Image processing

About

- Broad field that includes low-level operations as well as complex high-level algorithms
  - Low-level image processing
  - Computer vision
  - Computational photography
- Several procedures and concepts of that are frequently used in the context of multimedia systems
- Can also be applies to videos (frame by frame)

Image as a matrix

- 2D array (width x height)
- Channels
  - RGB = 3 channels
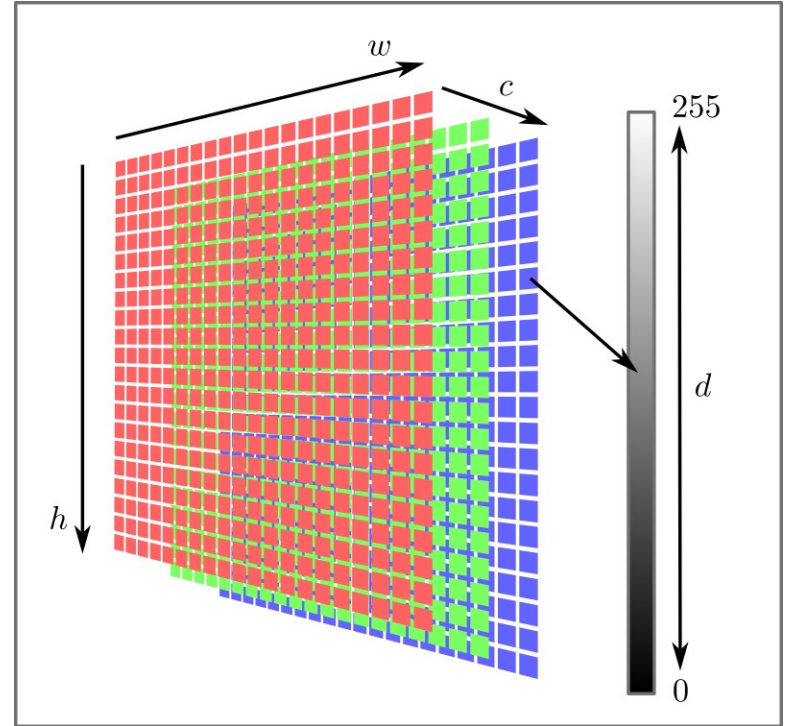 - Sampling resolution

Image as a function

- Image is a 2D function: $f : \mathcal{R}^2 \to \mathcal{R}$
  - Defined over a rectangle $[0, H] \times [0, W]$
  - Has a finite range $[0, 1]$
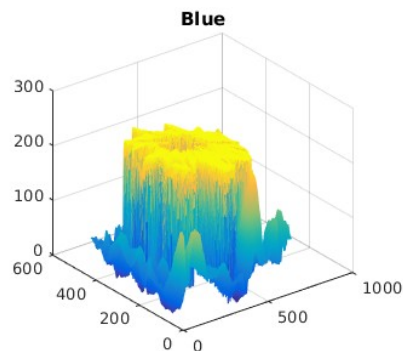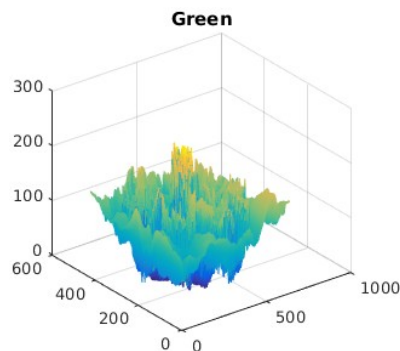 - Color image is defined as a triplet of functions
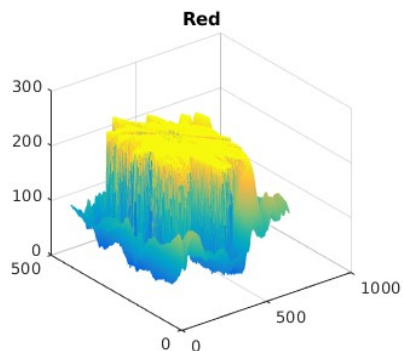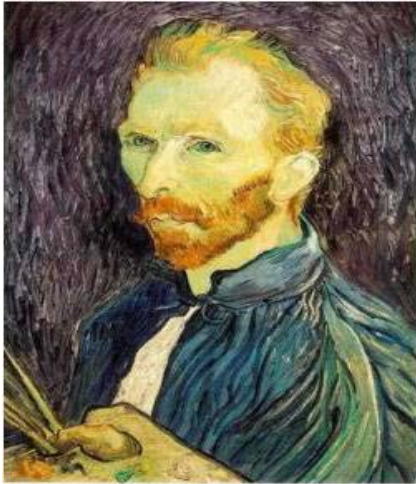
Image operations

- Intensity transformations
  - Pixel-wise
  - Histogram
  - Filtering
- Geometrical transformations
  - Linear
  - Local
- Complex operations
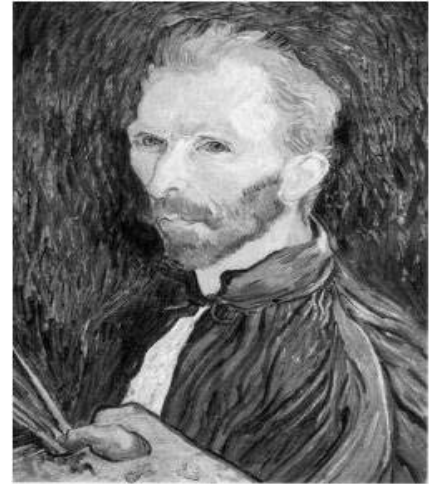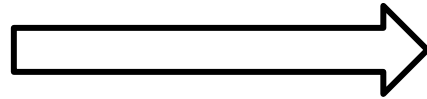 - Context-aware resizing
  - Compositing
  - Segmentation

Conversion to grayscale
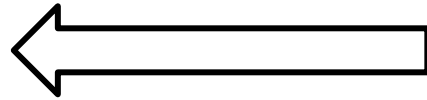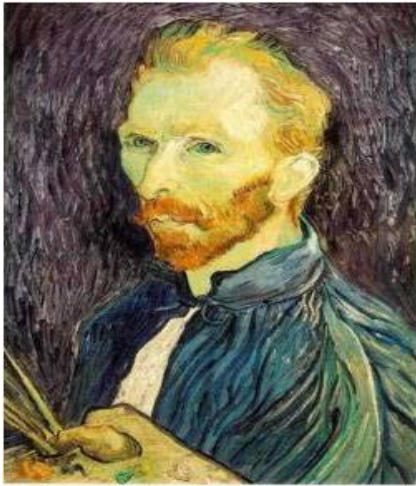
- From RGB: (weighted) averaging of channels

$$V = (R + G + B) / 3$$

$$V = 0.299\ R + 0.587\ G + 0.144\ B$$

Conversion from grayscale
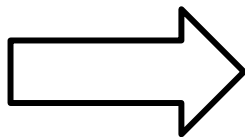
- How is gray represented in RGB color space?

Pixel-wise operation: negation

8-bit intensities are defined on interval from 0 to 255

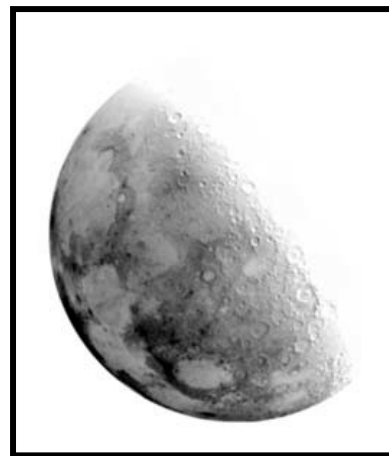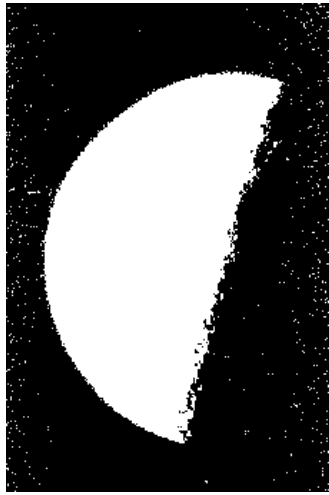Image negation of image A is B = (255 – A)

Image threshold

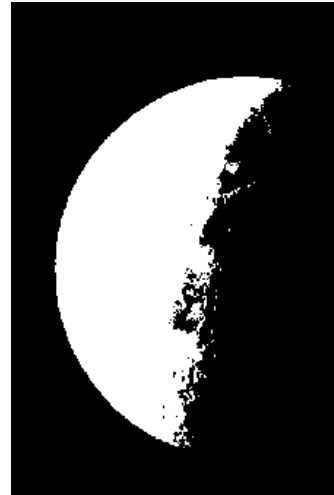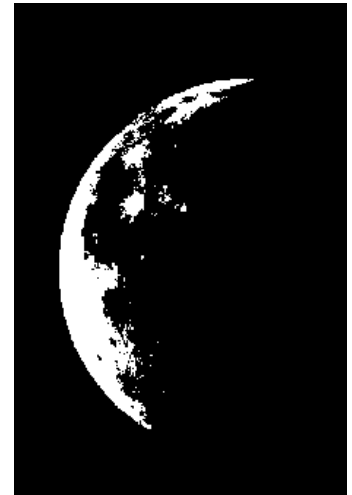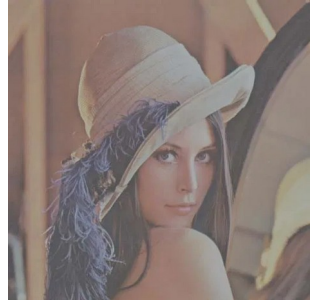Pixel values higher than value T are set to 1, others to 0
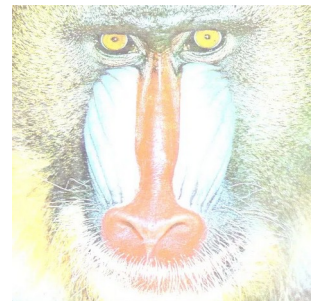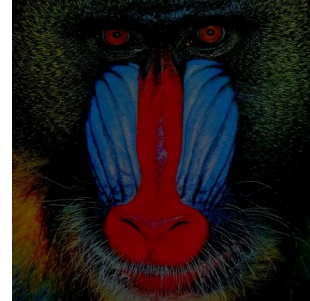

| I | I>10 | I>100 | I>200 |

Object segmentation: determining a suitable threshold is not trivial

Brightness and contrast

- **Brightness** – absolute value of all pixels

- **Contrast** – difference between minimum and maximum pixel
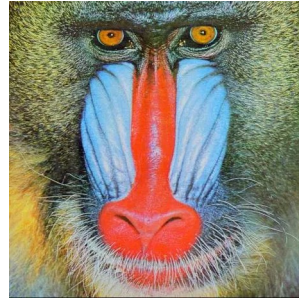
$$\mathrm{f}(\mathrm{x}) = \alpha x + \beta$$
$$\mathrm{f}(\mathrm{x}) = \alpha(x - 128) + 128 + \beta$$

Nonlinear transformations

- Parametric function that maps source values **r** to destination values **s**.
- Exponential function family $s = cr^{\gamma}$
- Parameter c is usually 1
- Parameter r is in [0, 1]

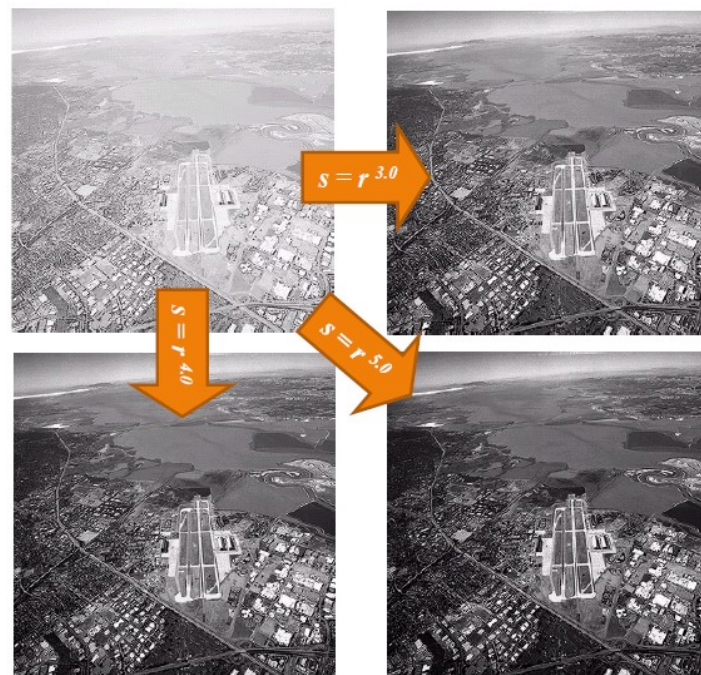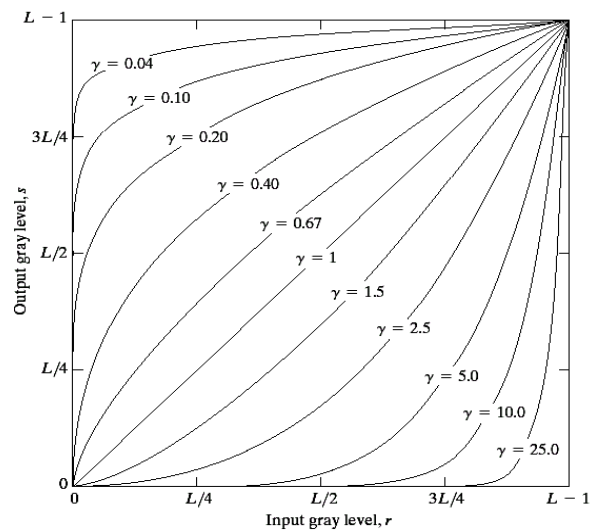# Distribution of values in images

- How to adjust values based on the image?

- Use image-specific statistics - histograms

Count: 1920000   Min: 0
Mean: 118.848   Max: 251
StdDev: 59.179   Mode: 184 (30513)

Histogram

- Frequency of different pixel values
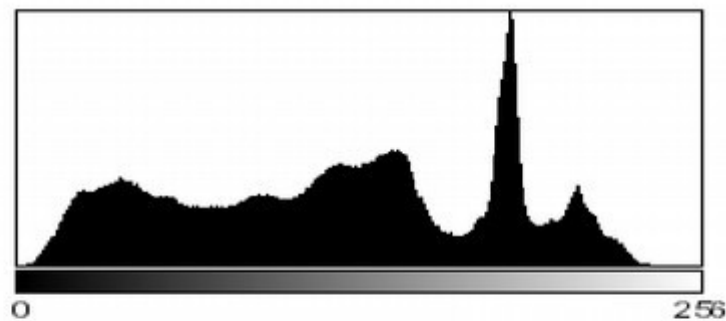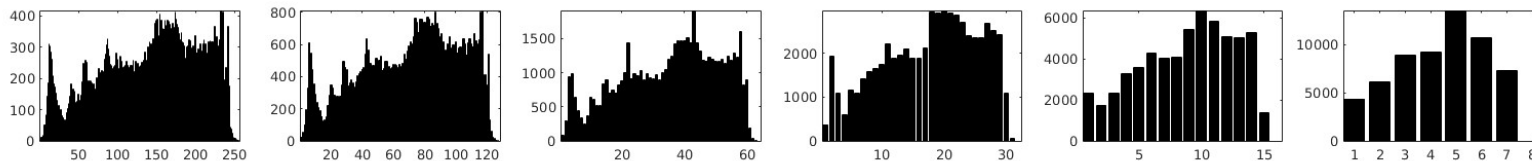 - How often they occur in image
  - Sub-sampling into cells/buckets
- Robust description
  - Rotation
  - Translation
  - Scale
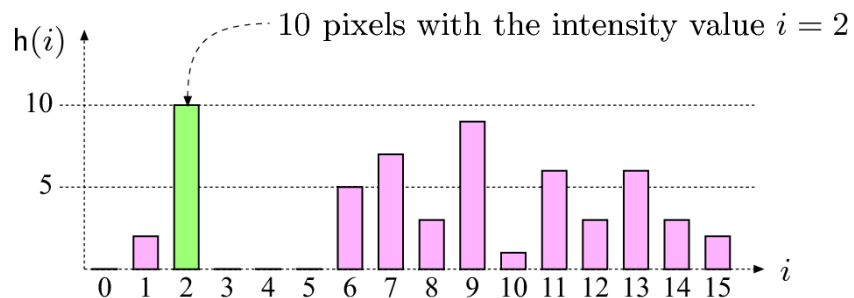
Histogram and image quality

- Increase/reduce brightness:
  - Histogram shifts left/right
- Increase/reduce contrast:
  - Histogram is shrinking/stretching

(a)   (b)   (c)
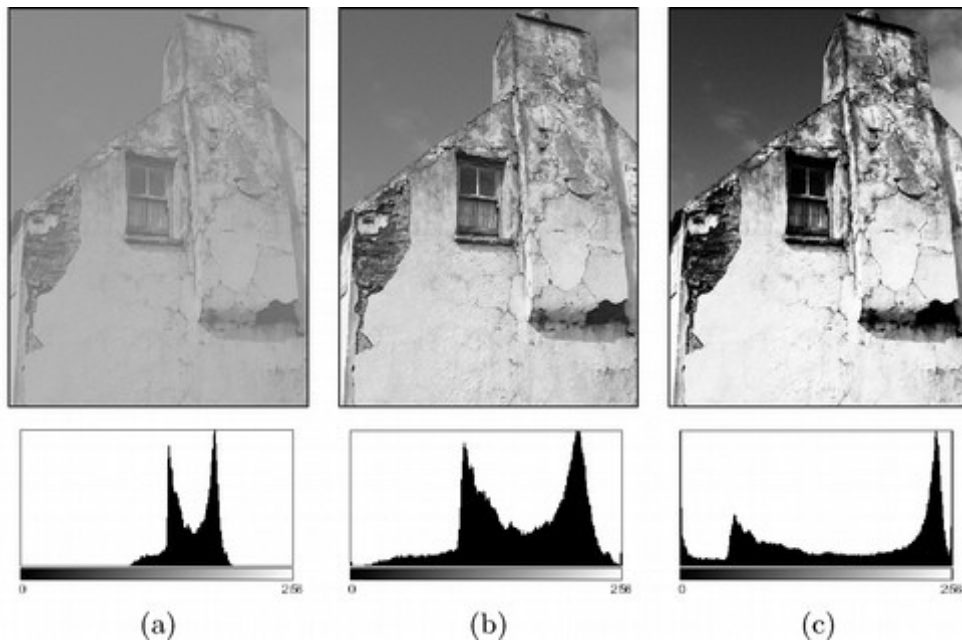
Histogram stretching

$$f_{ac}(a) = a_{min} + (a - a_{low}) \cdot \frac{a_{max} - a_{min}}{a_{high} - a_{low}}$$

Operation performed on each pixel individually.

input

histogram

output
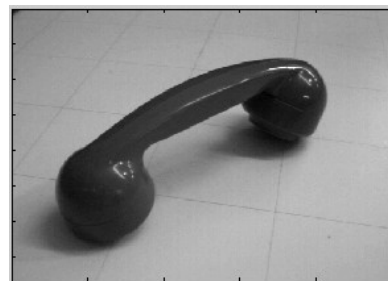
histogram
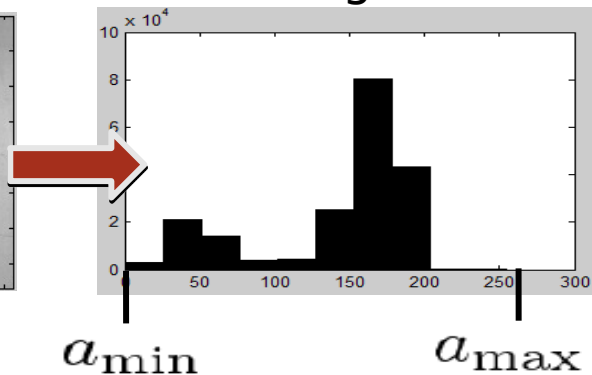
Histogram equalization

- Cumulative histogram - value dynamics
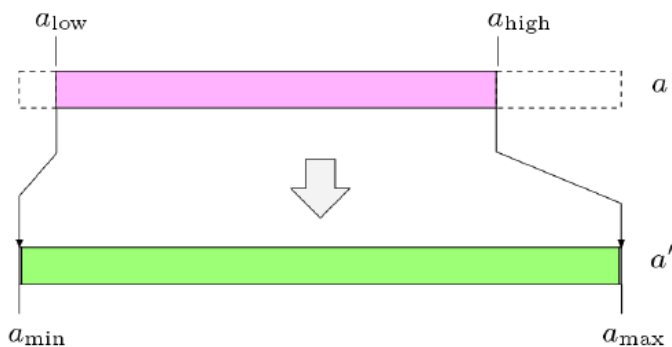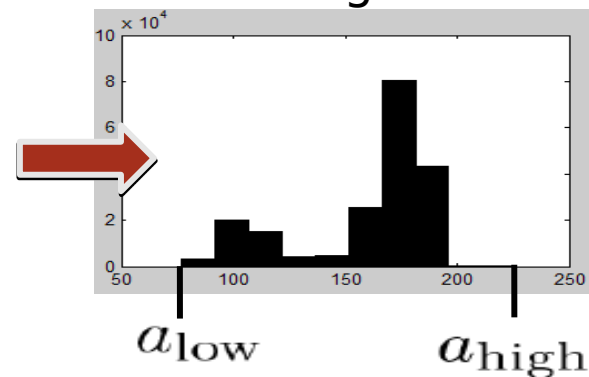- Desired dynamics is uniform
- Transform image values so that the cumulative histogram is diagonal.

$$H(i) = \begin{cases} h(0) & \text{for } i = 0 \\ H(i-1) + h(i) & \text{for } 0 < i < K \end{cases}$$
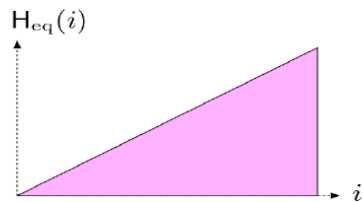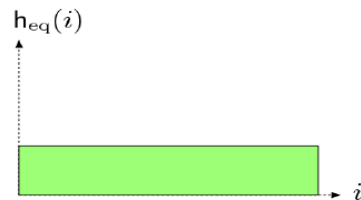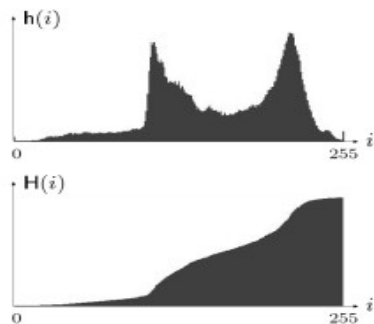
Equalization algorithm

- Compute 256-bin histogram of image I (h).

- Compute cumulative histogram (hc).

- Normalize (hc) with maximum value, multiply by 255 (hnc)

- Use (hnc) as a lookup table to transform individual pixels.
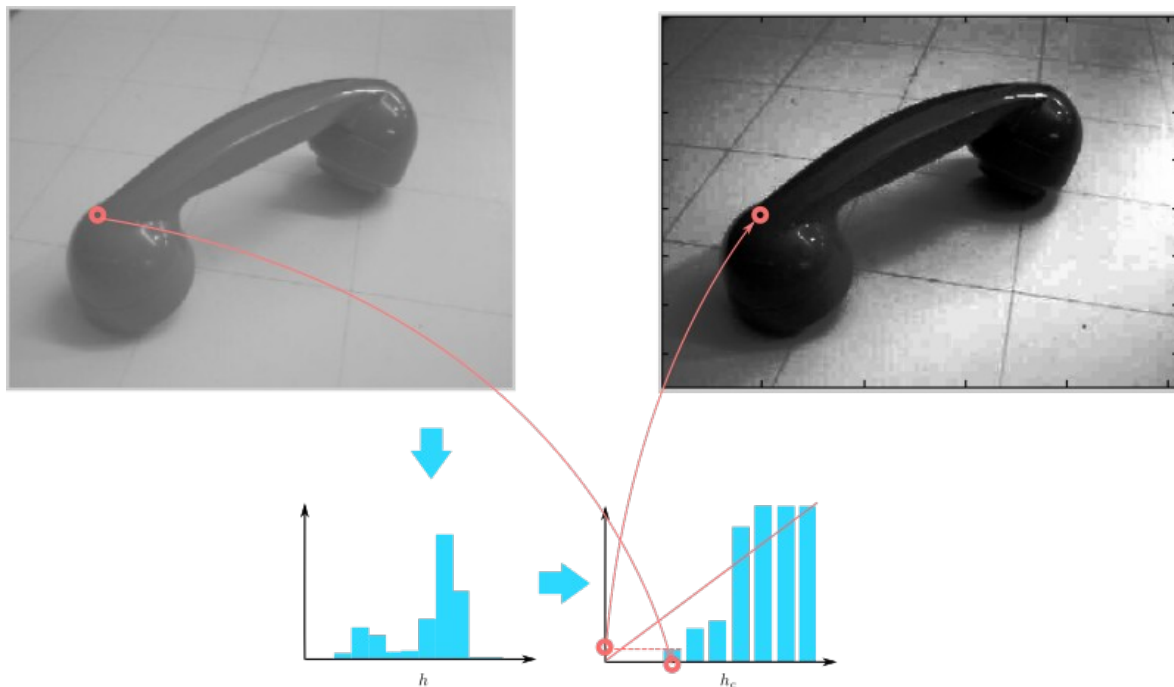
Histogram equalization in color images

Original

RGB Equalized Independently

Because each channel is transformed independently, the resulting color changes as well.

Luminance Equalization

Transform to color space with separate luminance channel, equalize only intensity

Histogram and thresholds

- Analyze histogram to find good threshold

- Bi-modal histogram

- Otsu method

  - Minimize variance of foreground and background
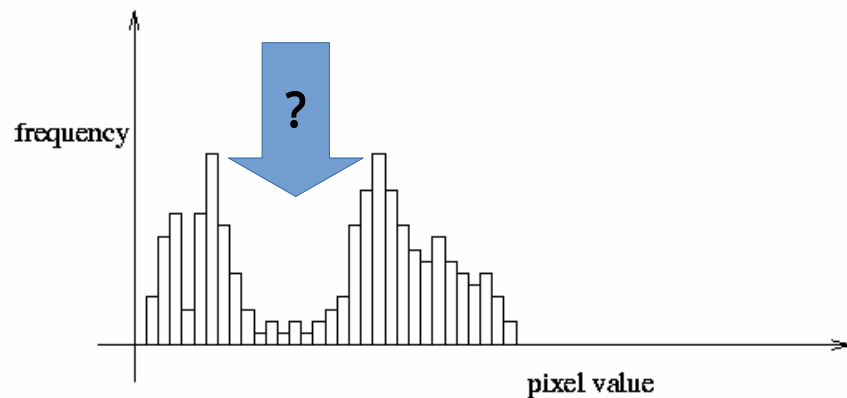
Color histogram

- 3 histograms
  - Each component separate
  - No correlation
  - Less space
- 3D histogram
  - Image color is a 3D index
  - More specific
  - More space
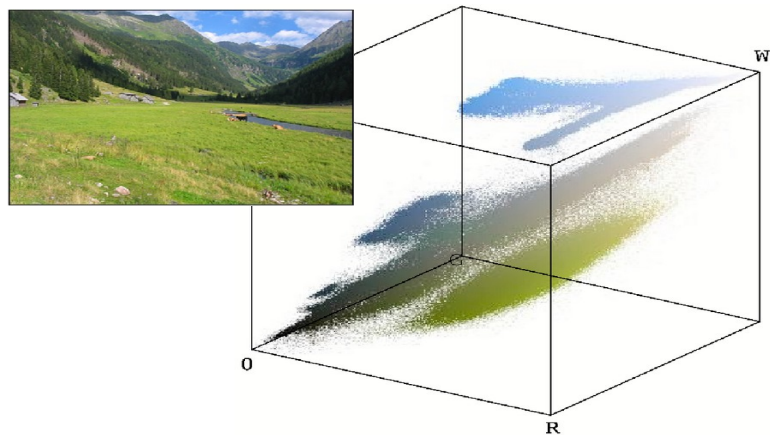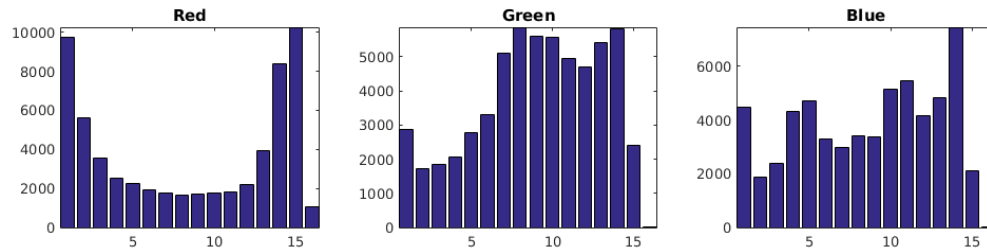
Filtering

- Resulting value dependent on the neighborhood
- Linear filters
  - Convolution / correlation
  - Kernel function (parameters)
  - Associativity, separability
- Nonlinear filters
  - Arbitrary (local) operation
  - Max, min
  - Median
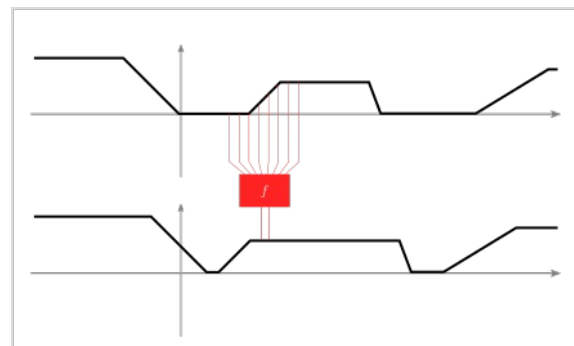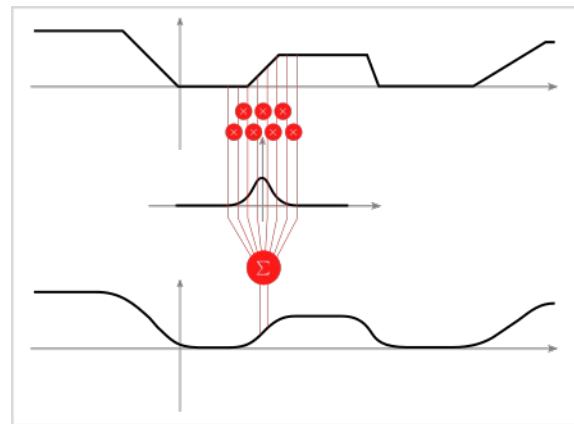
Simple, yet complex

- Single linear filter
  - Blur, directional blur
  - Edges, blobs
- Multiple linear filters that are learned
  - Extract specific properties
  - Data-driven
  - Convolutional neural networks

Linear filters

image
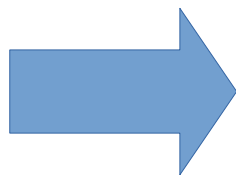
kernel
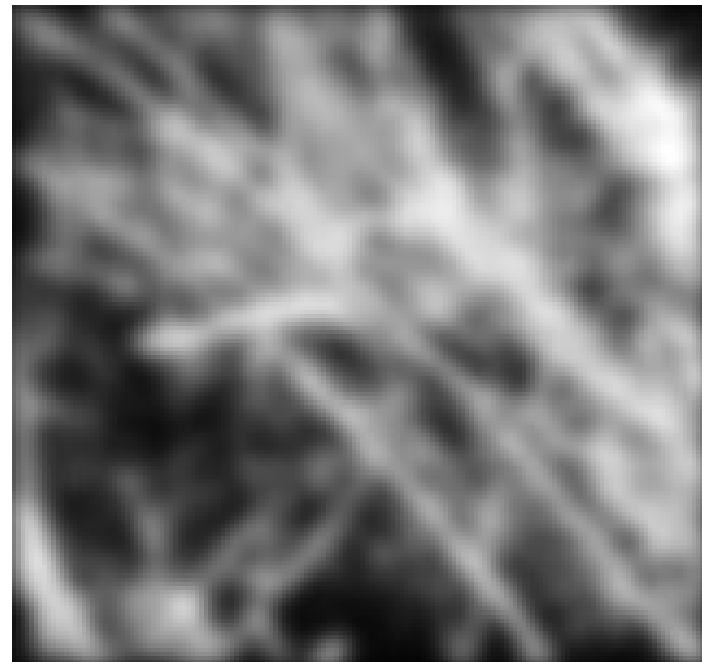
result

In a nutshell

- How to compute filter response in individual pixel?
  - Transpose kernel (convolution) and align its center with the pixel
  - Multiply corresponding elements and sum together

$$G[x, y] = \sum_{i,j} F[x + i, y, +j]g[i, j]$$

$F[x, y]$ $G[x, y]$
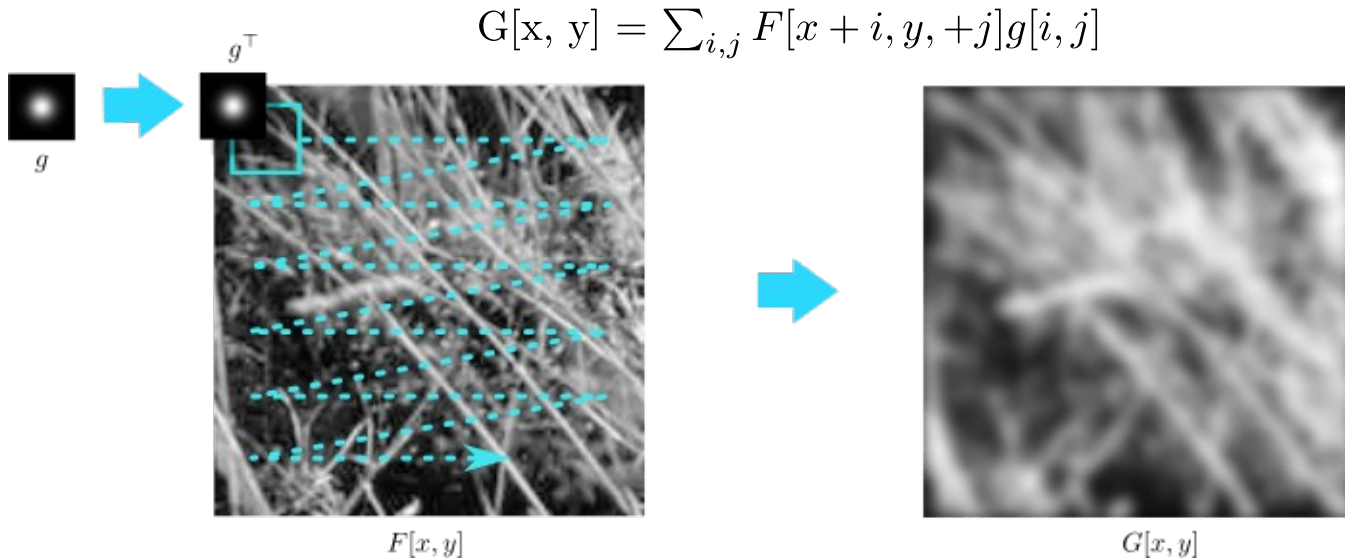
What to do with borders?

- Image is a finite signal
  - Filtering
  - Interpolation
- Data out of border has to be fabricated
- Different techniques
  - Based on use-case

constant

edge

wrap

reflect
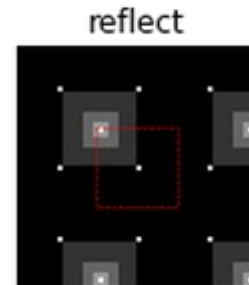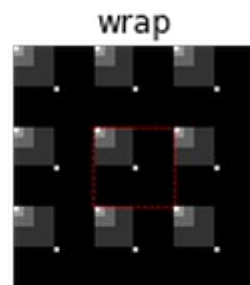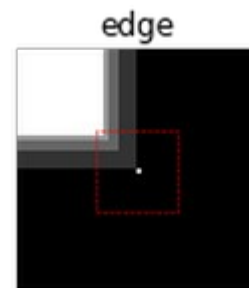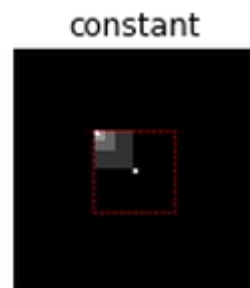
Weighted sum

$$F[x, y]$$
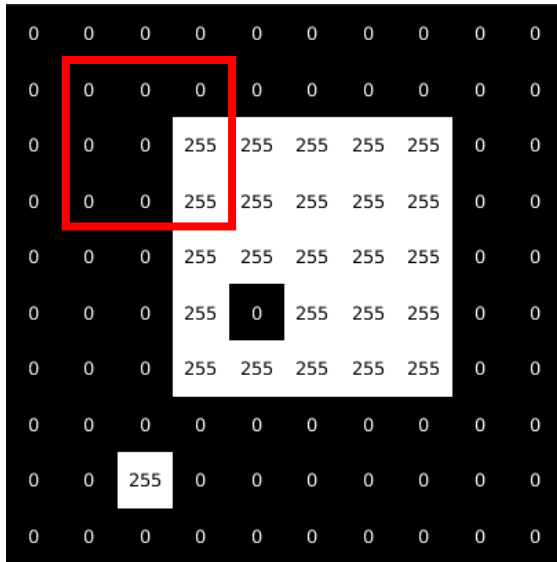
$$g$$

$$G[x, y]$$

what do to with the border?

Identity filter

$$\ast \quad \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \quad =$$

Shift filter

$$\begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 0 & 1 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$
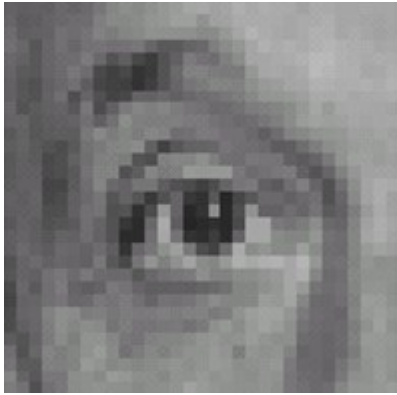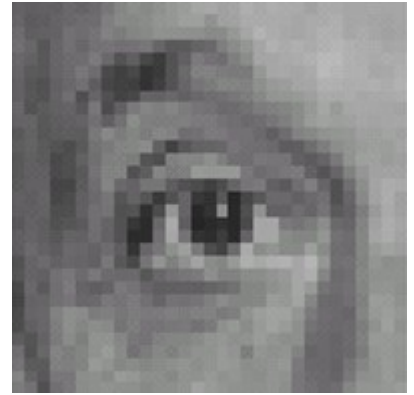
* (filter) =
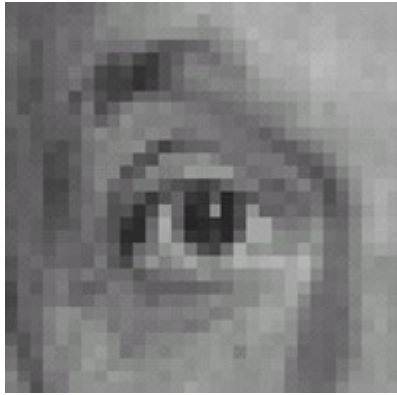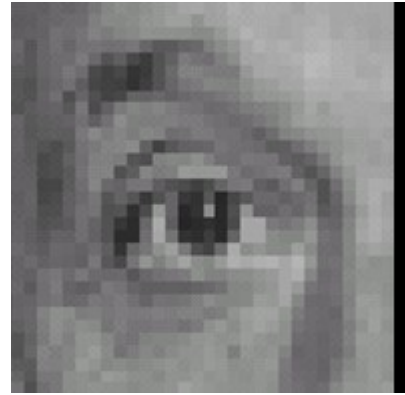
Gaussian kernel

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

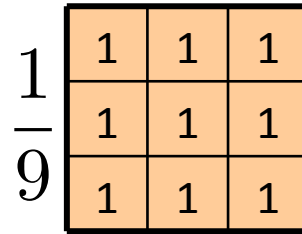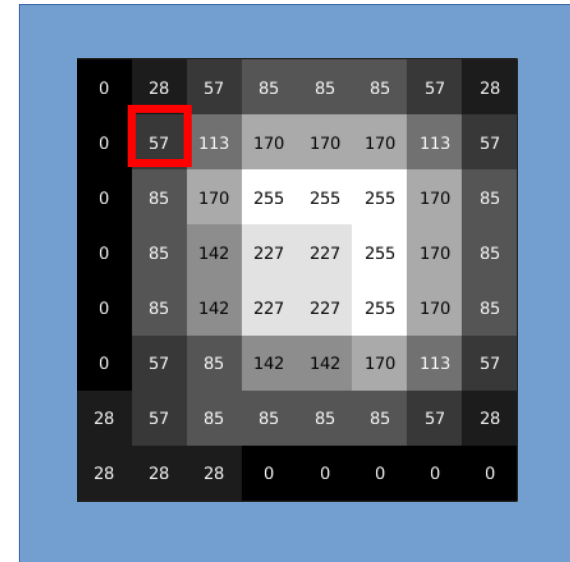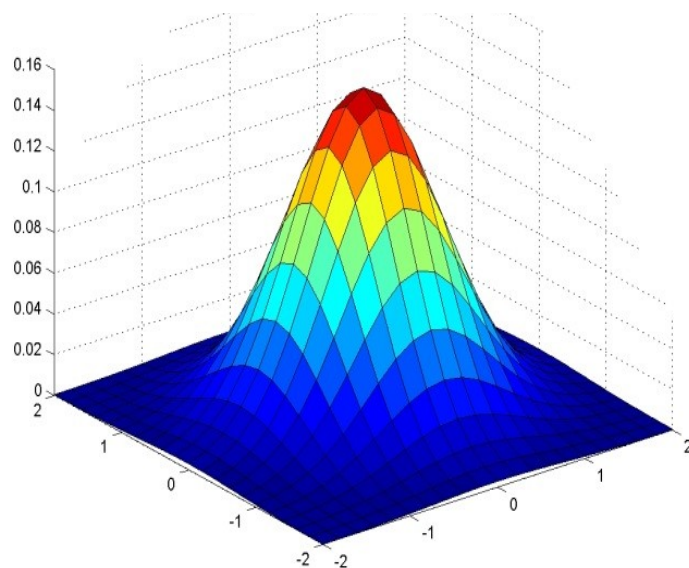| 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
0.00	0.00	0.01	0.02	0.01	0.00	0.00
0.00	0.01	0.06	0.10	0.06	0.01	0.00
0.00	0.02	0.10	0.16	0.10	0.02	0.00
0.00	0.01	0.06	0.10	0.06	0.01	0.00
0.00	0.00	0.01	0.02	0.01	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00

7 x 7, sum = 1

Constant before the exponential function ensures that the sum of the elements is always 1 (in continuous space).

Detecting edges

- Goal: find sudden changes in illumination in the image

- Ideal: line drawing by an artist (semantic knowledge)

image

values along the red line

first derivative
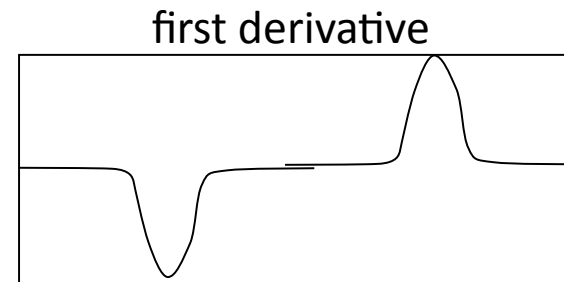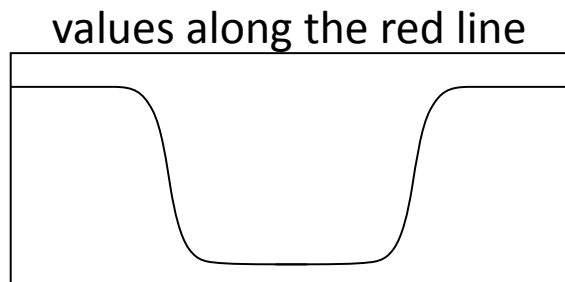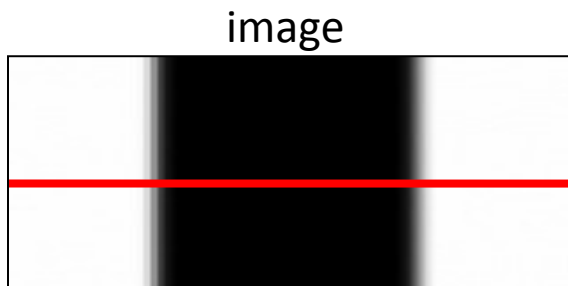
Using convolution

- Kernel can represent approximation of image derivation

- We use separate kernels for vertical and horizontal derivation

Prewitt

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

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

Sobel

-1	0	1
-2	0	2
-1	0	1

1	2	1
0	0	0
-1	-2	-1

Roberts

0	1
-1	0

1	0
0	-1

Derivative magnitude

$$\sqrt{d_x^2 + d_y^2}$$

Sharpening by blurring

Difference between low-pass filter and original is high-pass filter.

F

$F \star g$

$\star$

$-$

$+$

$F - (F \star g)$
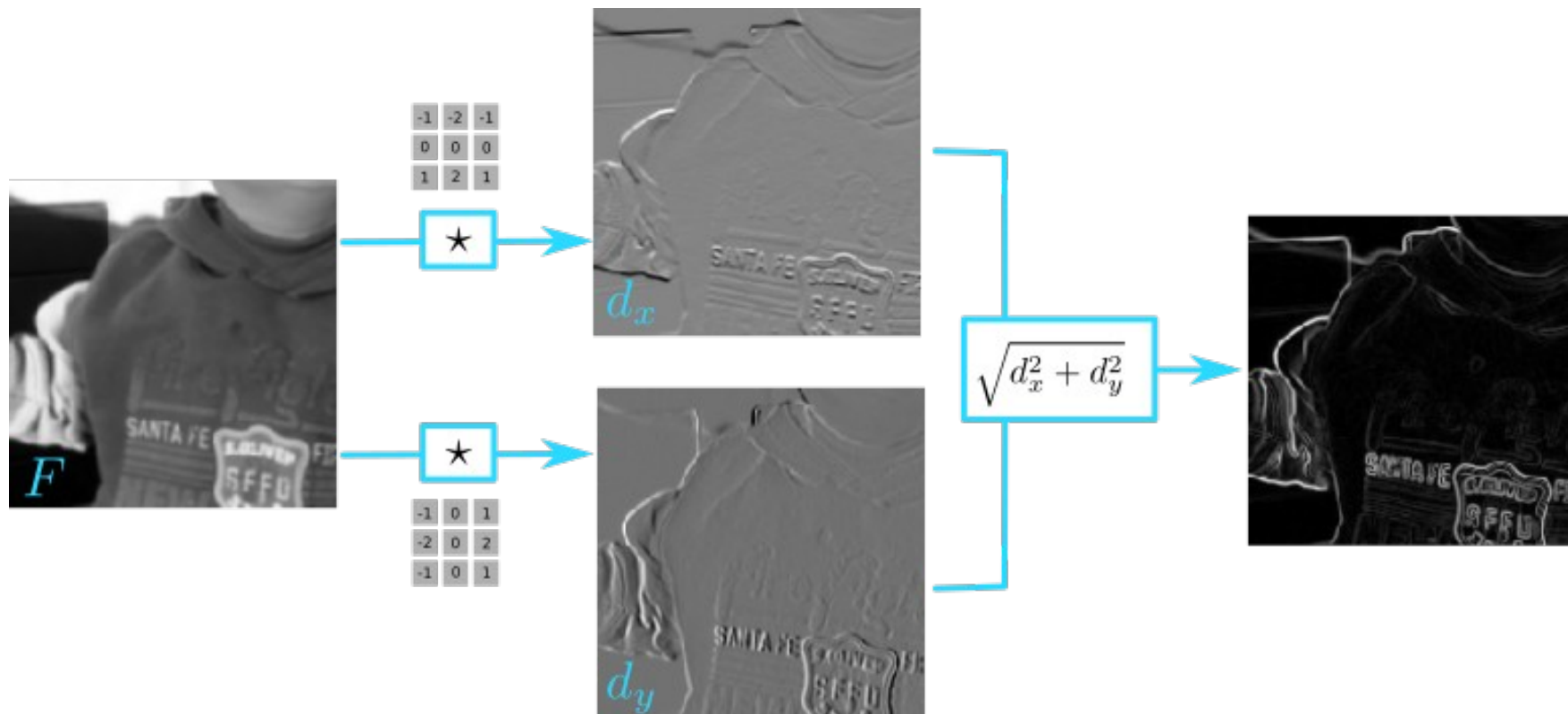
$(1 - \lambda)F + \lambda(F - (F \star g))$
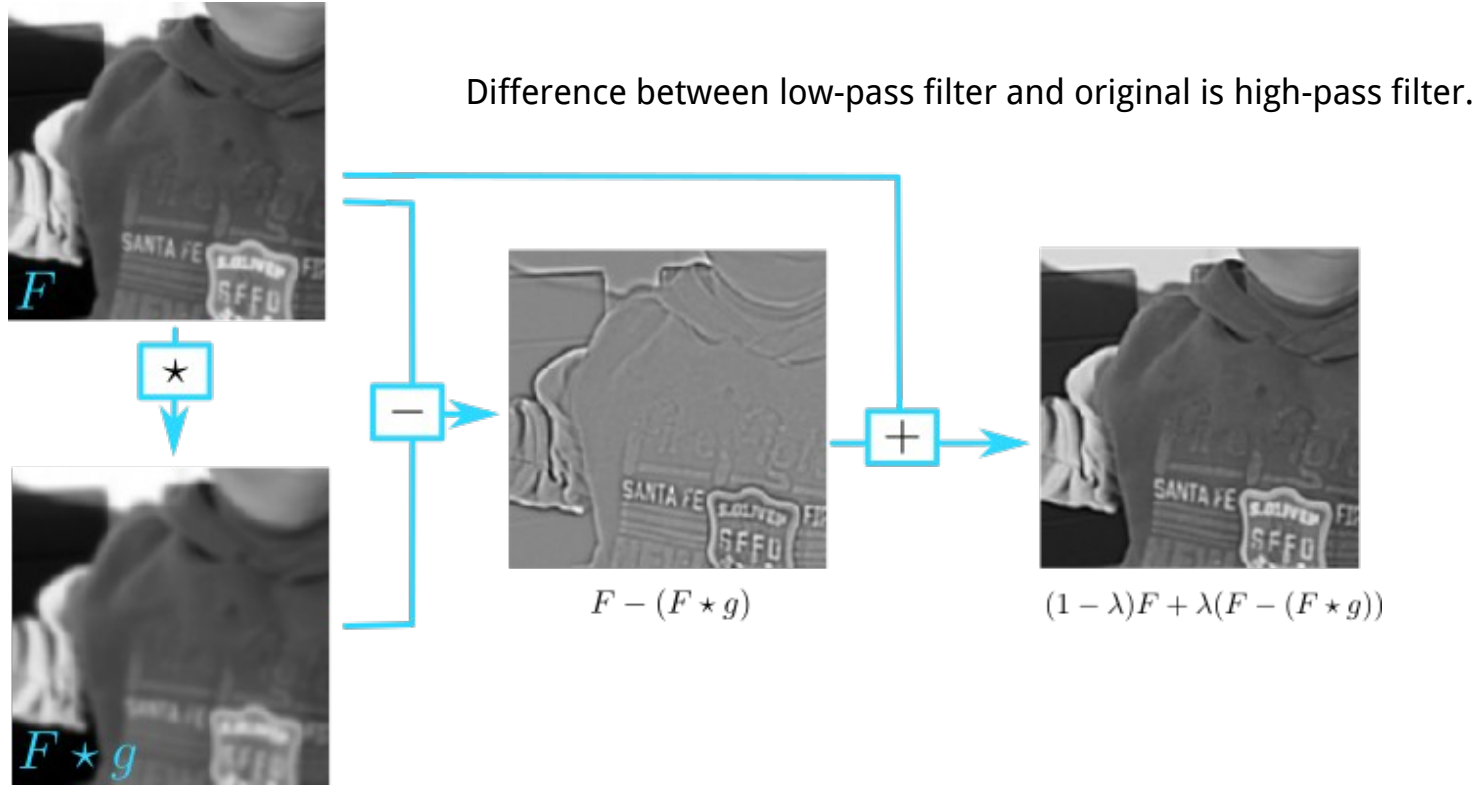
Non-linear filters

- Median: middle element by value

- Bilateral filter
  - Weights based on neighborhood
  - Preserves edges

- Morphological operations
  - Max: highest element in neighborhood
  - Min: lowest element in neighborhood
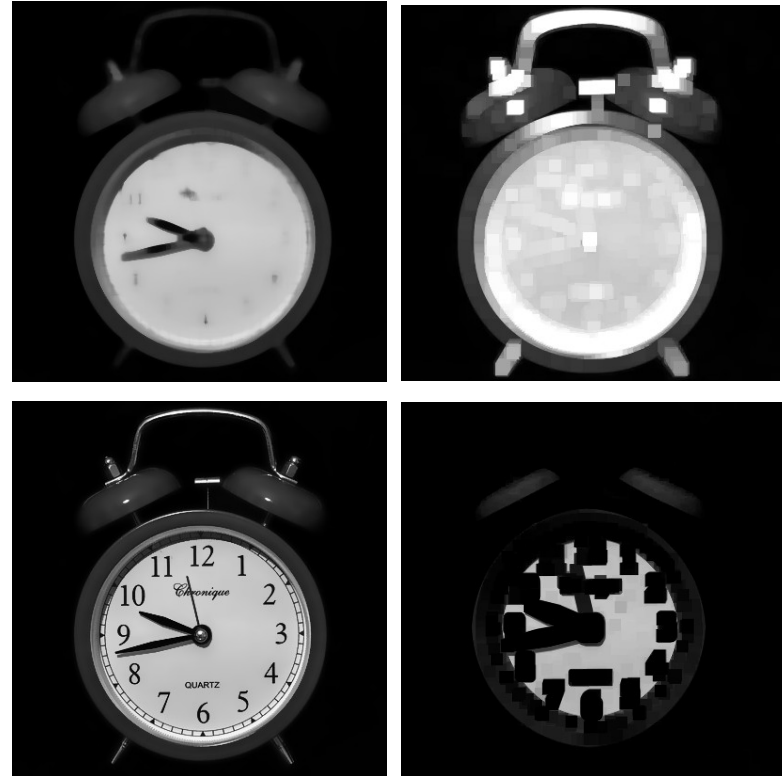
Gaussian noise removal
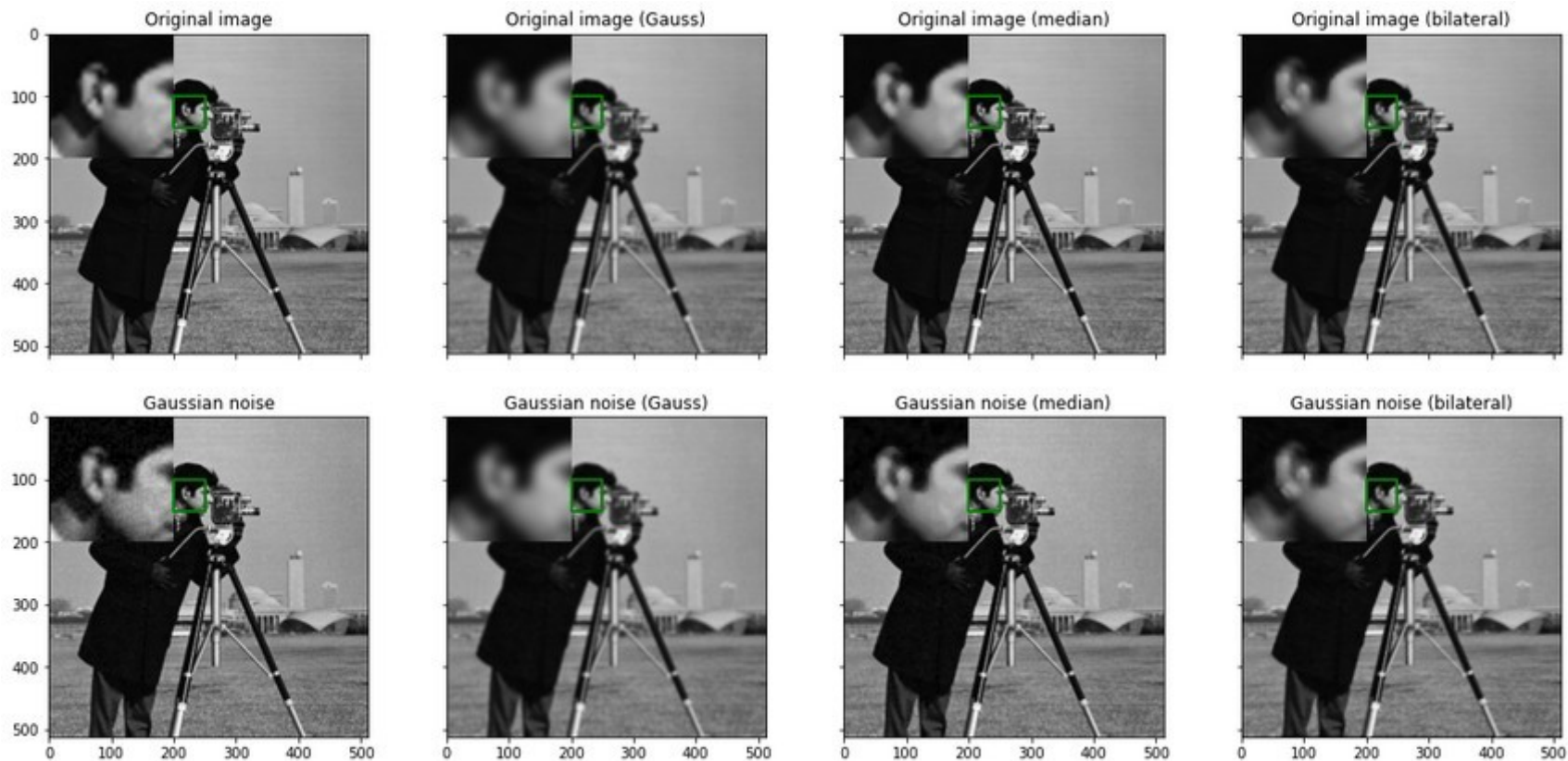
Salt&pepper noise removal
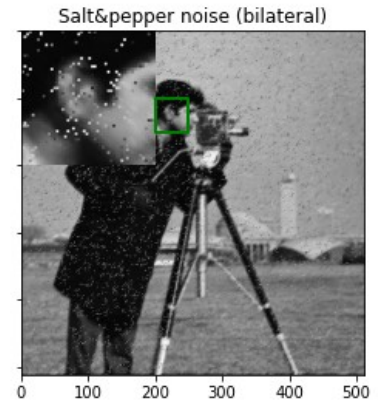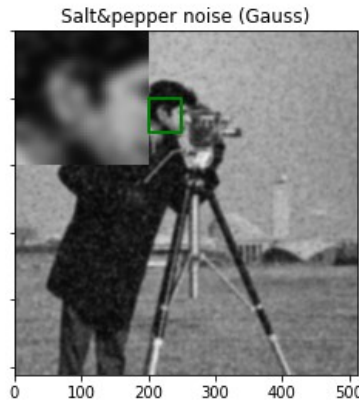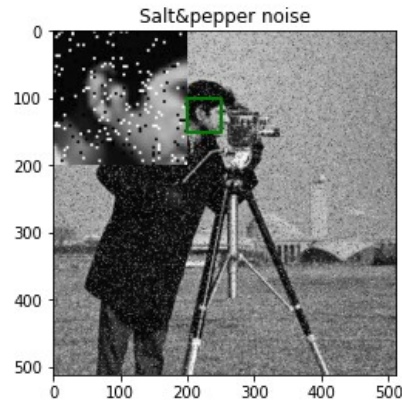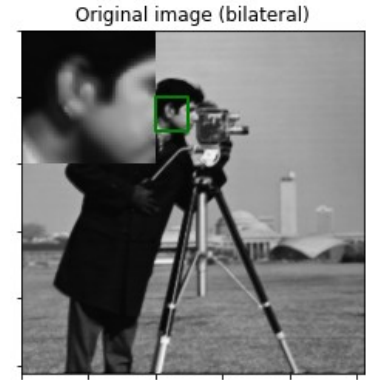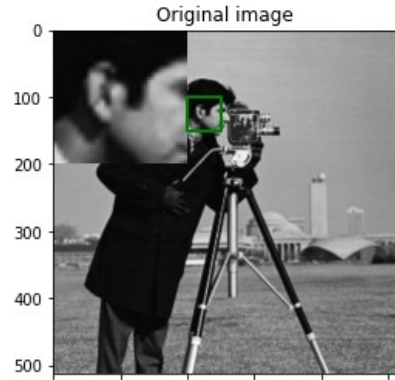
Geometry vs. intensity

- Image filtering is foremost an intensity transformation

$g(x) = T(f(x))$

f

T

g
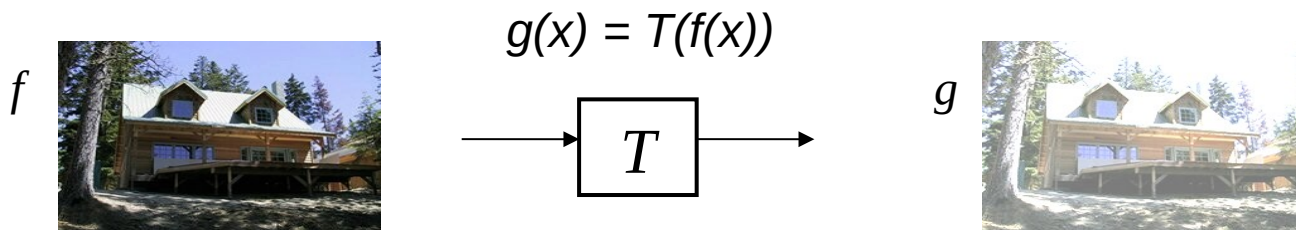
- Geometry transformation changes geometry of the image

$g(x) = f(T(x))$

f

T

g
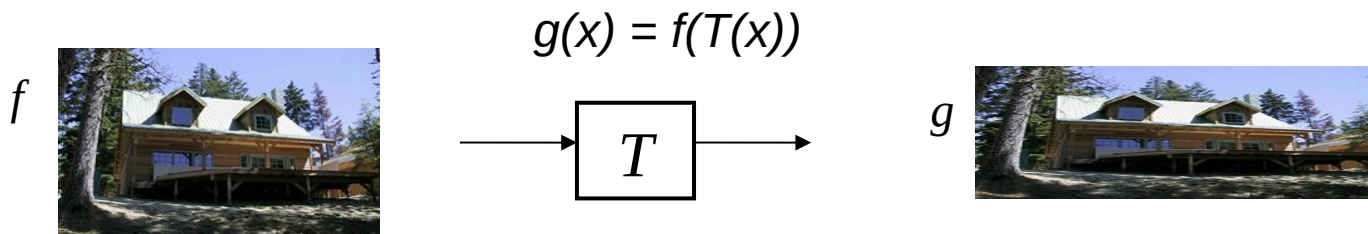
Parametric transformations
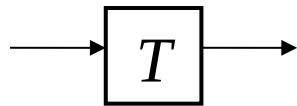
$$\mathbf{p} = (x,y) \qquad \mathbf{p}' = T(\mathbf{p}) \qquad \mathbf{p}' = (x',y')$$

- Transformation T changes coordinates of pixel p
- Global transformation changes all pixels in the same way

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{M} \begin{bmatrix} x \\ y \end{bmatrix} \qquad \mathbf{p}' = \mathbf{M} \cdot \mathbf{p}$$

Linear transformations

- Rotation:
$$x' = \cos\Theta x - \sin\Theta y$$
$$y' = \sin\Theta x + \cos\Theta y$$
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\Phi & -sin\Phi \\ sin\Phi & cos\Phi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Shear:
$$x' = x + \alpha_x y$$
$$y' = \alpha_y x + y$$
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & \alpha_x \\ \alpha_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Mirroring:
$$x' = -x$$
$$y' = -y$$
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scaling

- Multiply coordinates with a scalar
  - **Uniform** - same scalar for all axes
  - **Non-uniform** - different scalar

$$x' = \alpha x \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$
$$y' = \beta y$$

X x 2
Y x 2

X x 2
Y x 0.5
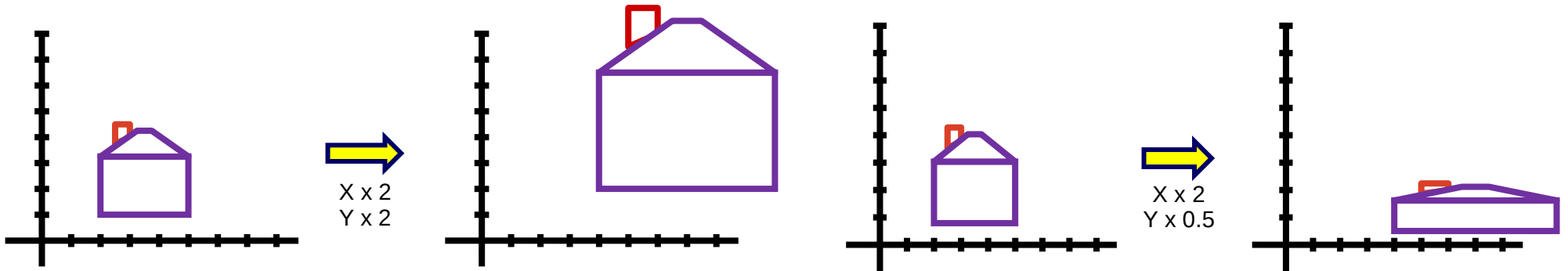
Translation

- Translation is not homogeneous in 2D space

$$x' = x + t_x$$
$$y' = y + t_y$$

- We can use homogeneous coordinates

$$\begin{bmatrix} x \\ y \end{bmatrix} \Longrightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} x' \\ y' \end{bmatrix} = \mathrm{M} \begin{bmatrix} x \\ y \end{bmatrix}$$

Homogeneous coordinates

- 2D space: transformation matrix of size 3x3

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \alpha_x & 0 \\ \alpha_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- General projective transform

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \lambda \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$
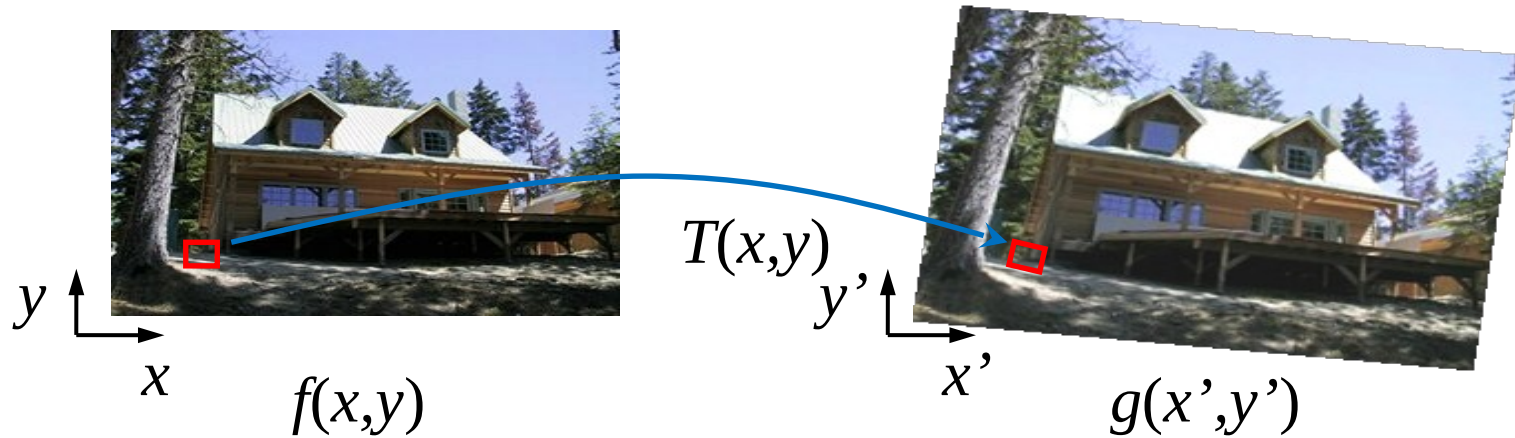
Common transformations

- Translation
- Euclidean transform (translation, rotation)
- Similarity transform (translation, rotation, scaling)
- Affine transform (maintains parallelism of straight lines)
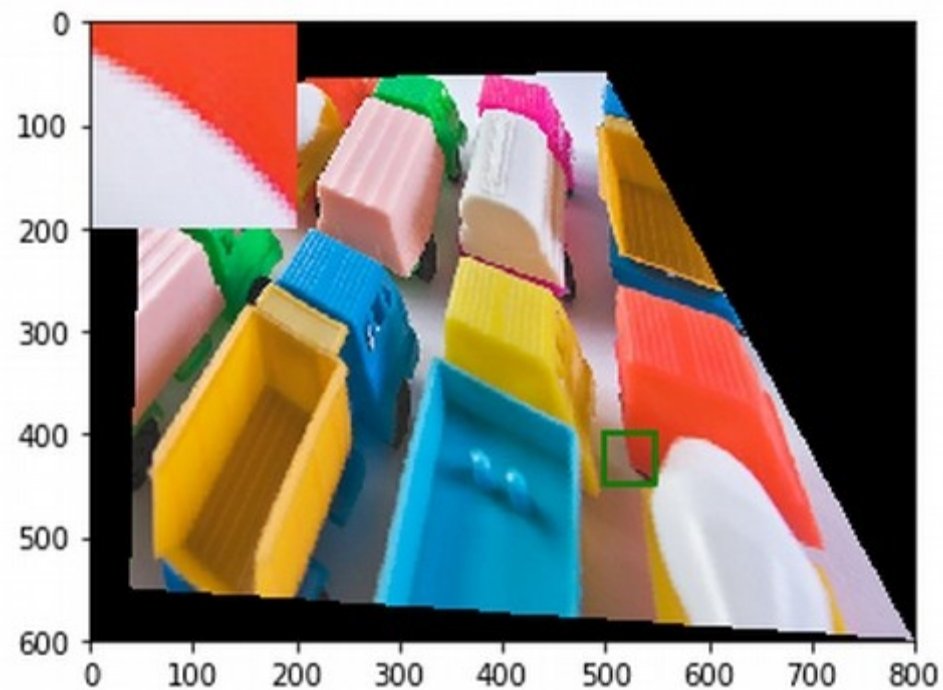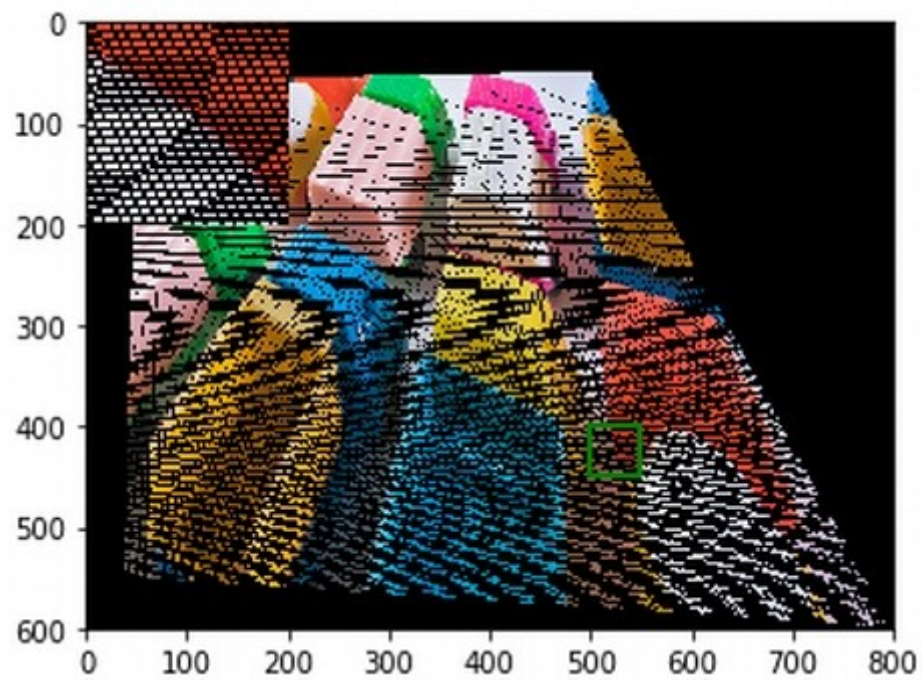- Projective transform
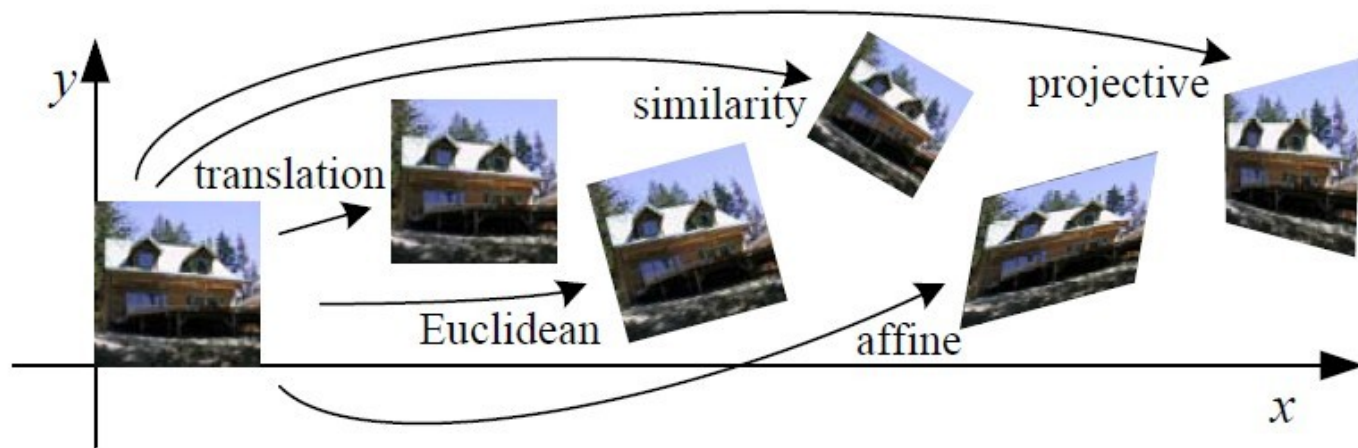
Warping

Given transform [x',y'] = T(x,y) and f(x,y), how to compute g(x',y') = f(T(x,y)) ?


$T(x,y)$

$f(x,y)$

$g(x',y')$

Naive approach

- For each pixel f with coordinates (x,y)
    - Compute transformed coordinates [x',y']=T(x,y).
    - Copy color of f(x,y) to new image at coordinates g(x',y')
- Why is it naive?
    - We visit all pixels in f(x,y)
   - Do we visit all pixels of g(x,y)?

Example

Asymmetry of discretization

- A single pixel of F is mapped to more pixels in G.

- Pixel in F is not mapped to any pixel in G.

- Visit all pixels in G
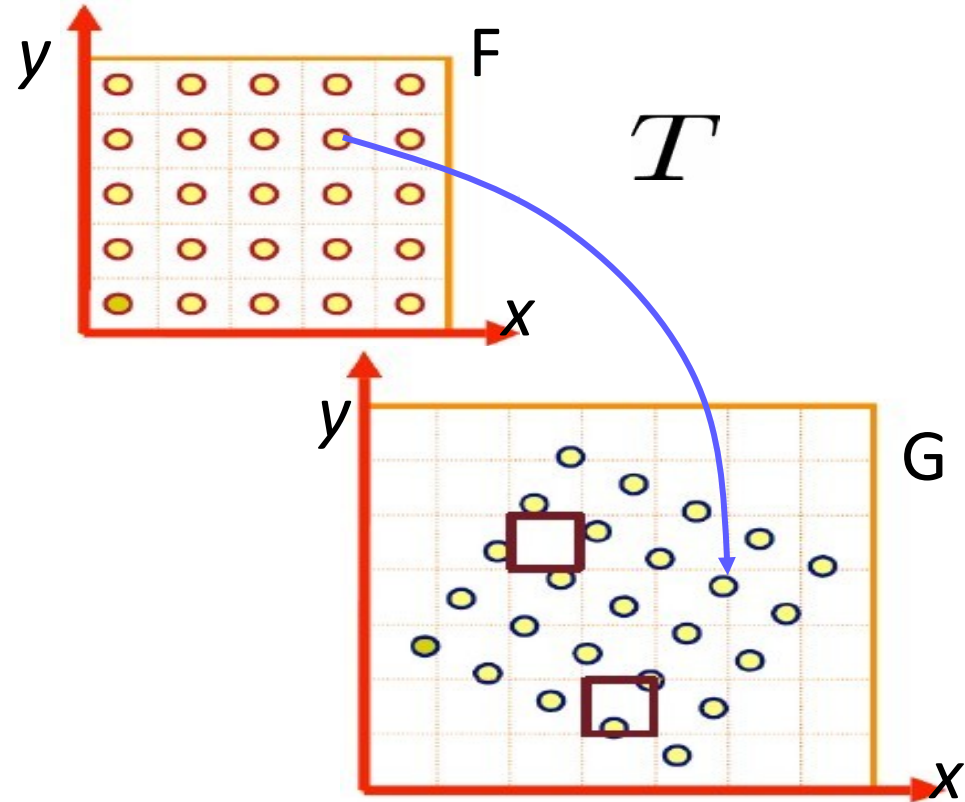
Inverse mapping approach

- For each pixel in g with coordinates (x′,y′)
  - Compute old coordinates using inverse transform [x,y]=T-1(x′,y′)
  - We copy pixel color of f(x,y) to g(x′,y′)
- We visit all pixels in g
- Pixel from g can transform to more than one pixel in f

$T^{-1}(x,y)$

y

x

$f(x,y)$

y'

x'

$g(x',y')$
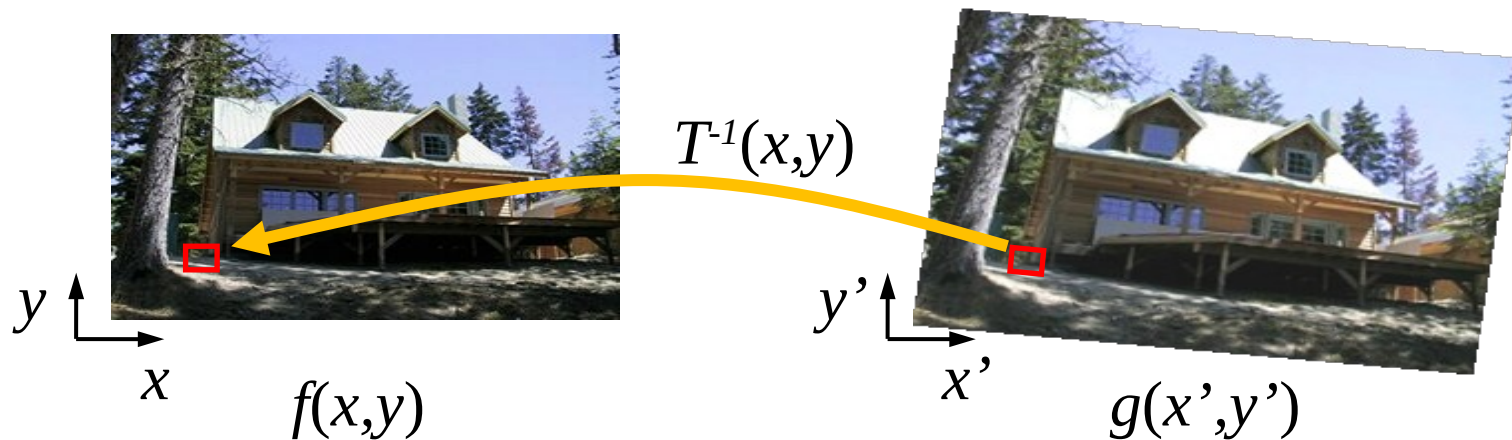
Resizing the image

- Adjusting (local or global) resolution in image.
- Two general cases:
    - Reduction / decimation / down-sampling – shrinking image
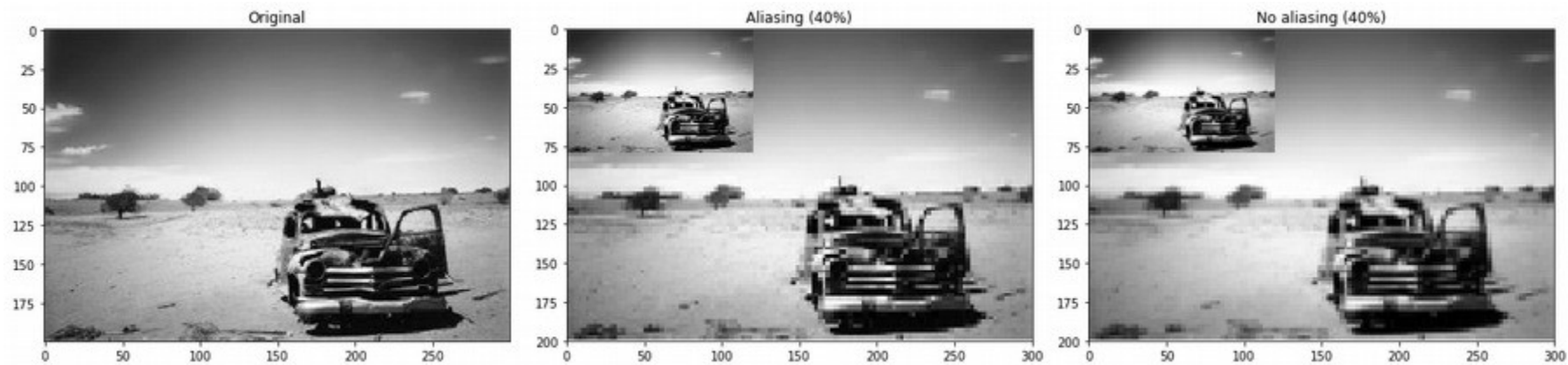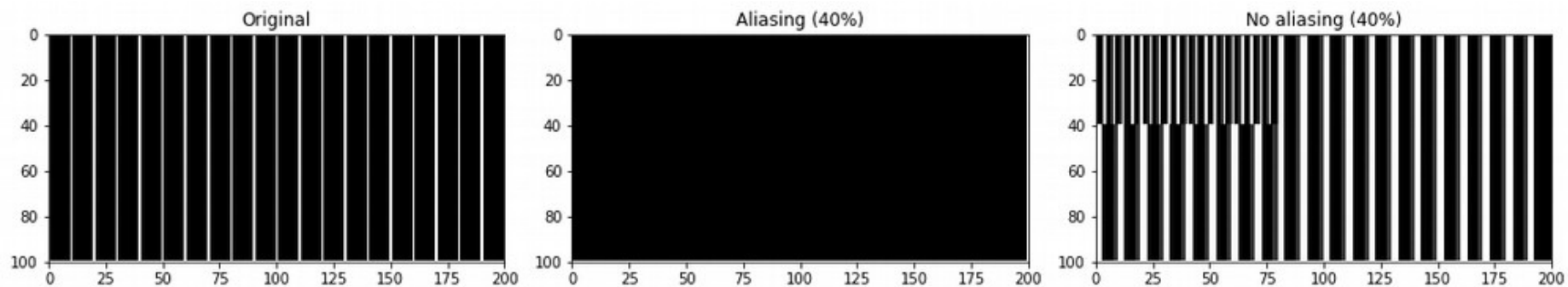   - Interpolation / up-sampling – enlarging image
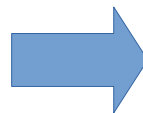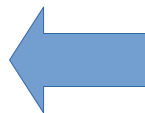
Decimation approaches

- Sampling values
  - Shannon sampling theorem $f_s \geq 2f_{max}$
  - Remove high frequencies
- Anti-aliasing
  - First remove high frequencies
  - Then subsample with appropriate frequency
- Filters used
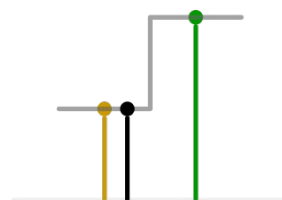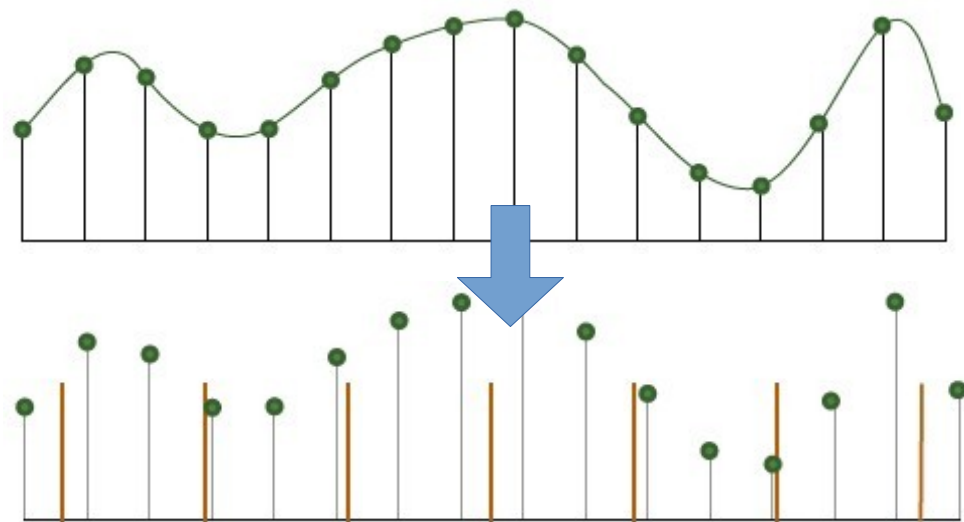 - Gaussian
  - Lanczos kernel

Good sampling
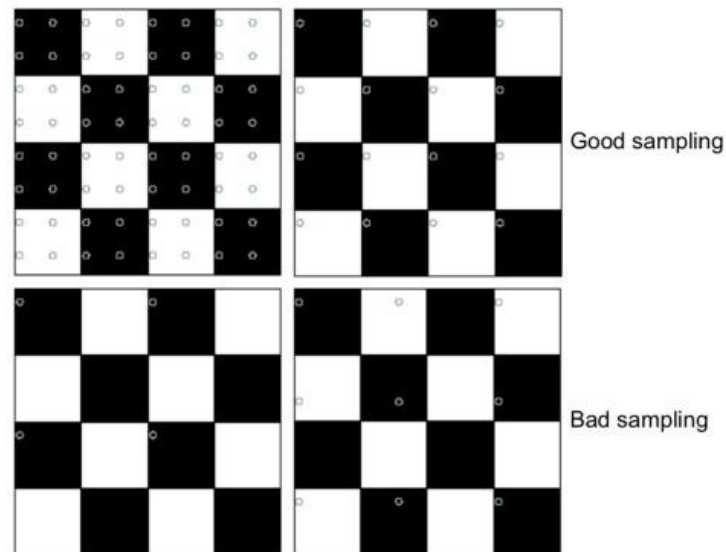
Bad sampling
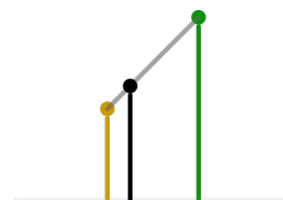
Maël Fabien

Decimation examples
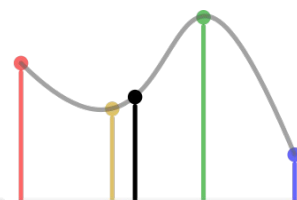
Up-sampling – guessing the values

- Nearest neighbor
  - Take value of the closest sample
- Linear interpolation
  - Use two values
  - Fit linear function
- Cubic interpolation
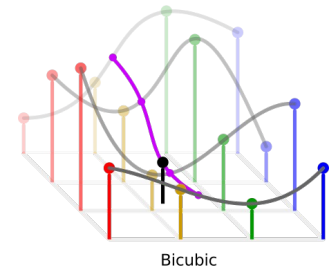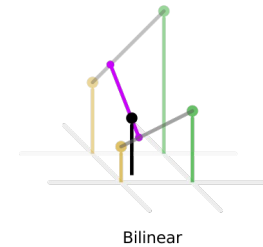 - Use four values
  - Fit third-degree polynomial
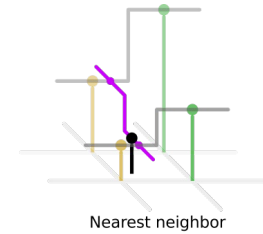

Nearest neighbor   Linear   Cubic
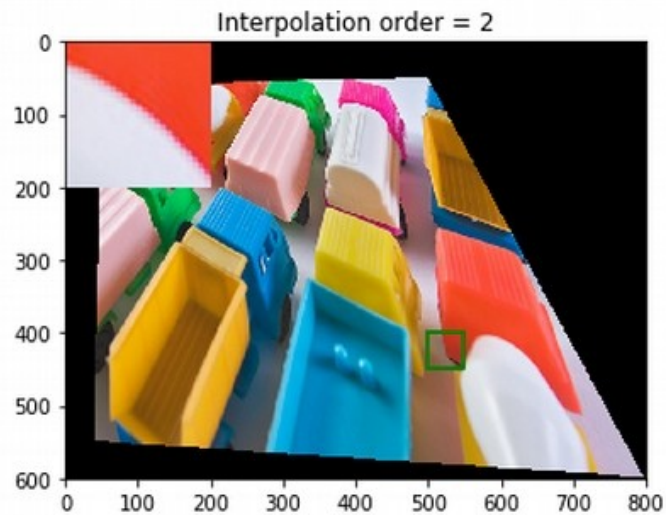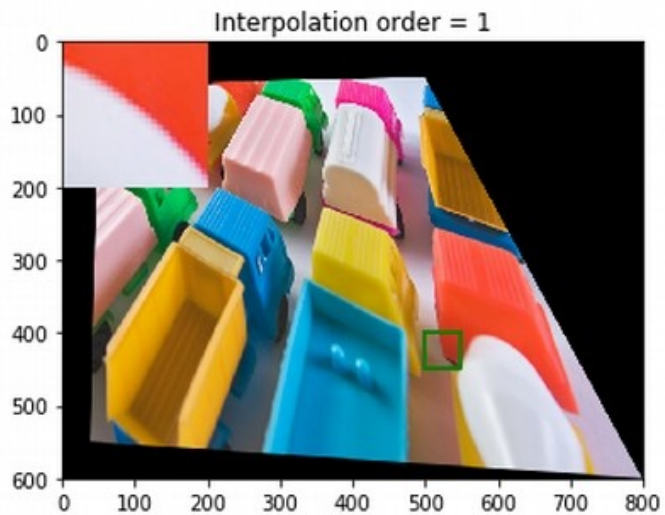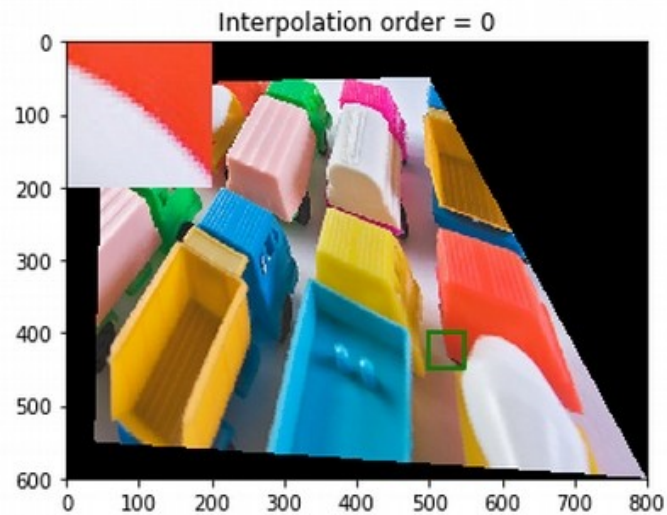
Author: Cmglee (CC-SA)

Interpolation in images

- Nearest neighbor
  - Take pixel closest to desired coordinates
- Bi-linear
  - Use four closest pixels
  - Estimating a plane
- Bi-cubic
  - Use 16 closest pixels
  - Estimating a polynomial surface
  - Slower

Nearest neighbor

Bilinear

Bicubic

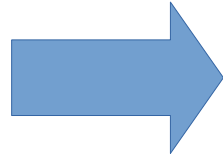Author: Cmglee (CC-SA)

Interpolation examples

Non-linear transformations

- Camera rectification
  - Counter lens distortions
  - Camera calibration model

- Locally-linear transformation
  - Local regions are transformed locally
  - Image morphing

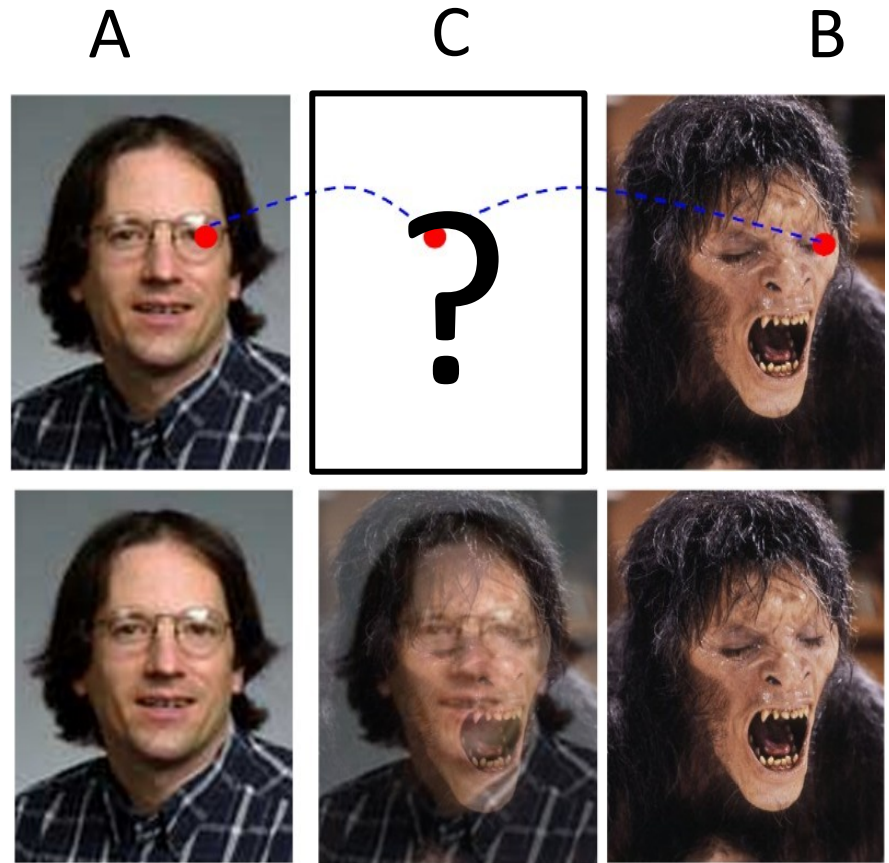# Image morphing

How to deform image A to image B?

Image morphing

- How to compute intermediate image C
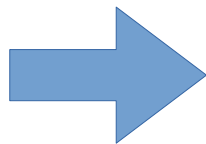
- Naive approach - weighted sum of pixels

$$C_t = \alpha_t A + (1 - \alpha_t)B$$
$$0 \leq \alpha_t \leq 1$$
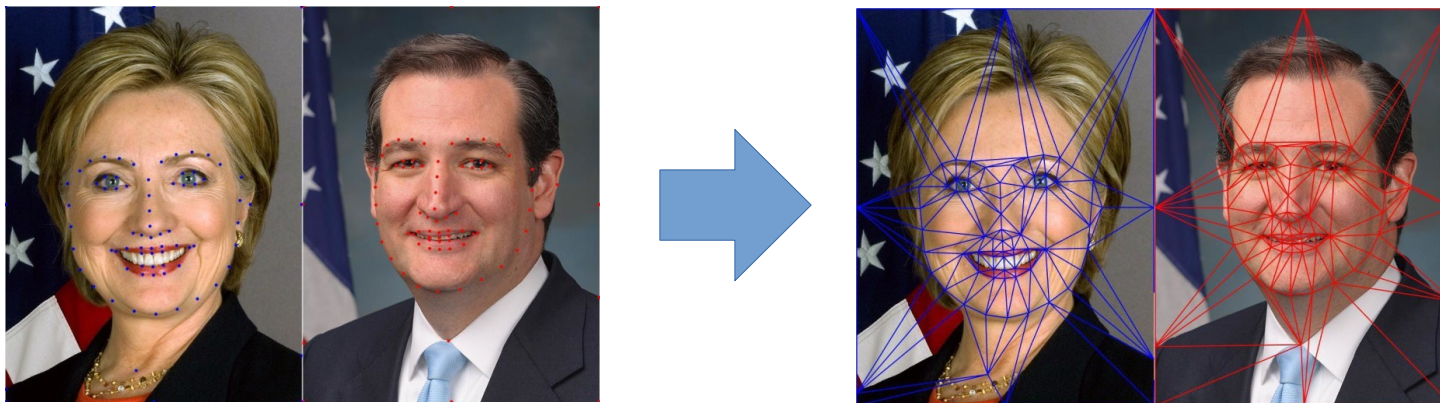
- Not realistic - not combining semantic parts

Dense deformation field

Mapping each pixel into its new location

Deformation field approximation

- Determining entire field is time-consuming
- Locally linear transformation
- Correspondences – control points
- Delaunay triangulation, interpolation

Control points
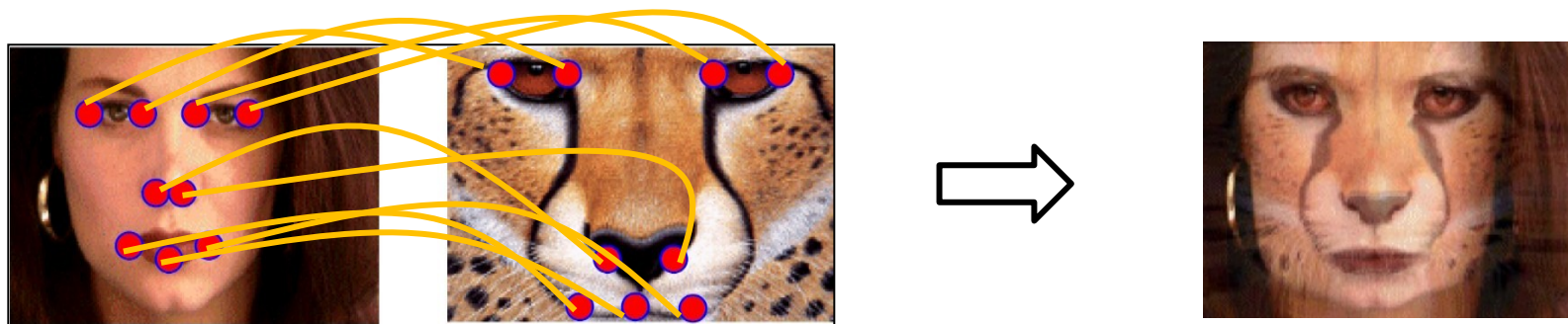
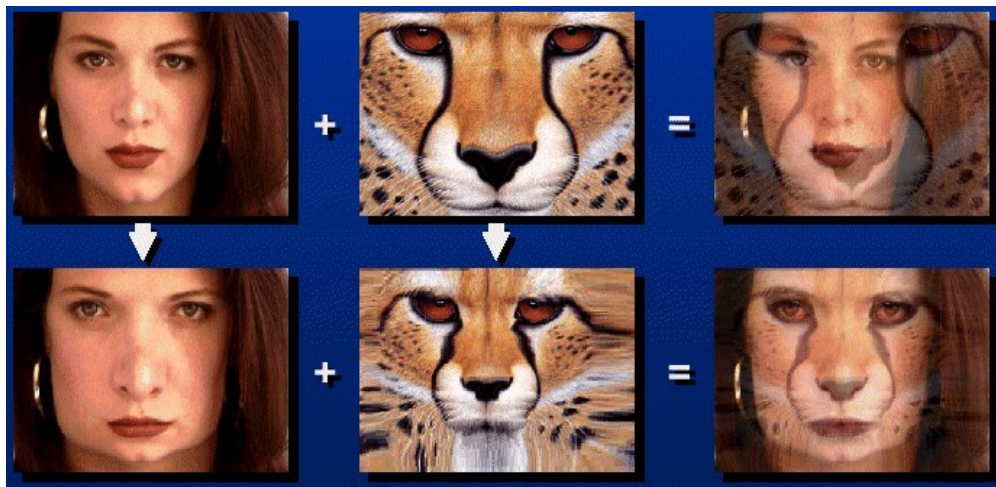Control points mark matching pixels in both images

Image morphing overview

- For each image Ct compute …
  - Interpolated position of control points
  - Two transformations: dA = A - Ct and dB = B - Ct
  - Blend colors of interpolated images

$$\mathbf{x}_i^C = \alpha_t \mathbf{x}_i^A + (1 - \alpha_t)\mathbf{x}_i^B$$
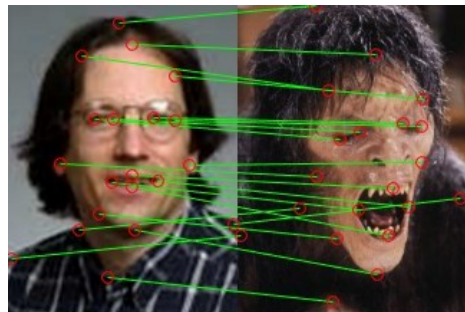
$$C_t = \alpha_t dA + (1 - \alpha_t)dB$$

Naive
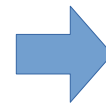
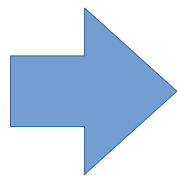Correct

Morphing example

Input images

Control points

Warped images

Blended image

One more morphing example

Content-aware resizing

- Change size, aspect
- Automatically preserve important structures

Important vs. unimportant content

Input

Scaling

Crop

Content-aware

"less important" content

Content-aware resizing

- General ideas
  - Adhere geometric constraints (size)
  - Preserve important structures
  - Reduce image artifacts
- Weakly conditioned problem
  - What is important? / Universal importance measure?
  - Would more people agree on the process?
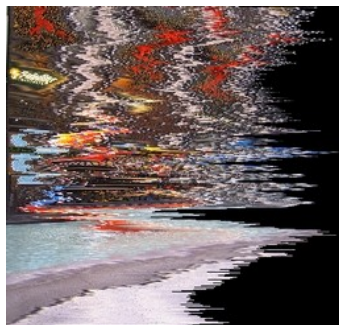 - Aesthetic rating (composition, ...)?

What is important in images?

- What do people consider important?

- Fast approximation - edges

How to remove?

Optimal (pixels with least energy)

Pixels with least energy in row

Columns with least energy

Seam carving

- We want to shrink image in one direction
- Basic idea: remove unimportant pixels
- Unimportant = little energy = little change = little edge
- Intuition
  - Preserve strong contours
  - Human perception is more sensitive to local changes
- Simple but achieves good performance

Image seam

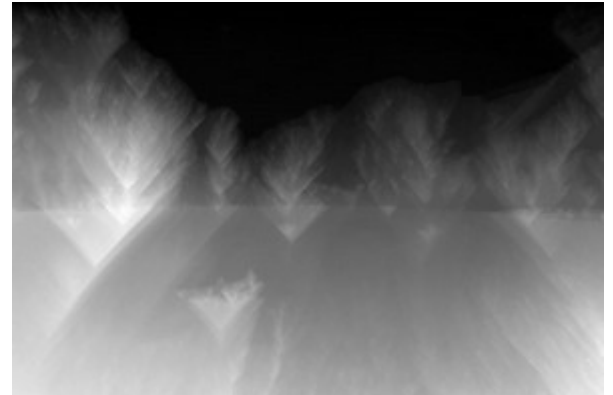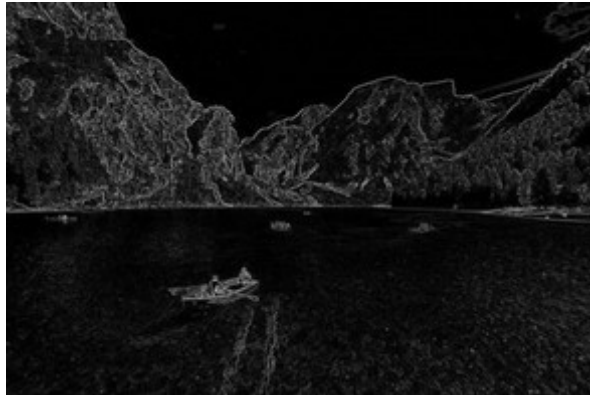- Connected path of pixels from the top to bottom

Determining optimal seam

- Optimal connected path of pixels from the top to the bottom that minimizes energy
    - Dynamic programming
    - Cumulative cost
    - Backtracking

5	8	12	3
9	2	3	9
7	3	4	2
4	5	7	8

$$M(i,j) = E(i,j) + min(M(i-1, j-1), M(i-1, j), M(i-1, j+1))$$

Removing a seam

- Compute edge energy
- Compute cumulative energy
- Determine optimal seam
- Remove seam

Examples – reducing width

Examples – reducing height

Examples – scaling down

Interactive demo: https://www.aryan.app/seam-carving/
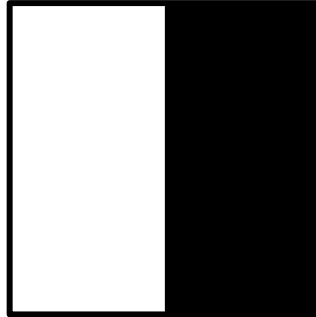
Image compositing

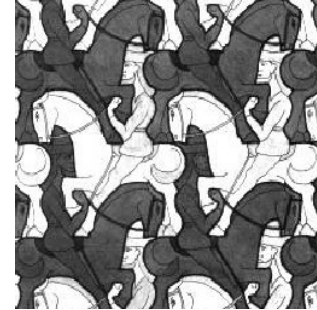Combine images by taking pixels from appropriate images
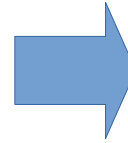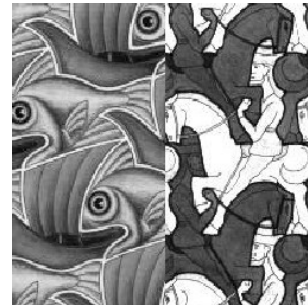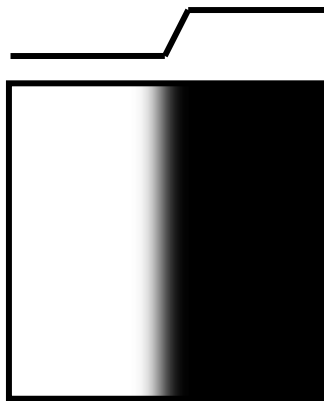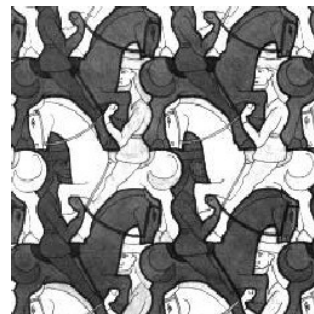
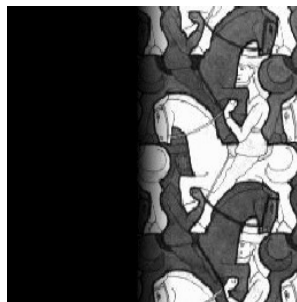# Naive combination – binary mask
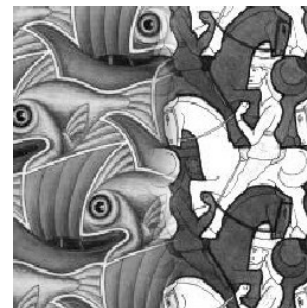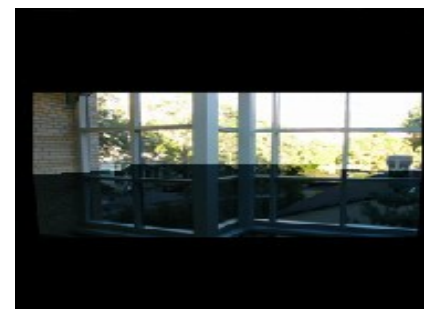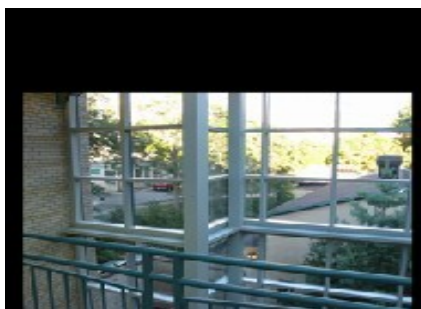

A

R

B

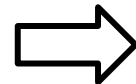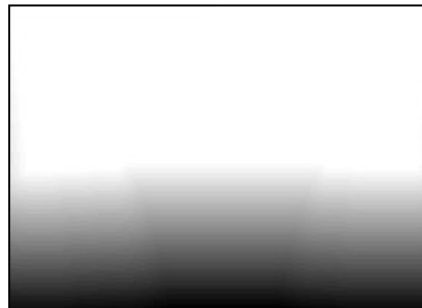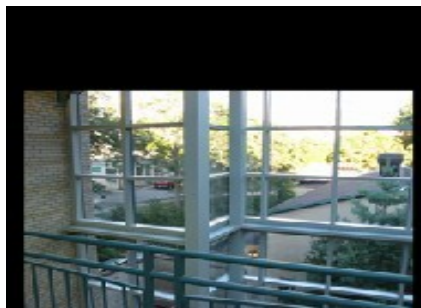$$RA + (1 - R)B = $$

Smooth alpha channel

A

R

B

 + **=**

Example

- Sharp transition – unreal image
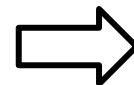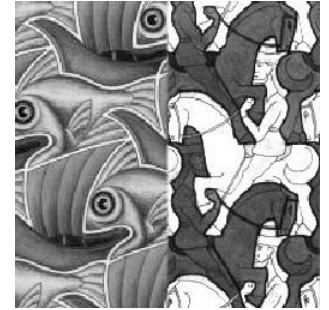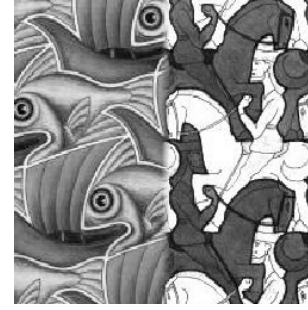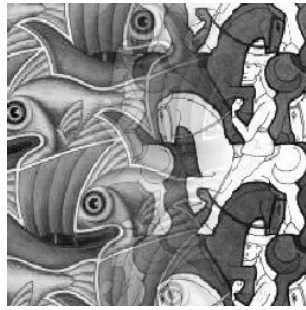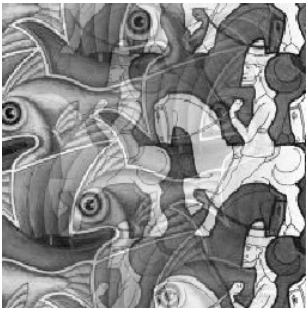

- Smooth transition – more realistic

Smoothing influence

Very smooth transition - ghosting

Sharp transition - cutoff

Frequency-aware blending

- Simple alpha mask blending
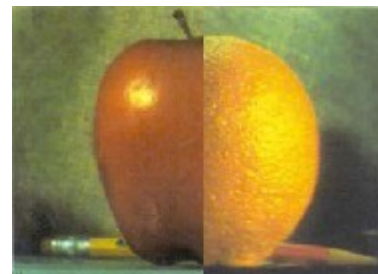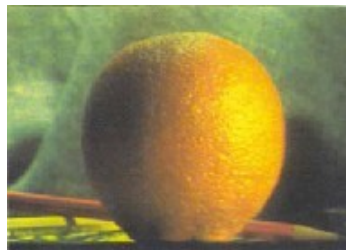
- More natural effect if we blend images by frequency bands
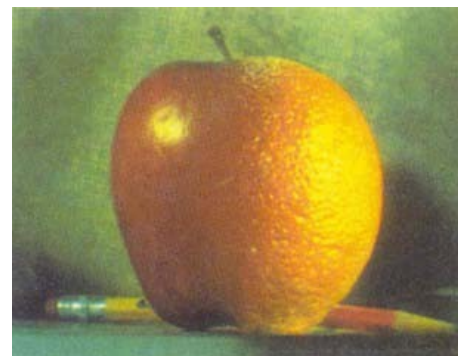
Image pyramids

- Multi-scale signal representation
  - Sequence of images
  - Each image only includes lower frequencies

- Gaussian pyramid
  - Smooth with Gaussian filter
  - Reduce resolution by half
  - Repeat

level k (= 1 pixel)

level k-1

level k-2

level 0 (= original image)
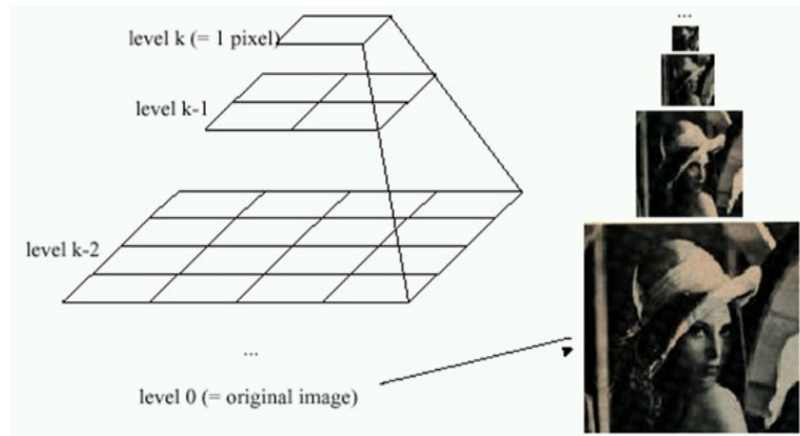
Gaussian pyramid

- Kernel size fixed

- Discard every second pixel

- Each layer removes frequency band

G 1/8

G 1/4

G 1/2

Laplacian pyramid

I_0 I_1 I_2 I_3 $L_3 = I_4$

High frequencies   Medium frequencies   Low frequencies
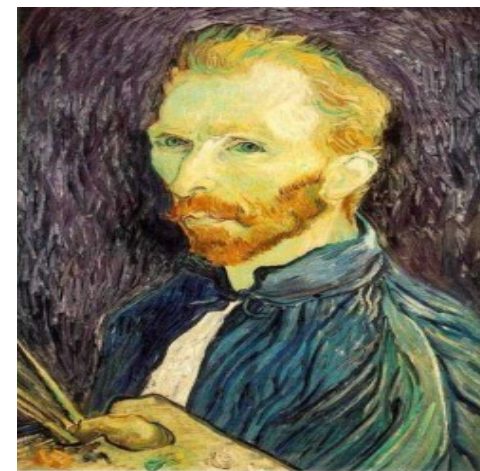
Collapsing the pyramid

$L_0 = I_0 - I_1$ $L_1 = I_1 - I_2$ $L_2 = I_2 - I_3$ $L_3 = I_3$

Reconstruction by collapsing pyramid

$$L_0 + L_1 + L_2 + L_3 = (I_0 - I_1) + (I_1 - I_2) + (I_2 - I_3) + I_3 = I_0$$

Laplacian blending algorithm

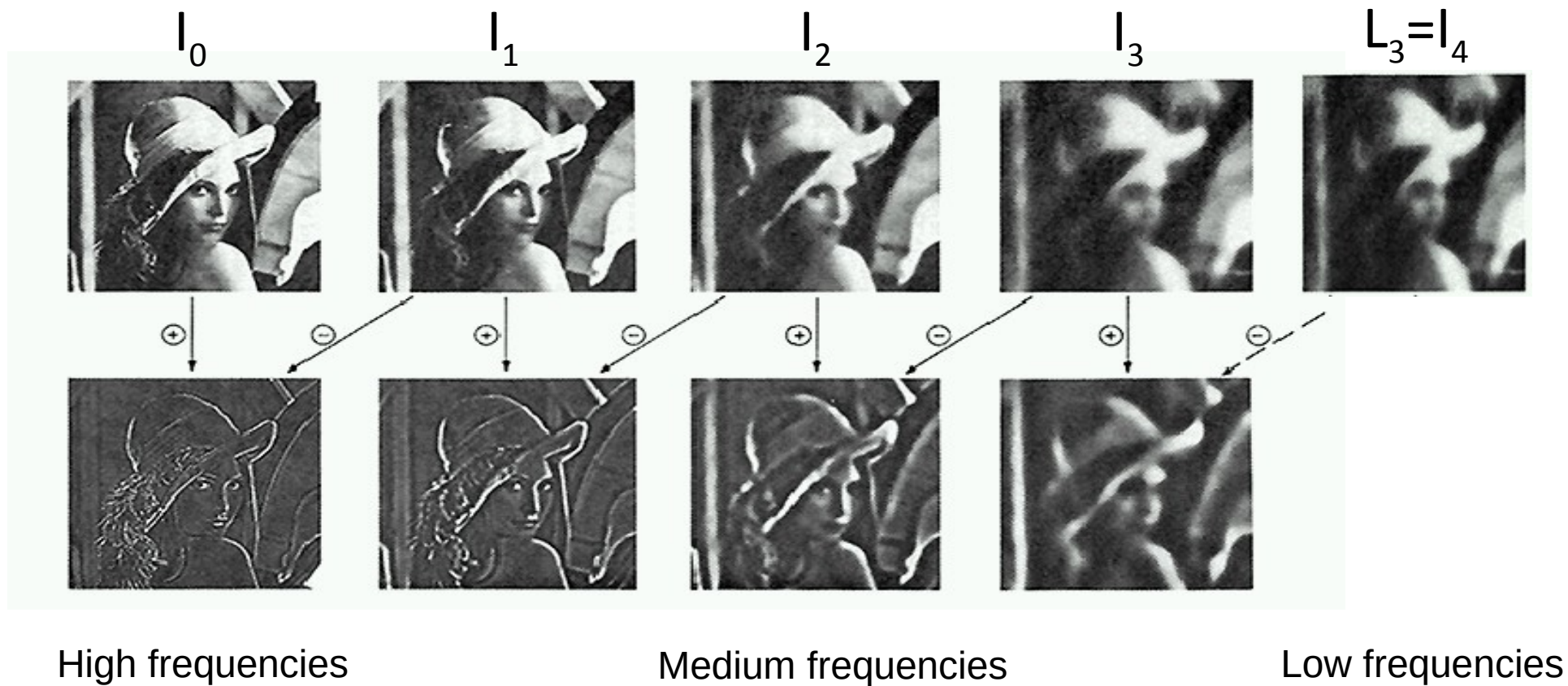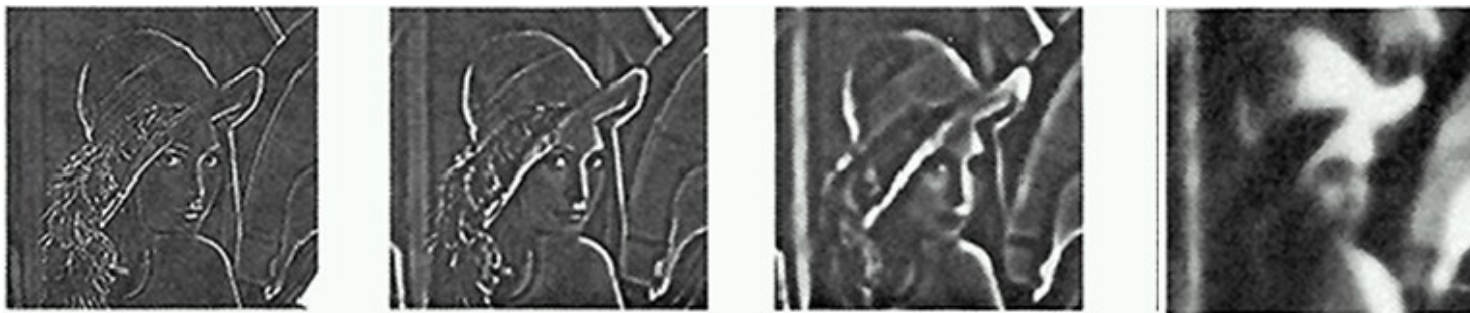- Generate Laplacian pyramids LA and LB for images A and B
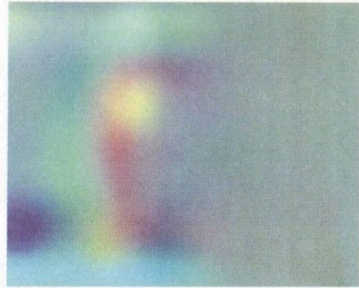
- Generate Gaussian pyramid GR for the alpha mask R

- Combine new Laplacian pyramid LS by combining corresponding layers from LA and LB using weights from the corresponding layer in GR:

$$LS_i = GR_i LA_i + (1 - GR_i)LB_i$$
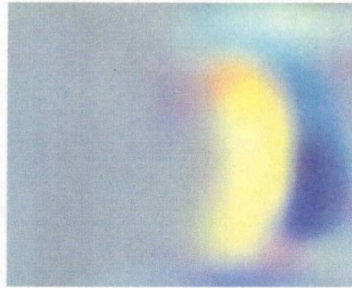
- Collapse pyramid LS into the resulting image S
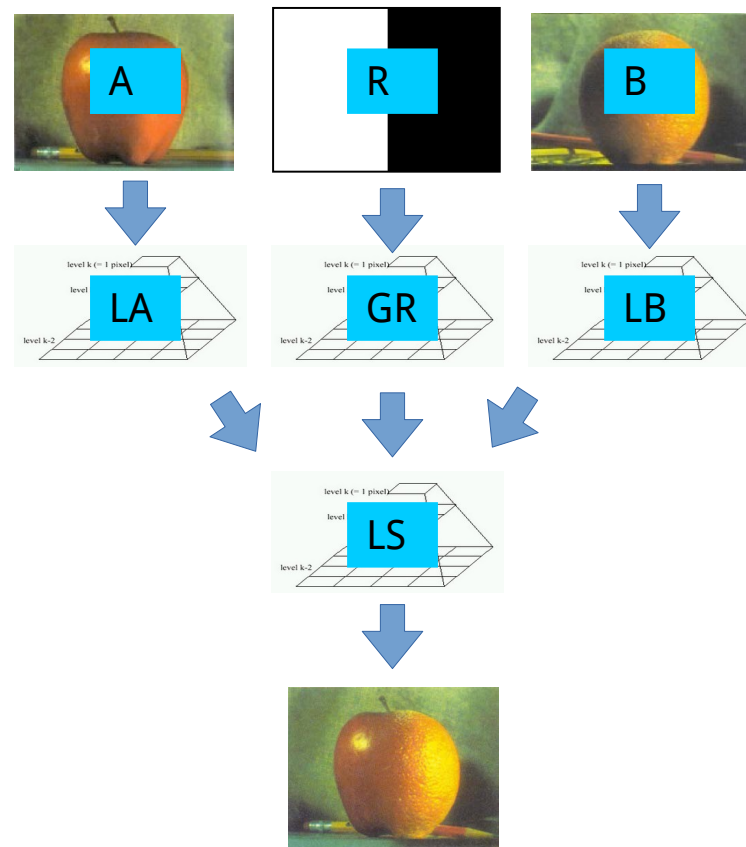
Layer 4
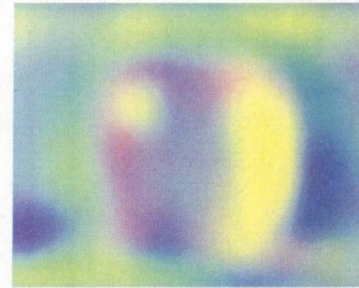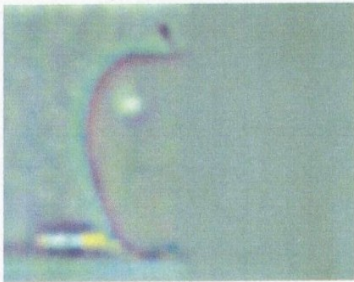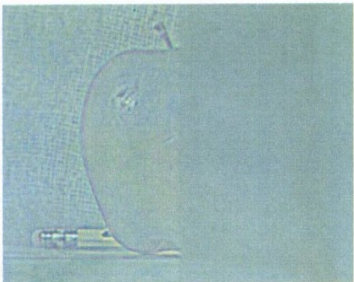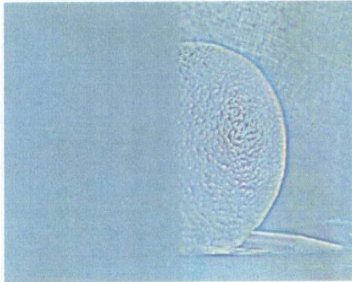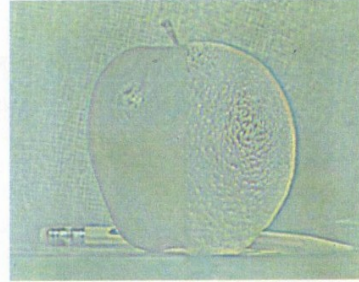
Layer 2

Layer 0

image A   image B   combined layer

Merging examples
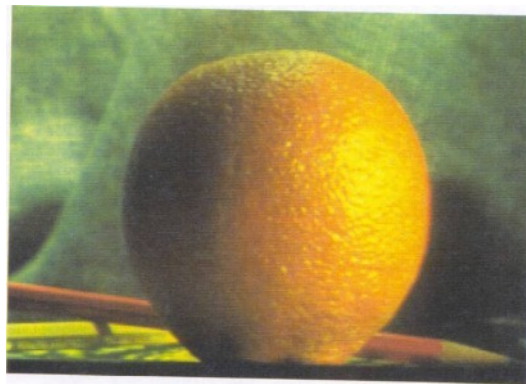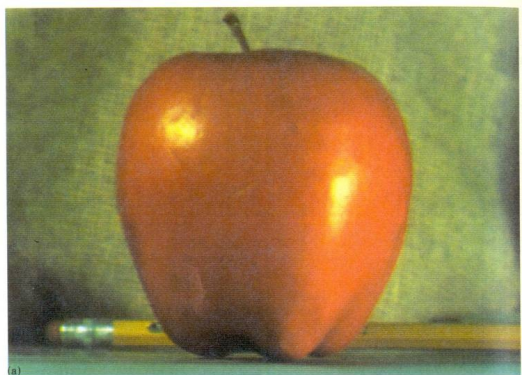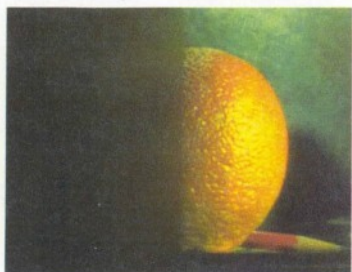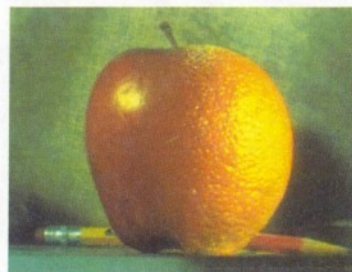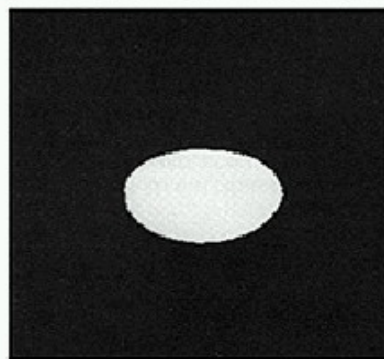
Compositing using imperfect masks

- Mask not following borders
  - User input
  - Low-resolution / speed
- Improve result using a deep model
  - Train using degraded masks as input
  - Compare to reference composite
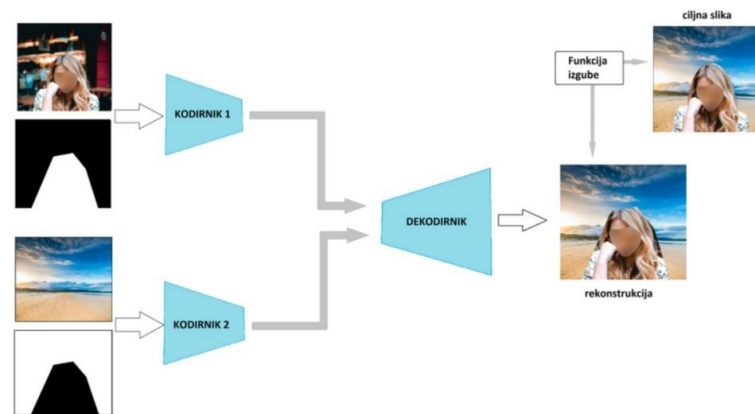
Examples
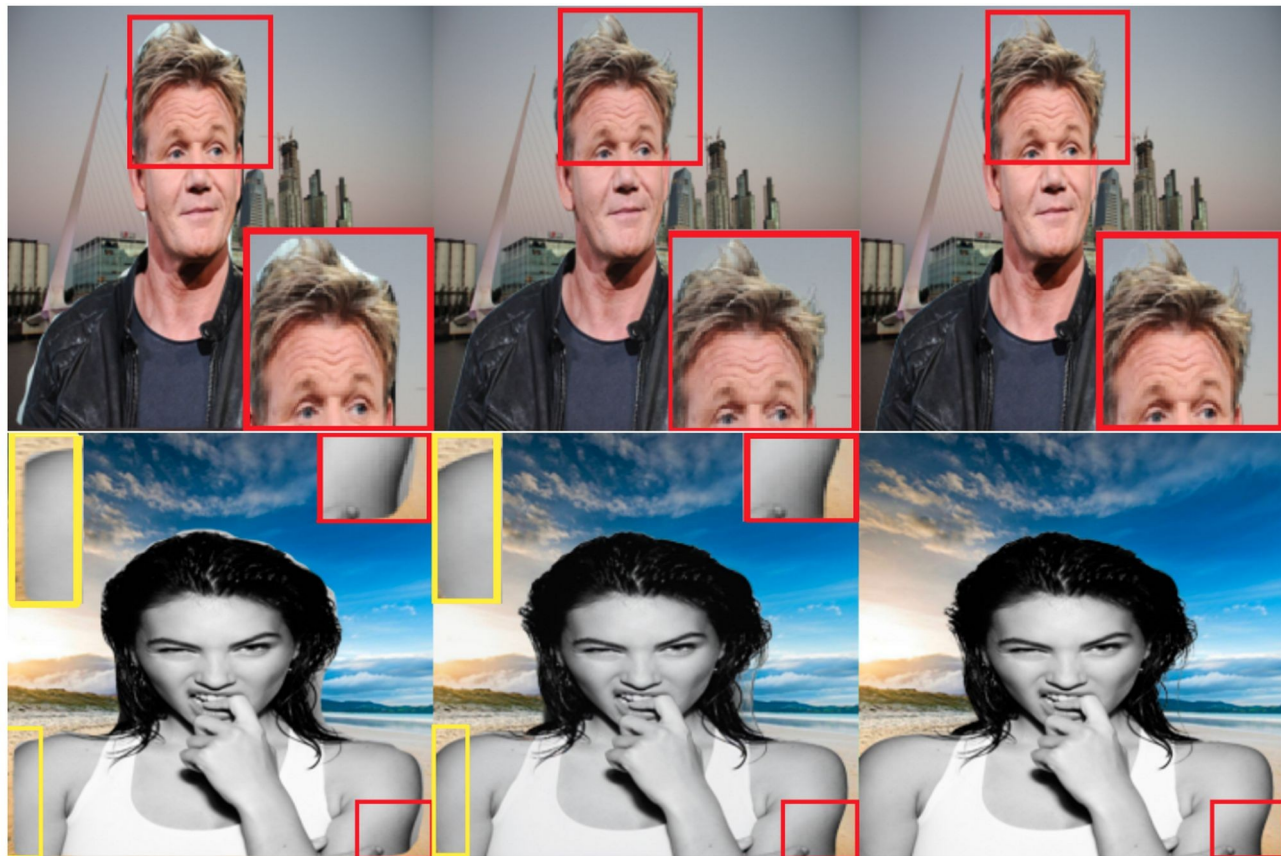
Merging examples - Autostitch

- Stitching panorama from multiple images
- Two-layer blending – high and low frequencies
- Only blend low frequencies, keep high frequencies intact

Low frequencies
High frequencies
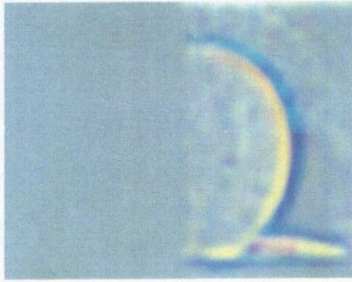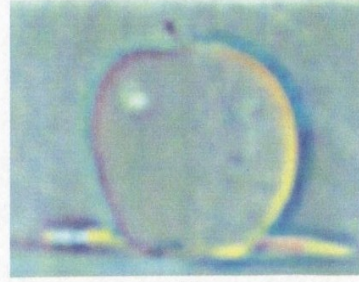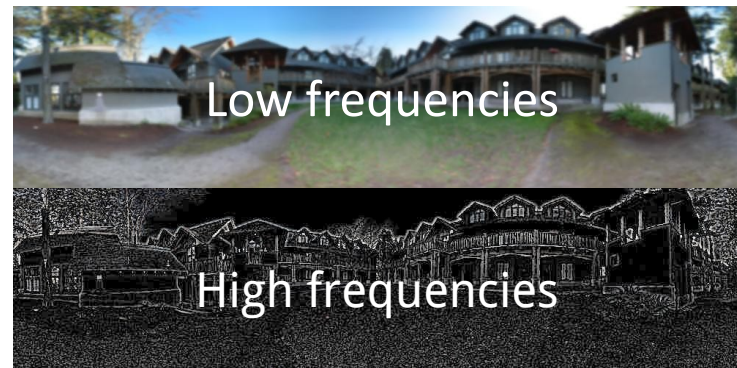

Matthew Brown and David G. Lowe, "Automatic panoramic image stitching using invariant features,"
International Journal of Computer Vision, 74, 1 (2007), pp. 59-73

Hybrid images

- Static images with two interpretations
  - Low frequencies – far away
  - High frequencies - nearby
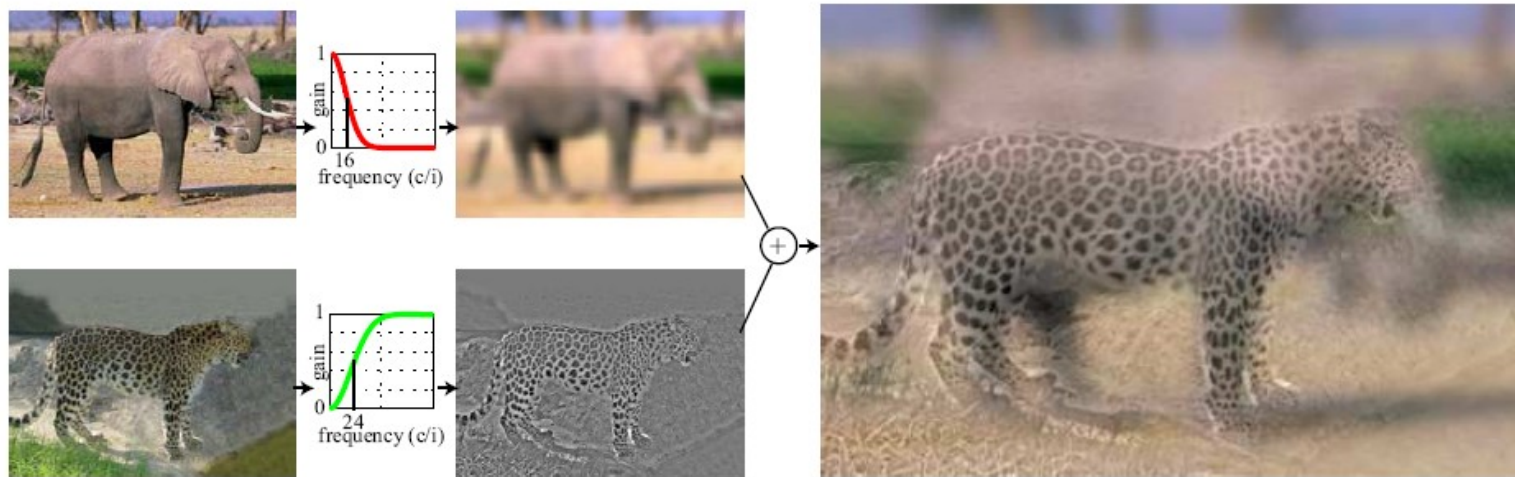
Hybrid images - examples

Interactive segmentation

- Determining segmentation mask is time-consuming
  - Determine per-pixel assignment
  - Easy to make mistakes
- Content-aware interactive segmentation
  - Approximately state interest
  - Algorithm refines the mask on per-pixel level

Segmentation with GrabCut

- Segmentation labels are highly structured
  - Two pixels that are similar are more likely in the same group
  - Two pixels that are near are more likely in the same group
- Formalized using Markov random field
- Solve MRF problem using Graph Cut

GrabCut algorithm

- Input: image, initial mask
- Compute local affinity for all pixels
- Iterate until convergence:
  - Estimate FG and BG models using mask
  - Compute model affinity for all pixels
  - Perform Graph cut for weights
  - Update mask with result

Image

Mask

Image derivatives (local similarity), used in $\psi_{i,j}(x_i, x_j)$

FG   BG

$p(x|F)$ and $p(x|B)$
$p(F|x) = \frac{p(x|F)}{p(x|F)+p(x|B)}$

Converged?

Graph cut

Segmentation

Y. Boykov et al., Fast Approximate Energy Minimization via Graph Cuts, IEEE TPAMI 2001
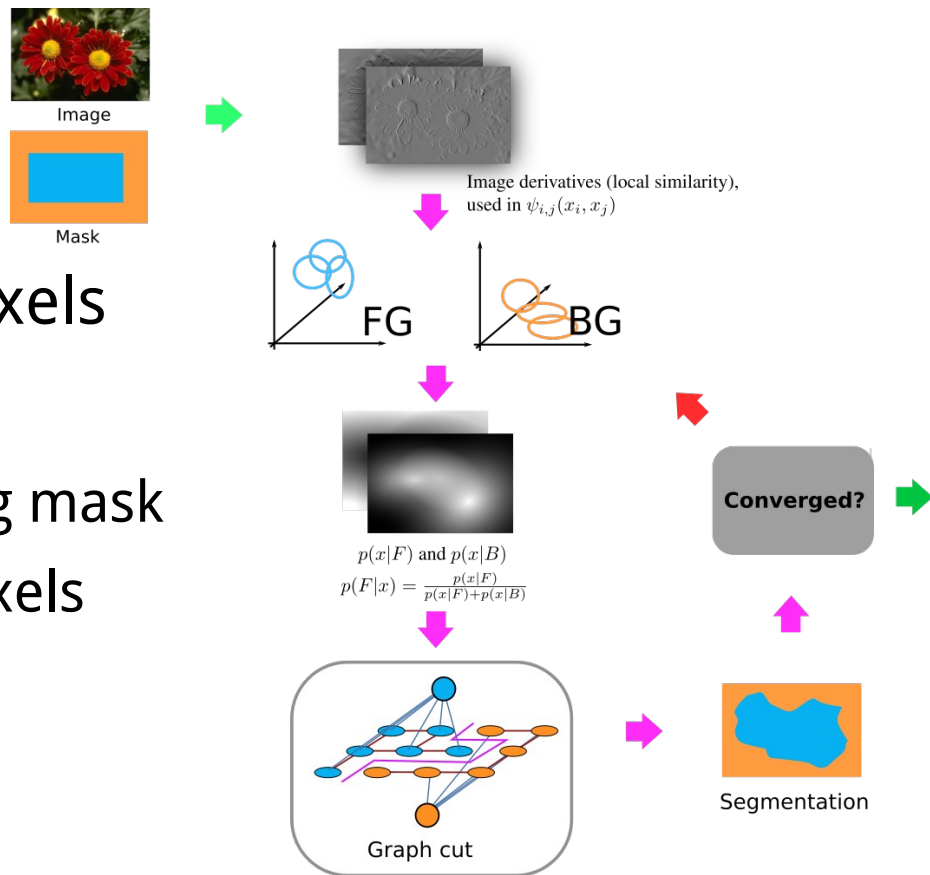
Interactive segmentation

- User selects initial estimate of the object
  - Foreground, background, don't know
- Perform GrabCut with initial mask
- Present result to user
- User can correct result and re-start with updated mask
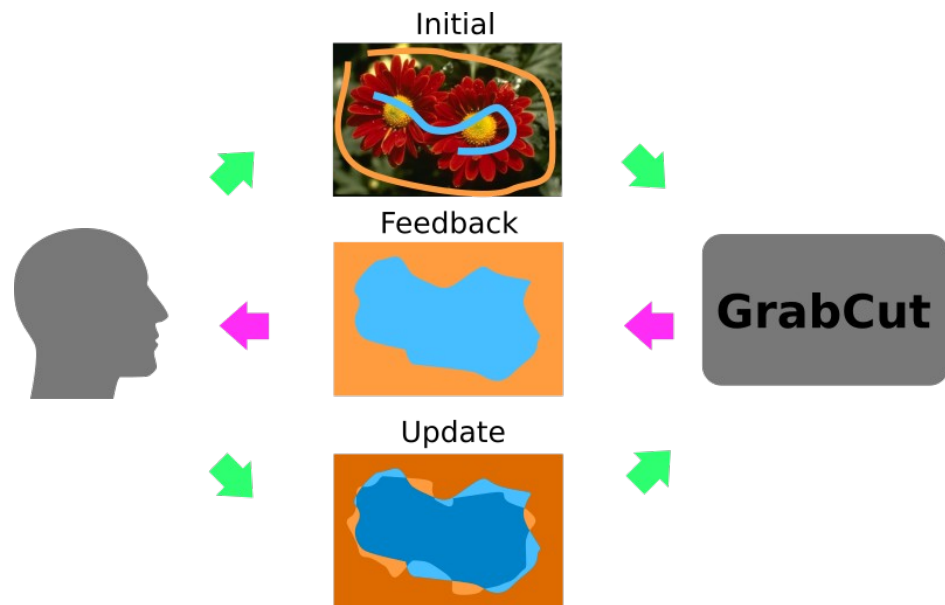
Initial

Feedback

GrabCut

Update

Rother et al. GrabCut: Interactive Foreground Extraction using Iterated Graph Cuts. SIGGRAPH, 2004

Examples

Online demo (non-interactive): https://chocopoule.github.io/grabcutweb/
Interactive GrabCut: https://github.com/louisfb01/iterative-grabcut

C. Rother, et al. GrabCut: Interactive Foreground Extraction using Iterated Graph Cuts. ACM Transactions on Graphics, 2004

Segmentation with deep learning

- Train model to segment objects with limited data
  - Bounding box
  - Extreme points
  - Center + corners
- Use rich image structure to determine boundaries - segmentation

DEXTR architecture

- Input
  - Image
  - Extreme points encoded as gaussians
- Output – binary segmentation
- Architecture – DeepLab v2
- Training
  - COCO dataset
  - Simulate clicks from segmentation

DEXTR examples