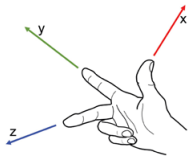
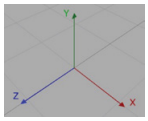
Basics

Coordinate System



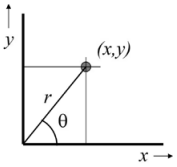


Polar coordinates

$x = r\cos\theta$

$y = r\sin\theta$

Angle θ: in radians

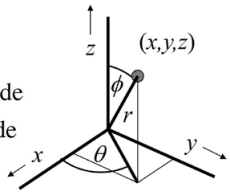


Spherical coordinates

$x = r\cos\theta\sin\varphi$ $\quad \theta$: azimuth or longitude

$y = r\sin\theta\sin\varphi$ $\quad \varphi$: elevation or latitude

$z = r\cos\varphi$



Normalform (x, y) → Polarform (r, φ)

$r = \sqrt{x^2 + y^2}$ $\quad \varphi = \begin{cases} \tan^{-1}\left(\frac{y}{x}\right) & \text{für } x \geq 0 \\ \tan^{-1}\left(\frac{y}{x}\right) + \pi & \text{für } x < 0 \end{cases}$

Polarform (r, φ) → Normalform (x, y)

$x = r \cdot \cos\varphi$ $\quad y = r \cdot \sin\varphi$

Transformations

Homogeneous Coordinates

Allows us to treat all transformation uniformly (express transformation in a single transformation-Matrix *M*). Homogen = Tranformation geht durch den Ursprung.

a 2D point: $\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$ ← Scale factor *w*

a 2D vector: $\begin{pmatrix} x \\ y \\ 0 \end{pmatrix}$ ← $p_1.w - p_2.w = 0$

$$M = \begin{matrix} x & y & z & \\ \end{matrix} \begin{bmatrix} 1 & 0 & 0 & m_{14} \\ 0 & 1 & 0 & m_{24} \\ 0 & 0 & 1 & m_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation, scaling, scewing, etc.

Translation

$\mathbf{P}' = \mathbf{MP}$

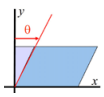
Rotation um *x*-Achse ... *y*-Achse ... *z*-Achse

$$R_x(\beta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta & 0 \\ 0 & \sin\beta & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_z(\beta) = \begin{bmatrix} \cos\beta & -\sin\beta & 0 & 0 \\ \sin\beta & \cos\beta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$\mathbf{R}(\theta) = \mathbf{R}_z(\theta_z)\,\mathbf{R}_y(\theta_y)\,\mathbf{R}_x(\theta_x)$ $\quad \theta_x\,\theta_y\,\theta_z$ are called the Euler angles.

Scaling $\mathbf{P}' = \mathbf{S}(s_x, s_y)\mathbf{P}$ $\quad \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$

Shearing $x' = x + fy,\ y' = y$ $\quad \mathbf{P}' = \begin{pmatrix} 1 & f & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\mathbf{P}$ with $f = \tan\theta$



Combining Transformations:

$$\mathbf{P}'' = \mathbf{M}_2(\mathbf{M}_1\mathbf{P}) = \mathbf{M}_2\mathbf{M}_1\mathbf{P}$$

Composite translations:

$$\mathbf{T}(t_{2x}, t_{2y})\mathbf{T}(t_{1x}, t_{1y}) = \mathbf{T}(t_{1x} + t_{2x}, t_{1y} + t_{2y})$$

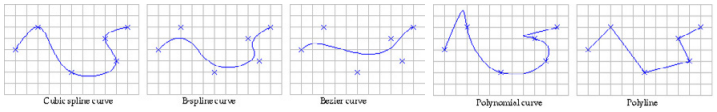
Composite rotations:

$$\mathbf{R}(\theta_2)R(\theta_1) = \mathbf{R}(\theta_1 + \theta_2)$$

Composite scaling:

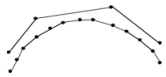
$$\mathbf{S}(s_{2x}, s_{2y})\mathbf{S}(s_{1x}, s_{1y}) = \mathbf{S}(s_{1x}s_{2x}, s_{1y}s_{2y})$$

2D Graphics



Polylines

Line segments



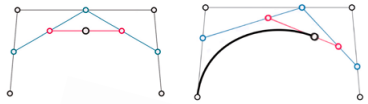
Spline

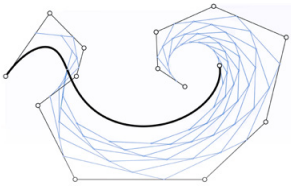
Parametric equation, Resolution-independent.
A spline is a function defined piecewise by polynomials.
Interpolation zwischen Punkten (kubische Splineinterpolation).

Bézier Curve

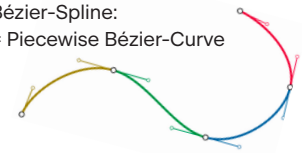
Parametric equation, Resolution-independent. Tangent to the curve at first/last control point. Control points = «attraction» point.

Qubic Bézier-Curve:

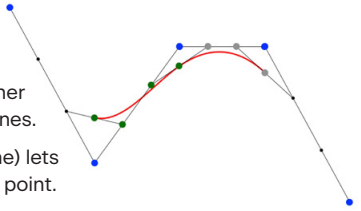




Bézier-Spline:
= Piecewise Bézier-Curve



B-Spline (Basis Spline)



Based on Bézier-Curve but are smoother (more C^2-continuous) than Bézier-Splines.

NURBS (Non-Uniform Rational B-Spline) lets you control the weight of each contorl point.

	Deg.	Cont.	Tangents	Interpol.	Use cases
Bézier	3	C^0/C^1	manual	some	shapes, fonts & vector graphics
Hermite	3	C^0/C^1	explicit	all	animation, physics sim & interpolation
Catmull-Rom	3	C^1	auto	all	animation & path smoothing
B-Spline	3	C^2	auto	none	curvature-sensitive shapes & animations, such as camera paths
Linear	1	C^0	auto	all	dense data & interpolation where smoothness doesn't matter

SVG

Path

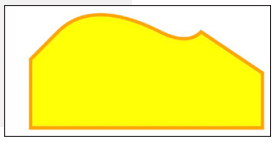
Cmd	Args	Effect
M,m	x,y	move
L,l	x,y	line
H,h	x	horizontal line
V,v	y	vertical line
Z,z		close

Grossbuchstaben: Absolute Position (x, y)
Kleinbucstaben: Relative Position (dx, dy)

Cmd	Arguments	Effect
A,a	rx, ry, x-axis-rotation, large-arc sweep x y	elliptical arc
Q,q	x1, y1, x, y	quadratic Bézier curve
T,t	x, y	quadratic Bézier curve
C,c	x1, y1, x2, y2, x, y	cubic Bézier curve
S,s	x2, y2, x, y	cubic Bézier curve

S: if it follows another S command or a C command, the first control point is assumed to be a reflection of the one used previously.

```
<svg xmlns="http://www.w3.org/2000/svg">
    <path stroke="orange" stroke-width="5" fill="yellow"
      d="M 40,180
         V 80
         l 40,-40
         q 50,-50 150,0 40,20 60,0
         l 90,60
         V 180
         z" />
</svg>
```



Information Architecture

Hierarchical Structuring

- **Generalization:** type hierarchy, categories, **kinf-of / is-a** relations
  Biology (categores of organisms), Shops (product classification)
- **Decomposition: part-of** relation
  Organization, Books (chapters, üaragraphs), file systems (directores, files)

Metadata

Data about data. Data without metadata (describes the data) is pretty useless.

Naming

Taxonomy: Definiert Vokabulare/Wörter
Thesaurus: Taxonomie mit Klassifizierung
Ontology: Addition of rules

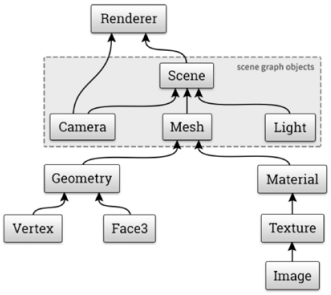
Categorization

More **precice** with **sub**categories:
Hyponyms (unterbegriffe), **Sub**ordinate (untergeordnet)

More **abstract** with **super**caegories:
Hypernyms (Oberbegriffe), **super**ordinate (übergeordnet)

3D Basics

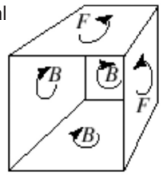
Scene Graph



Tesselation (Polygonalization)

Process of forming a polygonial mesh from a (mathematical described) curved surface. Replaces the surface by a number of trinagles and normals.

Due to normal direction the **sequence of vertices** matters to define front and back faces. To improve performance, back faces are culled **(culling)**, e.g. not draw.



Lambert Shading

«Farbabstufung»

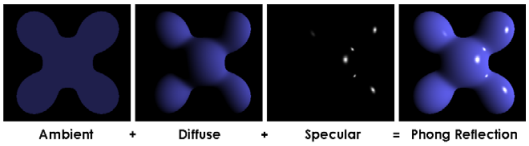
Lamberts Law

$$I_d = I_s \cdot \rho_d \cdot \max\left(\frac{\vec{s}.\vec{m}}{|\vec{s}|.|\vec{m}|}, 0\right)$$

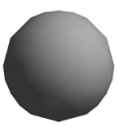
I_d = Intensity of reflected light
I_s = Intensity of light source
ρ_d = Diffuse reflexion coefficient
$\vec{m}$ = Surface normal
$\vec{s}$ = Direction to light source
$\vec{s}.\vec{m} = |\vec{s}|.|\vec{m}|\cos\alpha$

Phong

Interpolates **normals** on **each pixel.**



Gouraud uses **normals** on **each vertex.**



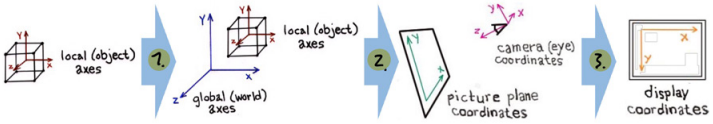
Problem:
Still has an edgy silhouette

z-Buffer (Depth Buffer)

Problem: A projection discards the depth information. The *z*-Buffer stores depth of last drawn pixel. A pixel is only drawn (saved to frame buffer) if it is closer to the camera than the current depth value (updates z-Buffer). Depth values is **logarithmic** in order to have better resolution at close distances.

3D Graphics Pipeline

1. Object/Model Matrix **2.** World Matrix **3.** View/Camera Matrix



1. Model Space
2. World Space
3. Eye/View Space: Camera coordinate system but without oersoective)
4. Perspective Space: Projection onto the image plane.
5. Clip Space
6. Window Space: Trasnform to device coordinate (pixels).
7. Shading/Rasterisation: z-Buffer is used to select pixels to draw

Blending is used to combine images.

Fragment vs Pixel

Pixel: Content (color) of the frame buffer (at a specific pixel location).
At the end of the pipeline.
Fragment: Set of data (color, texture, alpha, etc.) used to color a pixel.
Used through the pipeline.

Fog

Can make a scene more realistic.

$$C = f.C_o + (1\text{-}f).C_f$$

f = exp(-d*z)	GL_EXP
f = exp(-(d*z)^2)	GL_EXP2
f = (end-z)/(end-start)	GL_LINEAR

Computer Vision

Computer vision is concerned with the automatic **extraction,** analysis and understanding of useful **information** from a single image or a sequence of images: Segmentation, Object Detection/Identification, Motion Tracking, etc.

Digital Images and Videos

Dynamic Range

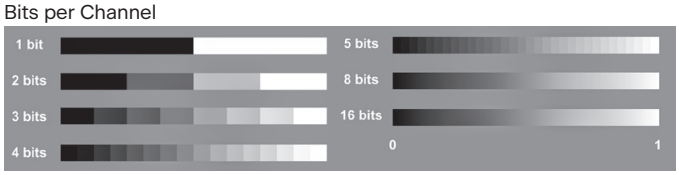
Describes the **ratio** between the highest and lowest values in an image.
Refers to the full range of tonal values.

Contrast

Refers to the **preceived difference** (not the the absolute range) in lightness between the light and dark areas of an image.

Bit Depth

Bits per Channel



Brightness, Lightness, Luminance and Lumen

Brightness: Subjective (perceptual) description of how bright something is.
No units, visual experience (e.g. screen brightness)

Lightness: Subjective (perceptual) description of color.
Scaled (e.g., 0–100%), used in color theory or design

Luminance: Physical, cd/m²
Quantifies how much light hits a surface

Lumen (lm): Physical, measures the total amount of perceived visible light emitted by a source (in all directions) = luminous flux. Bridges the gap between raw physical light energy and how humans actually perceive light.

Humans perceive the lightness of colors differently from their actual physical brightness. There are two main ways to convert a color image to grayscale: **perceptual** (luminance-based) and **non-perceptual** (color brightness/lightness-based).

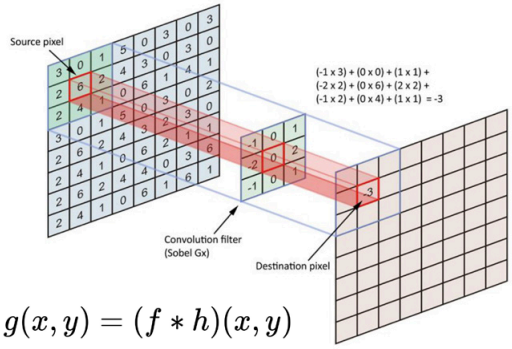
CODEC (Compress/Decompress)

Lossless Compression: Redundancy in the data.
Can be reconstructed exactly. PNG, GIF, TIF, RAW, BMP

Lossy Compression: Redundancy in human perception.
Some information lsot forever. JPG

Image Filtering

Discrete Convolution (Diskrete Faltung)



$$g(x,y) = (f * h)(x,y)$$

$$g(x,y) = \sum_{i=0}^{k_h-1} \sum_{j=0}^{k_w-1} f(x+i-m, y+j-n) \cdot h(i,j)$$

or

$$g(x,y) = \sum_{m=-M}^{M} \sum_{n=-N}^{N} f(x-m, y-n) \cdot h(m,n)$$

$g(x,y)$: The resulting convolved image or signal at position (x,y).

$f(x-m, y-n)$: The input signal or image, offset by m and n.

$h(m,n)$: The kernel (filter) values.

M, N: The half-sizes of the kernel (for a kernel of size $(2M+1) \times (2N+1)$)

Kernels

Edge detection

Sobel

Also used to compute image gradient (not just edges).

$$\mathbf{S}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A}$$ 

$$\mathbf{S}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$ 

Laplace

0	-1	0
-1	4	-1
0	-1	0

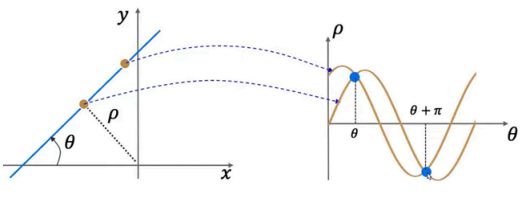
$* \mathbf{A}$ 

Hough Transform

Detects lines and circles.

Transforms a point in image space to a curve in parameter space.
A point in paramter space represents a line in image space.

Image Space | Parameter Space



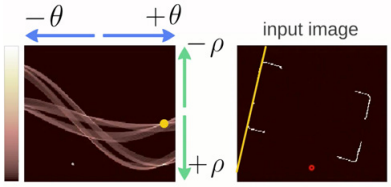
$x \sin\theta - y \cos\theta + \rho = 0$ $\quad$ $x \sin\theta - y \cos\theta + \rho = 0$

Step 1. Quantize parameter space (m,c)
Step 2. Create accumulator array $A(m,c)$
Step 3. Set $A(m,c) = 0$ for all (m,c)

Step 4. For each edge point (x_i, y_i),
$A(m,c) = A(m,c) + 1$
if (m,c) lies on the line: $c = -mx_i + y_i$

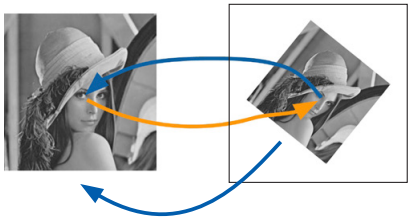
Step 5. Find local maxima in $A(m,c)$

Example



Transformations

Problem bei Forward Transformation: Kann Lücken im Ziel-Space geben.
Backward-Transformation garantiert ein Pixel.



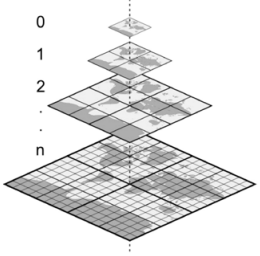
Forward-Transformation (x,y) = T(x',y')
Backward-Transformation (x',y') = T^{-1}(x,y) = Inverse Transformation

Optical Flow

Assumptions (about the source video):
- Brightness is constant
- Movement to be small → Can be approximated using Taylor series
- Spatial coherence (local/neighbouring pixels move in the same direction)

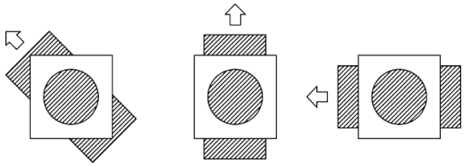
Exhaustive Search

Image Pyramide



Flow equation: $I(x,y,t) = I(x+\Delta x, y+\Delta y, t+\Delta t)$
We use the Sobel kernel to calculate the gradients Δx (u) and Δy (v).

Aperture Problem



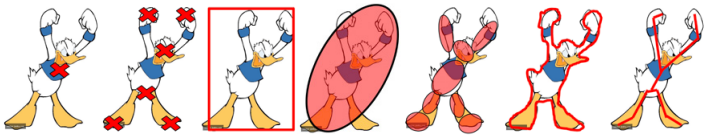
- Homogäne Flächen sind ungeeignet (z.B eine Wand)
- Optimal sind Muster mit Überschneidungen und Ecken

Which feature to track?
yes no

Motion Tracking

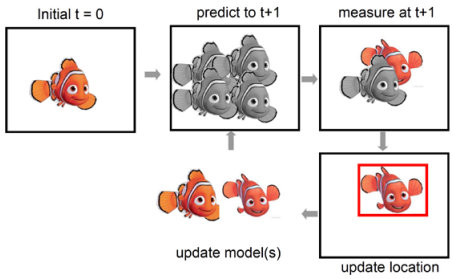
What to Track

center point, mutiple points, bounding box, parts, region, outilne, structure



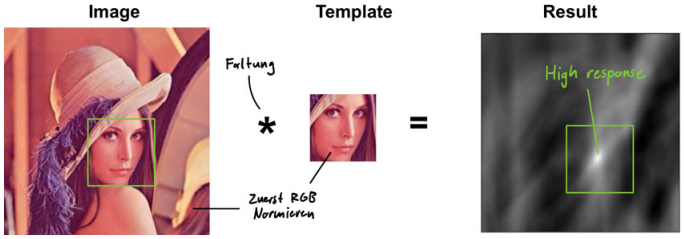
Tracking Approaches

Template Tracking



Problem: Drifting

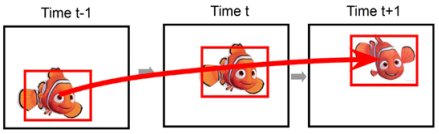
Template Matching



Background Modelling
Problem: Light conditions musn't change.

Input | Background | Foreground

Tracking by Detection
Detect object(s) independently in each frame

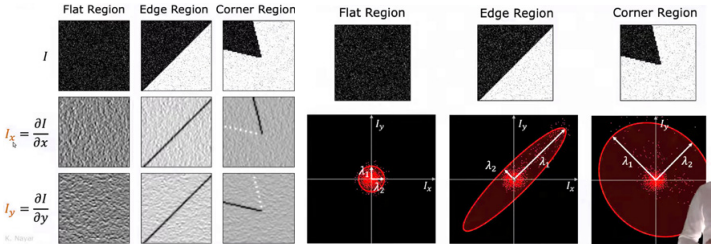


Tracking Issues

- Predition vs. correction
- Similar objects
- Fast motion (motion blur)
- Occlusion (object temporarily hide behind an object)
- Nonlinear dynamics (e.g. bouncing ball)

Harris Corner Detection

Edge = where two regions meet
Corner = where to edges meet.



The magnitude of these Eigenvalues (right image) decide, whether a region is a corner, an edge, or flat.

Idea: The sum of squared differences (SSD) within a small window (we assume motion is small) tells if the area is flat (uniform) or if it is descirptive.

$$f(\Delta x, \Delta y) = \sum_{(x_k, y_k) \in W} \left(I(x_k, y_k) - I(x_k + \Delta x, y_k + \Delta y)\right)^2$$

Problem: This method is slow. We can approximate $I(x_k + \Delta x, y_k + \Delta y)$ by a Taylor expansion, which can be written in matrix form:

$$f(\Delta x, \Delta y) \approx \sum_{(x,y) \in W} \left(I_x(x,y)\Delta x + I_y(x,y)\Delta y\right)^2 \approx \sum_{(x,y) \in W} \left(I_x \Delta x + I_y \Delta y\right)^2$$

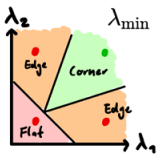
$$\approx \begin{pmatrix} \Delta x & \Delta y \end{pmatrix} M \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$

M is the structur tensor, which represent an ellipsoid.
The shape of the ellipse tells us if its a flat, a corner or an edge.

$$M = \sum_{(x,y) \in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

I_x and I_y are image gradients (derivatives of I), calculated using Sobel-kernel.
If the smallest Eigenvalue is 0, then its a corner (infinite long ellipsoid).

$$\lambda_{\min} \approx \frac{\lambda_1 \lambda_2}{(\lambda_1 + \lambda_2)} = \frac{\det(M)}{\operatorname{tr}(M)}$$



Harris response (threshold to determine if its a corner):

$$R = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2 = \det(M) - k \operatorname{tr}(M)^2$$

where k is an empirically determined constant; $k \in [0.04, 0.06]$

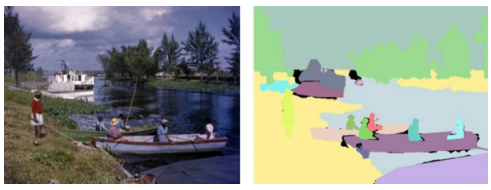
Harris Corner Detection is used in **KLT Point Tracking** (Kanade–Lucas–Tomasi).

Segmentation

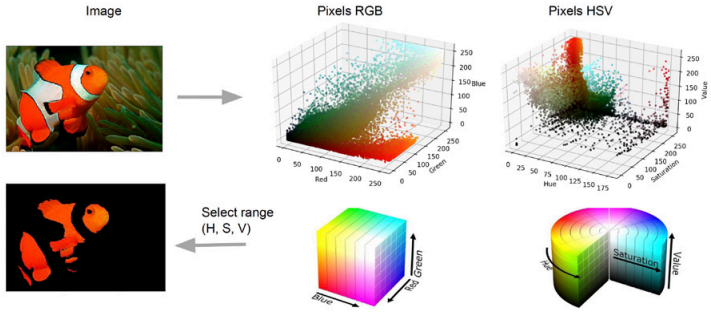
Segmentation through motion, color, texture,shape, context

Semantic Segmentation (Zuordnung)

Jedes Pixel wird einer Klasse zugeordnet.



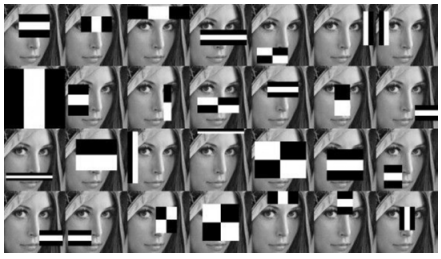
Color Segmentation



Example taken from: https://realpython.com/python-opencv-color-spaces/

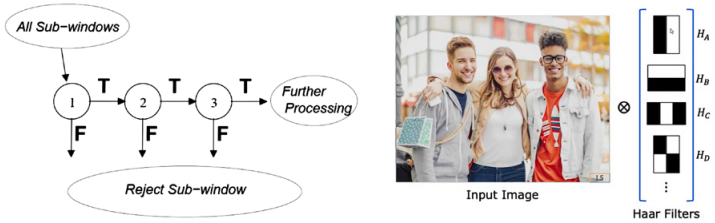
Face Detection

Cascaded Classifier



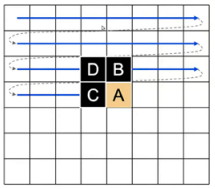
Haar filters (values 1 and -1) substract one part of an image from another part of an image. Do this with different sizes to detect different face sizes. There filters are learned.

In order to classify a face, all filters must be passed.
Early rejection of negative examples.



Integral image makes this process really fast. The Integral image let's you calculate the area of a region in an image.

Construct Integra Image



Raster Scanning

Let I_A and II_A be the values of Image and Integral Image, respectively, at pixel A.

$II_A = II_B + II_C - II_D + I_A$

Calculate the area of an image

$$Sum = II_P - II_Q - II_S + II_R$$

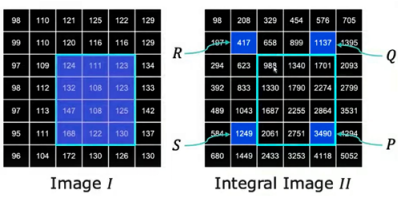


Image I | Integral Image II

Random Forests and **Boosting** (ML) is used to add more features and selection = Ensemble method (composition of indvidual models): Transforms a weak learning algorithm into a strong one.

Modern Computer Vision

Machine Learning

Paradigms

1. **Supervised Learning:** Labeled Data (features)
   Goal is to earn a function that maps input features to output labels.
2. **Unsupervised Learning:** Unlabeled Data. No output values.
   Clustering (goal is class identification)
3. **Reinforcement Learning:** Learning policies + rewards (feedback)

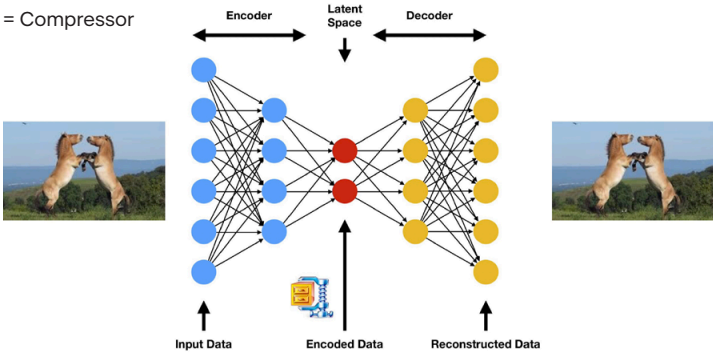
Techniques

Boosting: Method to transform a weak learning algorithm into a strong one.

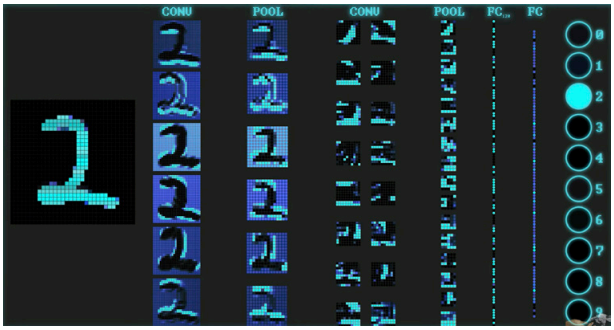
Bagging: Methode gegen overfitting. **Idee:** Jeder «bag» (train-data wird aufgeteilt) overfitted in andere Richtung. Der Durchschnitt der bags gleicht sich wieder aus.

Autoencoder

= Compressor

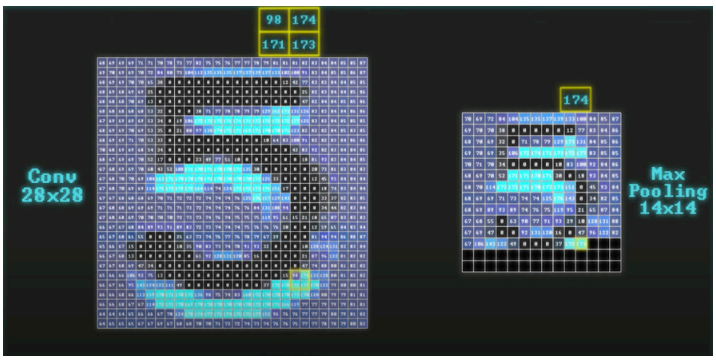


Convolutional Neural Network



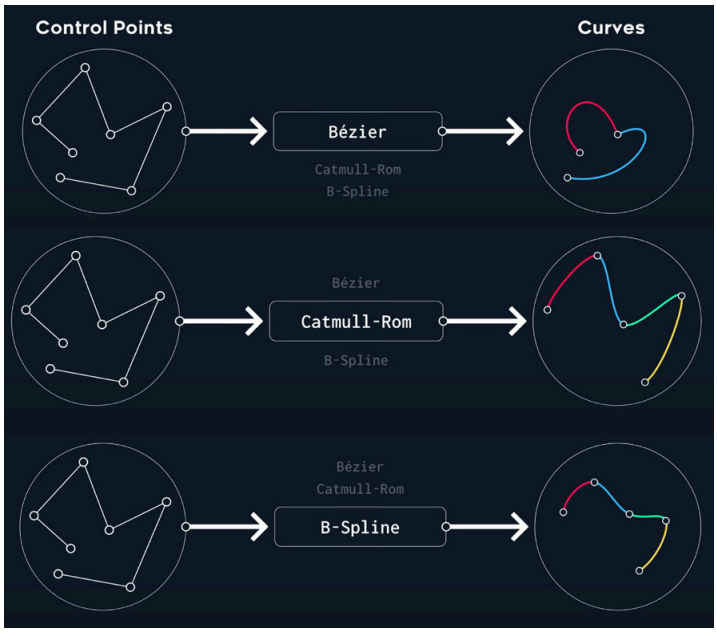
Convolutional Layer
First performs a convolution (linear filter) before applying an non-linear activation function (e.g. ReLU). Each convolutional layer detects more complicating shapes. Each node in the layer is a convolutional filter.

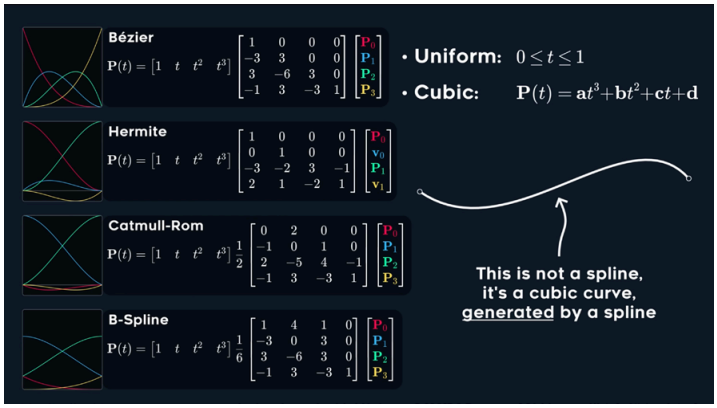
Pooling
Reduces the size of the input data without loosing important information (such as lines, corners, etc.), e.g. from from 28×28 to 14×14 pixels.
It does this by a sliding window 2×2 (non-linear convolution) and keeps the max value = **Max Pooling.**



Appendix

Splines = Curve Generators





Affine transformation: Preserves lines and parallelism, but not necessarily Euclidean distances and angles.