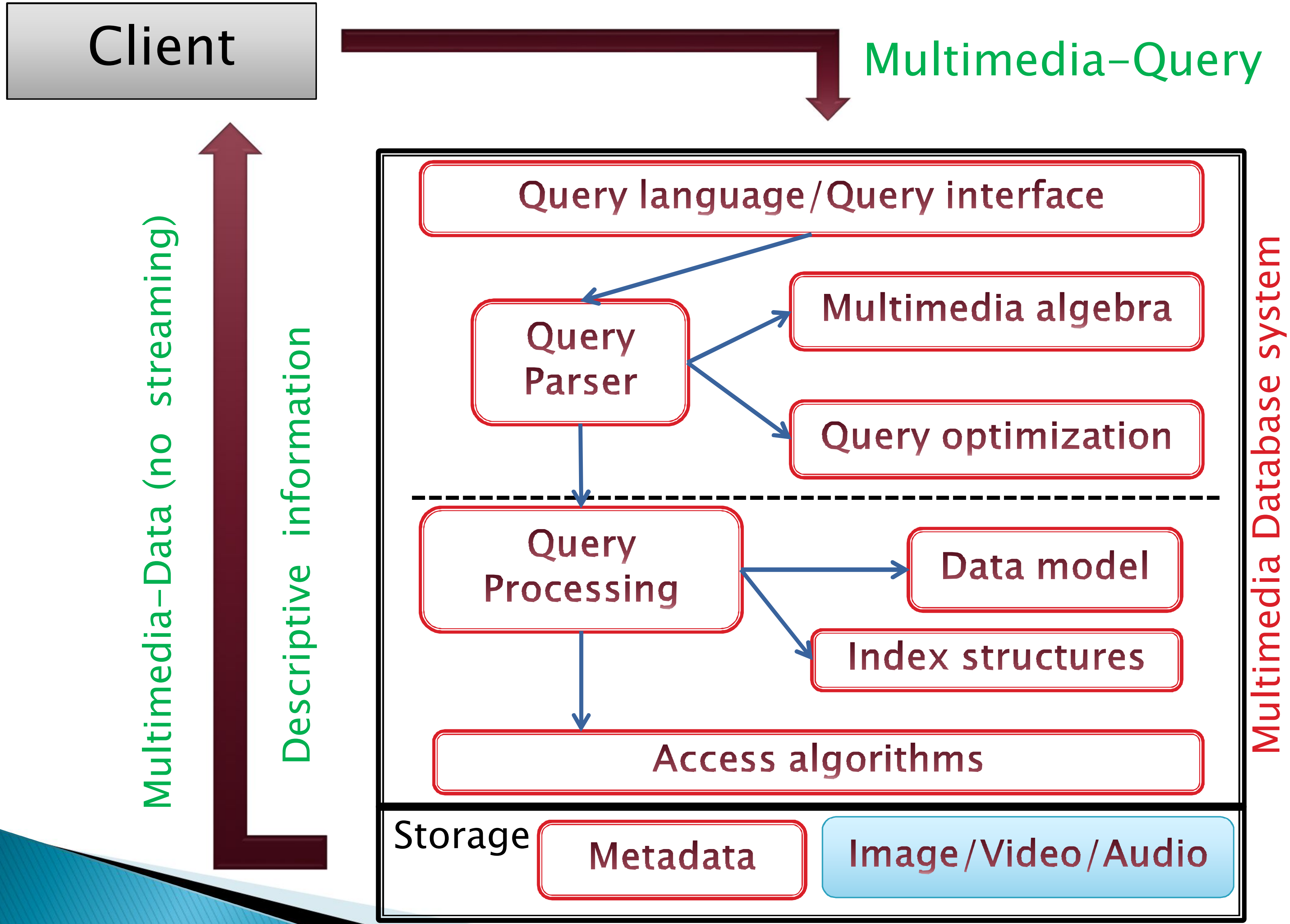
Multimedia Databases

The Image Medium

Prof. Dr. Michael Granitzer
Prof. Dr. Harald Kosch

Table of Contents
Image



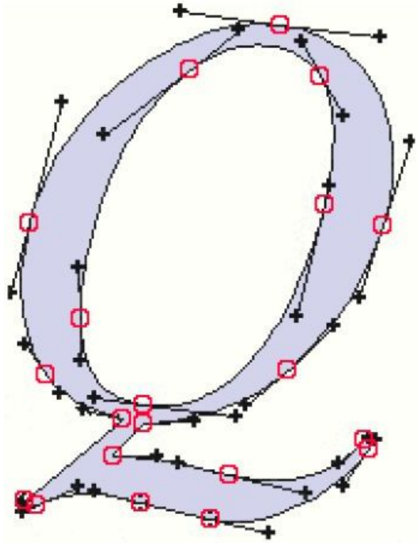
Client
Multimedia-Query
Query language/Query interface
Query Parser
Multimedia algebra
Query optimization
Query Processing
Data model
Index structures
Access algorithms
Storage
Metadata
Image/Video/Audio
Multimedia Database system
Multimedia-Data (no streaming)
Descriptive information

Table of Contents
Image

Vector Graphics – Generalities I

Vector graphics are mathematically and programmatically defined drawing instructions within a coordinate system.

- Geometric transformations can be easily and exactly applied to vector graphics :
  - Scale, rotate, move
  - Single image elements can be separated:
    - Levels, groups, object
  - Attributes of image elements can be modified:
    - Color of an area, thickness of a line etc.



Vector graphics – Generalities II

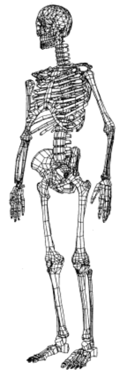
- Vector graphics formats
  - PostScript (.ps, .eps). PDF
  - Windows Metafile (*.wmf, *.emf)
  - Corel Draw (*.cdr)
  - Scalable Vector Graphics (*.svg)
  - VRML (3D)
  - And more…
- Drawbacks of Vector Graphics: the images must be drawn in order to be visible:
  - Rendering (reproduction, interpretation)

Representation of graphics

Computer generated Images based on models

Mostly Geometric models

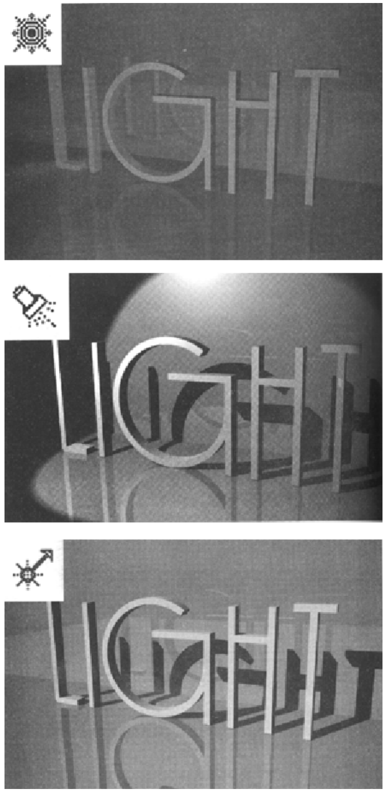
- Graphic libraries of 2D- and 3D-elementary objects (primitives)
- Associated functions, e.g.: rotation



Operations for graphics

Highly dependent on the model

- Editing
  - Primitive and structures (spatial relations)
- Shading
  - Lighting effects on surfaces (mirroring etc.)
- Mapping
  - Textures
  - Irregularities of the surface, reflection
- Lighting
  - Position of light sources
- Display
  - 3D- to 2D-representation (model $\rightarrow$ screen)
- Rendering
  - Generation of an image based on the model and parameters (Resolution etc.)



Scalable Vector Graphics (SVG)

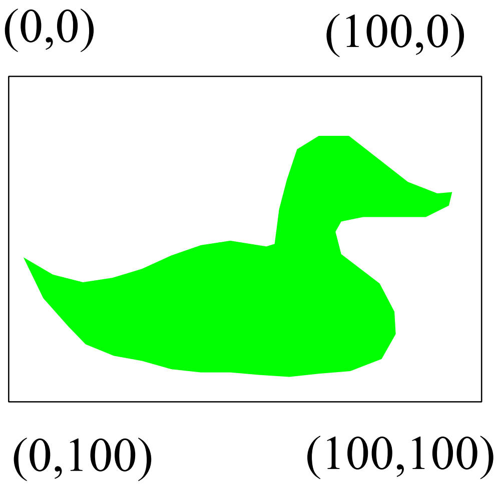
- Language for 2D-Graphics in XML
  - Can be combined with other Web standards
- Three types of graphical objects
  - Shapes (Paths of curves and straight lines)
  - Images (Raster graphics)
  - Text
- Graphical objects may be
  - grouped
  - „styled“ (CSS)
  - transformed
  - combined
- Goodies:
  - Global transformations
  - „Clipping paths“ (flexibly cut images)
  - Alpha masks (Objects transparency)
  - Filter effects
  - Object templates
- SVG drawings are potentially
  - interactive and
  - Dynamic

https://wiki.selfhtml.org/wiki/SVG

http://www.w3.org/Graphics/SVG/
Current v2011

SVG Coordinate System

- (0,0) is top left
- User Coordinate System default in screen-pixel units (e.g. 100 units in SVG are 100 pixel on screen)
- Units can be change



- Width / head / units defined in header:

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xmlns="http://www.w3.org/2000/svg"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    width="320" height="220">
```

SVG Example

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xmlns="http://www.w3.org/2000/svg"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    width="320" height="220">
  <rect width="320" height="220" fill="white" stroke="black""/>
   <g transform="translate(10 10)">
     <g stroke="none" fill="lime">
       <path d="M 0 112 L 20 124 L 40 129 L 60 126 L 80 120
          L 100 111 L 120 104 L 140 101 L 164 105 L 170 103
          L 173 80 L 178 60 L 185 39 L 200 30 L 220 30
          L 260 61 L 280 69 L 290 68 L 288 77 L 272 85
          L 250 85 L 230 85 L 215 88 L 211 95 L 215 110
          L 228 120 L 241 130 L 251 149 L 252 164 L 242 181
          L 221 189 L 200 191 L 180 193 L 160 192 L 140 190
          L 120 190 L 100 188 L 80 182 L 61 179 L 42 171
          L 30 159 L 13 140 Z"/>
          </g> </g>
</svg>
```

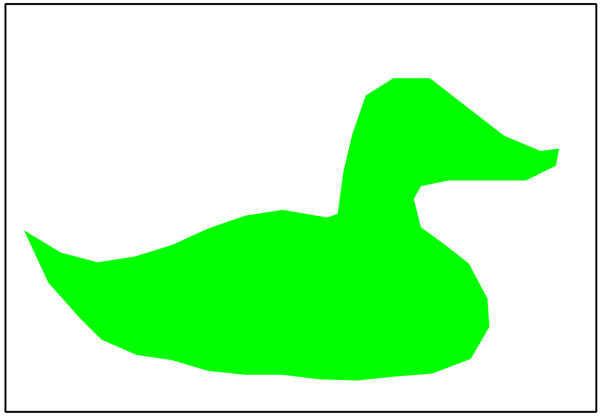


Table of Contents
Image

Bezier Curves

Properties of Parametric Curves

- Parametric curves are intended to provide the generality of polygons but with fewer parameters for smooth surfaces
  - Polygon have as many parameters as there are vertices (at least)
- Fewer parameters makes it faster to create a curve, and easier to edit an existing curve
- Parametric curves are easier to animate than polygon meshes

Parametric Curves

- The parametric form for a line:

$$x = x_0 t + (1-t)x_1$$
$$y = y_0 t + (1-t)y_1$$
$$z = z_0 t + (1-t)z_1$$

- x, y and z are each given by an equation that involves:
  - the parameter *t*
  - Some user specified control points, like x_0 and x_1

Hermite Spline (1)

- A *spline* is a parametric curve defined by *control points*
  - The term spline dates from engineering drawing, where a spline was a piece of flexible wood used to draw smooth curves
  - The control points are *adjusted by the user* to control the shape of the curve
- A *Hermite spline* is a curve for which the user provides:
  - The endpoints of the curve
  - The parametric derivatives of the curve at the endpoints
    - The parametric derivatives are dx/dt, dy/dt, dz/dt

Hermite Spline (2)

- Say the user provides

$$x_0, x_1, x_0', x_1'$$

- A cubic spline has degree 3, and is of the form:

$$x = at^3 + bt^2 + ct + d$$

- We have constraints to determine a,b,c,d
  - The curve must pass through x_0 when $t=0$
  - The derivative must be x'_0 when $t=0$
  - The curve must pass through x_1 when $t=1$
  - The derivative must be x'_1 when $t=1$

Hermite Spline (3)

- Solving for the unknowns gives:

$$
\begin{aligned}
a &= -2x_1 + 2x_0 + x_1' + x_0' \\
b &= 3x_1 - 3x_0 - x_1' - 2x_0' \\
c &= x_0' \\
d &= x_0
\end{aligned}
$$

Bezier Curves (1)

- Different choices of basis functions give different curves
  - In Hermite case, two control points define endpoints, and two more define parametric derivatives
- For Bezier curves, two control points define endpoints, and two control the tangents at the endpoints in a geometric way

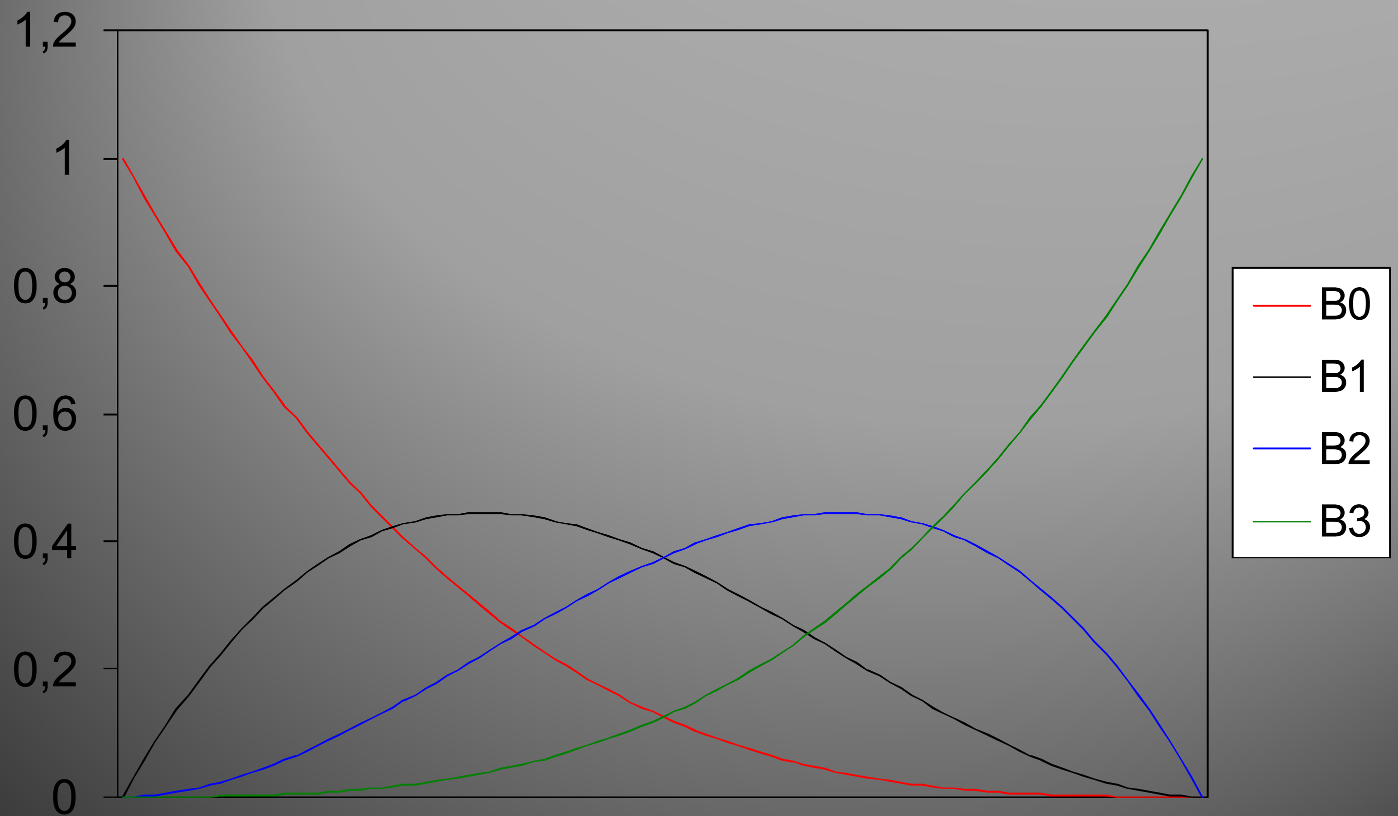
Bezier Curves (2)

- The user supplies d control points, $\boldsymbol{p}_i$
- Write the curve as:

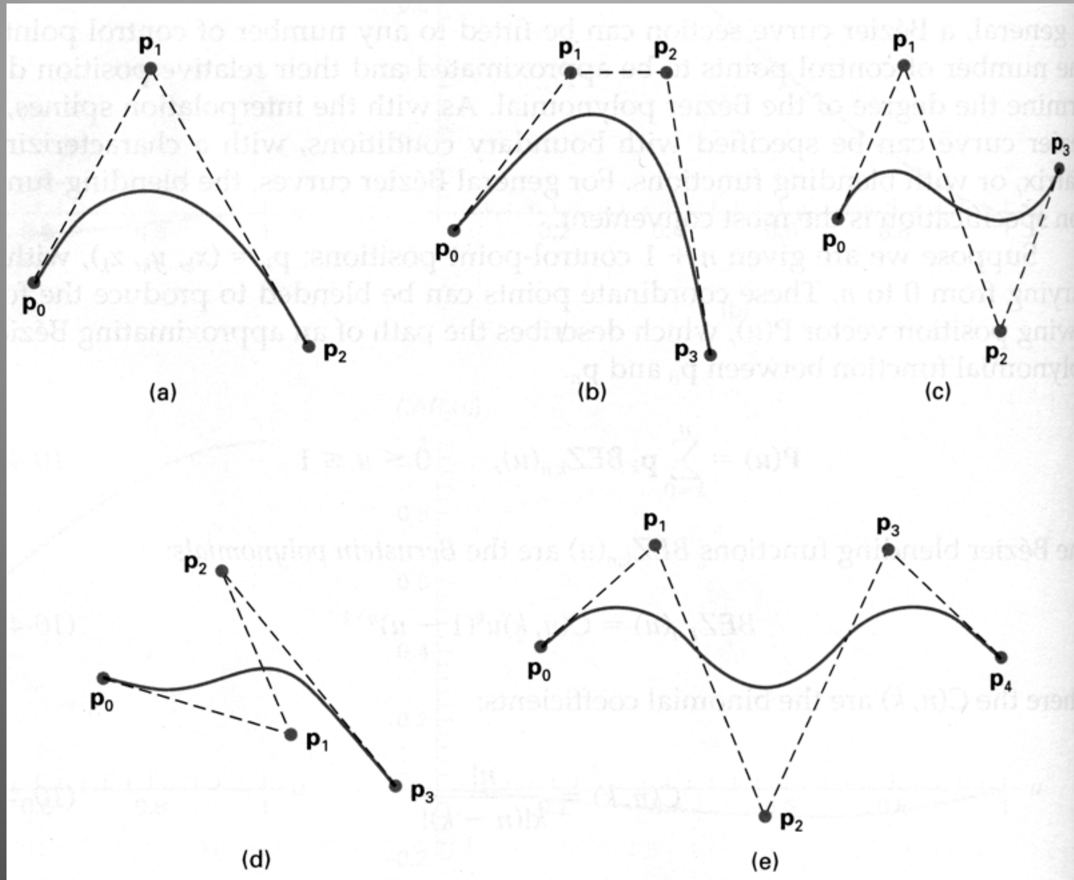
$$\mathbf{x}(t) = \sum_{i=0}^{d} \mathbf{p}_i B_i^d(t) \qquad B_i^d(t) = \binom{d}{i} t^i (1-t)^{d-i}$$

- The functions B_i^d are the *Bernstein polynomials* of degree d

Bezier Basis Functions for $d=3$



Some Bezier Curves

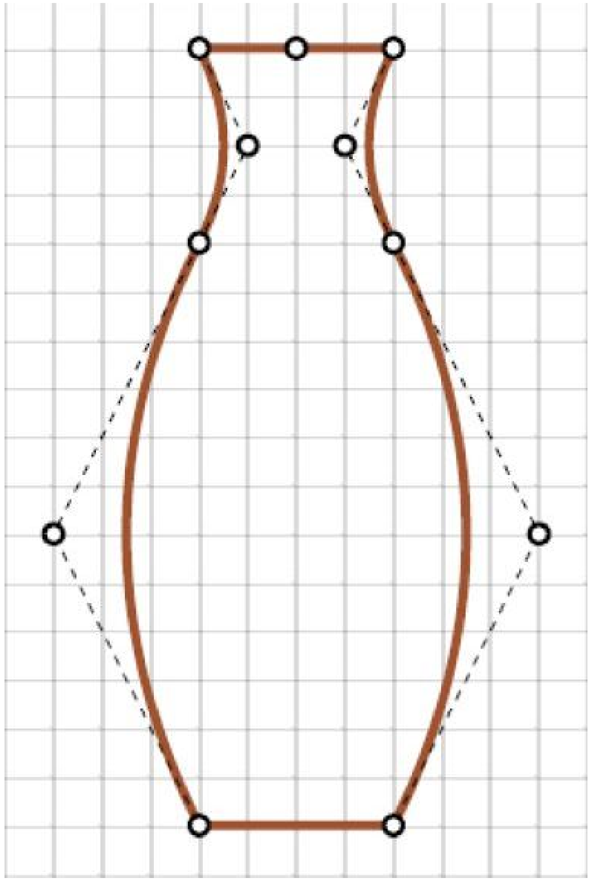


Bezier Curve Properties

- The first and last control points are interpolated
- The tangent to the curve at the first control point is along the line joining the first and second control points
- The tangent at the last control point is along the line joining the second last and last control points
- The curve lies entirely within the convex hull of its control points

SVG: Splines Example I

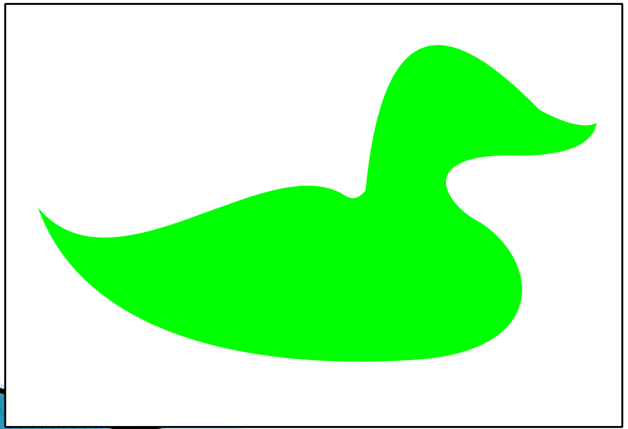
```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
  "http://www.w3.org/TR/2001/REC-SVG-
20010904/DTD/svg10.dtd">

<svg width="12cm" height="7.2cm"
       viewBox="0 0 1000 600"
       xmlns="http://www.w3.org/2000/svg" >

  <path stroke="sienna" stroke-width="2"
           fill="none"
   d="M 80,180
       Q 50,120 80,60
       Q 90, 40 80,20
       Q 100, 20 120,20
       Q 110, 40 120,60
       Q 150,120 120,180Z" />
</svg>
```



SVG Example II – Bezier Paths

With Berzier curves

```
<path d="M 0 312
C 40 360 120 280 160 306 C 160 306 165 310 170 303
C 180 200 220 220 260 261 C 260 261 280 273 290 268
C 288 280 272 285 250 285 C 195 283 210 310 230 320
C 260 340 265 385 200 391 C 150 395 30 395 0 312 Z"/>
```



Without Berzier curves

```
<path d="M 0 112 L 20 124 L 40 129 L 60 126 L 80 120
  L 100 111 L 120 104 L 140 101 L 164 105 L 170 103
  L 173 80 L 178 60 L 185 39 L 200 30 L 220 30
  L 260 61 L 280 69 L 290 68 L 288 77 L 272 85
  L 250 85 L 230 85 L 215 88 L 211 95 L 215 110
  L 228 120 L 241 130 L 251 149 L 252 164 L 242 181
  L 221 189 L 200 191 L 180 193 L 160 192 L 140 190
  L 120 190 L 100 188 L 80 182 L 61 179 L 42 171
  L 30 159 L 13 140 Z"/>
```

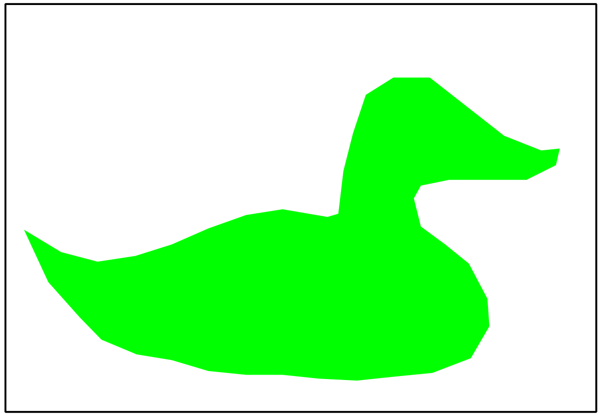
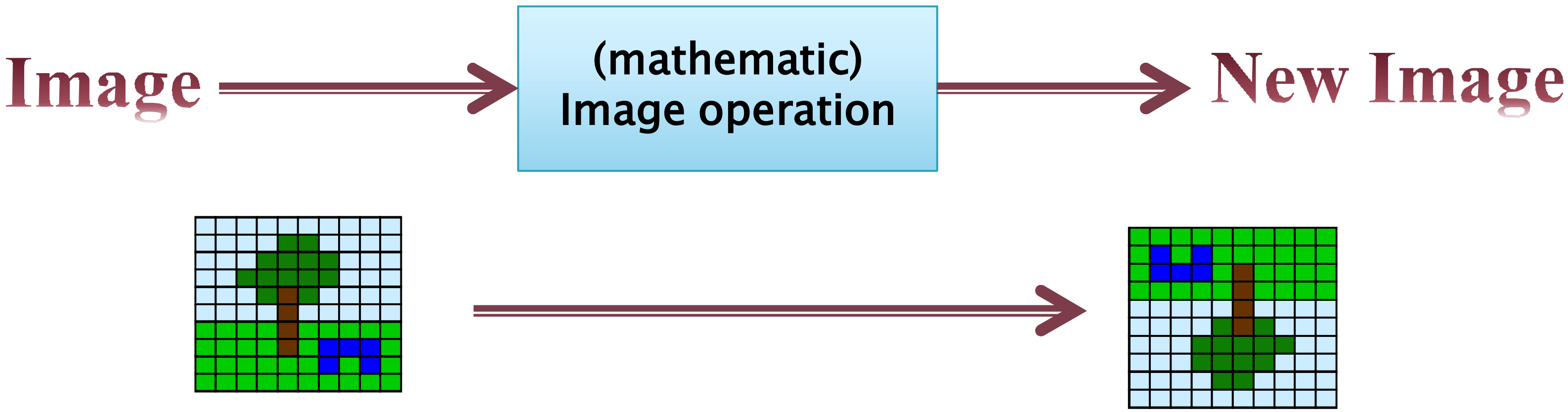


Table of Contents
Image

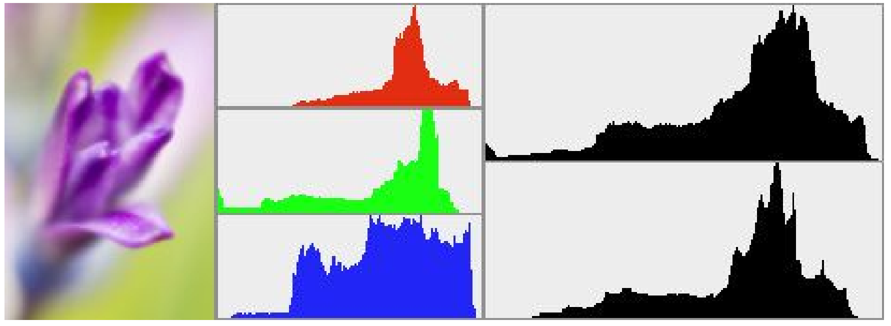
Types of Image Manipulation



- Image Point Operations
  - Applied to an Image
- Neighborhood operations/Filter
- Geometric Operations

Color-Histogram

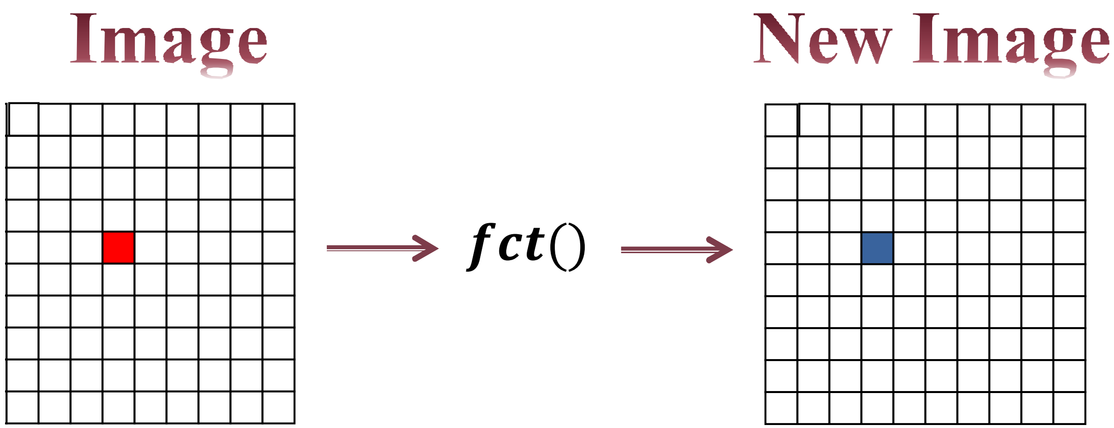
- $h_c(i)$: Number of Pixels with a particular color value i for a channel c
  - Allows to identify basic properties of images (e.g. low contrast images)
  - Basis for point operations (e.g. contrast enhancements)



http://blog.epicedits.com/2007/04/14/working-with-image-histograms/

Image Point Operations

Image Point Operation



$$PixelNew(x, y) = fct(Pixel(x, y))$$

- Brightness and contrast adjustement
- Color corrections, tonal value corrections
- Image overlays, etc.

Image Point Operation II

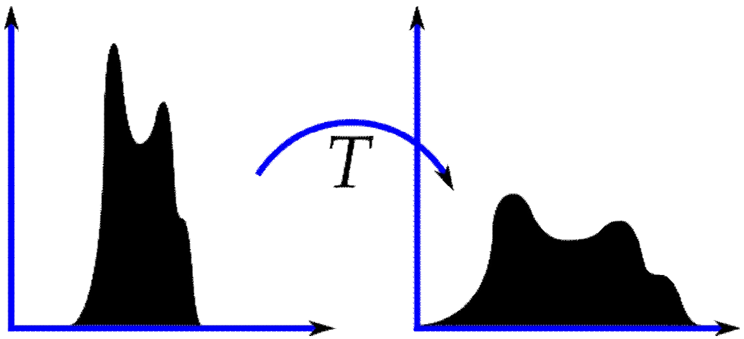
- Simple Example:

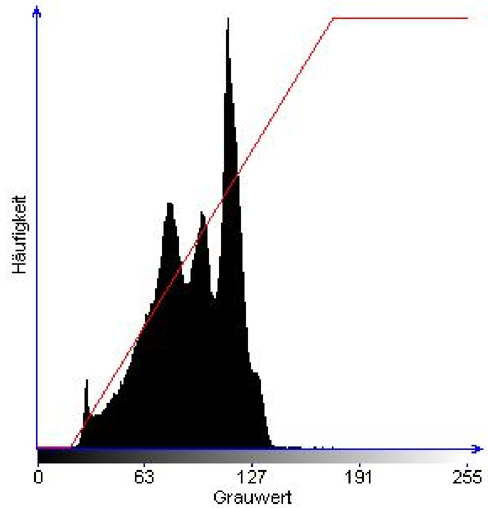
  Negative of a greyscale image

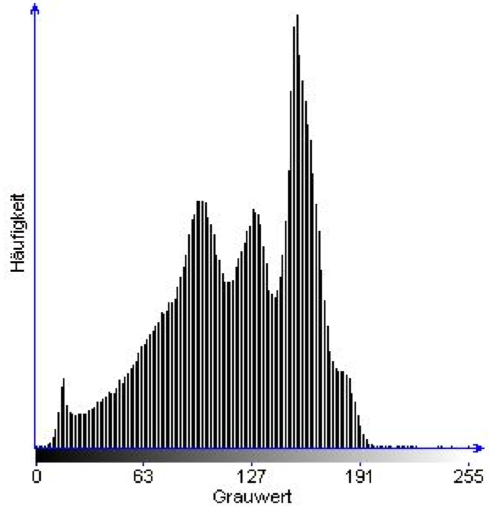
$$PixelNew(x, y) = 255 - Pixel(x, y)$$

Histogramm Spreading

New Histogramm $h_c(i)$





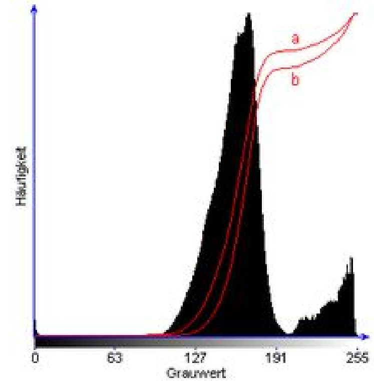


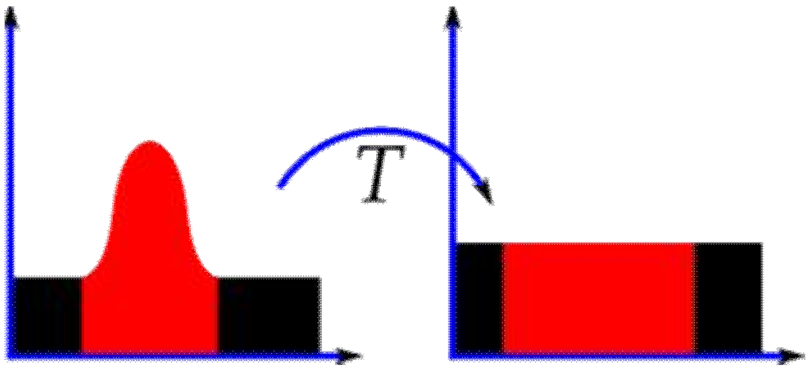
Histogramm Equalization

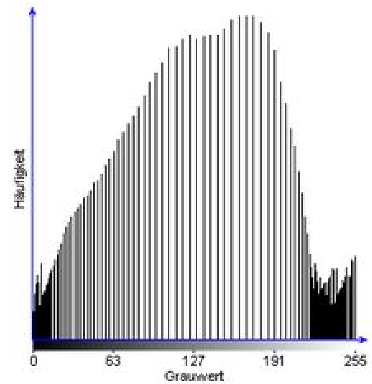
- Cumulative Histogramm

$$H(i) = \sum_{j=0}^{i} h_c(j)$$

- New Histogramm





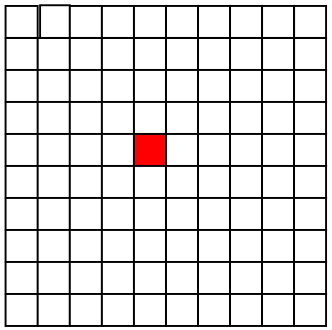


Filter

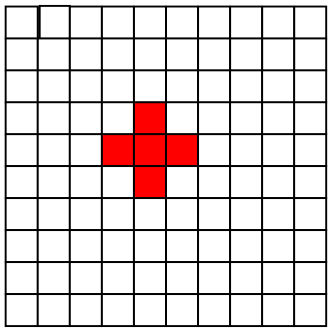
Neighborhood operations/Filter

- Neighborhood is defined by the following formula:

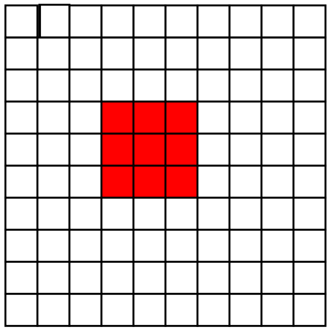
$$N_c(x, y) = \{(i, j): 0 < (i - x)^2 + (j - y)^2 \leq c\}$$



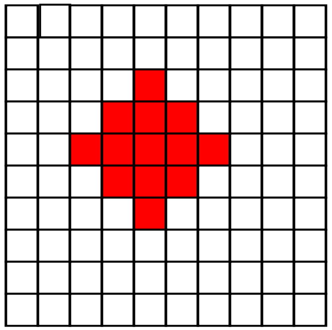
c=0



c=1

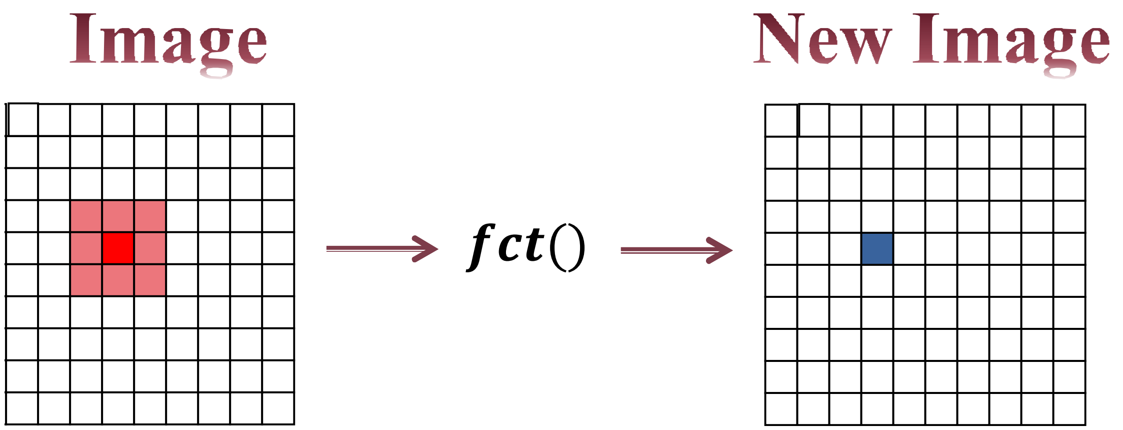


c=2



c=3

Filter =Neighborhood operations

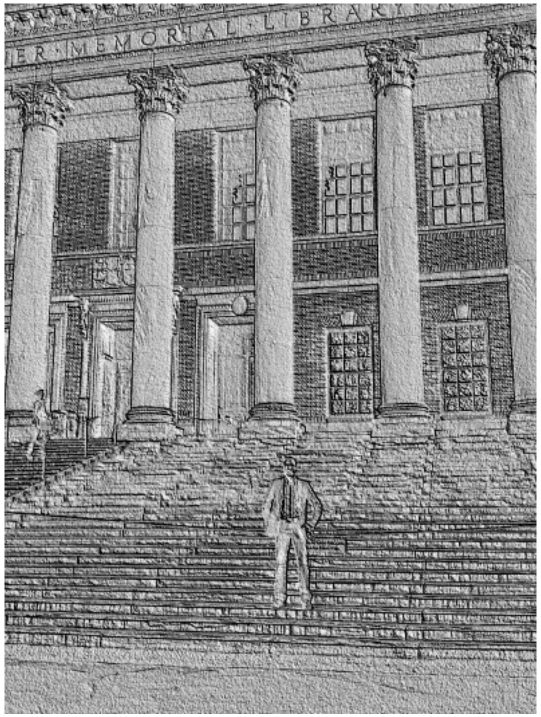


$$B^{new}(x, y) = fkt(N_c(x, y))$$

- Enhancement and blurring
- Denoising
- Edge detection

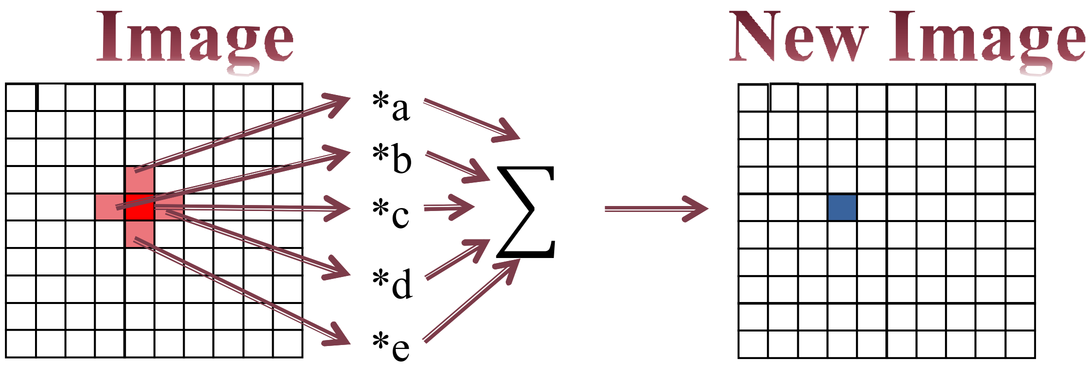
Filter Example:
Vertical relief generation

$$B^{new}(x, y) = 2 * B(x, y) - 2 * B(x, y - 1) + 128$$

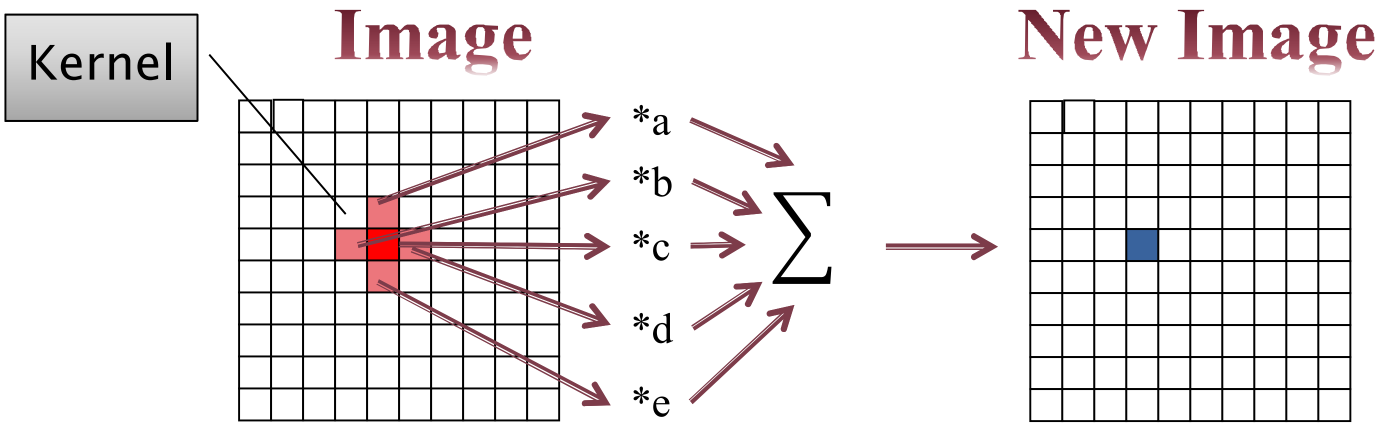


Linear Filter = Convolution

- Linear Filter: Each (Pixel-)value is replaced by a linear weighted combination of the (Pixel-)values of its „neighborhood“ (the kernel region).



Linear Filter



$$B^{new}(x,y) = a * B(x, y-1) + b * B(x-1, y) + c * B(x,y) + d * B(x+1, y) + e * B(x, y+1)$$

M x N Kernel with Folding

$$K = \begin{bmatrix} 0 & a & 0 \\ b & c & d \\ 0 & e & 0 \end{bmatrix}$$

$$B^{new}(x,y) = \mathbf{K} * \boldsymbol{B}(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=-M}^{M} \sum_{j=-N}^{N} K(i,j) \cdot B(x-i, x-j)$$

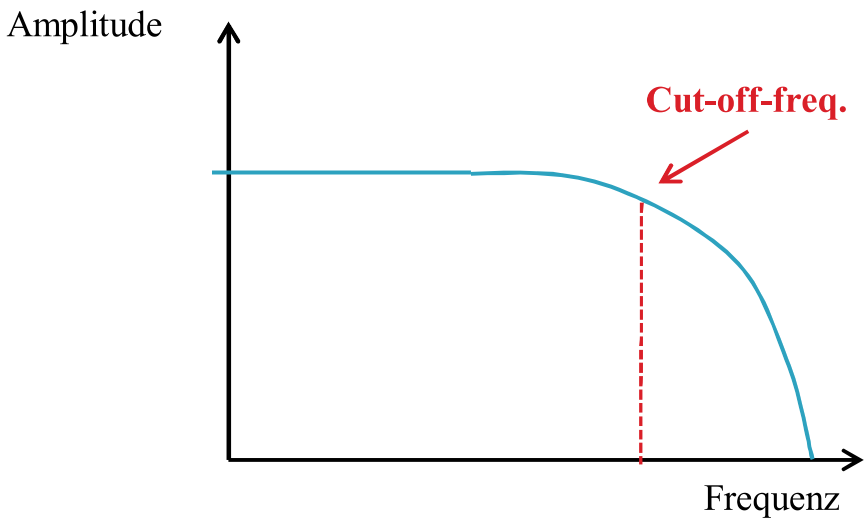
Mode of operation of linear filters

- Image point operations (gradation curves, K=1x1) scale Pixel-(values) i.e. they modify the amplitude ratios

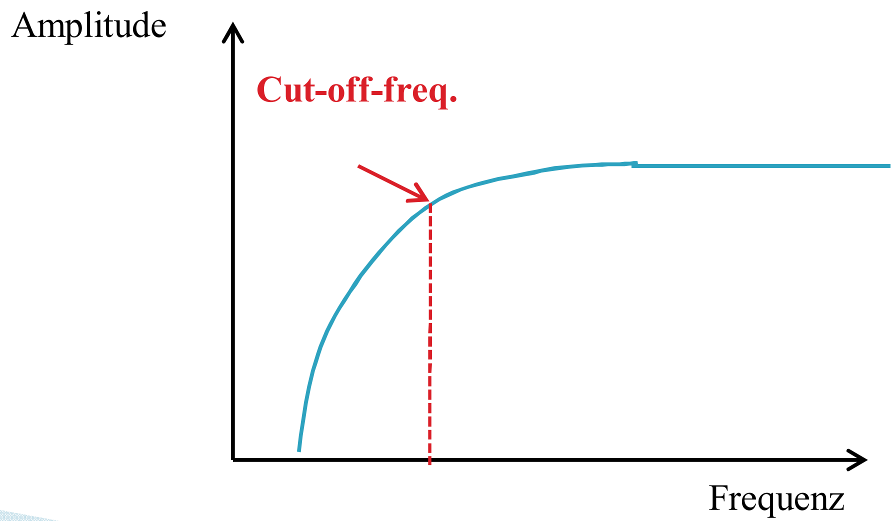
- Linear neighborhood operators (Filter) scale the spectral domains i.e. they modify the frequency ratios

Filter properties

- ## Low-pass filter
  - Frequencies above a specific one, the cut-off-frequency are reduced, whereas lower frequencies can „pass“

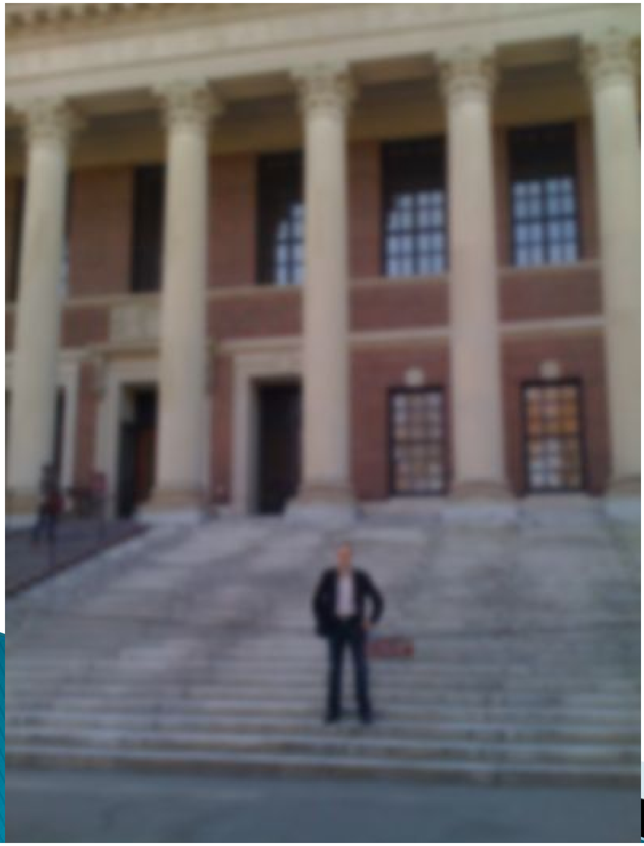


- ## High-pass-Filter
  - Frequencies below the cut-off-frequency are reduced, the higher one can pass.



Example
Low- and high-pass filter

Low-pass filter



High-pass filter

The End