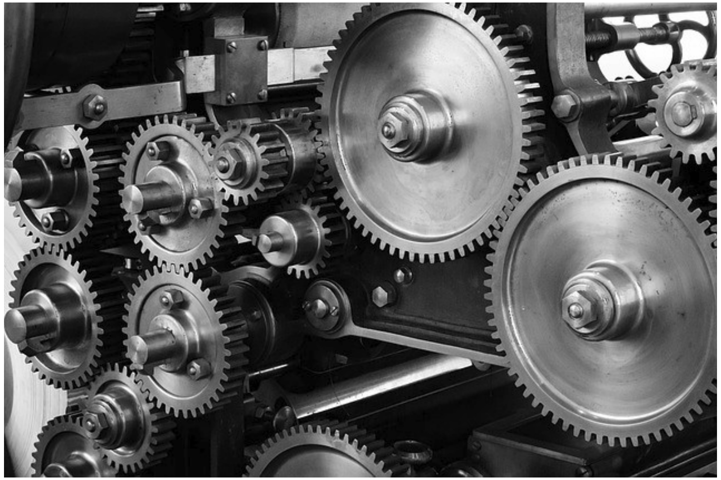


Image processing

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

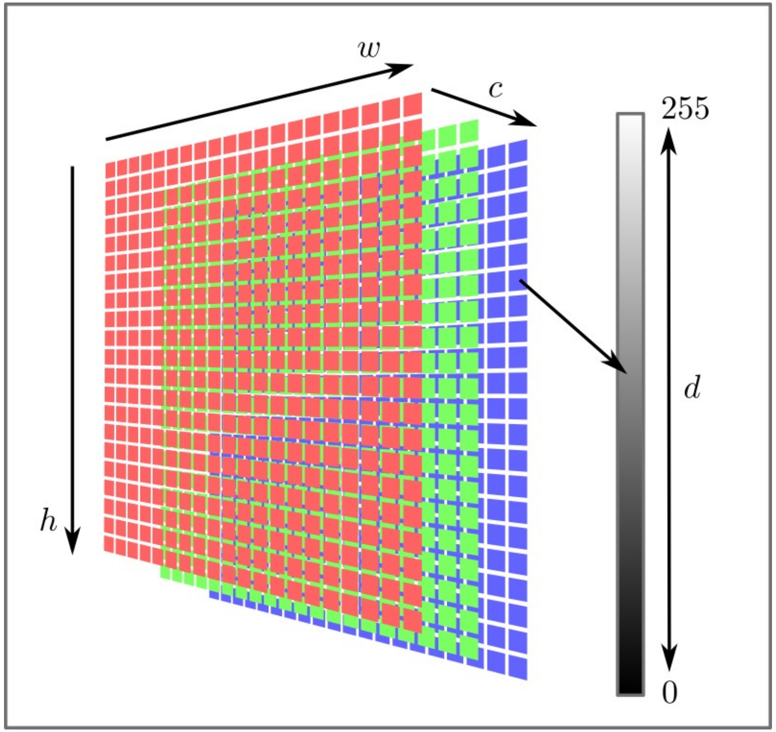
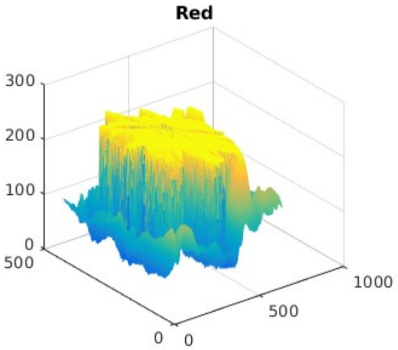
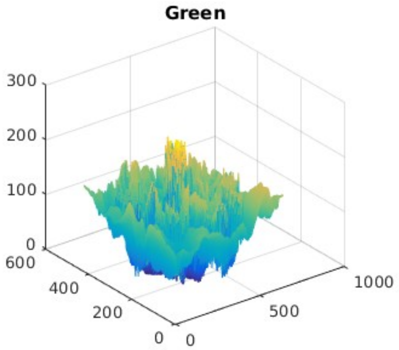


Image as a function

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





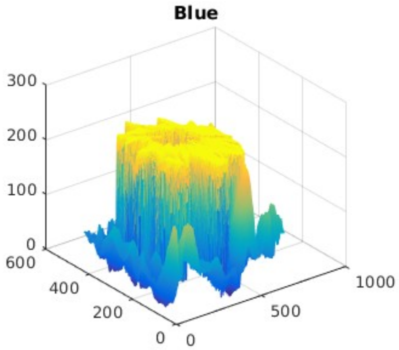
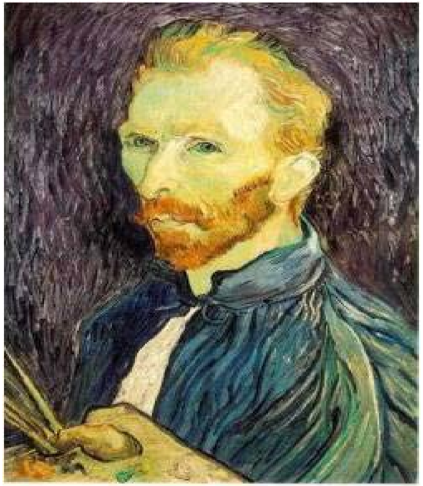


Image operations

- Intensity
  - Pixel-wise
  - Histogram
  - Filtering
- Geometrical
  - Linear
  - Local
- Complex / combined / custom

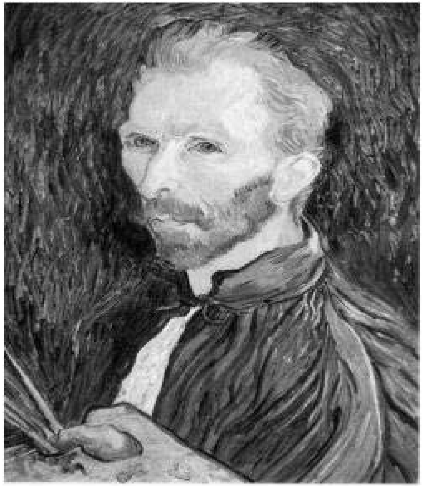
Conversion to grayscale

- From RGB: (weighted) averaging of channels



$$V = R + G + B$$

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



- With different weights we can set the importance of channels

Pixel-wise operation: negation

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

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

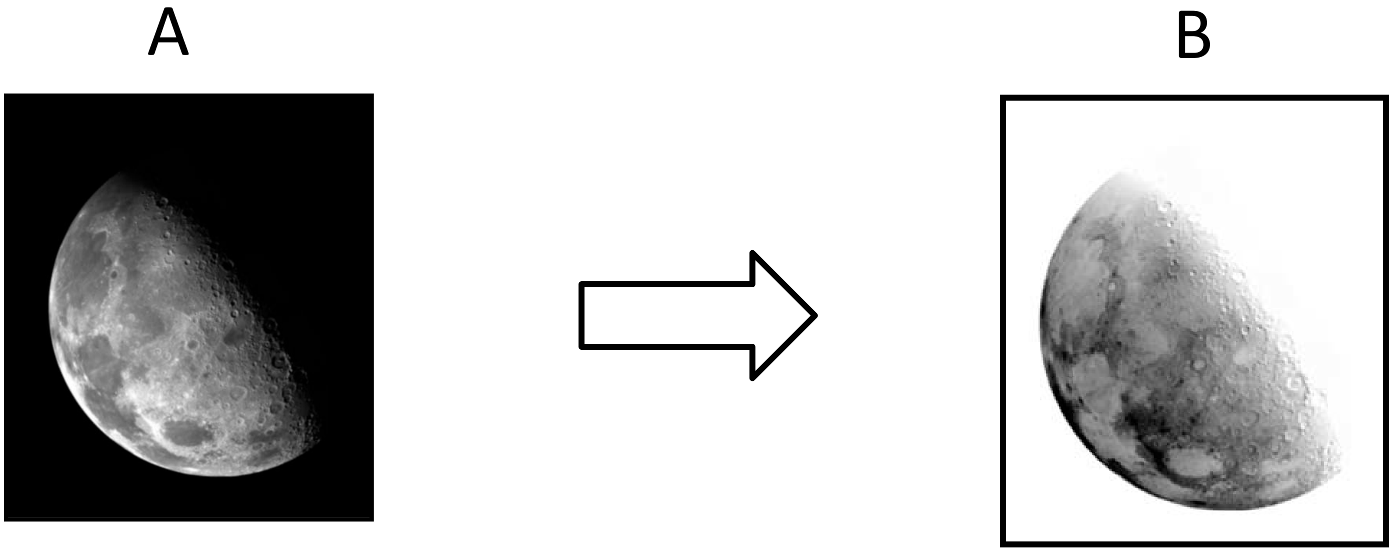
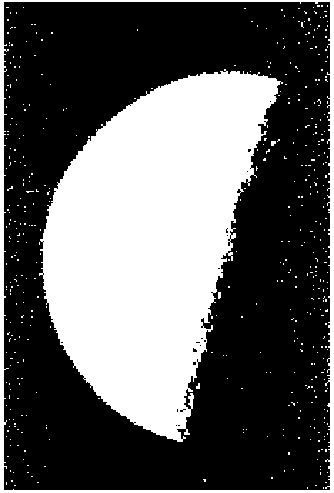


Image threshold

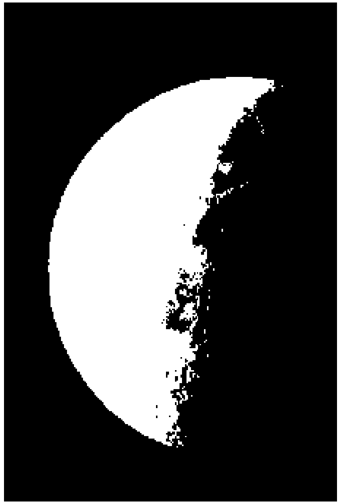
Pixel values higher than value T are set to 1, others to 0

I

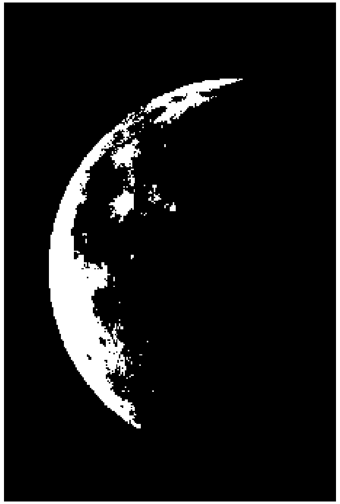
I>10



I>100



I>200



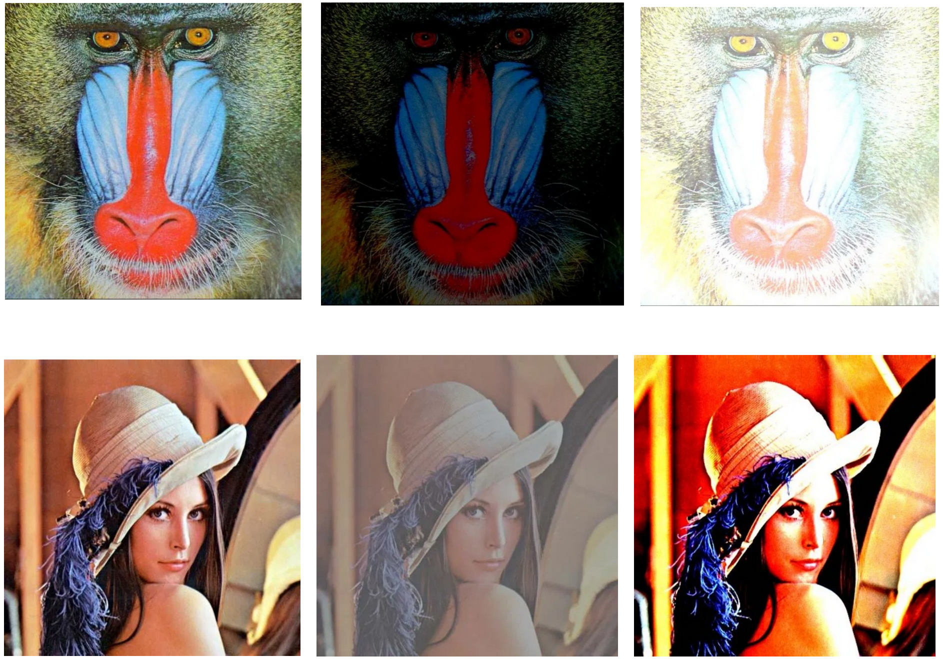
Determining an optimal threshold is not trivial

Brightness and contrast

- **Brightness** - intensity of a pixel relative to another pixel
- **Contrast** – difference between minimum and maximum pixel

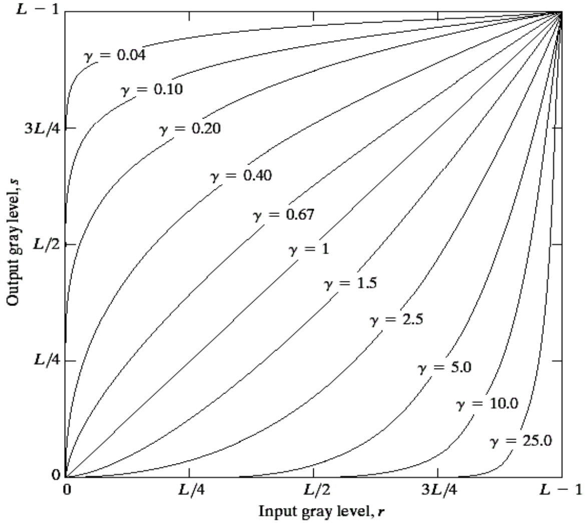
$$\mathrm{f(x)} = \alpha x + \beta$$

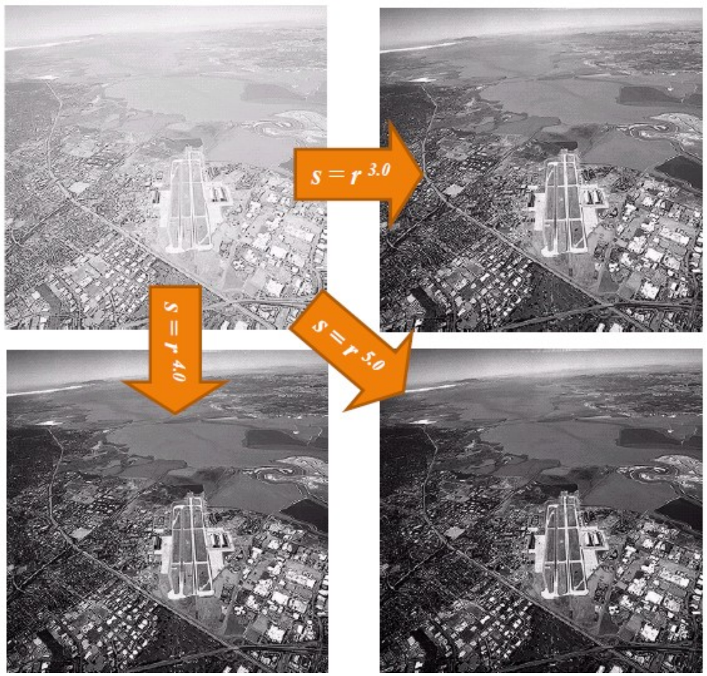
$$\mathrm{f(x)} = \alpha(x - 128) + 128 + b$$



Nonlinear transformations

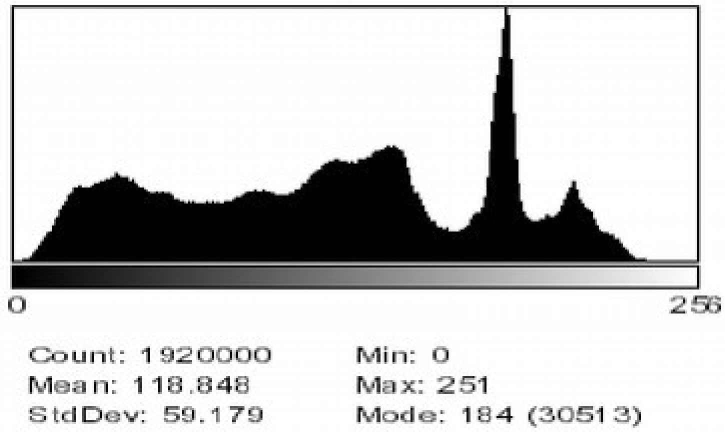
- Parametric function that maps source values **r** to destination values **s**.
- Exponential function family $s = cr^{\gamma}$
- Parameter c is usually 1
- Parameter r is in [0, 1]





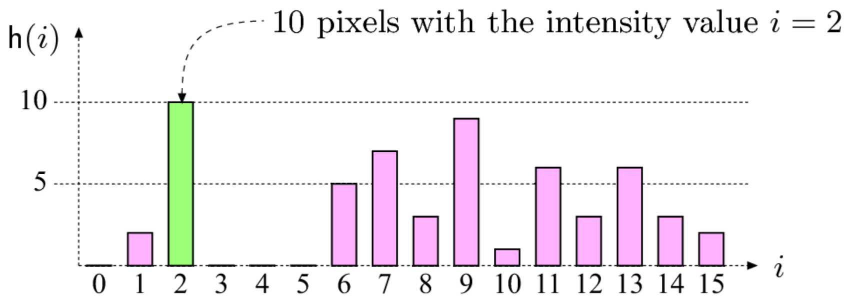
Distribution of values in images

- How to adjust values based on the image?
- Use image-specific statistics - histograms

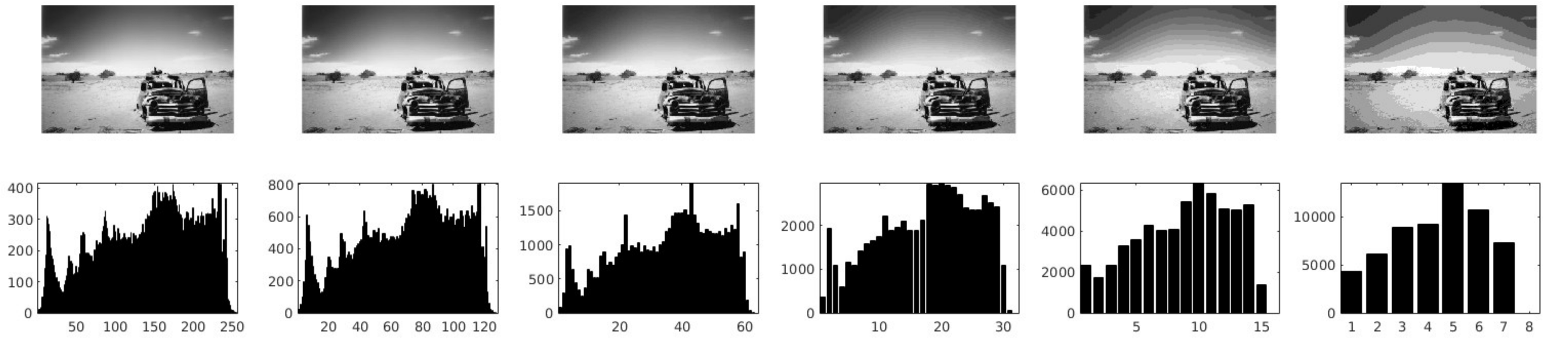


Histogram

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

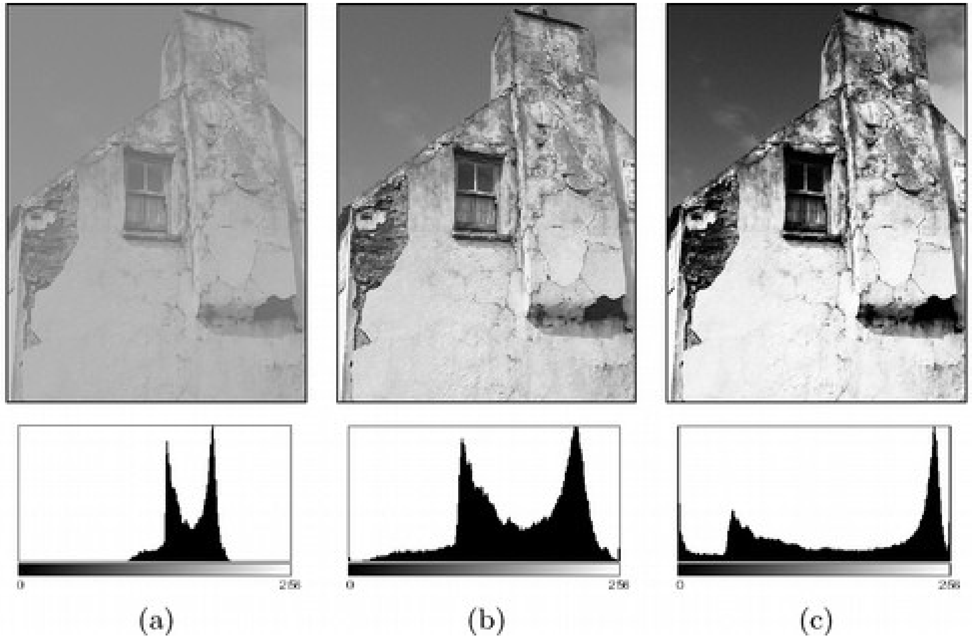


h(i)	0	2	10	0	0	0	5	7	3	9	1	6	3	6	3	2
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15



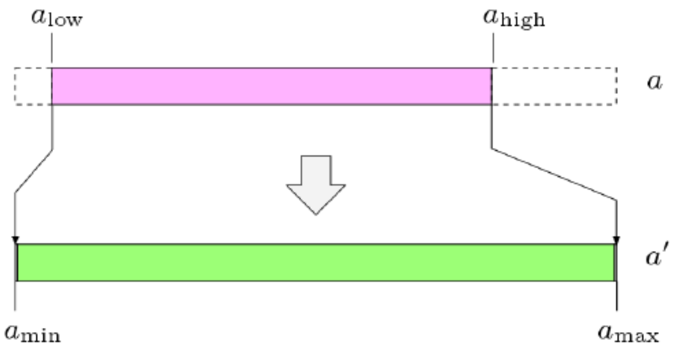
Histogram and image quality

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



Histogram stretching

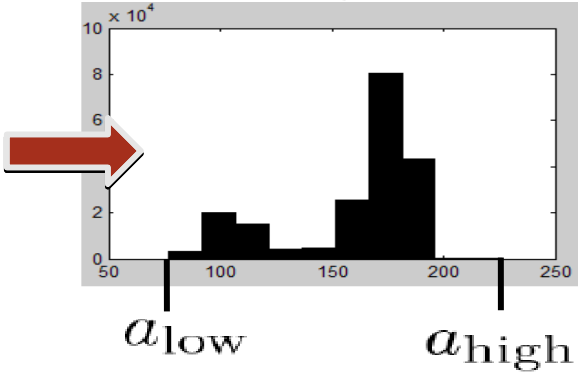
$$f_{ac}(a) = a_{\min} + (a - a_{\text{low}}) \cdot \frac{a_{\max} - a_{\min}}{a_{\text{high}} - a_{\text{low}}}$$



Operation performed on each pixel individually.

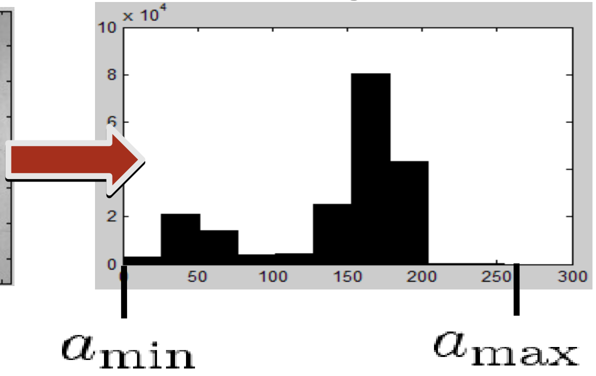
input

histogram



output

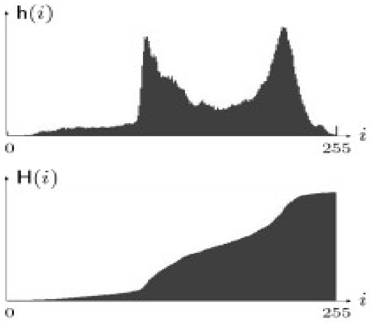
histogram

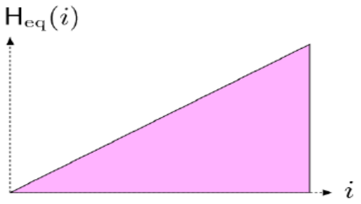


Histogram equalization

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

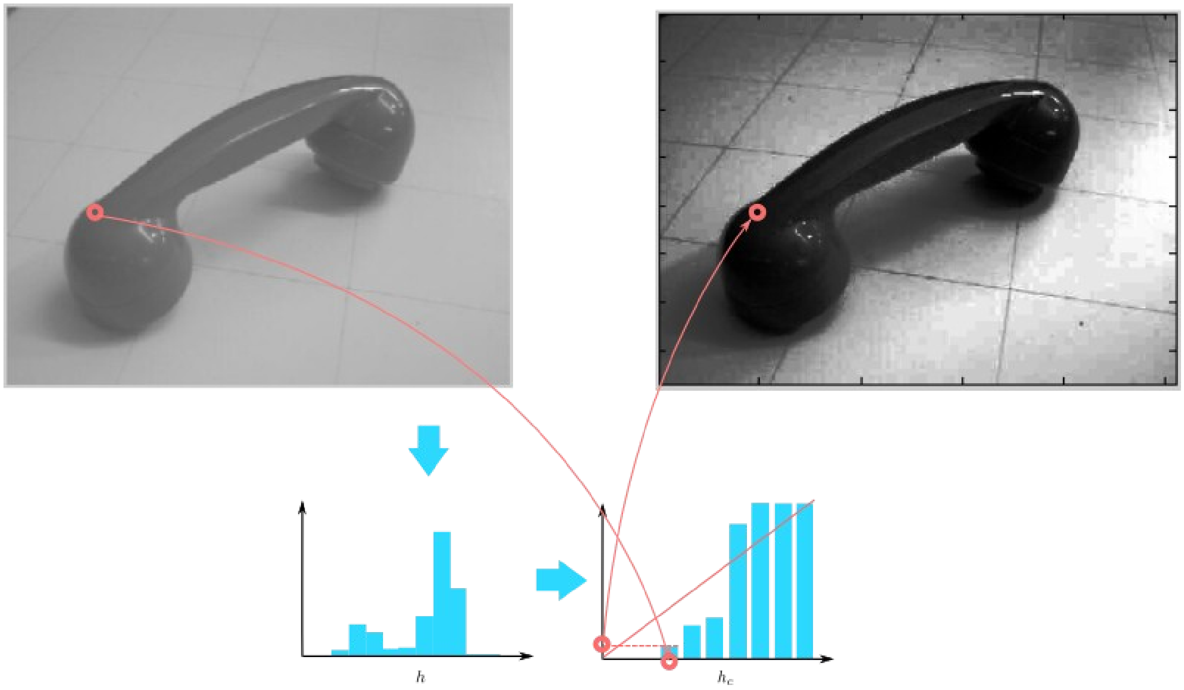
$$\mathsf{H}(i) = \begin{cases} \mathsf{h}(0) & \text{for } i = 0 \\ \mathsf{H}(i-1) + \mathsf{h}(i) & \text{for } 0 < i < K \end{cases}$$





Equalization algorithm

- Compute 256-bin histogram of image I (h).
- Compute cumulative histogram.
- Normalize cumulative histogram with maximum value.
- Multiply normalized cumulative histogram with 255 (hc).
- Use hm as a lookup table to transform individual pixels.



Histogram equalization in color images

Original

RGB Equalized Independently

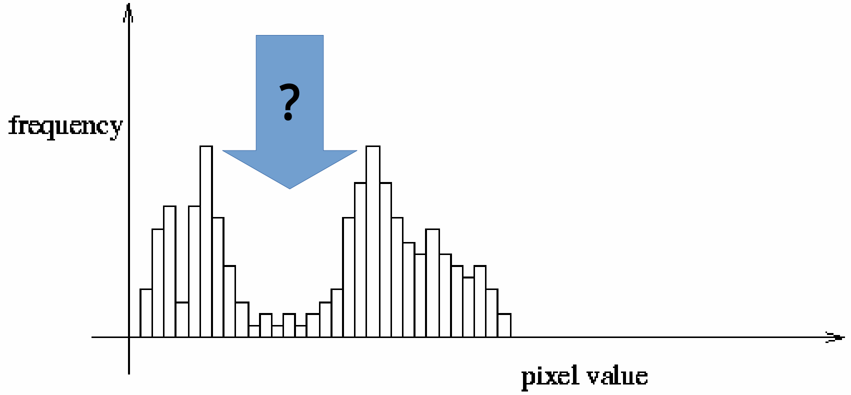
Because each channel is transformed independently, the resulting color changes as well.

Luminance Equalization

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

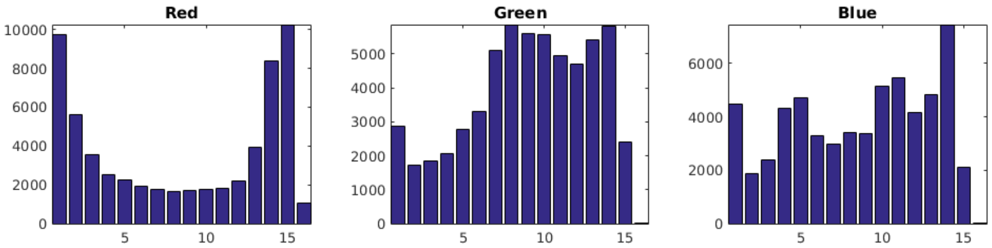
Histogram and thresholding

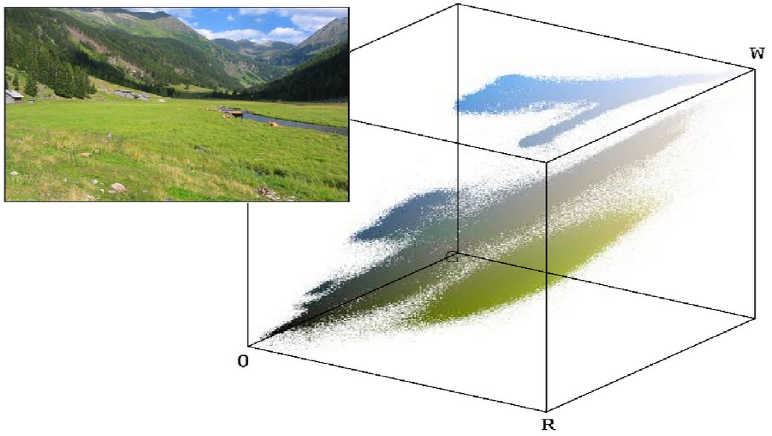
- Analyze histogram to find good threshold
- Bi-modal histogram
- Otsu method
  - Minimize variance of foreground and background



Color histogram

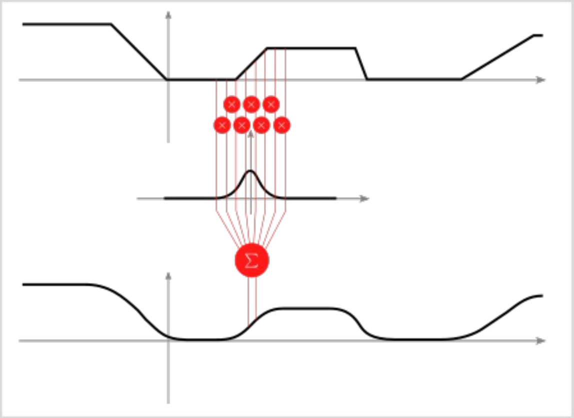
- 3 histograms
  - Each component sepa
  - No correlation
  - Less space
- 3D histogram
  - Image color is a 3D index
  - More specific
  - More space

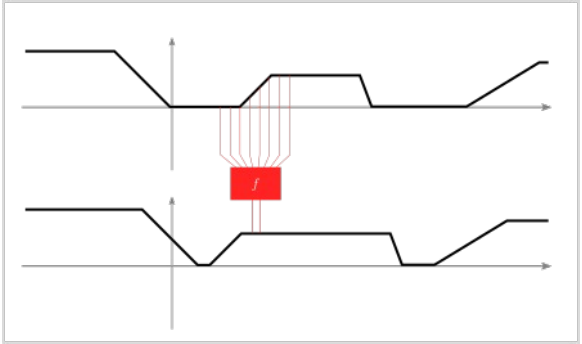




Filtering

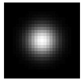
- Resulting value dependent on the neighborhood
- Linear filters
  - Convolution / correlation
  - Associativity
  - Separability
- Nonlinear filters
  - Arbitrary (local) operation
  - Max, min
  - Median





Linear filters

image



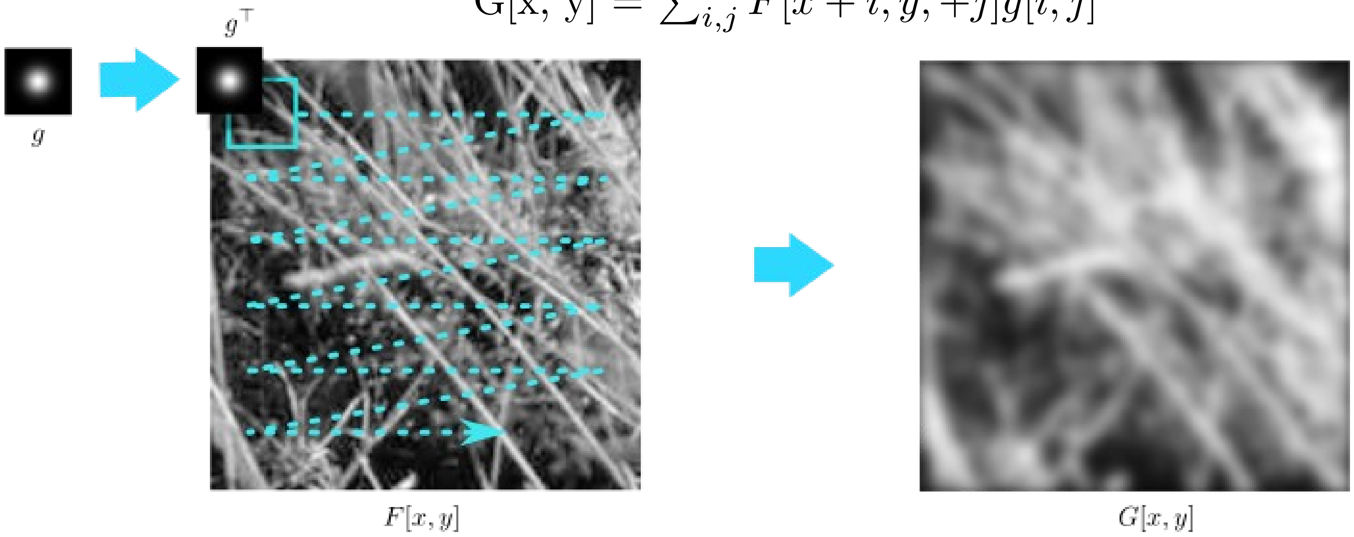
kernel

result

In a nutshell

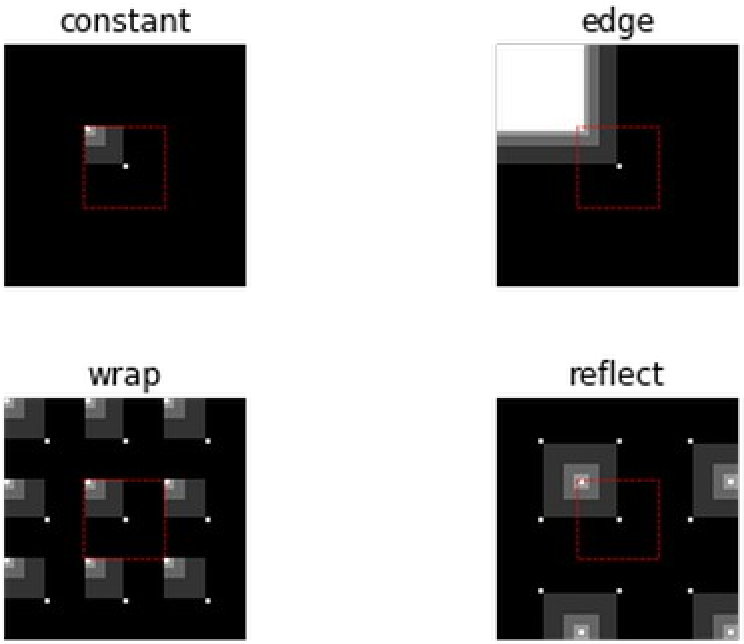
- How to compute filter response in individual pixel?
  - Transpose kernel (convolution) and align its center with the pixel
  - Multiply corresponding elements and sum together

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



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



Weighted sum

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	255	255	255	255	255	0	0
0	0	0	255	255	255	255	255	0	0
0	0	0	255	255	255	255	255	0	0
0	0	0	255	0	255	255	255	0	0
0	0	0	255	255	255	255	255	0	0
0	0	0	0	0	0	0	0	0	0
0	0	255	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

g

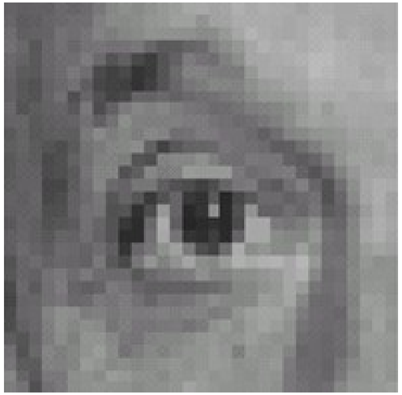
$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$G[x, y]$

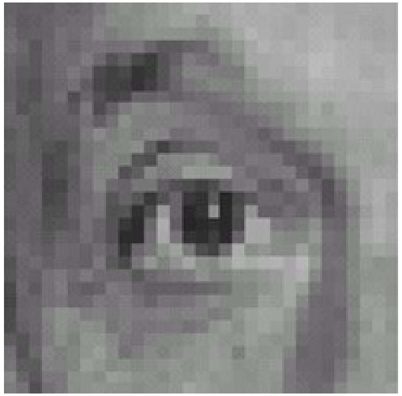
0	28	57	85	85	85	57	28
0	57	113	170	170	170	113	57
0	85	170	255	255	255	170	85
0	85	142	227	227	255	170	85
0	85	142	227	227	255	170	85
0	57	85	142	142	170	113	57
28	57	85	85	85	85	57	28
28	28	28	0	0	0	0	0

what do to with the border?

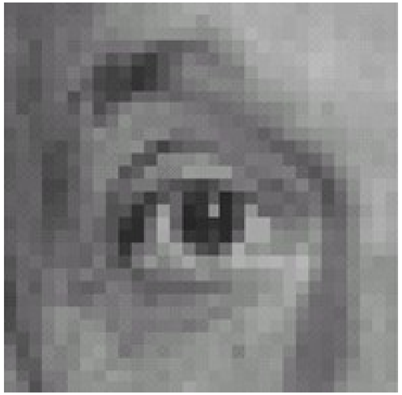
Identity filter



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



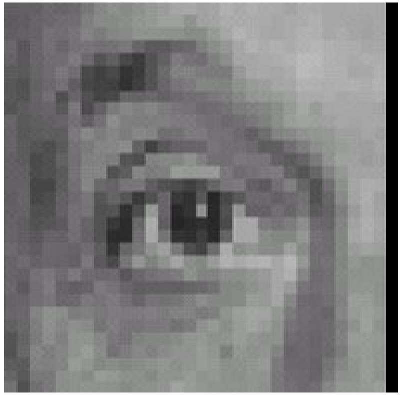
Shift filter



$*$

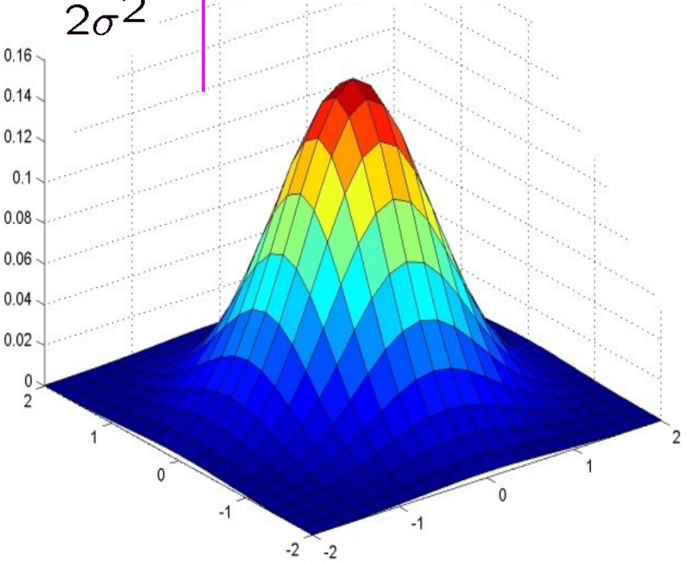
0	0	0
0	0	1
0	0	0

$=$



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

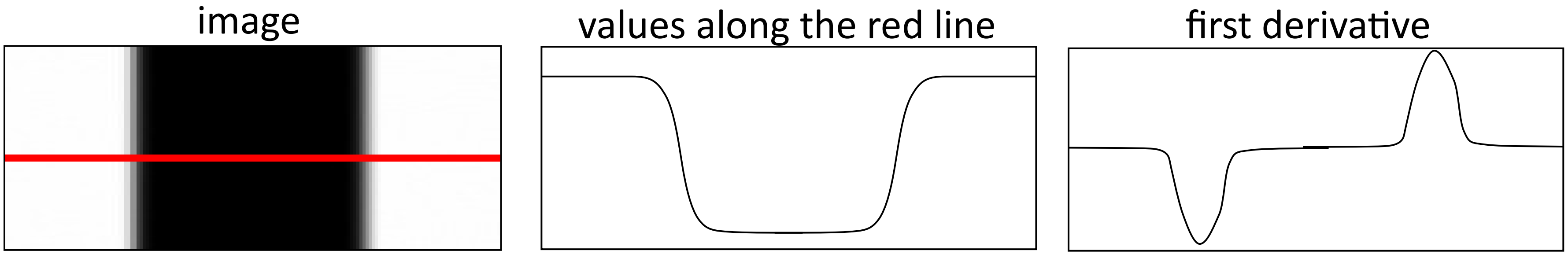
7 x 7, 📅 = 1

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

Source: C. Rasmussen

Detecting edges

- Goal: find sudden changes in illumination in the image
- Ideal: line drawing by an artist (semantic knowledge)



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

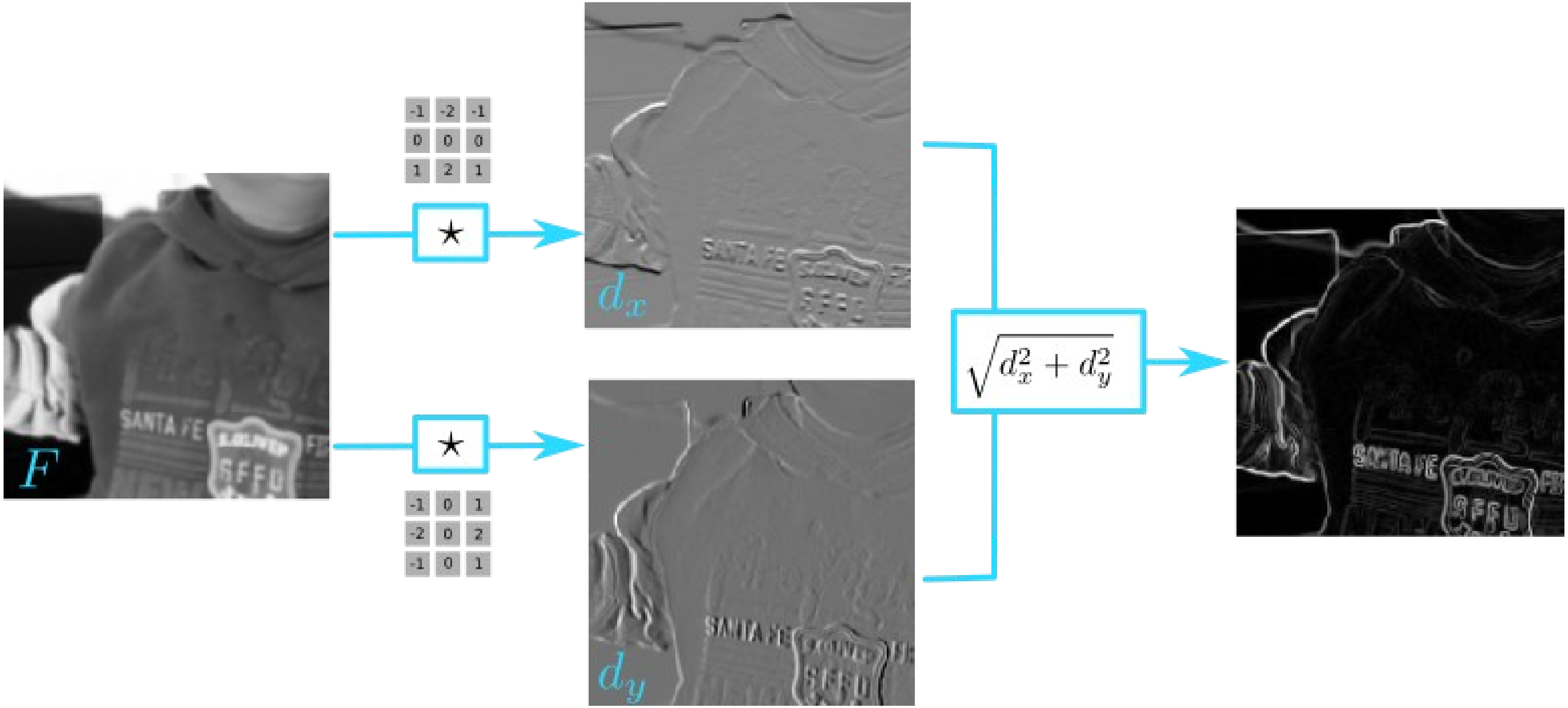
1	2	1
0	0	0
-1	-2	-1

Roberts

0	1
-1	0

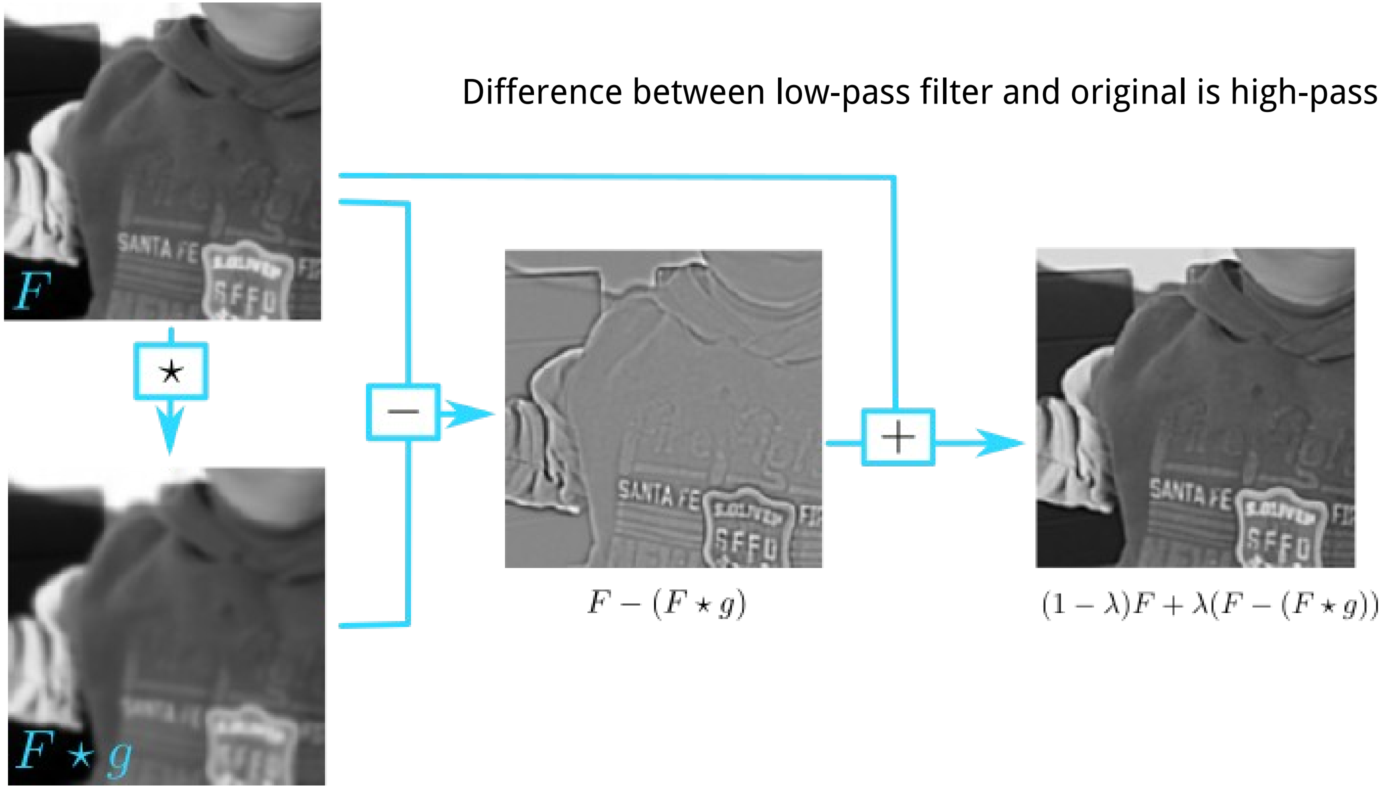
1	0
0	-1

Derivative magnitude



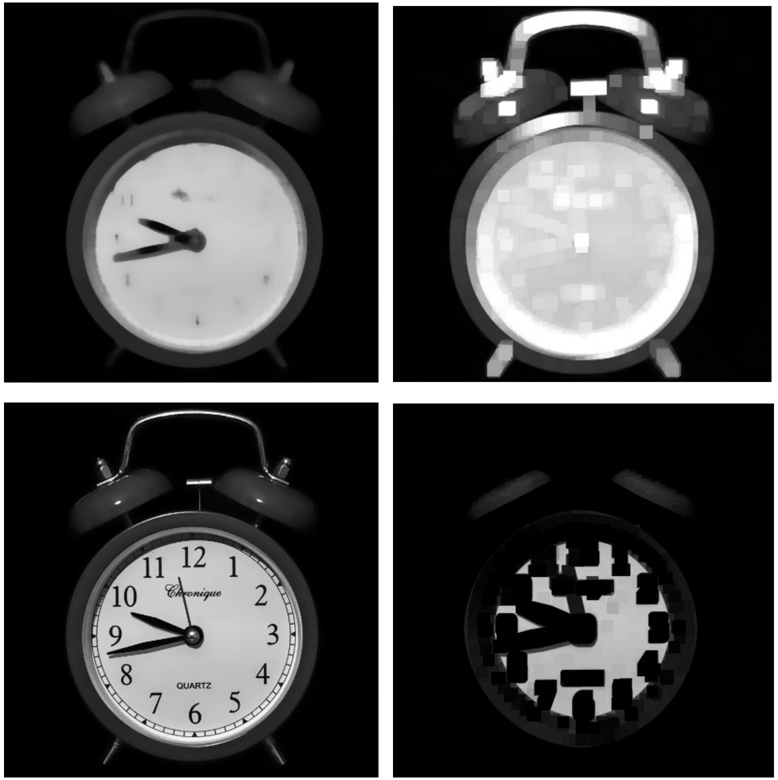
Sharpening by blurring

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

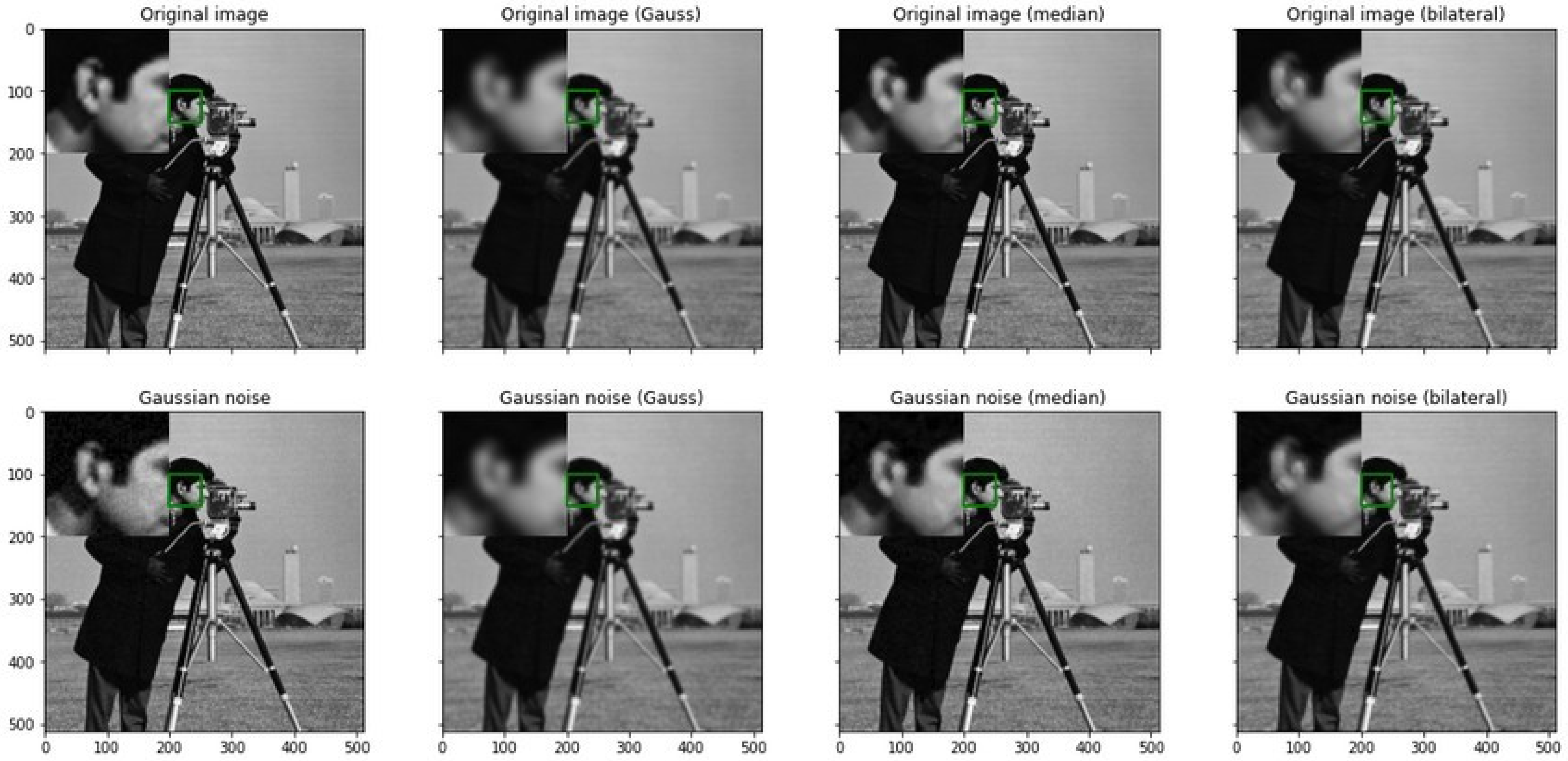


Non-linear filters

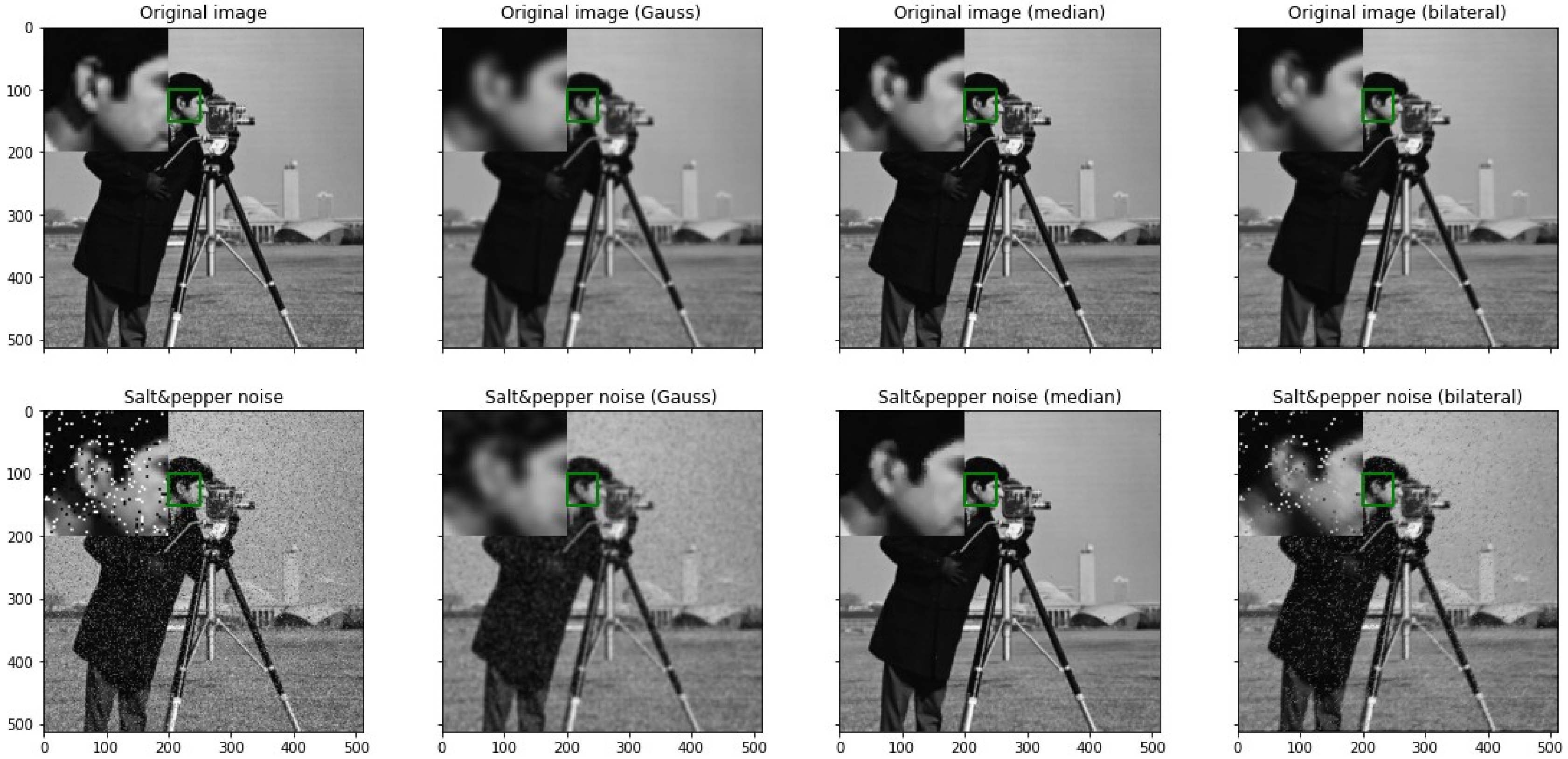
- Median: middle element by value
- Bilateral filter
  - Weights based on neighborhood
  - Preserves edges
- Morphological operations
  - Max: highest element in neighborhood
  - Min: lowest element in neighborhood



Gaussian noise removal



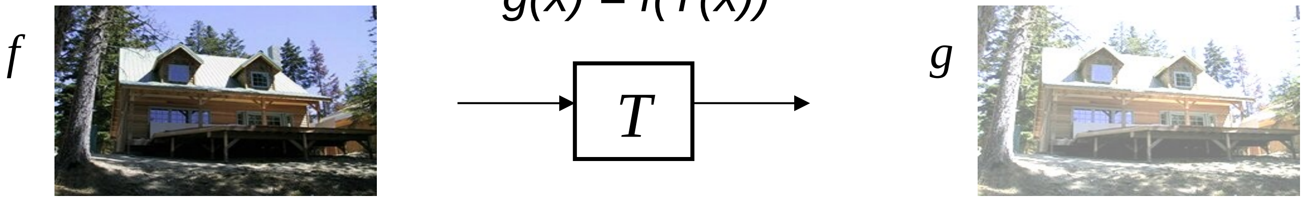
Salt&pepper noise removal



Geometry vs. intensity

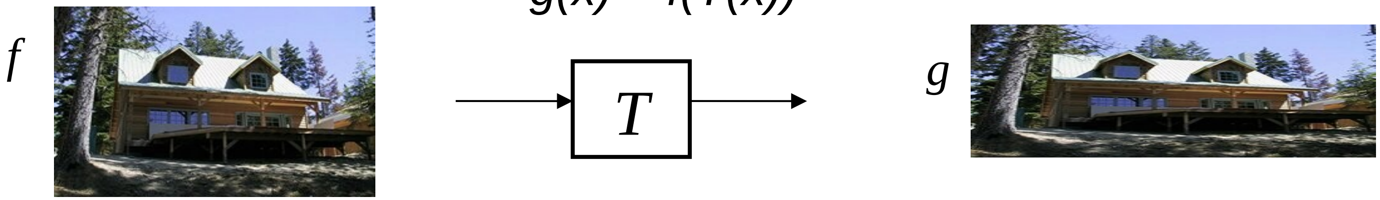
- Image filtering is foremost an intensity transformation

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

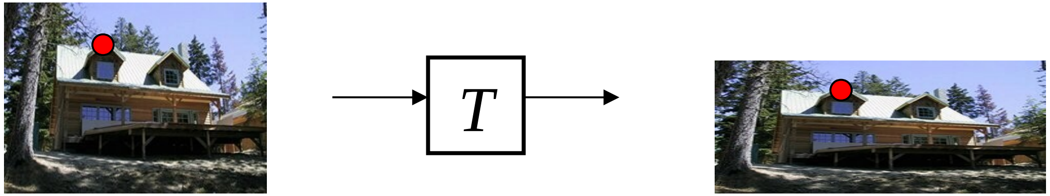


- Geometry transformation changes geometry of the image

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



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

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

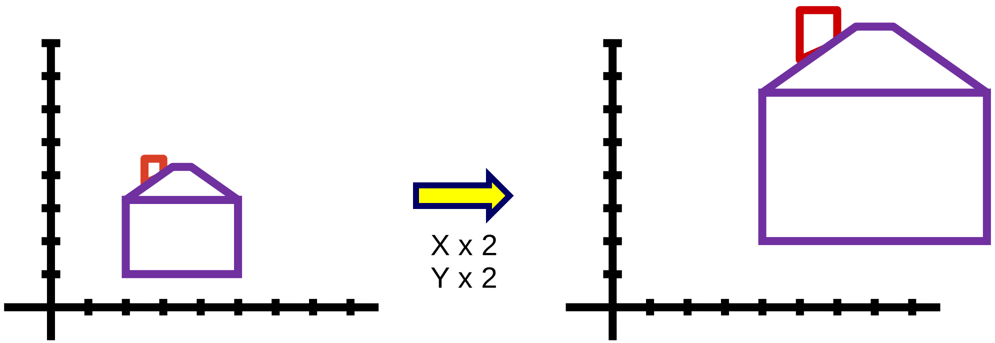
- Mirroring: $\begin{aligned} x' &= -x \\ y' &= -y \end{aligned}$ $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

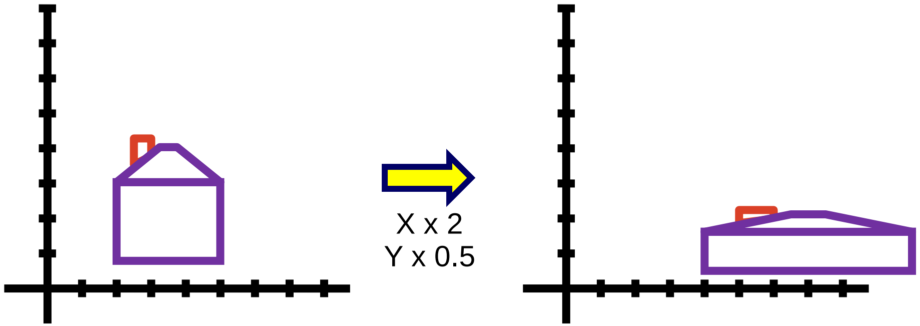
Scaling

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

$$x' = \alpha x \qquad y' = \beta y$$

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





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

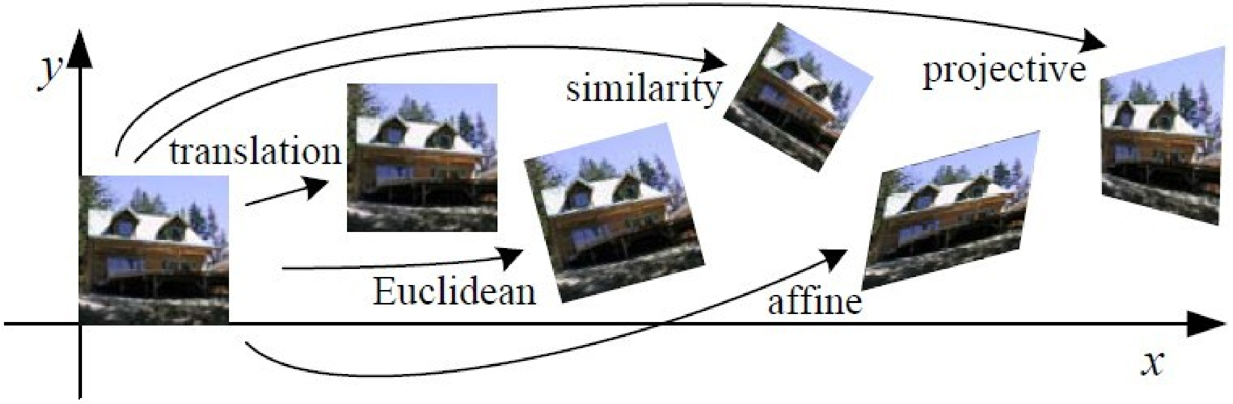
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \alpha_x & 0 \\ \alpha_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- General projective transform

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

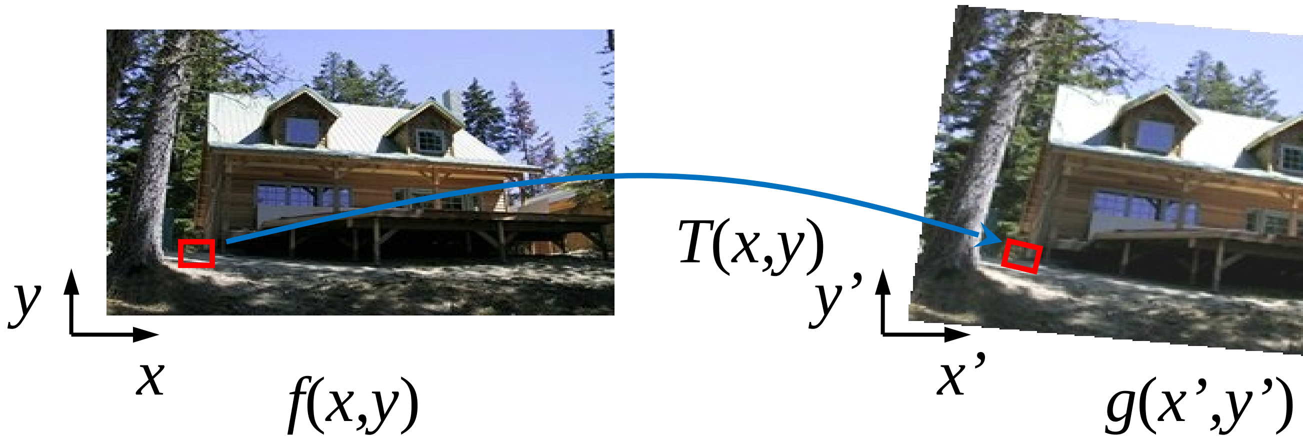
Common transformations

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



Warping

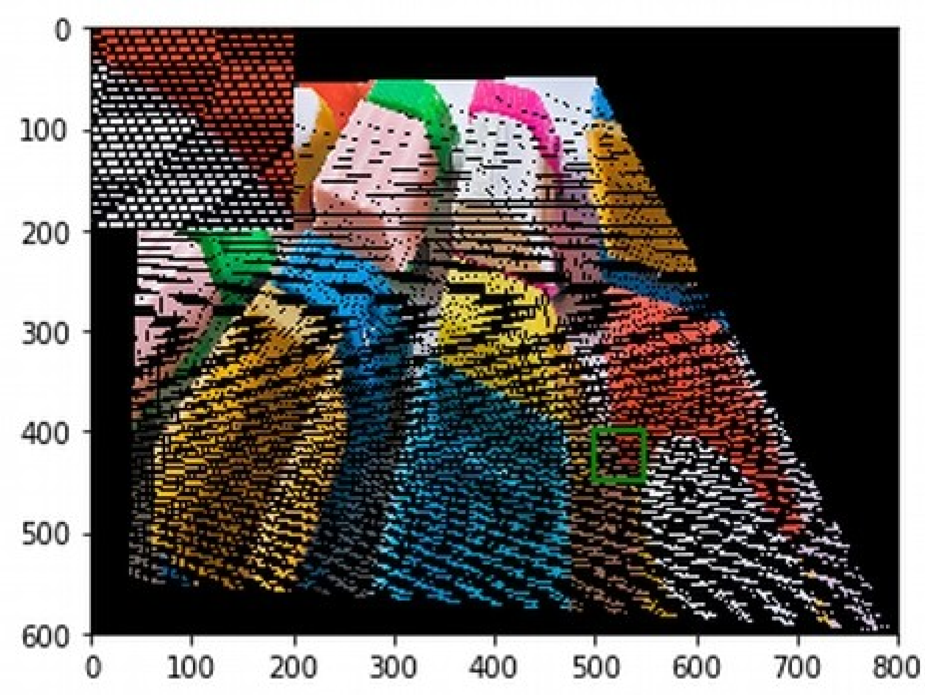
Given transform [x',y'] = T(x,y) and f(x,y), how to compute g(x',y') = f(T(x,y)) ?



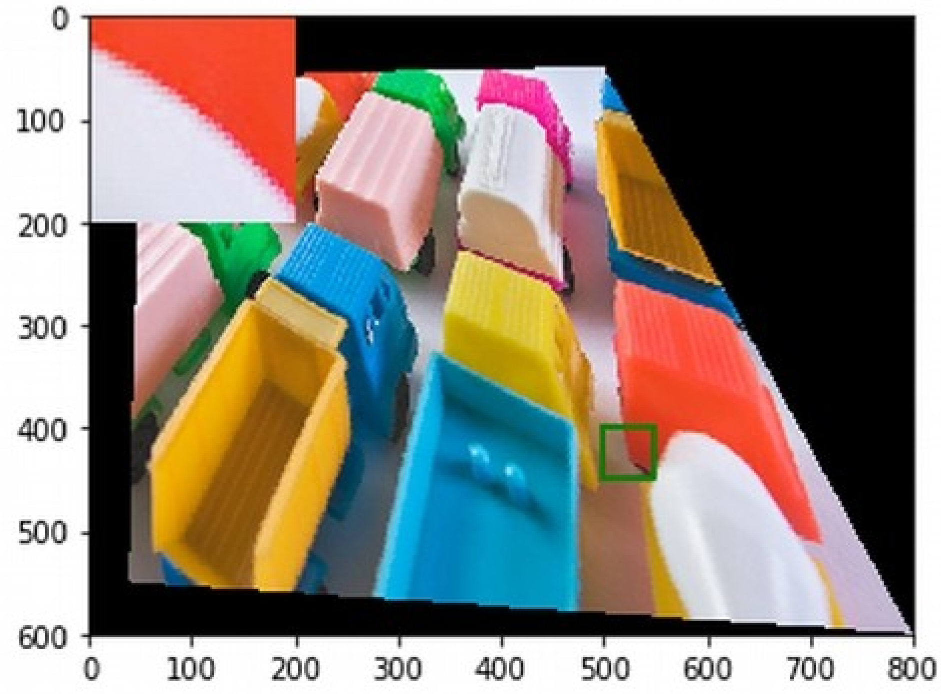
Naive approach

- 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



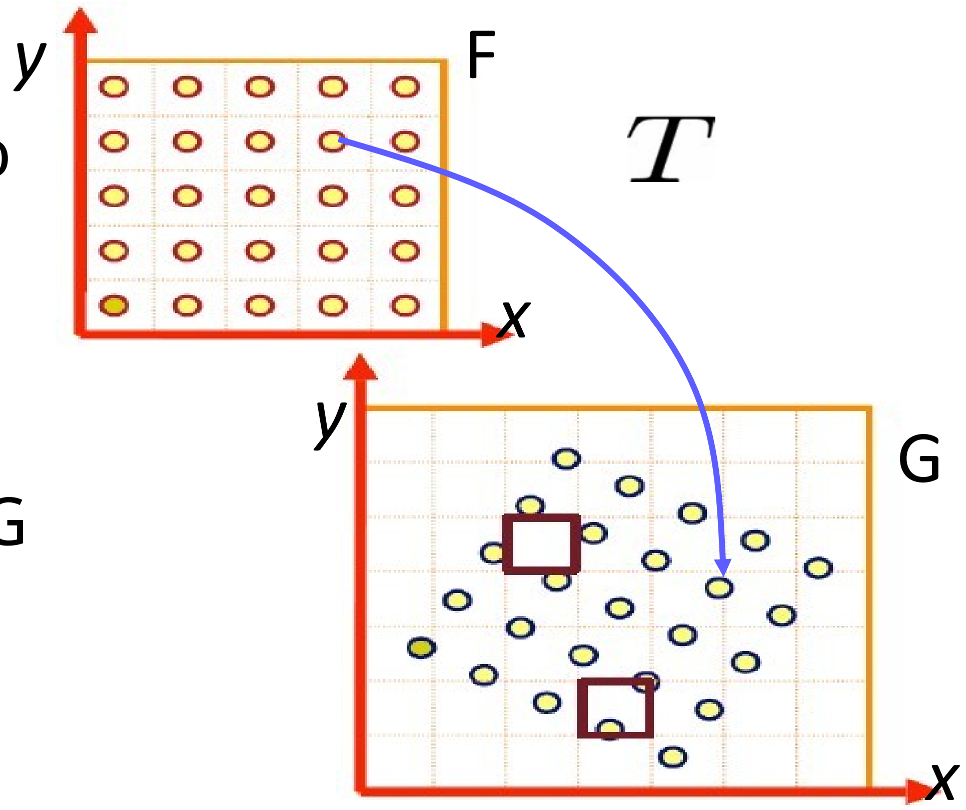
0
100
200
300
400
500
600
0
100
200
300
400
500
600
700
800



0
100
200
300
400
500
600
0
100
200
300
400
500
600
700
800

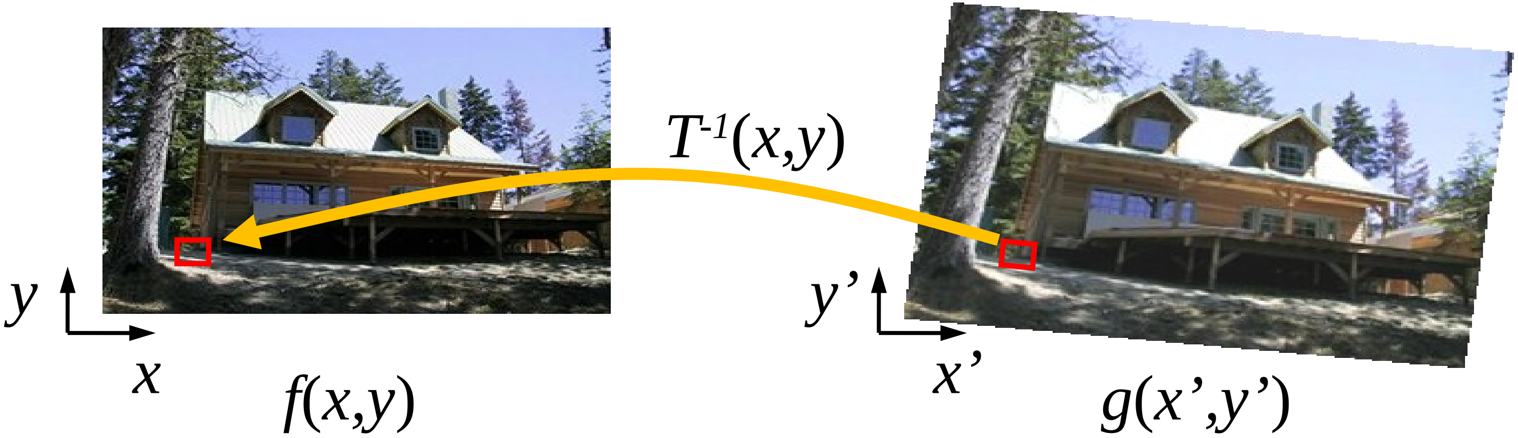
Problems

- A single pixel of F is mapped to more pixels in G.
- Pixel in F is not mapped to any pixel in G.
- Guarantee to 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



Resizing the image

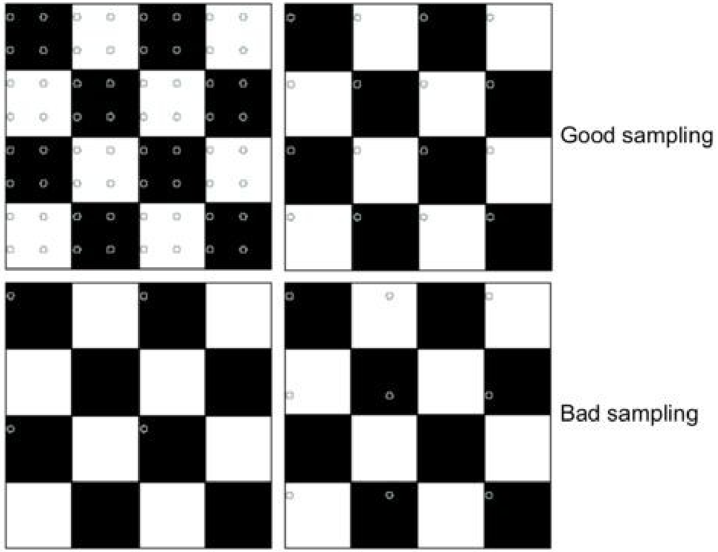
- Adjusting (local or global) resolution in image.
- Two general cases:
  - Reduction / decimation / down-sampling – shrinking image
  - Interpolation / up-sampling – enlarging image

Decimation approaches

- Sampling values
  - Shannon sampling theorem
  - Remove high frequencies

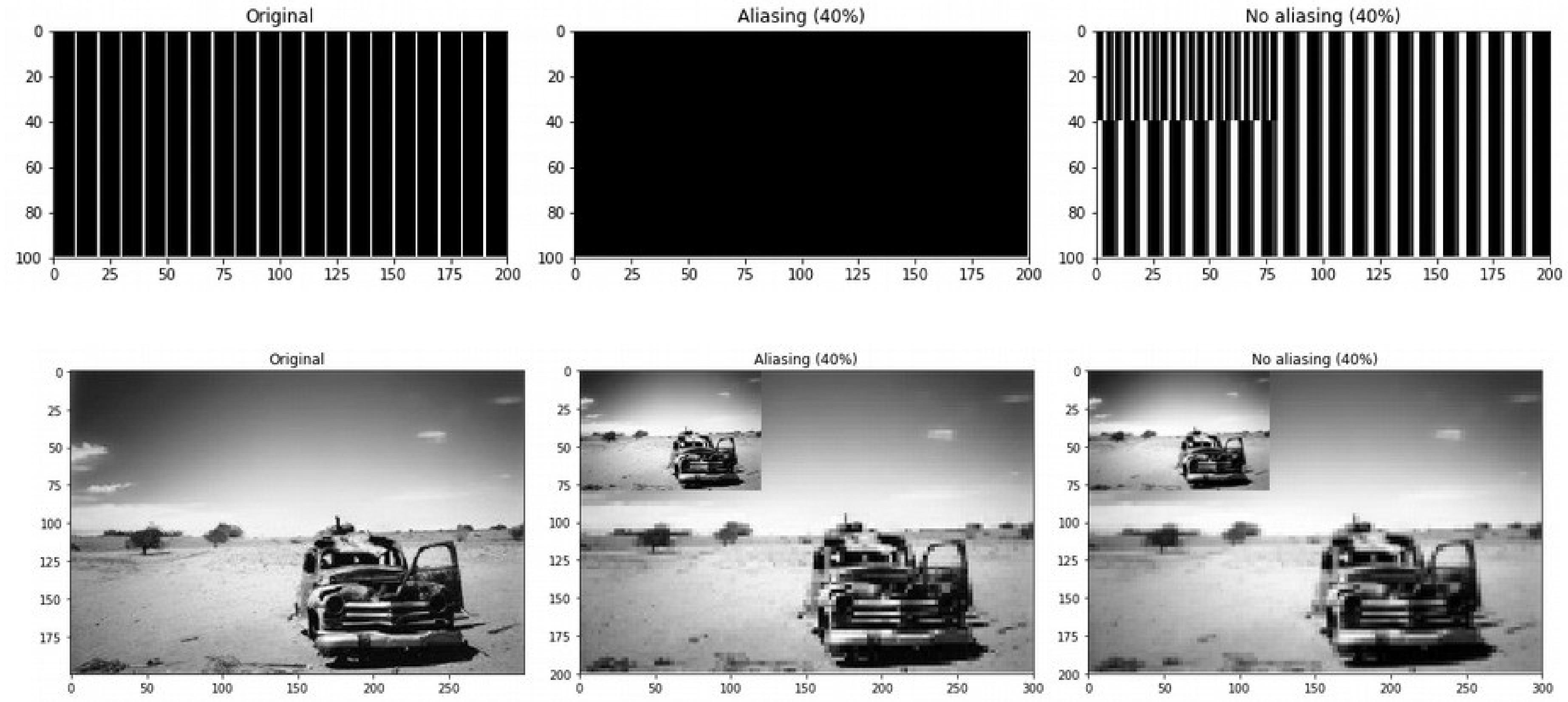
$$f_s \geq 2f_{max}$$

- Anti-aliasing
  - First remove high frequencies
  - Then subsample with appropriate frequency



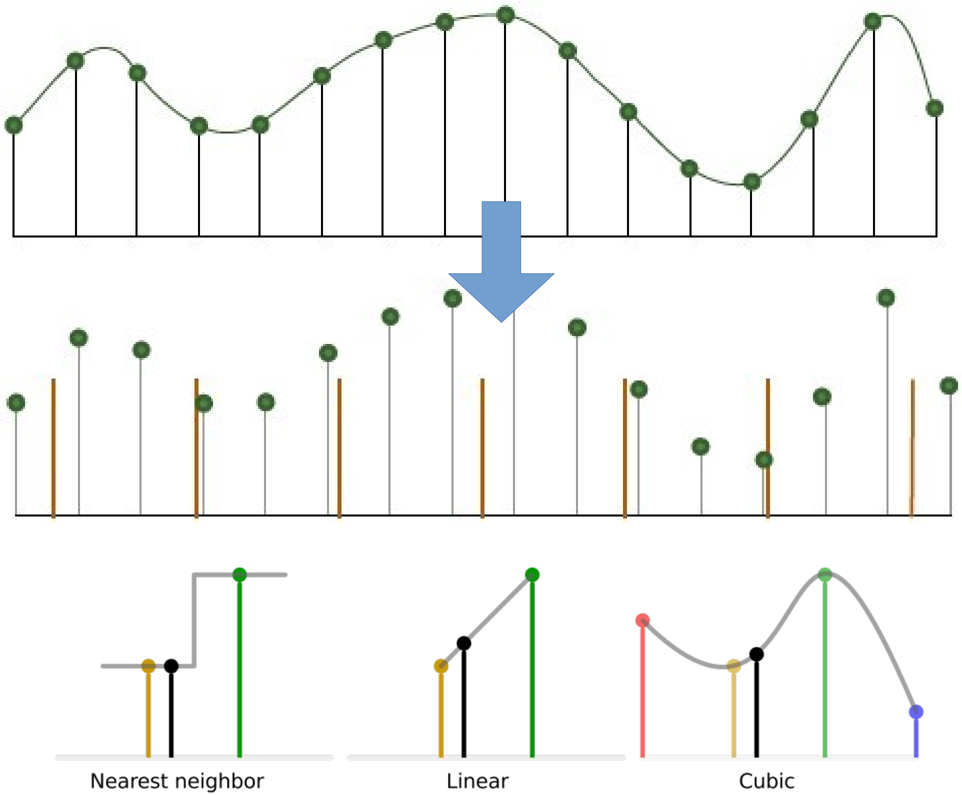
Maël Fabien

Decimation examples



Up-sampling – guessing the values

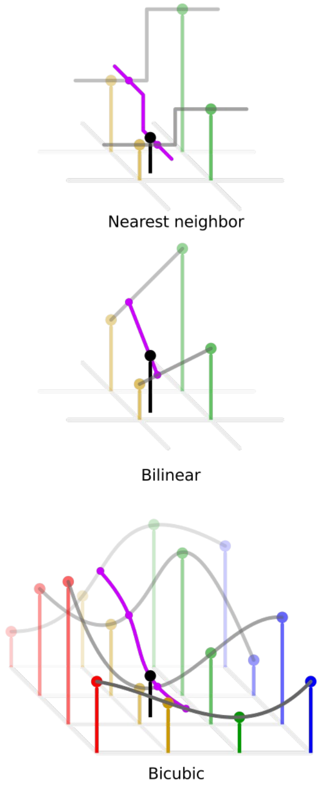
- Nearest neighbor
  - Take value of the closest sample
- Linear interpolation
  - Use two values
  - Fit linear function
- Cubic interpolation
  - Use four values
  - Fit third-degree polynomial
- Lanczos kernel
  - Approximation of ideal low-pass filter



Author: Cmglee (CC-SA)

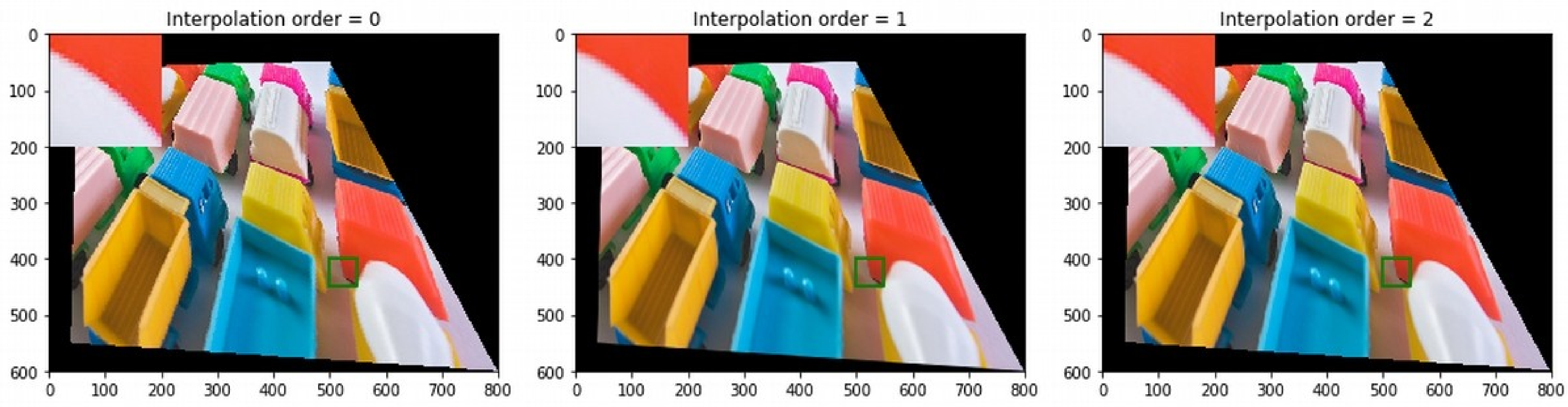
Interpolation in images

- Nearest neighbour
  - Take pixel closest to desired coordinates
- Bilinear
  - Use four closest pixels
  - Estimating a plane
- Bicubic
  - Use 16 closest pixels
  - Estimating a polynomial surface
  - Slower



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

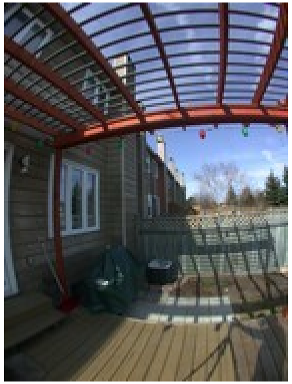


Image morphing

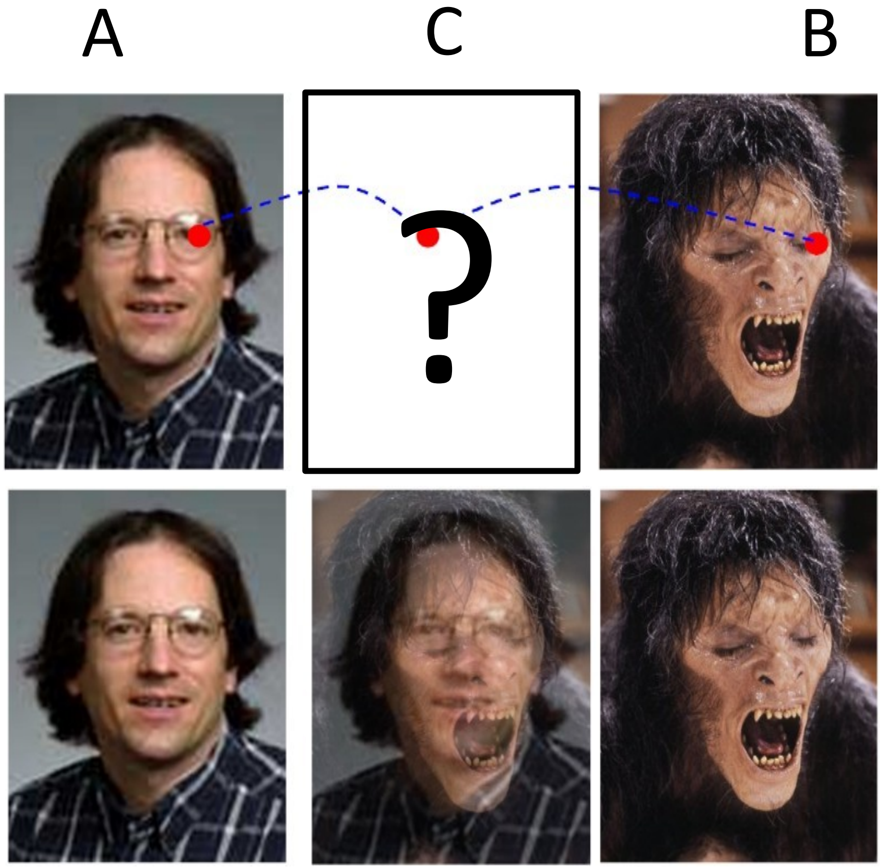
How to deform image A to image B?

Image morphing

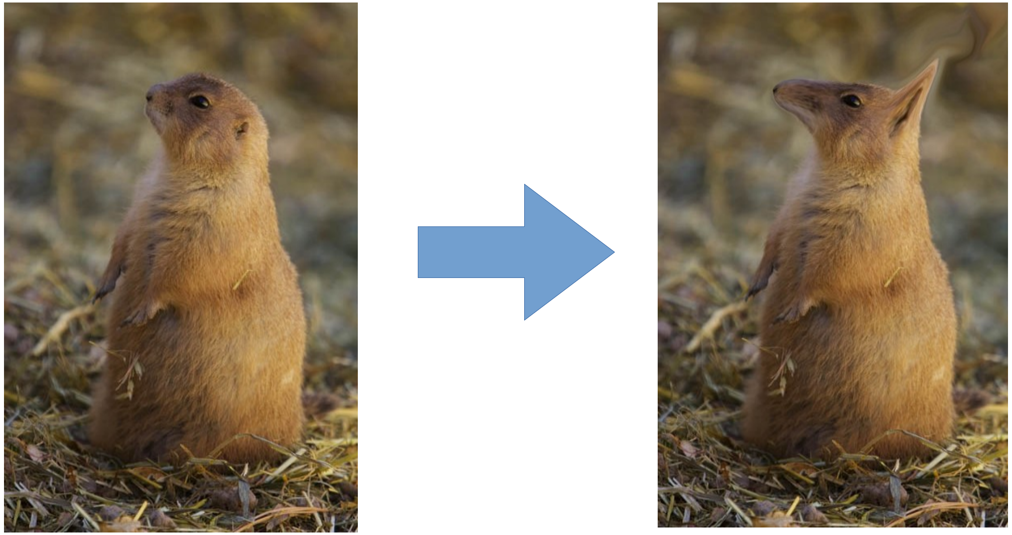
- How to compute intermediate image C
- Naive approach - weighted sum of pixels

$$C_t = \alpha_t A + (1 - \alpha_t) B$$
$$0 \leq \alpha_t \leq 1$$

- Not realistic - not combining semantic parts



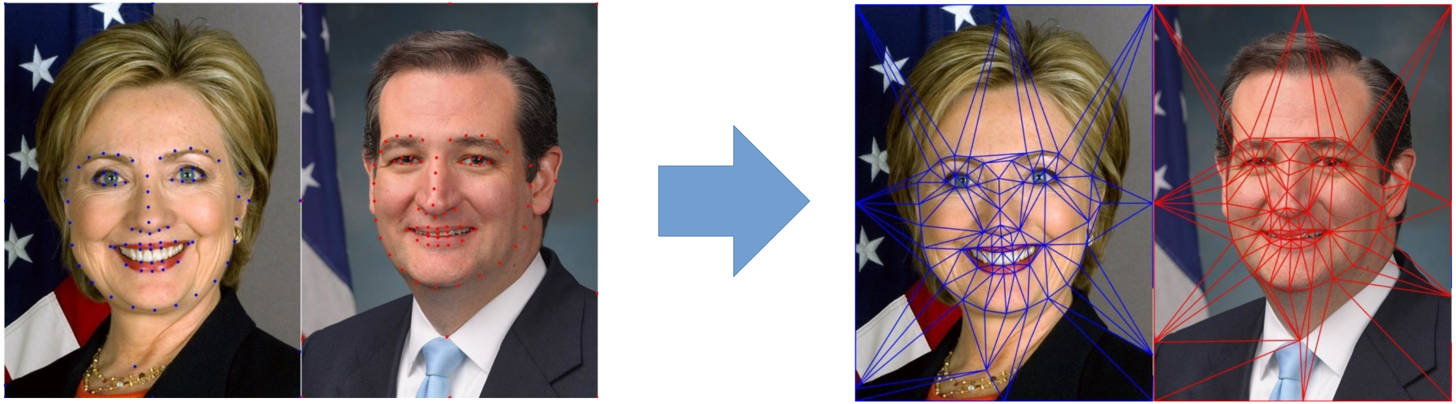
Dense deformation field



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

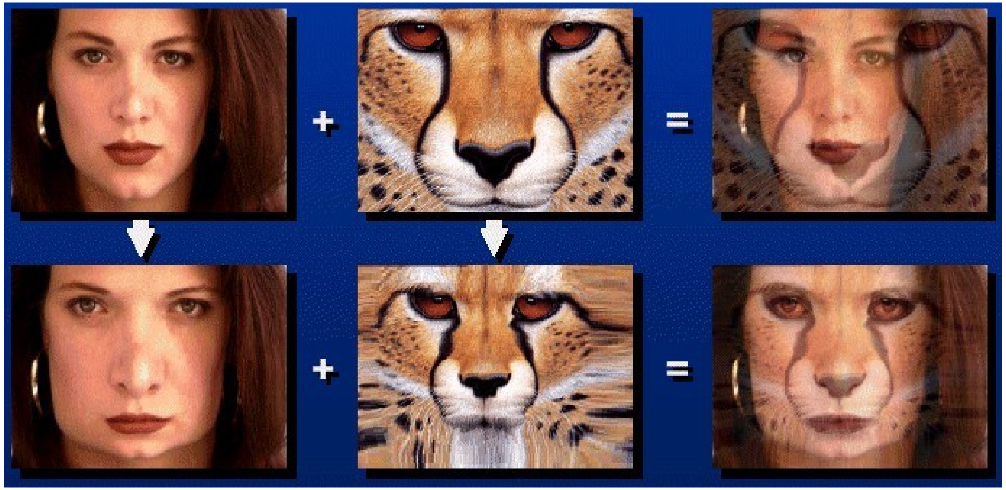
Control points mark matching pixels in both images

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

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

$$C_t = \alpha_t dA + (1-\alpha_t) dB$$

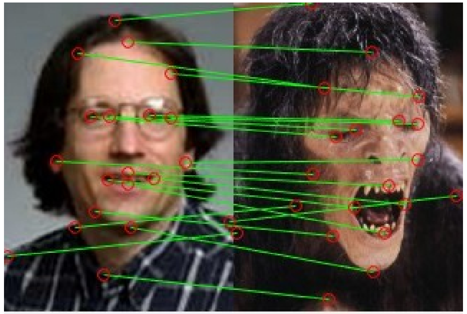


Naive

Correct

Morphing example

Input images



Control points

Warped images

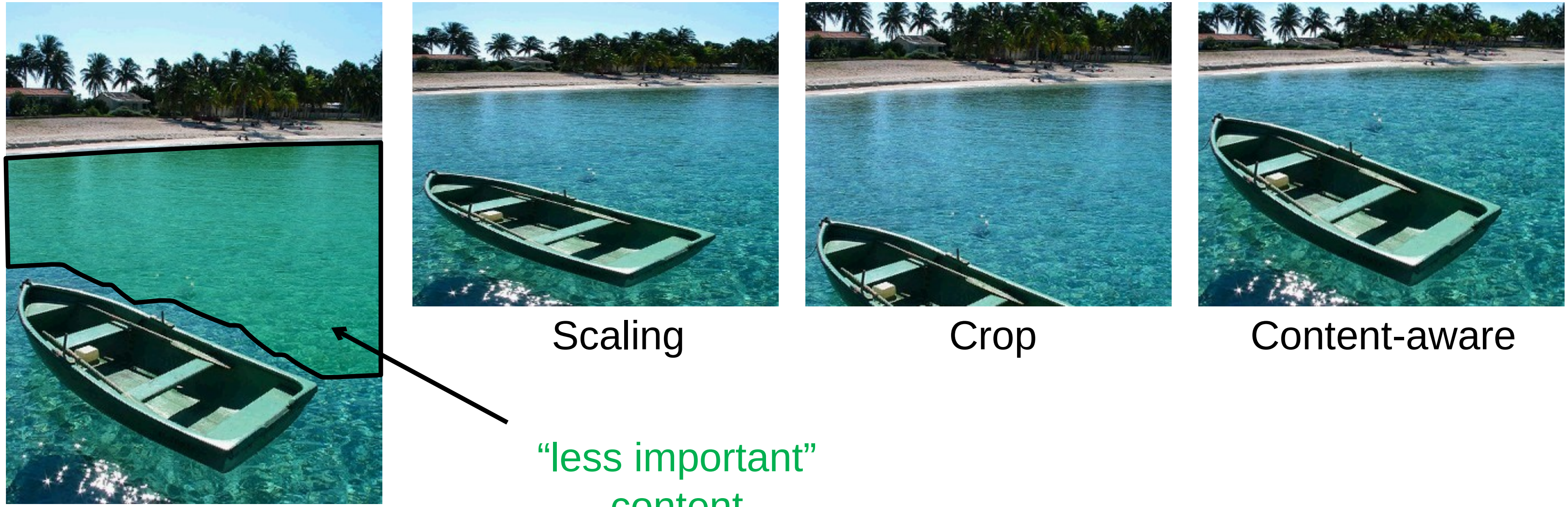
Blended image

One more morphing example

Content-aware resizing

- Change size, aspect
- Automatically preserve important structures

Important vs. unimportant content



Scaling

Crop

Content-aware

“less important” content

Input

Content-aware resizing

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

What is important?

- What do people consider important?

Judd et al. ICCV09 *Learning to predict where people look*

- Fast approximation - edges

How to remove?

slide credit: Michael Rubinstein

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

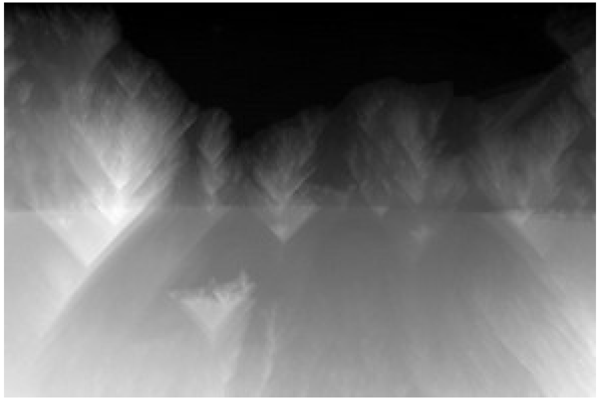
- Optimal connected path of pixels from the top to the bottom that minimizes energy
  - Dynamic programming
  - Cummulative cost
  - Backtracking

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



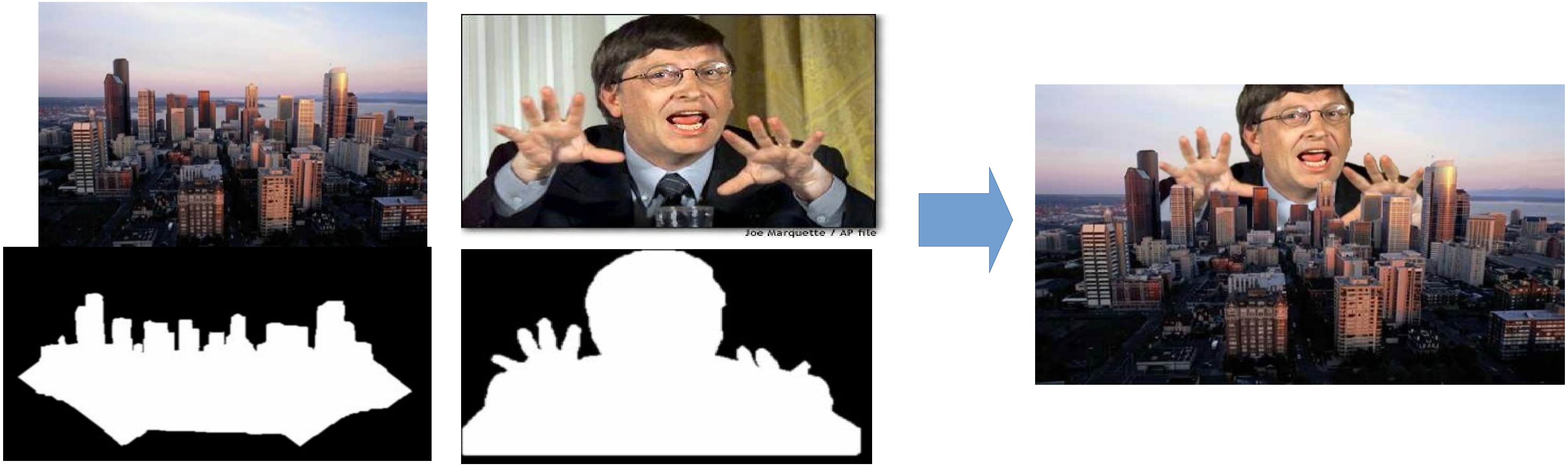
Examples – reducing width

Examples – reducing height

Examples – scaling down

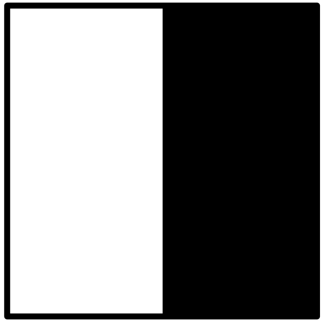
Image merging

Combine images by taking pixels from appropriate images

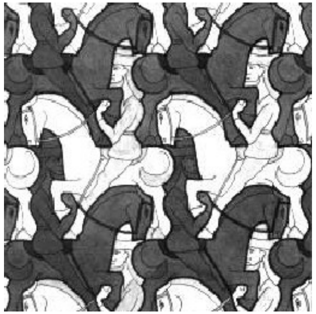


Naive combination – binary mask

A



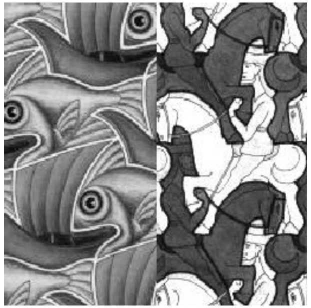
R



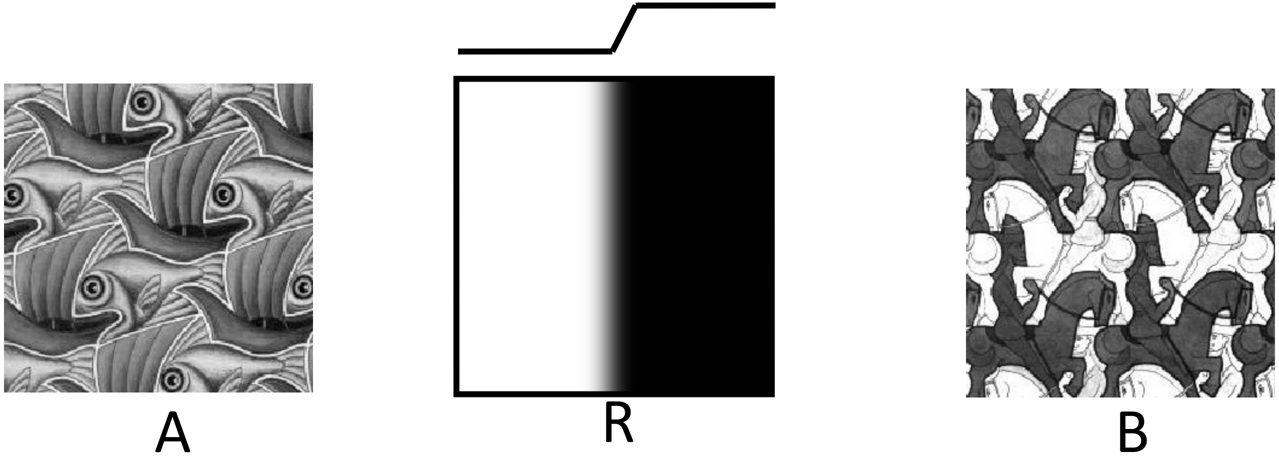
B

Combine images A and B using alpha channel R

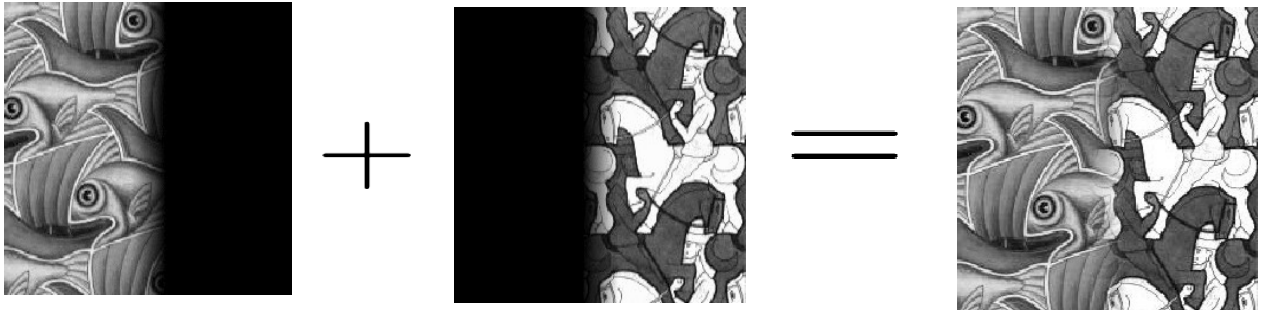
$$RA + (1 - R)B =$$



Smooth alpha channel

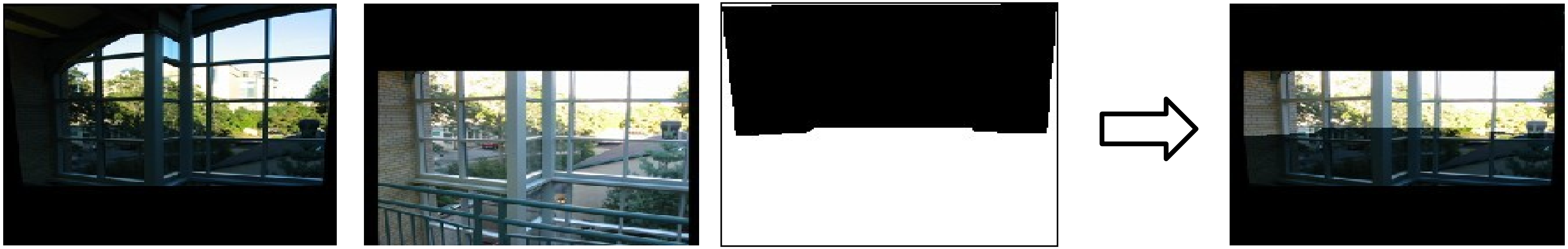


Blending the transition border

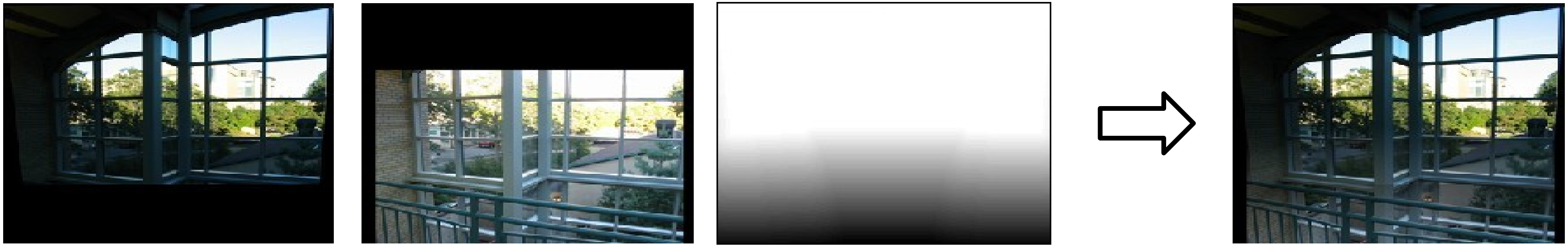


Example

- Sharp transition – unreal image

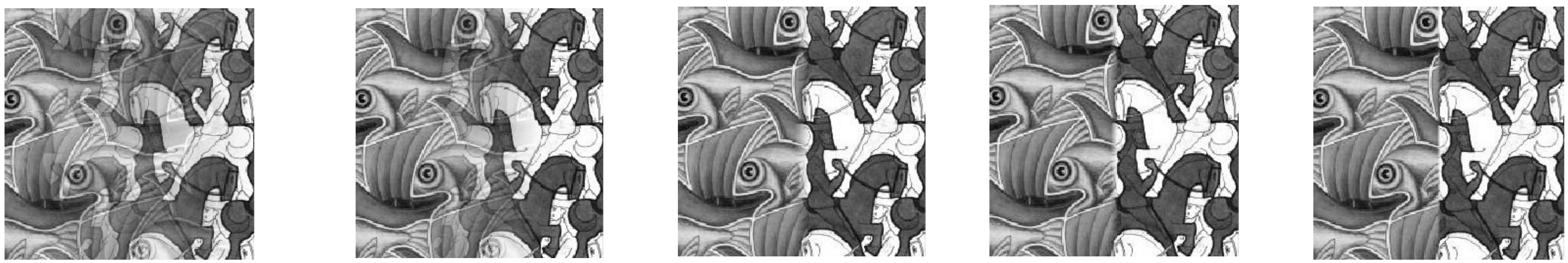


- Smooth transition – more realistic



Smoothing influence

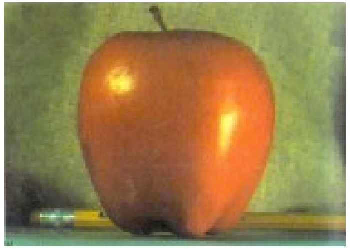
Very smooth transition - ghosting

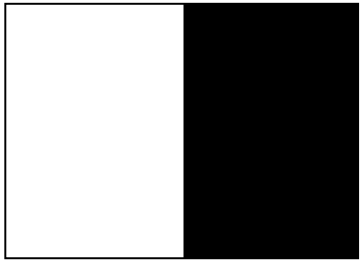
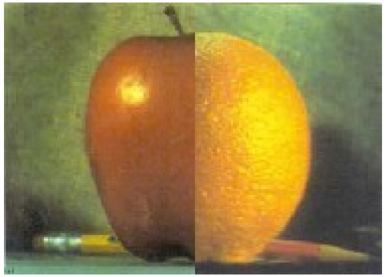


Sharp transition - cutoff

Frequency-aware blending

- Simple alpha mask blending



- More natural effect if we blend images by frequency bands

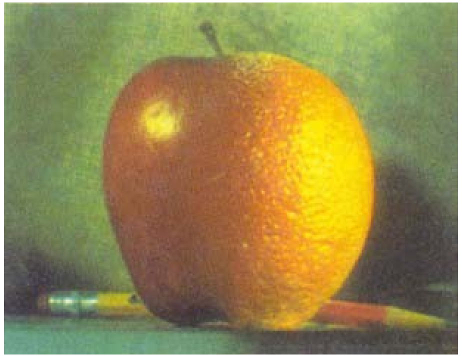
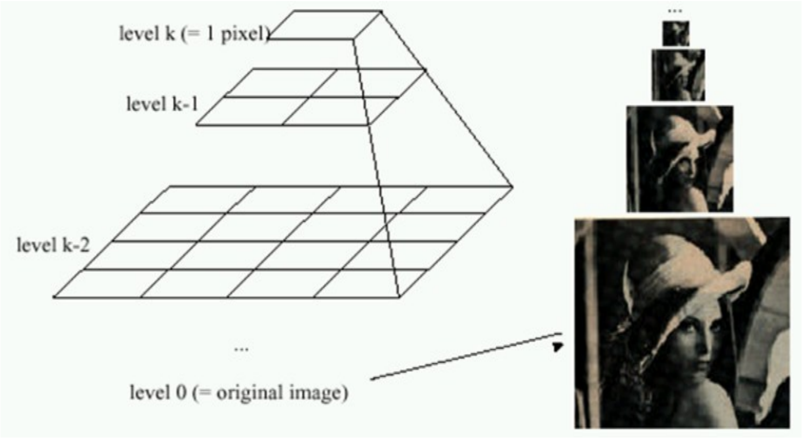


Image pyramids

- Multi-scale signal representation
  - Sequence of images
  - Each image only includes lower frequencies
- Gaussian pyramid
  - Smooth with Gaussian filter
  - Reduce resolution by half
  - Repeat

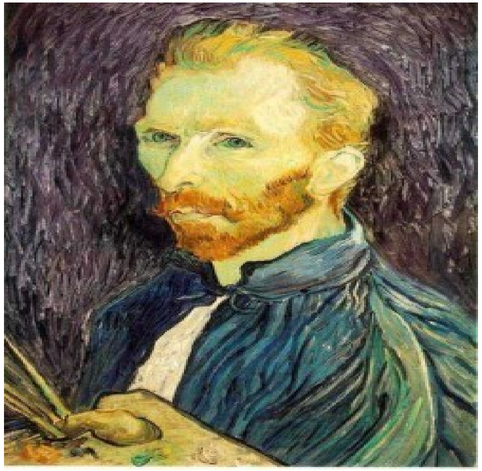


Gaussian pyramid

- Kernel size fixed
- Discard every second pixel
- Each layer removes frequency band

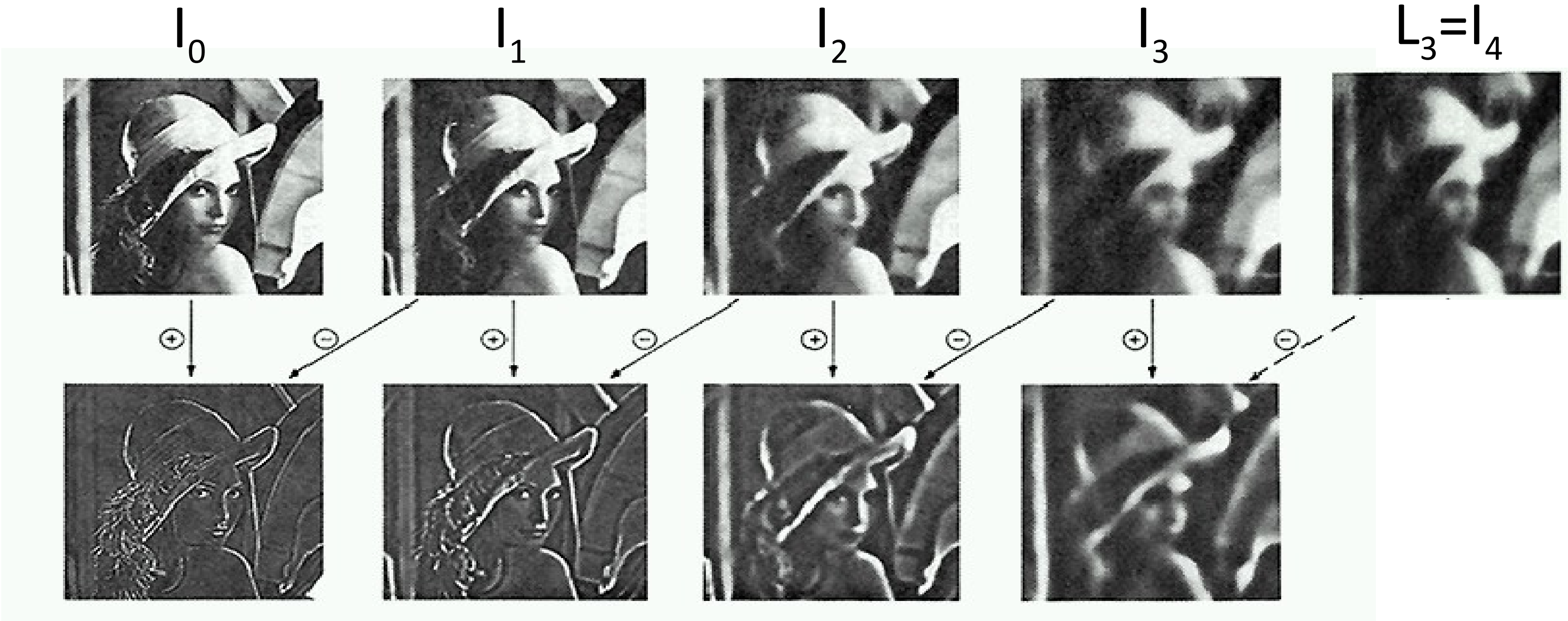
G 1/8

G 1/4



G 1/2

Laplacian pyramid

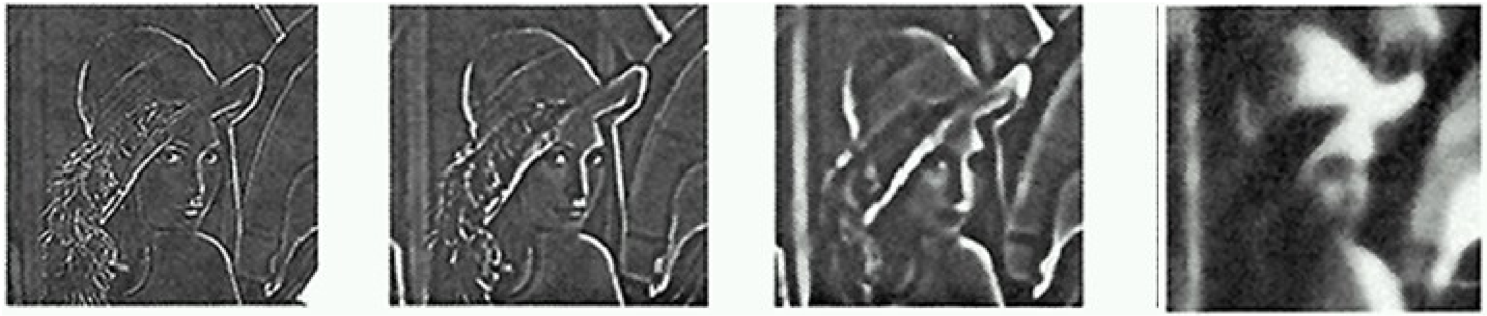


High frequencies

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

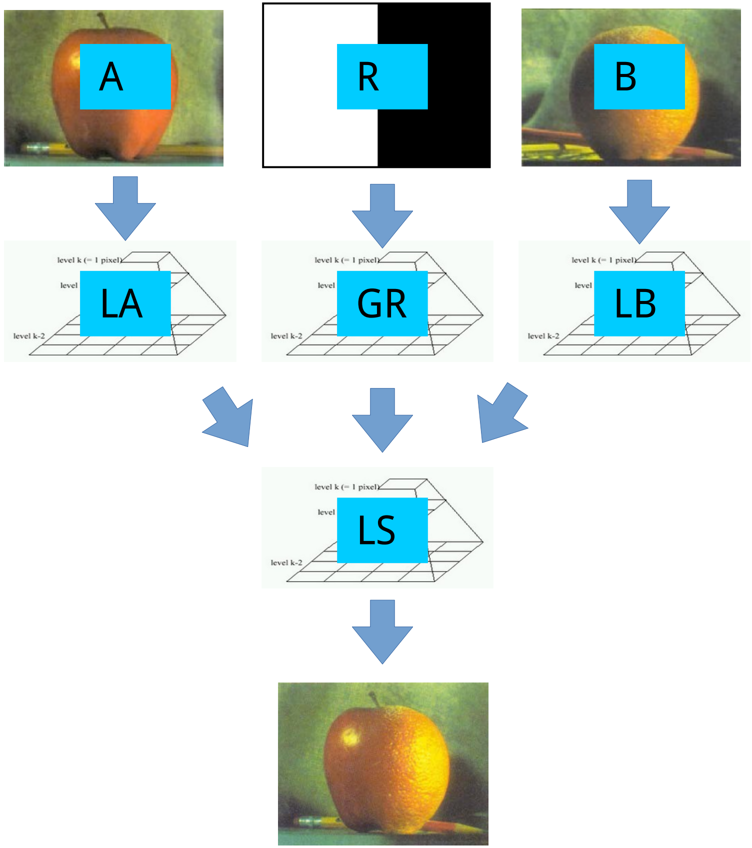
$$L_0 + L_1 + L_2 + L_3 = (I_0 - I_1) + (I_1 - I_2) + (I_2 - I_3) + I_3 = I_0$$

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

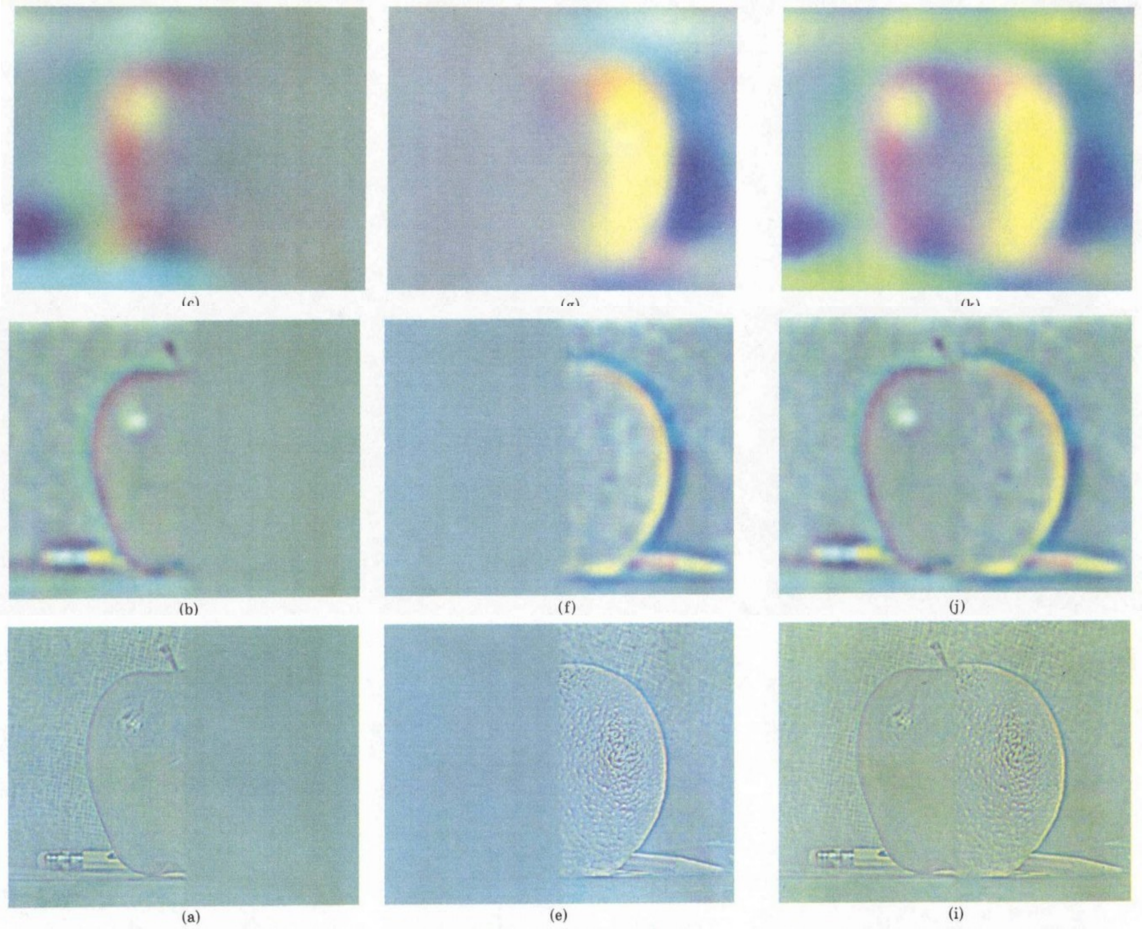
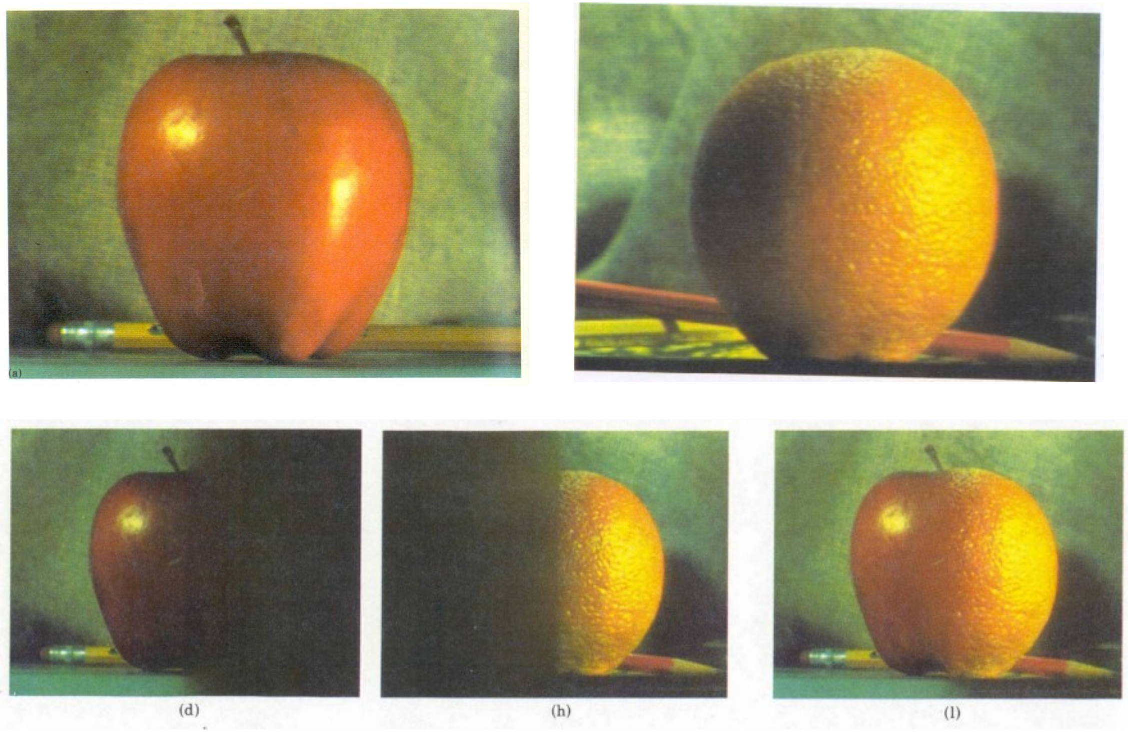


image A

image B

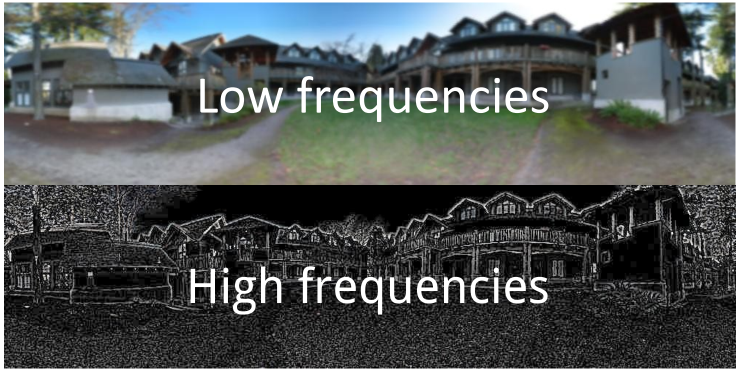
combined layer

Merging examples



Merging examples - Autostitch

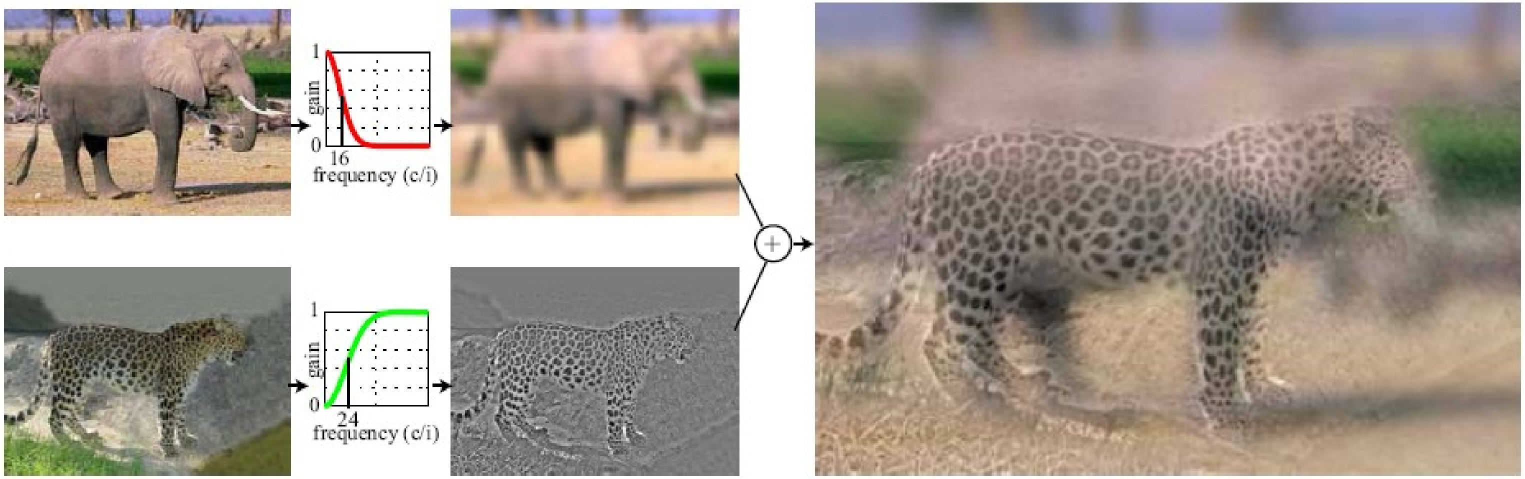
- Stitching panorama from multiple images
- Two-layer blending – high and low frequencies
- Only blend low frequencies, keep high frequencies intact



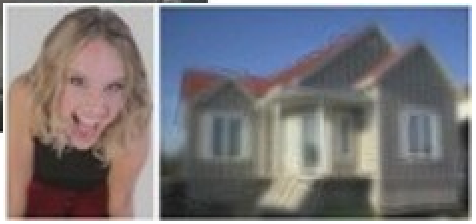
Matthew Brown and David G. Lowe, "Automatic panoramic image stitching using invariant features," International Journal of Computer Vision, 74, 1 (2007), pp. 59-73

Hybrid images

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



Hybrid images - examples



A. Oliva, A. Torralba, P.G. Schyns, “Hybrid Images,” SIGGRAPH 2006

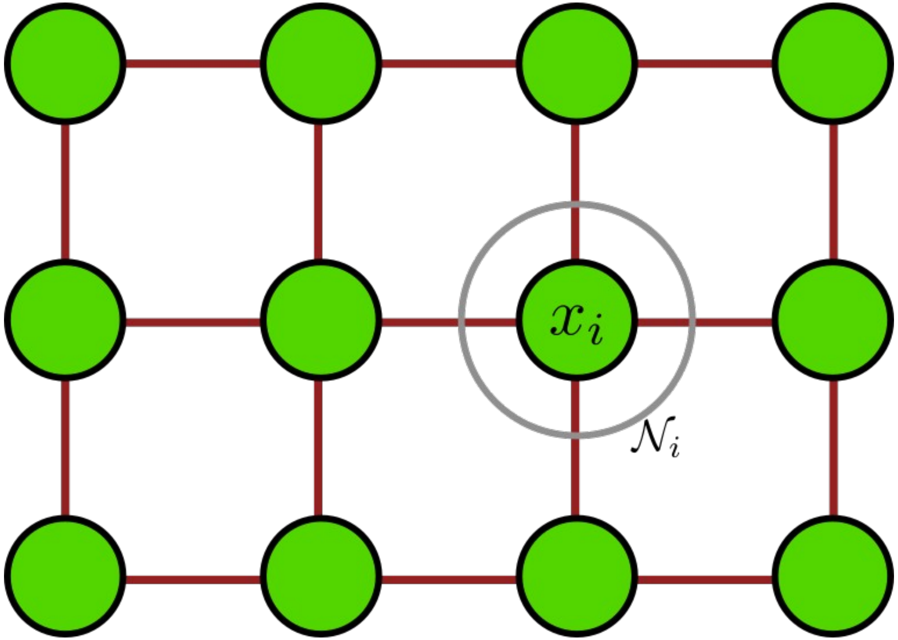
Interactive segmentation

- 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 in color are more likely in the same cluster
  - Two pixels that are near are more likely in the same cluster
- Formalized using Markov random field
- Solve MRF problem using Graph Cut

Markov random field



$$\mathrm{E(x)} = \sum_{i\in\mathcal{V}} \psi_i(x_i) + \sum_{i\in\mathcal{V}, j\in\mathcal{N}_i} \psi_{i,j}(y_i, y_j)$$

data term **smoothness term**

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathcal{L}} E(\mathbf{x})$$

Maximize

$$\psi_i(x_i) = \begin{cases} -\log(1 - p(F|x_i)) & x_i = 0 \\ -\log(p(F|x_i)) & x_i = 1 \end{cases}$$

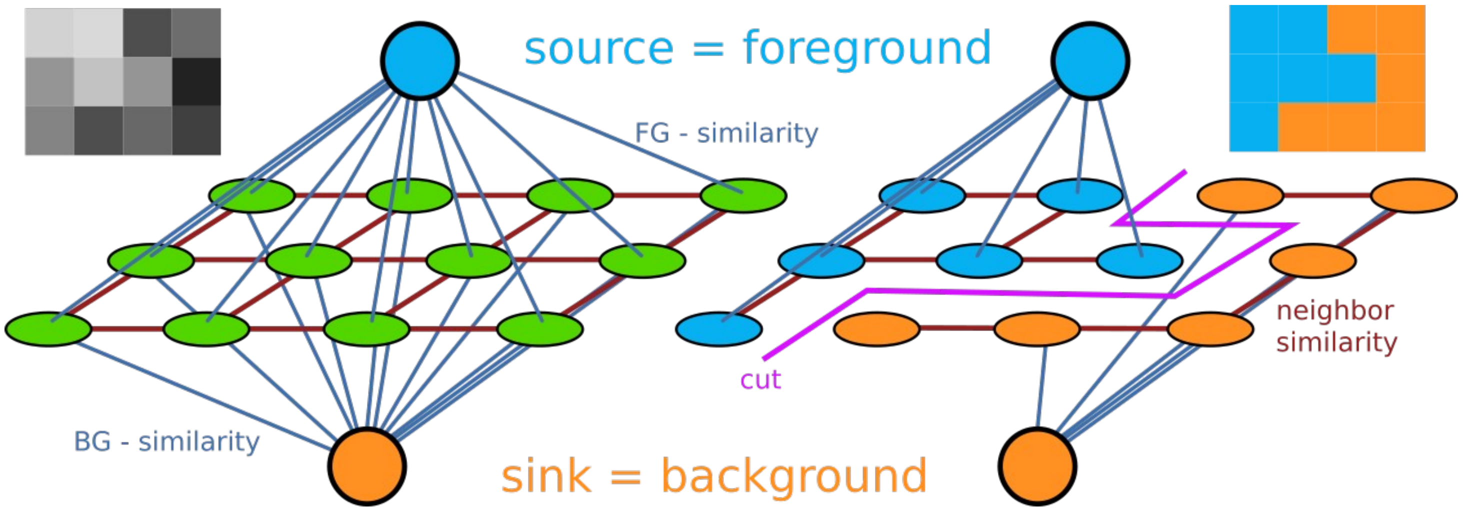
Penalize assignment of label if the other label is more likely.

$$\psi_{i,j}(x_i, x_j) = \begin{cases} \lambda_1 + \lambda_2 \exp(-\beta(I_i - I_j)^2), & x_i \neq x_j \\ 0, & x_i = x_j \end{cases}$$

Penalize assignment of different labels if pixels are visually similar.

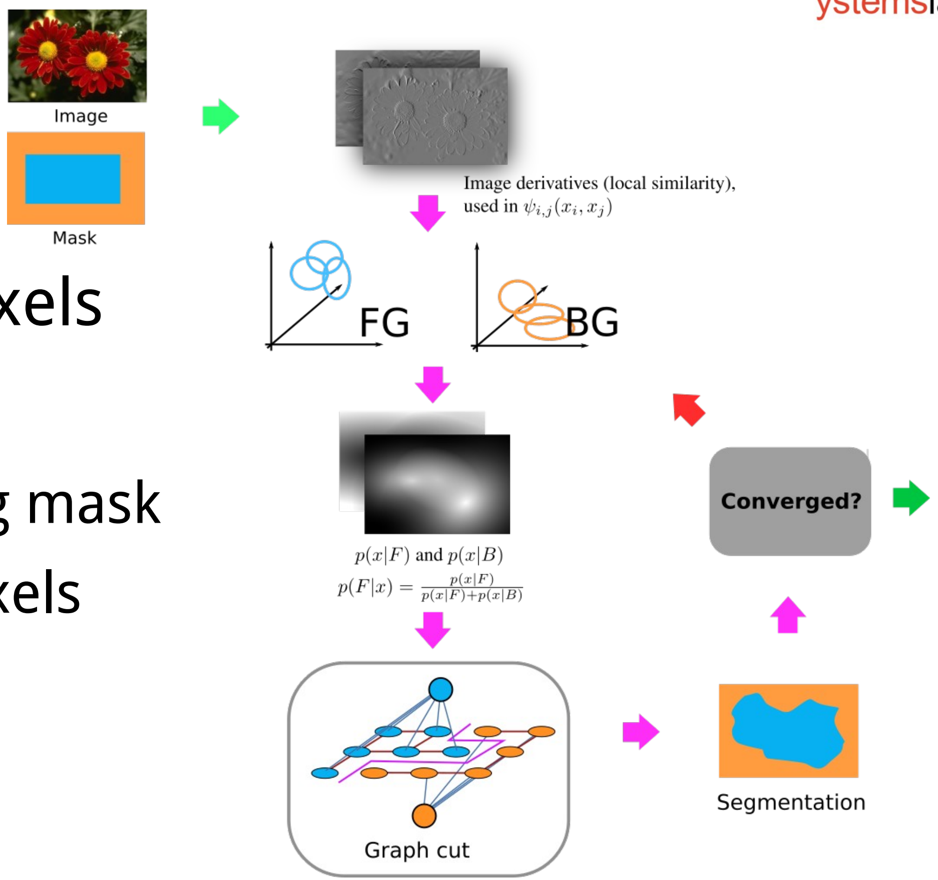
Max-flow min-cut theorem

- **Maximum flow through a network** is the **sum of flow through edges** that, if removed, would disconnect source from the sink
- Ford-Fulkerson algorithm



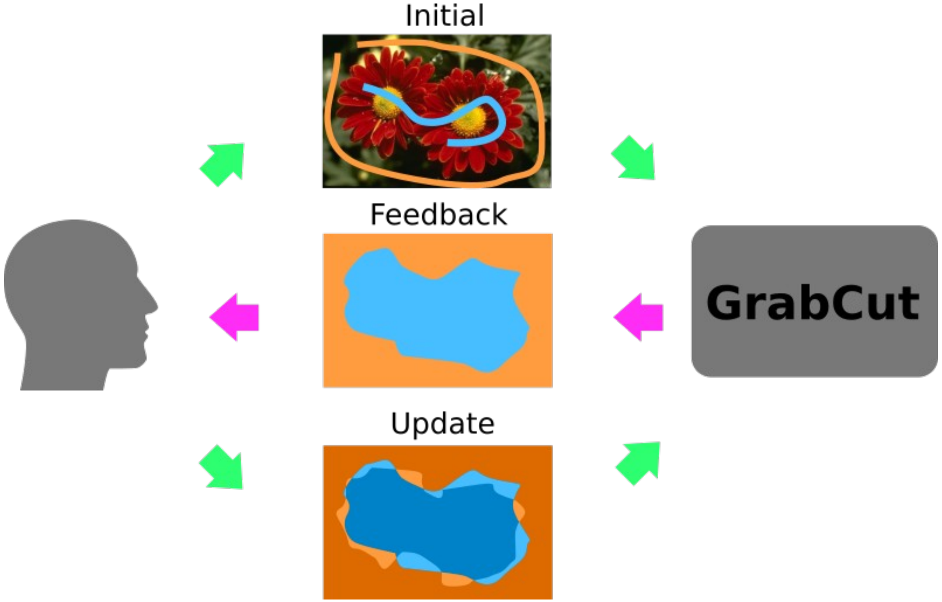
GrabCut algorithm

- 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

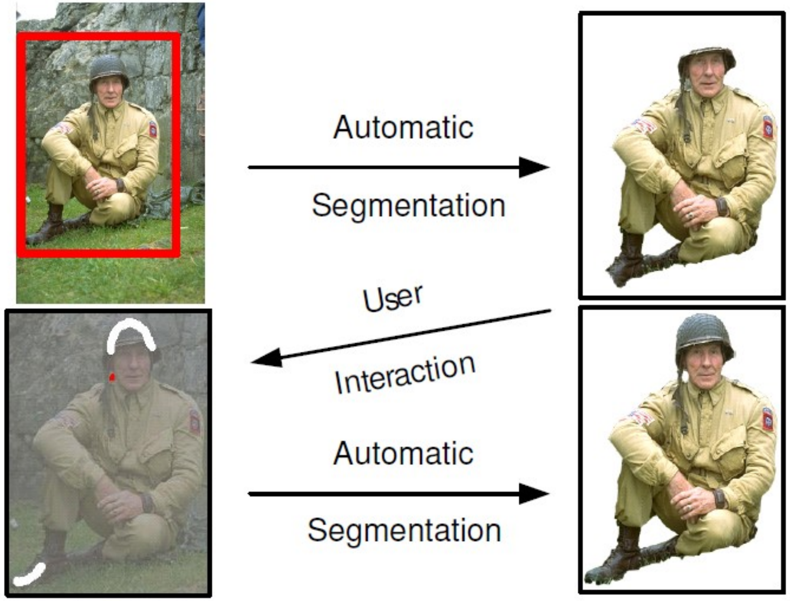


Interactive segmentation

- 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



Examples

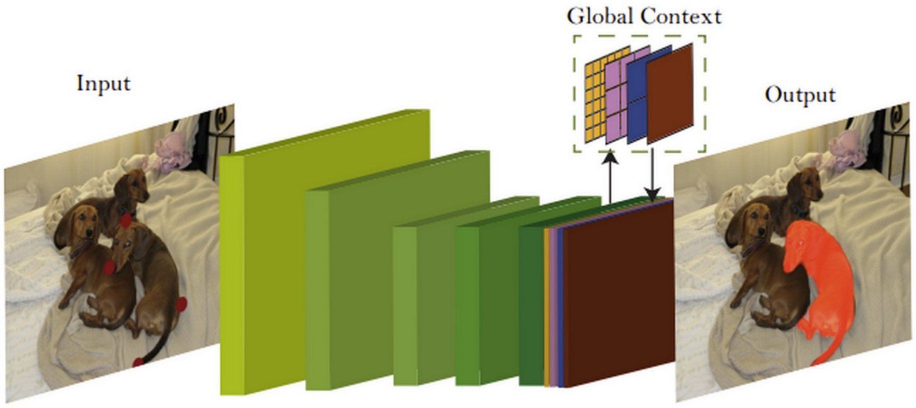

Automatic
Segmentation
User
Interaction
Automatic
Segmentation

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