,y)
- K should always be normalized (L1-Norm = 1), i.e. $\sum_i \sum_j K_{ij} = 1$

4.2 Correlation

$$I' = K \circ I$$
$$I'(x,y) = \sum_{(i,j)\in N(x,y)} K(i,j)I(x+i,y+j)$$

Correlation is template-matching

4.3 Convolution

$$I' = K * I$$
$$I'(x,y) = \sum_{(i,j)\in N(x,y)} K(i,j)I(x-i,y-j)$$

Convolution is associative and commutative, i.e. $(f * g) * I = g * (f * I)$

4.3.1 Differences correlation vs. convolution

Correlation	**Convolution**

Template-matching function (gather)	Point spread function (scatter)
For each pixel, gather values from neighbouring pixels	For each pixel, scatter it to neighbouring pixels
Computationally, a kernel using correlation can be used with convolution if it is mirrored along the midpoint and vice versa. This is only possible, if K is shift-invariant	
If K(I,j) == K(-I,-j) then Correlation == Convolution	

4.4 Image boundary conditions

Since correlation and convolution kernels work on a neighbourhood, the image must be artificially extended at the boundaries (add rows and columns). Different strategies exist:

- Clipping boundary condition. (Add rows and columns of zeros)
- Periodic boundary condition. (Copy last row in front of first row etc.)
- Zero gradient boundary condition (Copy first row in front of itself)

4.5 Separable kernels

A kernel $K(m,n)$ is separable, if and only if $\exists f(m), g(n)$, such that

$$K(m,n) = f(m)g(n)$$

Assuming the image has dimension $M \times N$ and the filter has dimension $P \times Q$, the 2D convolution takes approximately MNPQ mul-adds. Two 1D convolutions take only MN(P+Q) mul-adds.

4.6 Gaussian Kernel (Lowpass filter)

$$g(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$
$$= \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{x^2}{2\sigma^2}\right)\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{y^2}{2\sigma^2}\right) = g(x)g(y)$$

- Kernel width should be > 3σ
- Rotationally symmetric
- Has a single maximum
- Still one single maximum in frequency domain (no corruption from higher frequencies)
- Efficiently implementable
- Convolution with a Gaussian with standard deviation σ is a Gaussian with standard deviation $\sqrt{2}\sigma$

4.7 Differential filters (Highpass filters)

Prewitt operator
$$\begin{bmatrix}-1 & 0 & 1\\ -1 & 0 & 1\\ -1 & 0 & 1\end{bmatrix} = \begin{bmatrix}1\\1\\1\end{bmatrix} * \begin{bmatrix}-1 & 0 & 1\end{bmatrix}$$

Sobel operator
$$\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix} = \begin{bmatrix}1\\2\\1\end{bmatrix} * \begin{bmatrix}-1 & 0 & 1\end{bmatrix}$$

Centered difference operator
$$\begin{bmatrix}-1 & 0 & 1\end{bmatrix}$$

Laplacian (2nd derivative)
$$\begin{bmatrix}0 & 1 & 0\\ 1 & -4 & 1\\ 0 & 1 & 0\end{bmatrix} = \begin{bmatrix}1\\1\\1\end{bmatrix} * \begin{bmatrix}-1 & 0 & 1\end{bmatrix}$$

Image sharpening: $I' = I + \alpha|k * I|$, where k is a highpass filter and $\alpha \in [0,1]$

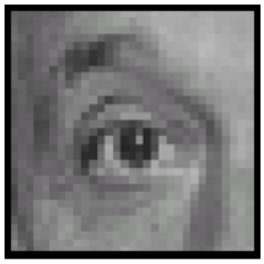
4.8 Low and high frequencies in an image

Lowpass filter	**Highpass filter**	**Bandpass filter**

Low frequencies: Slow grey level changes	High frequencies: Fast grey level changes (such as edges and noise)	Between high and low frequencies.
Blurs image by attenuating (abschwächen) sharp edges	Sharpens image by enhancing edges. Areas of rather constant grey levels (low freqs) are suppressed.	Retains middle range of frequencies (enhance edges, but reduce noise)
Retain frequencies lower than D_0	Retain frequencies higher than D_1	Retain freqs higher than D_0 and lower than D_1
Gaussian filter	Differential filters, such as Laplace	Combination of lowpass and highpass filters, but watch out that there is really a band left.

In general, a filter which has sharp edges detects sharp edges and a filter which is rather smooth will also smooth the image.

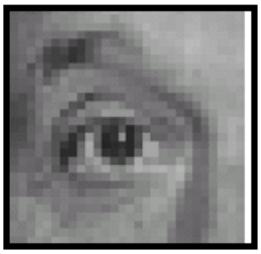
4.9 Examples



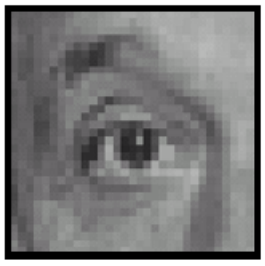
Original

$$\begin{matrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{matrix}$$

(use convolution)

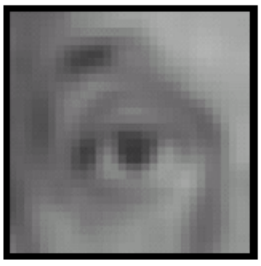


Shifted left
By 1 pixel

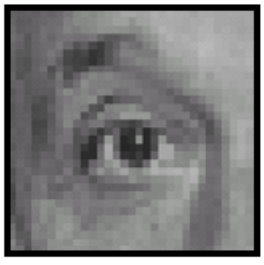


Original

$$\frac{1}{9}\begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$



Blur (with a
box filter)



Original

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{matrix} - \frac{1}{9}\begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

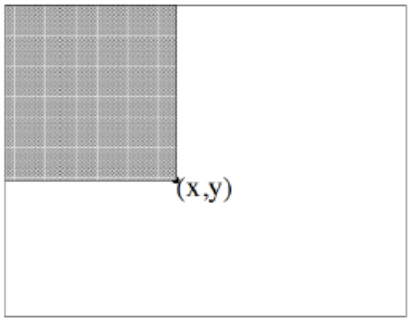
Sharpening filter
- Accentuates differences with local average

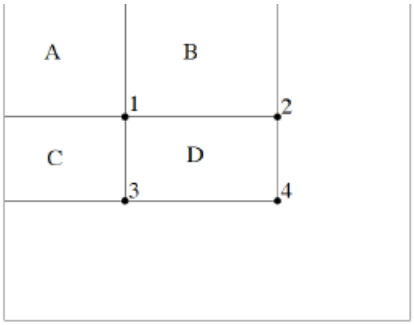
4.10 Integral images

The integral $II(x,y) = \iint_{(0,0)}^{(x,y)} I(x',y')dx'dy'$ can be computed efficiently for many evaluations by storing intermediate integrals for all $(x,y) \in I$, since

$$\iint_{(a,b)}^{(x,y)} I(x',y')dx'dy' = II(x,y) - II(a,y) - II(x,b) + II(a,b)$$

Building the integral image II can be done efficiently by relying on intermediate results.





5 Image features

5.1 Edge detection

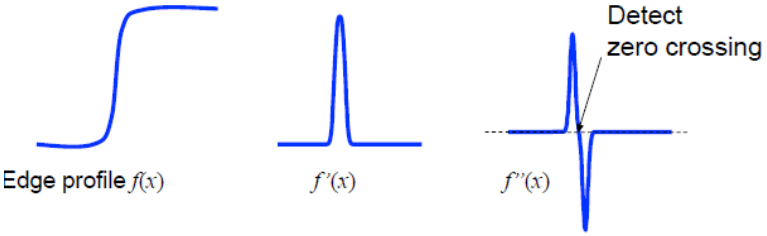
5.1.1 Differential operators

See above for differential Kernels such as Prewitt, Sobel etc.

5.1.2 Laplacian of Gaussian operator

$$L \circ G(x,y) = \frac{1}{\pi\sigma^4}\left(1 - \frac{x^2+y^2}{2\sigma^2}\right)\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

- Edges are recognized as zero crossings of the Laplacian.
- Pure Laplacian is very sensitive to both strong and weak edges
- Blurring with Gaussian helps to suppress weak edges



5.1.3 Canny edge detector

1. Apply derivative of Gaussian operator to smooth the image.
2. Compute gradient magnitude (using a differential operator) and angle for every coordinate

$$M(x,y) = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

$$\alpha(x,y) = \tan^{-1}\left(\frac{\partial f}{\partial x} \Big/ \frac{\partial f}{\partial y}\right)$$

3. Quantize the angle to four directions: 0, 45, 90, 135

4. Apply nonmaxima suppression to gradient magnitude image in order to thin the edge. In a 8-neighbourhood, keep M(x,y) only if it is bigger than both of its neighbours in edge normal direction.
5. Remove M(x,y) which have no neighbours with the same angle.
6. Double-thresholding of gradient magnitude (Hysteresis). If M(x,y) > T_high then walk along the edge on which M(x,y) is and take all M(x,y) > T_low

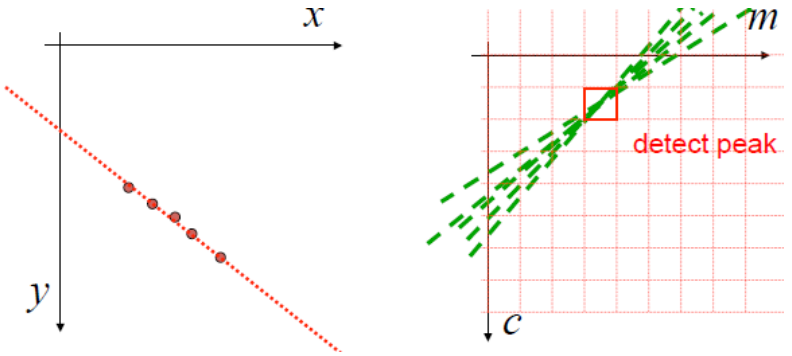
5.2 Hough transform

Generalized template matching technique. Detect a template given a number of observations.

Example straight lines

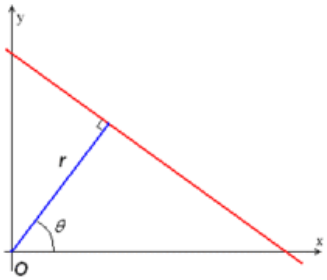
- Pattern: y = mx+c
- Observations: (x,y)
- Parameters: (m,c)

If the observations follow the same pattern, there will be a peak in the parameters plane. This peak is the "solution".



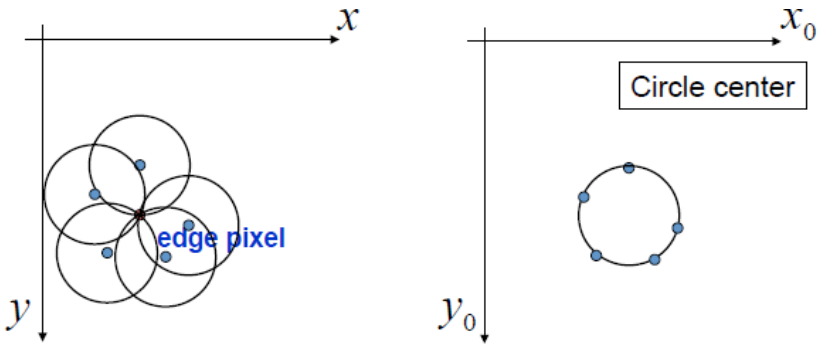
The parameterization $y = mx + c$ is indefinite for vertical lines like $x = a$. A better parameterization for the line is given by radius and angle

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right)x + \left(\frac{r}{\sin\theta}\right)$$
$$r = x\cos\theta + y\sin\theta$$



Example circles with radius r

- Pattern: $r^2 = (x - x_0)^2 + (y - y_0)^2$
- Observations: (x, y)
- Parameters: (x_0, y_0)



5.3 Corner detection (Harris detector)

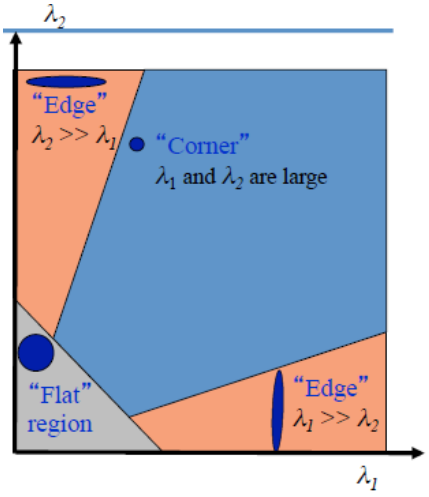
Idea: Both horizontal and vertical derivative must be large for a region to be classified as a corner.

Compute **local displacement sensitivity**

$$S(\Delta x, \Delta y) = \sum_{x,y \in window} \left[f_x(x,y)\Delta x + f_y(x,y)\Delta y \right]^2$$

$$= (\Delta x \quad \Delta y) \left(\underbrace{\sum_{x,y \in window} \begin{bmatrix} f_x^2(x,y) & f_x(x,y)f_y(x,y) \\ f_x(x,y)f_y(x,y) & f_y^2(x,y) \end{bmatrix}}_{M} \right) \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$

This measures approximately the sum of squared errors of neighbouring pixels. Note: $f_x = \frac{df}{dx}$, $f_x^2 = \left(\frac{df}{dx}\right)^2$, it's not the second derivative.



Measure of cornerness

$$C(x,y) = \det(M) - k \cdot \left(trace(M)\right)^2$$
$$= \lambda_1\lambda_2 - k \cdot (\lambda_1 + \lambda_2)^2$$

The Harris corner detector is

- Invariant to brightness offset
- Invariant to shift and rotation
- Not invariant to scaling

6 Fourier Transform

- Observation: Natural images have structures, especially alternating patterns.
- **Aliasing**: Shrinking an image can't be done by taking every second pixel. If the image happens to be an alternating black-white pattern, every other pixel would always be a white pixel.
- An image is a multidimensional vector space: There is a basis function for every pixel.
- **Fourier transform**: Interpret an image as a sum of different sine and cosine waves. The Fourier basis element is $e^{-i2\pi(ux+vy)} = \cos 2\pi(ux+vy) + i \sin 2\pi(ux+vy)$. The Fourier Dtransform is therefore nothing else than a change in basis. The Image is expressed with spatial frequencies and orientation instead of exact positions.
- **Gibbs phenomenon**: A problem of the Fourier transform is the representation of hard edges. This is practically impossible. Instead over and undershooting is observed at edges (ringing).

- **Convolution theorem**: $G * H = \hat{G} \cdot \hat{H}$ $G \cdot H = \hat{G} * \hat{H}$. Convolution can be interpreted as a copy and shift operation, but also look at the animations at http://en.wikipedia.org/wiki/Convolution
- The Fourier transform is **linear** and **invertible**.

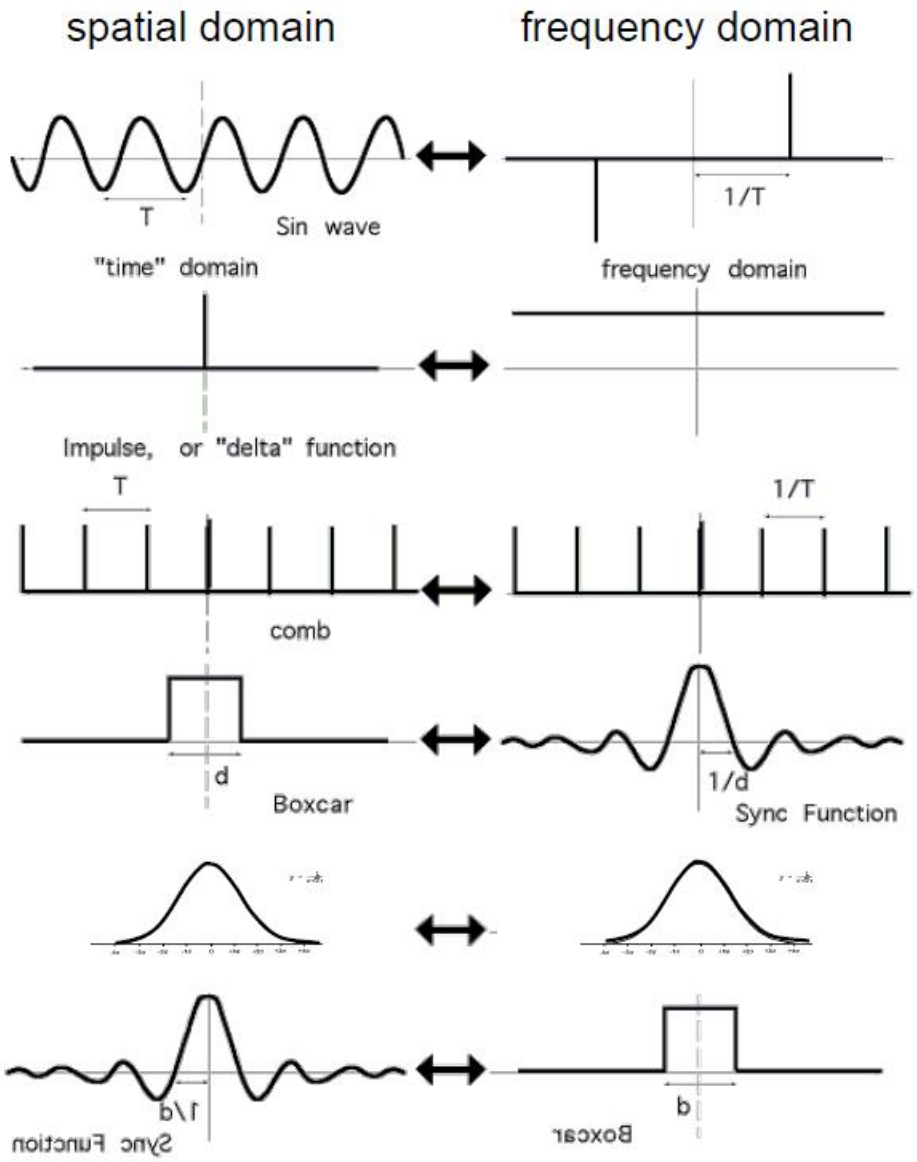
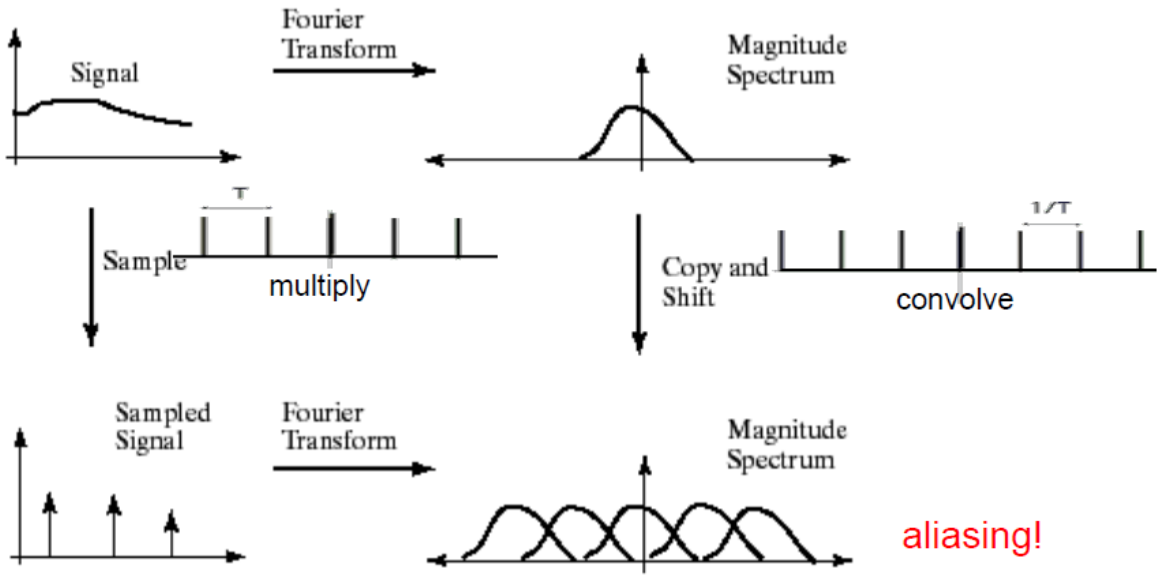


Abbildung 3 Various Fourier transforms

6.1 Proper sampling



- Sampling a signal => Transform many shifted Dirac-Delta-functions.
- Fourier transforms **overlap** => Impossible to reconstruct data. The overlaps are the precise reason for aliasing.
- Before sampling, perform **low-pass filtering** to get a non-overlapping Fourier transform.
- **Nyquist sampling theorem**: The sampling frequency must be at least twice the highest frequency in the image, i.e.

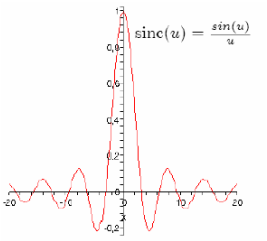
$$f_s \geq 2f_i$$

6.2 Image restoration (Blurring)

- Assume blurring is a result of some kernel $h(x)$ convolving the image.
- Then, if we knew $h(x)$ we could deblur the image by convolving it with $h^{-1}(x)$
- $(h^{-1} * h) = \delta(x)$ $\mathcal{F}[h^{-1}] \cdot \mathcal{F}[h] = 1$ $\mathcal{F}[h^{-1}] = 1/\mathcal{F}[h]$

6.2.1 Motion blur

- Problem: For motion blur, $h(x)$is a boxcar function and has limited support => $\mathcal{F}[h](x)$ is a sinc-function, which has a lot of roots (Nullstellen). Vanishing frequencies can't be recovered.
- Problem: As $\mathcal{F}[h](x)$ is a $sinc$ function, $\mathcal{F}[h^{-1}]$ amplifies noise for $\mathcal{F}[h](x) \ll 1$.
- Solution for noise amplification: Use regularized reconstruction filter: $\mathcal{F}[h^{-1}] = \frac{\mathcal{F}[h](x)}{|\mathcal{F}[h](x)|^2+\epsilon}$



6.2.2 Space-time super resolution

- Temporal blur is often a strict box function. => Fourier transform is a sinc-function with infinite support and amplitudes well above zero. This is good, since high frequencies are therefore reconstructible. Super-resolution in time is therefore possible.
- Spatial blur is often a Gaussian => Fourier transform is also a Gaussian. This is bad, since high frequencies are very soon suppressed. Super-resolution in space is therefore impossible.

7 Unitary transforms

A unitary transform is a unitary (i.e. $A^{-1} = A^{*T}$, aka orthonormal for real matrices) matrix A with dimension $MN \times MN$. The $M \times N$ image matrix I is stacked column-wise to a vector f, such that

$$c = Af$$

- For any unitary transform, $\|c\|^2 = c^H c = f^H A^H A f = \|f\|$, i.e. vector lengths ("energies") are conserved. Every unitary transform is only a rotation of the coordinate system.

7.1 Principal component analysis (PCA) aka Karhunen-Loeve transform

Cf. HPC summary

Images are reshaped to a column vector. Total number of pixels of an image = # dimensions. Number of images = # observations.

- PCA boils down to a SVD, i.e. $A = U\Sigma V^T = SVD(I)$
- SVD is the best-possible rank-k approximation of a matrix
- Mean squared error by choosing only the first k coefficients is minimal.

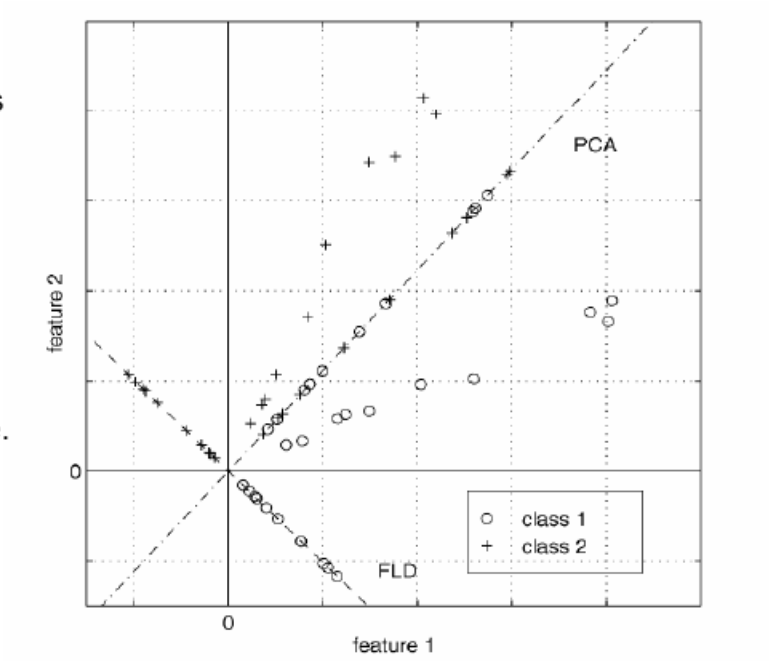
7.2 Face recognition

1. Perform a PCA on a set of test faces I and obtain the principal components P, such that $PC = I$. (In fact, PCA boils down to performing a SVD on I, i.e. $I = U\Sigma V^T$. Here, $P = U, C = \Sigma V^T$)
2. Discard some less important (low singular value) principal components and multiply with original test faces, i.e. $C_k = U_k^T I$, where the k most important principal components were considered. ➔ Test faces are now lower dimensional.
3. For each trial face (e.g. an image on which face recognition should be carried out), multiply it with the most important principal components (e.g. make it lower dimensional) and compare it with all low-dimensional test faces.
4. Comparison is done with simple Eulerian distances on the C matrices.
5. Apply some threshold to determine whether an image is likely to be face or not.

➔In this application, PCA reduces the computational cost of the Eulerian distance comparison a lot.

7.3 Fisher transform

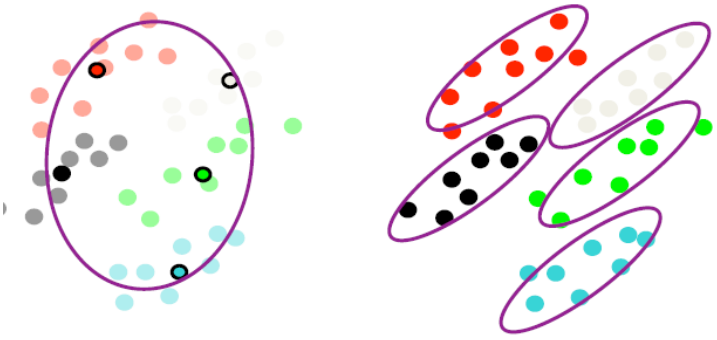
2d example:
Samples for 2 classes are projected onto 1d subspace using the KLT (aka PCA) or Fisher LDA (FLD). PCA preserves maximum energy, but the 2 classes are no longer distinguishable. FLD separates the classes by choosing a better 1d subspace.



[Belhumeur, Hespanha, Kriegman, 1997]

Between Individual Variance

Within-Individual Variance



- Idea: Sort images into classes (magically). Find directions which maximize between individual variance, minimize within individual variance.
- This can be solved using the eigenvectors w_i with k largest eigenvalues from the following generalized Eigenvalue problem:

$$R_B w_i = \lambda_i R_W w_i$$

Where R_B = between class scatter and R_W = within class scatter

$$W_{opt} = \arg\max_W \left(\frac{\det\left(W R_B W^H\right)}{\det\left(W R_W W^H\right)} \right)$$

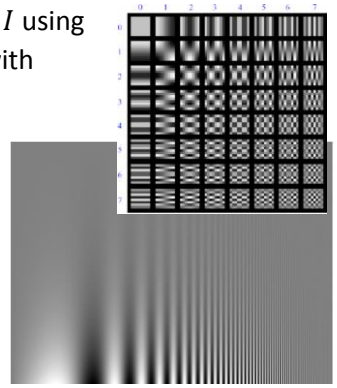
$$R_B = \sum_{i=1}^{c} N_i \left(\overline{\mu_i} - \overline{\mu}\right)\left(\overline{\mu_i} - \overline{\mu}\right)^H$$

Samples in class i

Mean in class i

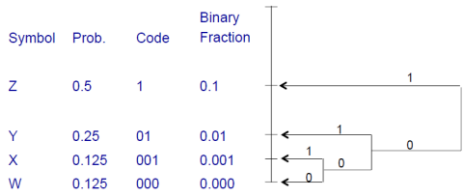
$$R_W = \sum_{i=1}^{c} \sum_{\overline{\Gamma_l} \in Class(i)} \left(\overline{\Gamma_l} - \overline{\mu_i}\right)\left(\overline{\Gamma_l} - \overline{\mu_i}\right)^H$$

7.4 JPEG image compression



1st compression: Discrete Cosine transform (DCT). Decompose image I using a frequency basis (i.e. $I = \sum_i c_i f_i$). This decomposition can be done with DCT. Discard very high (and thus uninformative) frequencies.

2nd compression: Campbell-Robson contrast sensitivity curve. Human perception is not equally sensitive to light intensity with respect to all spatial wavelengths. Changes at middle wavelengths are well perceivable, at high wavelengths not.
Therefore, use different **quantisation** of intensity for different wavelengths. Encode high wavelengths coarser than middle wavelengths (according to the Campbell-Robson curve).



3rd compression: Entropy coding. Since there is some sort of a probability distribution in wavelength distribution, a **Huffman code** can be applied. (I.e. more frequent symbols have shorter codes, to minimize overall length of code).
The average code word length is $H = -\sum p_i \log_2 p_i$, where p_i are the probabilities of the symbols.

8 Image pyramids

Store an image in multiple resolutions in order to apply coarse-to-fine strategies, such as

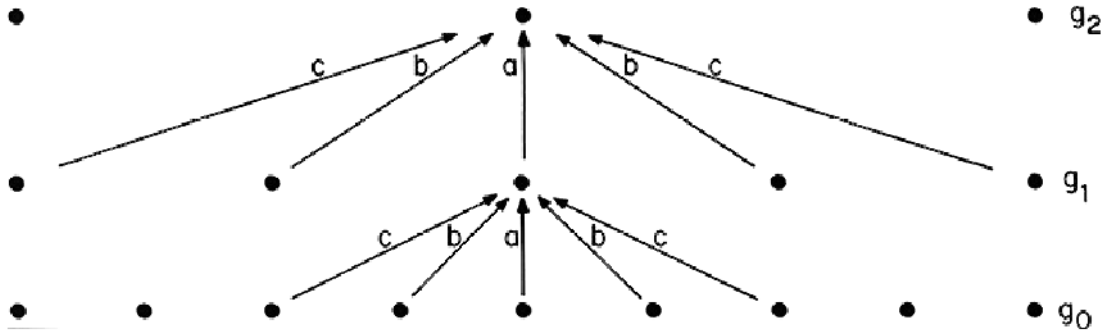
- Search for correspondence
- Edge tracking
- Control of detail and computational cost in template matching

8.1 Gaussian pyramid

Basic level: Image itself

Lower level: For every lower level, compute the Gaussian of the top-level and downsample with factor 2

GAUSSIAN PYRAMID



g_0 = IMAGE

g_L = REDUCE $[g_{L-1}]$

8.2 Laplacian pyramid

The Laplacian pyramid is the difference of Gaussians pyramid. On level j, it is built as Gaussian pyramid level j minus Gaussian pyramid level j+1, where level 0 represents the image itself. On the coarsest level, the Laplacian level is the same as the Gaussian level.

Therefore, each level of the Laplacian pyramid acts as a band-pass filter, since it represents frequencies unrepresented at other levels.

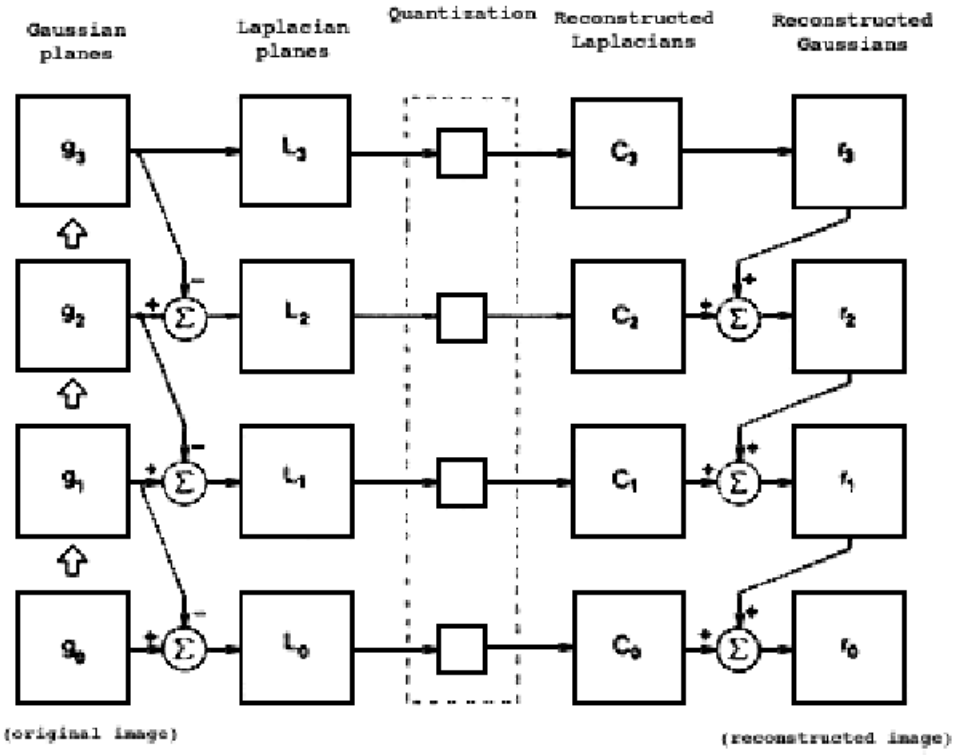


Abbildung 4 Application for compression using Gaussian and Laplacian pyramids. Starting from g0, the Gaussian pyramid levels are built. Then, the Laplacian planes are computed from the Gaussian planes. Compression actually takes place in the quantization stage. G0 can be reconstructed by iteratively reconstructing the Gaussian planes from the coarsest level.

8.3 2D discrete wavelet transform / Haar transform

The Haar transform provides another basis for images.

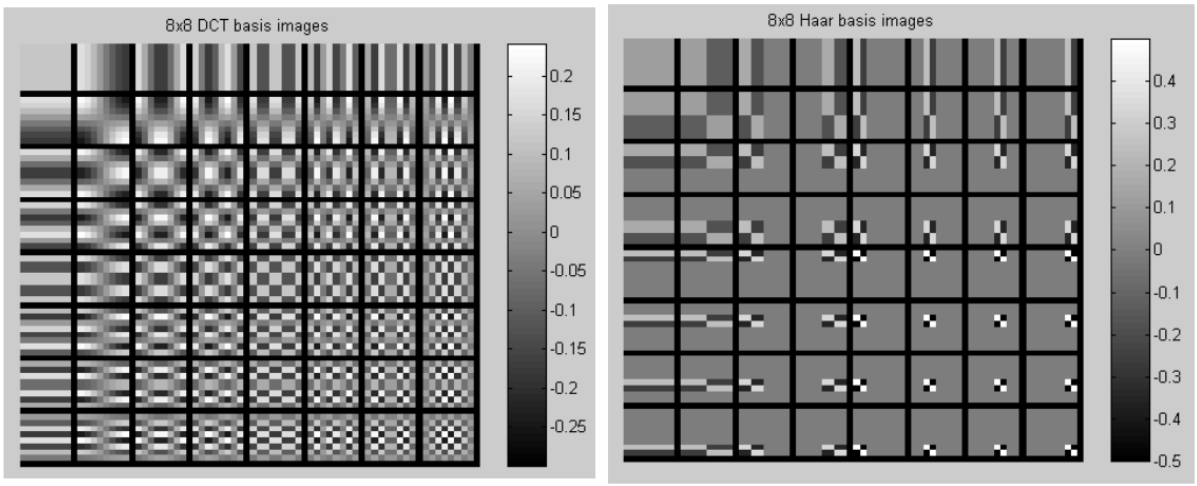


Abbildung 5 Basis images of Discrete Cosine transform (left) and Haar transform (right)

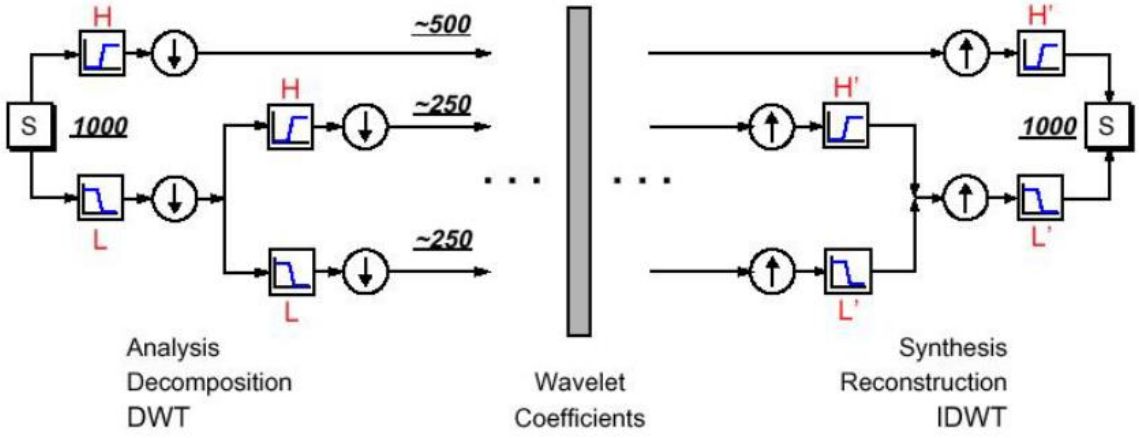
The basis images of the Haar transform are not only local in frequency, but also in position.

DCT	Wavelet-transform
Captures information about frequencies only	Captures information about frequencies and positions
$O(n \log n)$	$O(n)$
Over / undershoot is typical artefact (ringing)	Less over / undershoot

8.4 Filterbanks (Fast Wavelet transform)

Wavelets can be recursively built from wavelets of the next higher resolution level. In fact, every wavelet is computed by a series of two operations and after every operation, another wavelet is built.

As operations, different possibilities exist, usually a low-pass and a high-pass filter. By applying the filters recursively, a set of basis images can be derived (procedurally). That means, that not the whole basis must be specified, but only the two operations.



9 Optical flow

Definition: Apparent motion of brightness patterns. Note: This is not necessarily the same as the motion field, although this is what we mostly look for.

9.1 Optical flow constraint equation (Material derivative)

$$\frac{DI}{Dt} = \frac{\partial I}{\partial t} + \frac{dx}{dt}\frac{\partial I}{\partial x} + \frac{dy}{dt}\frac{\partial I}{\partial t} = 0$$
$$\Leftrightarrow \frac{DI}{Dt} = I_t + u \cdot I_x + v \cdot I_y = 0$$

Unknowns: u, v

Note: This is somewhat the inverse problem to the standard fluid dynamics advection. While in fluid dynamics, the velocity field is usually known and I^{n+1} is sought, here, I^{n+1} is known but the velocity field is sought.

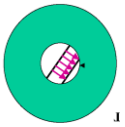
9.2 Aperture problem

- If a (infinite) line moves, it is impossible to tell in which exact direction it moves. It looks the same whether it moves up and down or left and right.
- Mathematically: Solution of optical flow constraint equation is not unique, since there is only one equation for two unknowns.

- Normal flow, Horn & Schunck and Lucas Kanade are approaches to provide a second equation.

9.3 Normal flow

$$\frac{DI}{Dt} = 0$$
$$\nabla I \cdot U = 0$$



9.4 Horn & Schunck algorithm

- $e_s = \iint u_x^2 + u_y^2 + v_x^2 + v_y^2 \, dxdy$ This term is the L2 norm and solves the aperture problem.
- $e_c = \iint \left(I_x u + I_y v + I_t\right)^2 dxdy$
- Minimize $e_s + \lambda e_c$

Use Euler-Langrage equations, this leads to a coupled system of PDEs.

9.4.1 Remarks

- Information spreads from corner-type patterns
- Errors at boundaries (L1 norm would be better for discontinuities)
- Example of regularisations

9.5 Lucas Kanade approach

- Integrate over patches: Assume a single velocity for all pixels inside an image patch (e.g. 8x8 pixels). This solves the aperture problem, since it leads to an overdetermined system of equations.

$$\begin{pmatrix} I_{x1} & I_{y1} \\ \vdots & \vdots \\ I_{x64} & I_{y64} \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} -I_{t1} \\ \vdots \\ -I_{t64} \end{pmatrix} \Leftrightarrow A\boldsymbol{v} = \boldsymbol{b}$$

The solution of this system can be found with least squares:

$$A^T A v = A^T b$$
$$\underbrace{\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}}_{A^T A} \underbrace{\begin{pmatrix} u \\ v \end{pmatrix}}_{\boldsymbol{v}} = \underbrace{-\begin{pmatrix} \sum I_x I_t \\ \sum I_y I_t \end{pmatrix}}_{A^T b}$$

- Alternatively, this procedure can be viewed as an energy minimisation of the velocities, leading to the same LSE.

$$E(u,v) = \sum_{x,y \in \Omega} \left(I_x(x,y)u + I_y(x,y)v + I_t\right)^2$$
$$\frac{dE}{du}(u,v) = \sum 2I_x\left(I_x u + I_y v + I_t\right) = 0$$
$$\frac{dE}{dv}(u,v) = \sum 2I_y\left(I_x u + I_y v + I_t\right) = 0$$

- LSE similar to corner-detection algorithm
- The solution of this LSE yields the **normal flow**.
- The LHS is singular, if all gradient vectors have the same direction (e.g. along an edge)
- The patch must contain at least 2 elements, otherwise the LSE is underdetermined.

9.5.1 Implementation notes

- I_x and I_y are computed using the average of both images, i.e. $I_x = \frac{\partial}{\partial x}\frac{1}{2}(I^n + I^{n+1})$

- I_t is computed using the difference, i.e. $I_t = \frac{\partial}{\partial t}(I^{n+1} - I^n)$
- The derivatives can be computed with standard difference stencils (i.e. convolution)

9.5.2 Iterative refinement

- Take first image and send it forward in time using optical flow field (aka. Warping).
- From the second image, don't take the same window, but the one translated with the optical flow.
- Fails if intensity structure within window is poor
- Fails when displacement is large. (Better strategy: template matching, with sum of squares distance)

9.5.3 Coarse-to-Fine strategy

Do iterative refinement from optical flow prediction of coarse images to fine images.

9.6 Techniques for general flow transformations

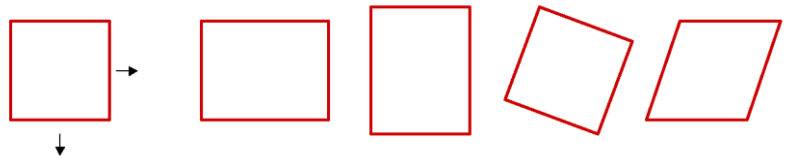
General flow transformations: Not only translation, but also rotation, perspective, any other transformation.

9.6.1 Translation (c.f. Lucas-Kanade)

$$E(\boldsymbol{h}) = \sum_{x \in \mathbb{R}^2} [I(\boldsymbol{x} + \boldsymbol{h}) - I_0(x)]^2$$

where $\boldsymbol{h} = (\delta x, \delta y)^T$

9.6.2 Affine transformation



$$E(\boldsymbol{A}, \boldsymbol{h}) = \sum_{x \in \mathbb{R}^2} [I(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{h}) - I_0(x)]^2$$

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

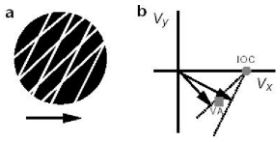
9.6.3 Planar perspective

Same technique as for affine transformation.

$$A = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & 1 \end{bmatrix}$$

9.7 Bayesian flow

Normal flow fusion of several moving lines is not always unique. There are two approaches in human perception.



1. Vector average (of normal motions)
2. Intersection of constraints (Lucas-Kanade)

Adjust Lucas-Kanade with uncertainty parameters σ, ratio of observation and σ_p, prior Gaussian speed.

$$\begin{bmatrix} \sum I_x^2 + \frac{\sigma^2}{\sigma_p^2} & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 + \frac{\sigma^2}{\sigma_p^2} \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = -\begin{pmatrix} \sum I_x I_t \\ \sum I_y I_t \end{pmatrix}$$

10 Video compression

- Spatial correlation between neighbouring pixels
- Temporal correlation between frames

Temporal redundancy reduction

Group of figures (I,P,B)

Block-matching algorithm

Motion vector field

PSNR vs. MOS

11 Radon transform

http://en.wikipedia.org/wiki/Radon_transform

11.1 Application of tomography

Tomography is applicable to Radon transform, if the logarithm is taken on the outputs, since rays are reflected / absorbed in a multiplicative manner, while we need additive elements. The conversion from multiplicative to additive is achieved by the logarithm.

11.2 Radon transform

$$Rf(L) = \int_L f(\boldsymbol{x}) |d\boldsymbol{x}|$$

The Radon transform is the integral of a function over all lines in the space.

Typically, the lines are parameterized as follows

$$\big(x(t), y(t)\big) = \big((t \sin\vartheta + s \cos\vartheta), (-t \cos\vartheta + s \sin\vartheta)\big)$$

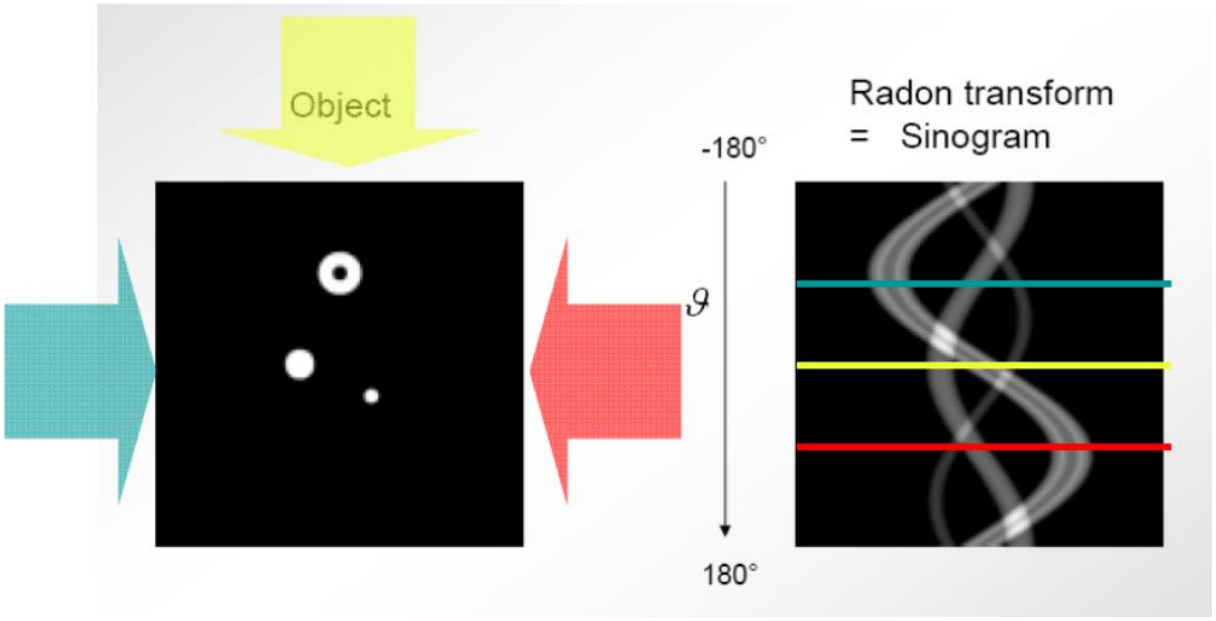
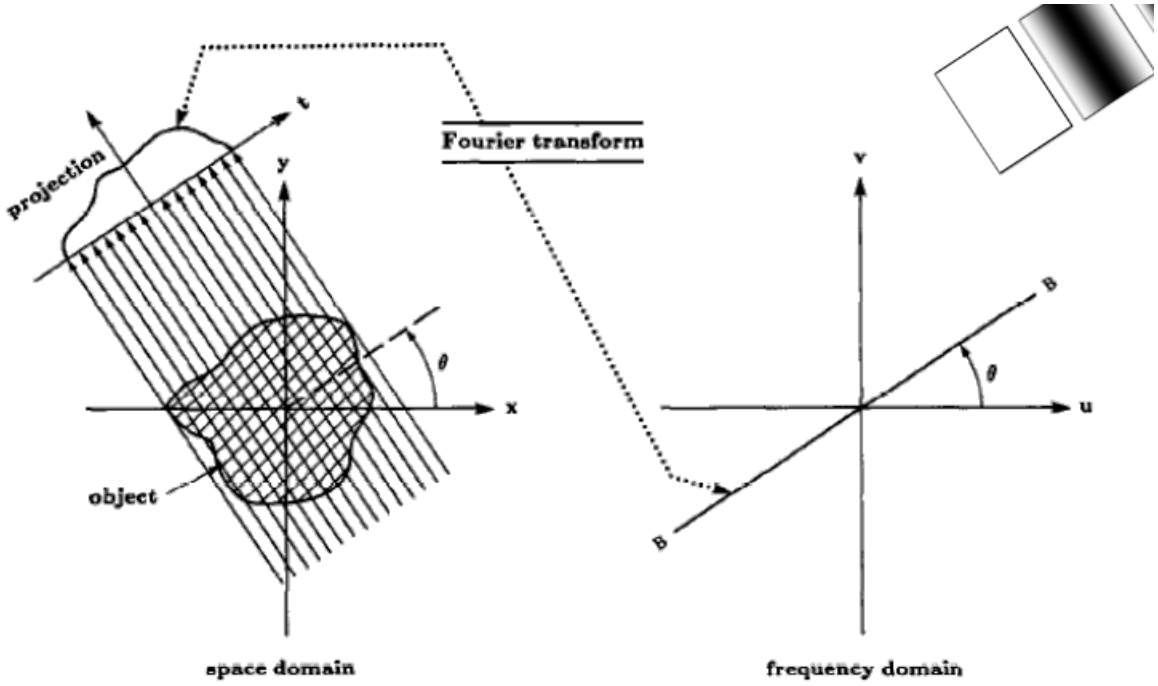


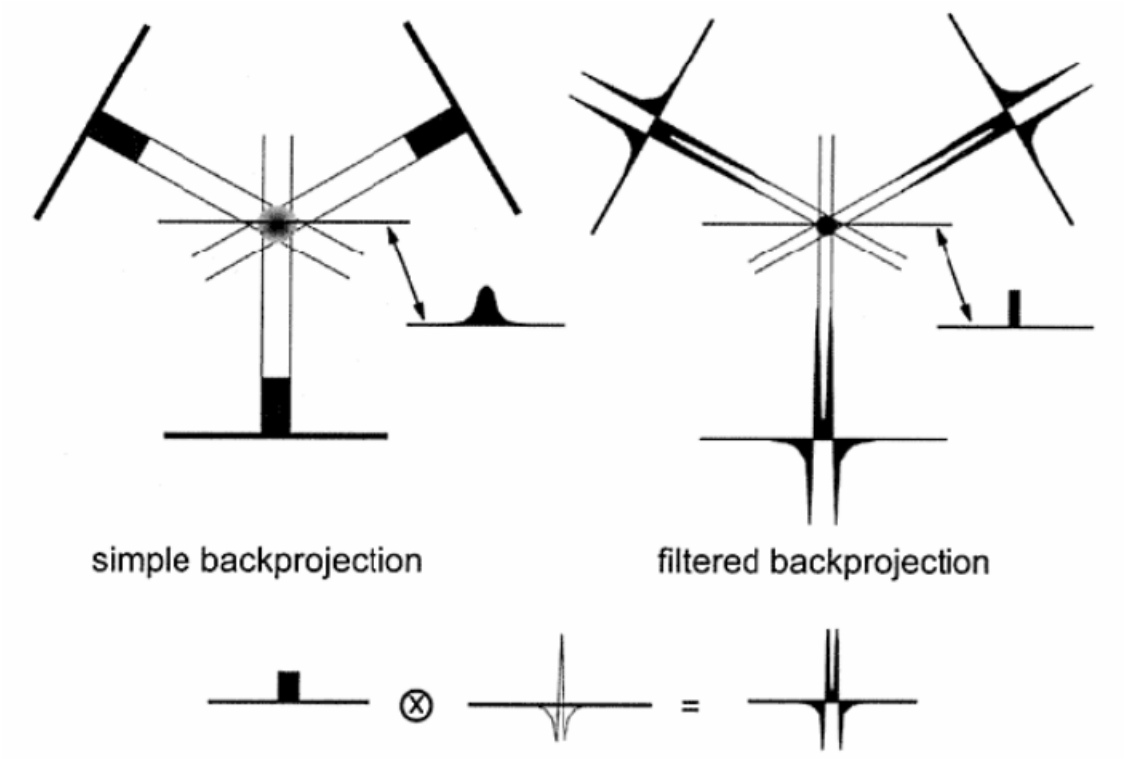
Abbildung 6 Radon transform. A line (in the space of all lines) is parameterized by the angle ϑ. S (on the x-axis of the sinogram) goes from one corner to the other. So, the point in the middle of the yellow line corresponds to a line integral over the object, where the line is exactly vertical and exactly in the middle (right beneath the 'e')

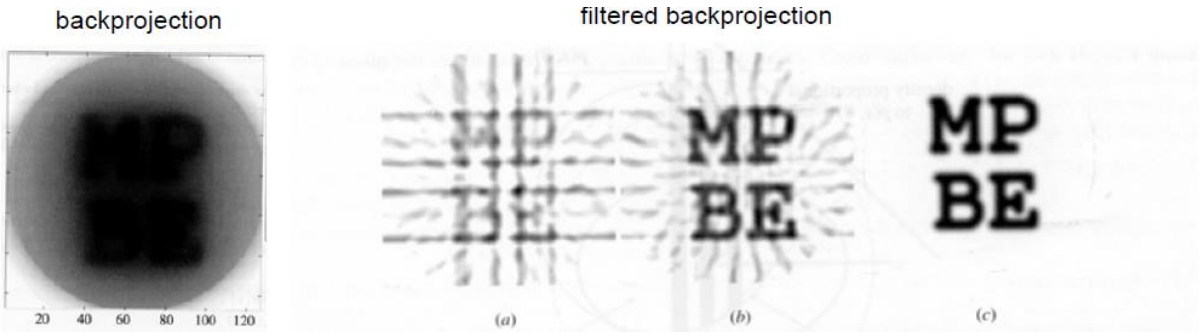
11.3 Fourier slice theorem



The Fourier transform of a projection with angle θ is only a line with angle θ in the frequency domain.

11.4 Filtered backprojection





The Radon transform is not perfect, and leads to blur in the backtransform. Therefore it is filtered in the frequency domain and then backprojected

11.5 Algorithm Computer tomography with filtered backprojection

1. Measure projections (scan around object)
2. Fourier transform projections (to find lines in the frequency domain)
3. Multiply Fourier transform with weighting factor
4. Do the inverse Fourier transform