Univerza v Ljubljani
Fakulteta za elektrotehniko

Computer Vision 06 – Image processing and analysis 1a

doc. dr. Janez Perš
(with contributions by prof. Stanislav Kovačič)

Laboratory for Machine Intelligence
Faculty of Electrical Engineering
University of Ljubljana

Quick recap of the previous lectures

- Geometric aspects of image formation
  - WHERE does the image appear on the sensor/film/screen
- Photometric aspects of image formation
  - HOW bright is the image that appears on the sensor
- Color (last week)
  - Human perception of color
  - Sensing of color in digital cameras
  - Reproducing color & color matching functions
    - Because neither people nor cameras capture full spectrum!
    - That's the reason why we need color matching
  - Color spaces
    - There are other color spaces besides RGB

Outline

- Algorithms for image processing and analysis
  - I assume you have been waiting for this!
- Terminology & overview
  - Image processing
  - Image enhancement
  - Image restoration
  - Geometric transformations
  - Image analysis
- Point operations
- Histogramming
- local operations, filtering

Image processing

Image processing operation

h(x,y), k(x,y), w(x,y), g(x,y)
h(i,j), k(i,j), w(i,j), g(i,j)

Input Image
$f_I(x,y)$, $f_I(i,j)$
I(i,j), …

Output Image
$f_O(x,y)$, $f_O(i,j)$
J(i,j),…

Various notations are in use ….

- The purpose of image processing is:
  - to produce new image that is in *some aspect* better than the original image
  - And thus *more appropriate* for further processing.
  - Sometimes people call this “Image preprocessing”
    - Because the input is the image, and the output is image as well
    - Therefore it comes before the *core algorithms* of computer vision

Image processing

- Typical goals:
  - to reduce noise, improve S/N ratio
    - Blurring, filtering
  - to improve brightness/contrast, to compensate variations in brightness in contrast
  - to save space/bandwidth, i.e. image compression, before transmitting the image to the processing unit
    - Lossy (JPEG)
    - Lossless (GIF, PNG)
  - to normalize the image, geometrically or photometrically
    - Remember Task 2 from your Lab Assignment 2
    - What happened to the chessboard?

Image (pre) processing

Input Image → Preprocessing operation → Output Image

- Two basic approaches:
  - Image *enhancement*: enhancement model
  - Image *restoration*: degradation -> restoration model
- In CV, IP is one of the first steps within processing pipeline.
  - Therefore, we often call it preprocessing.
- In machine vision/industrial applications
  - We try to avoid enhancement/degradation!
  - The images should be captured at best quality possible!

Image enhancement

- We have input image that needs *improvement*
  - Therefore, we suitably process the input image to obtain *improved* output image.
  - We perform enhancement most often when the images are meant to be observed by eyes (not for further processing)
- Examples
  - Pixel value manipulation, linear, nonlinear to improve visibility of details, contrast, color, ...
  - image filtering, in spatial or frequency domain, linear or non-linear to improve S/N ratio, morphological filtering, ...

Image enhancement – example

- Manipulating the brightness and contrast
  - Task: to make low contrast details visible

Before | After

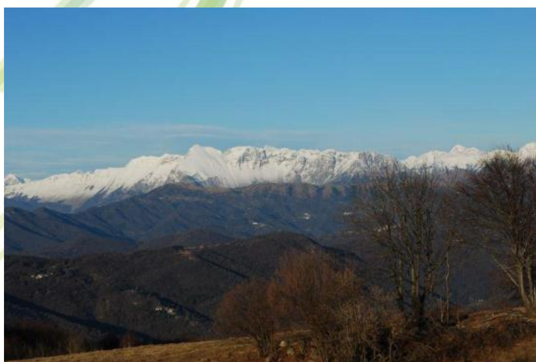
  - Why do we do it mainly for human observation?
    - Good CV algorithms don't care about absolute pixel values
    - Contrast of 5 pixels is as good as contrast of 50 pixels

Image restoration

- We have *degraded* image
  - We possibly know (or can estimate) the actual process of degradation
- We want to infer the *original image before degradation*, based on
  - our image and degradation model
- Image restoration approaches:
  - Unsharp masking to improve contrast
  - (Blind) deconvolution
  - Wiener filtering
- Sometimes restoration is treated as a special case of enhancement.

Geometric transformations

Image filtering, etc., work on pixel values (image range)



f(x,y)

$$g(x,y) = H(f(x,y))$$

$$f(x,y) \rightarrow \boxed{H} \rightarrow g(x,y)$$

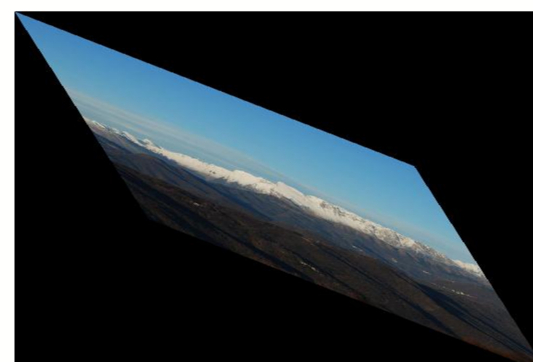
g(x,y)

Geometric image transformations work on pixel positions (image domain)

f(x,y)

$$g(u,v) = f(T(x,y))$$

$$f(x,y) \rightarrow \boxed{T} \rightarrow g(u,v)$$



g(u,v)

Image analysis

Input Image
f(x,y), f(i,j), I(i,j) → Image analysis → Description of an image

- Task of image analysis
  - to extract *meaningful information* from images
- Approaches:
  - Segmentation: divide image into meaningful regions
  - Object detection, localization
  - Feature detection & extraction
  - edge, corner detection
  - Shape, texture, appearance analysis
- Goal:
  - Qualitative and/or quantitative description of image

Wrap-up on IP and IA

- Various categorizations of image processing & image analysis exist
  - Sometimes image enhancement and restoration are put into the same category.
  - There is not clear distinction between IP and IA
- In computer vision, IP is used as one of the first steps – if it is used at all
- IA in computer vision one of the most important components, and sometimes sufficient for solving a problem at hand.

Questions?

Simple preprocessing operations

Image preprocessing

- Point operations
  - Pixel -> pixel
  - e.g. arithmetic and logic operations, +, -, *, /, AND, OR,...
- Local operations
  - Local area -> pixel
  - e. g. linear / nonlinear filtering to suppress noise
- Global operations
  - Whole image -> result
  - Result depends on the whole image
  - e.g. Image histogram

Point operations

- Value of the output pixel $I_{out}(i,j)$ depends solely on the value of the input pixel $I_{in}(i,j)$

  $I_{out}(i,j) = T(I_{in}(i,j))$

  $I_{out}(i,j)$ – output image

  $I_{in}(i,j)$ – input image

  T – Pixel transformation

- Example

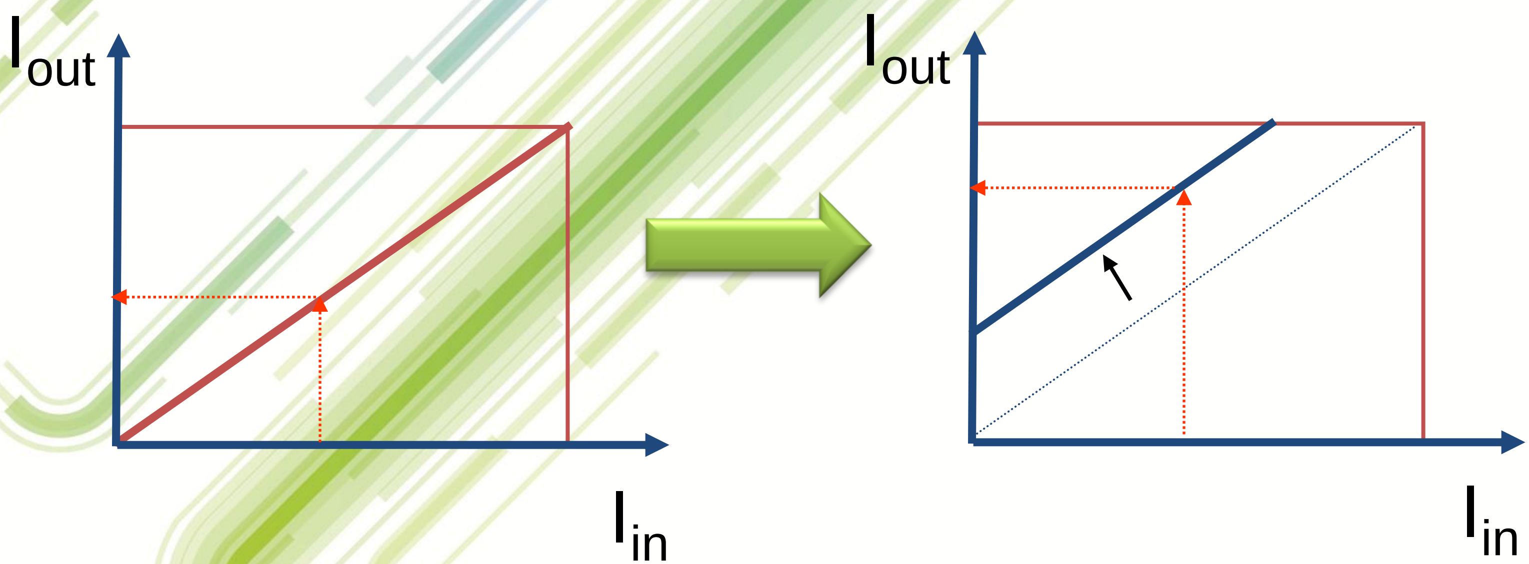
  $I_{out}(i,j) = a*I_{in}(i,j)+b$

  a – contrast adjustment, b – intensity adjustment

  – In theory, *a* and *b* could be different for each pixel, $a(i,j)$, $b(i,j)$

Brightness

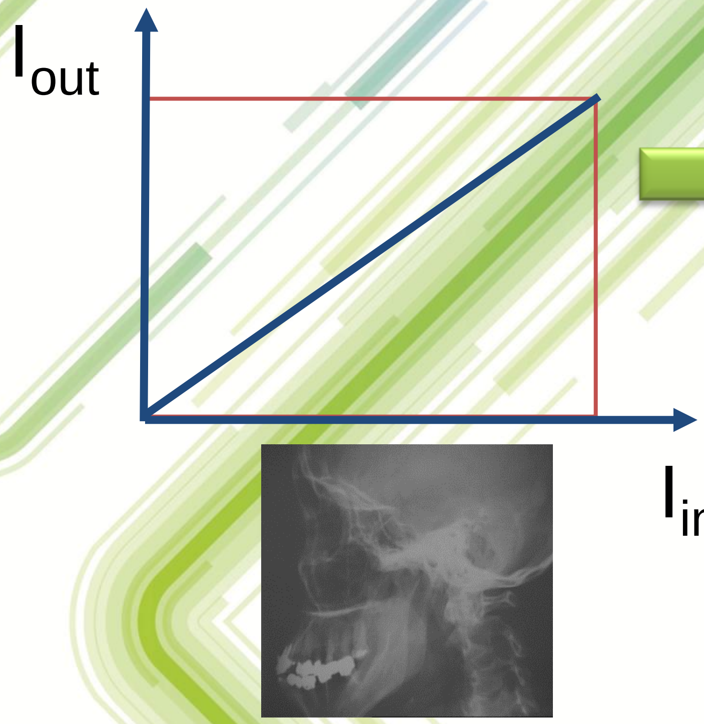
- Brightness change: $I_{out}(i,j)=I_{in}(i,j)+k$

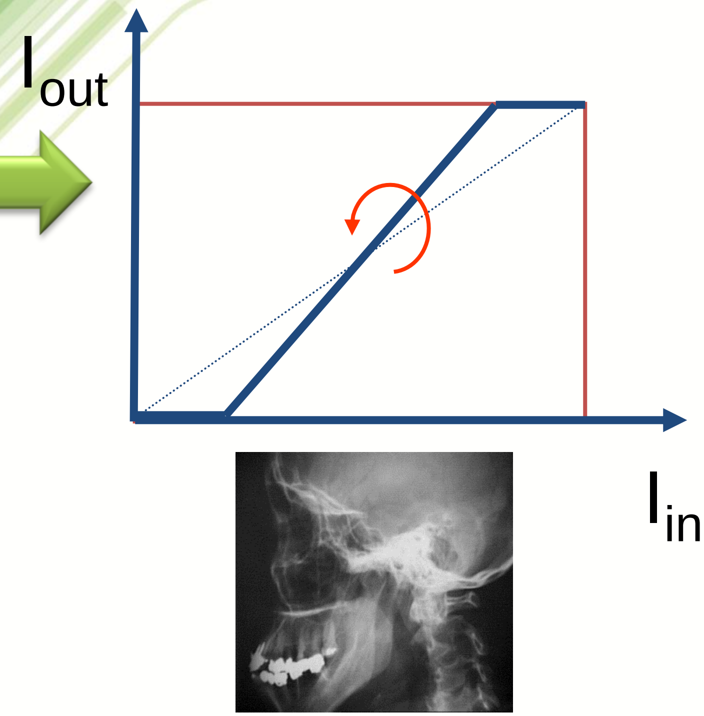


- $I_{out}(i,j)$ – output brightness (grayness) levels
- $I_{in}(i,j)$ – input brightness (grayness) levels

Contrast

- (linear) contrast change: $I_{out}(i,j)=k*I_{in}(i,j)$



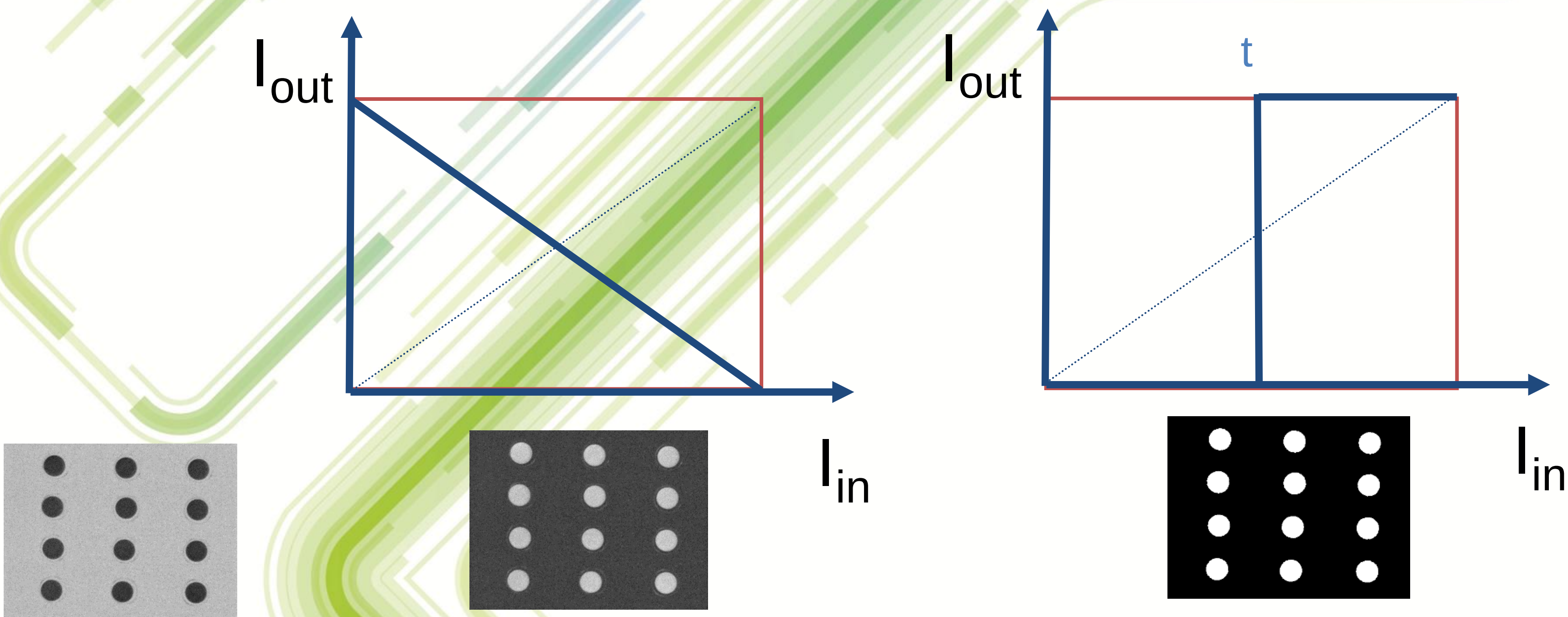


Remember: as with brightness, no information gain!

Inversion and thresholding

Inversion: $I_{out}(i,j)=255-I_{in}(i,j)$
(8 bits per pixel)

Thresholding: $I_{out}(i,j) = (I_{in}(i,j)>t)$
(t=threshold)

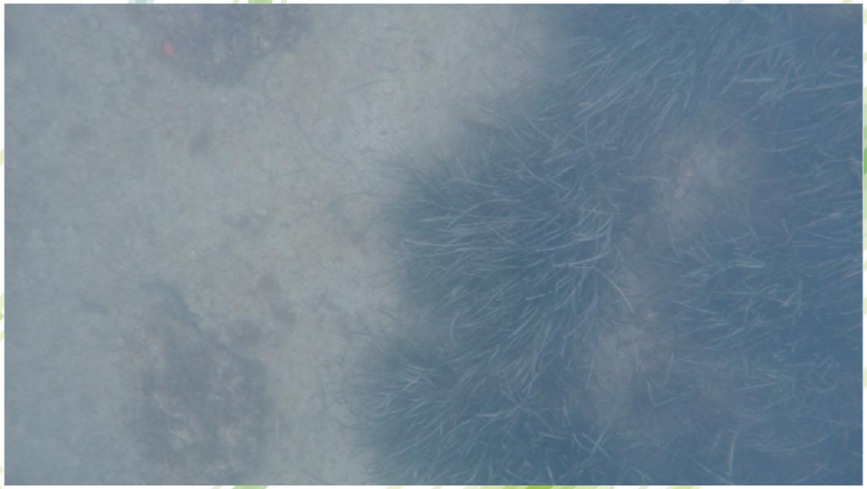


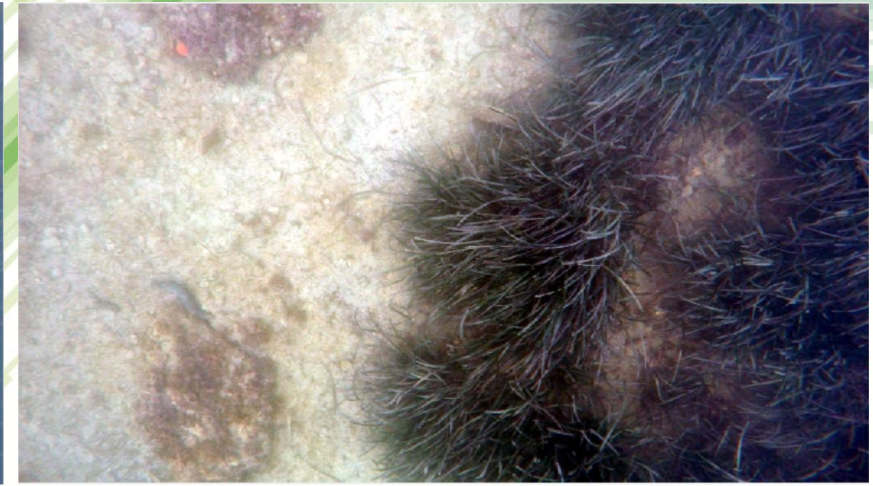
General linear transformation

$$I_{out} = (I_{in} - c)\frac{b-a}{d-c} + a$$

- I_{out} – output brightness (grayness) level
- I_{in} – input brightness (grayness) level
- d – max input value
- c – min input value
- b – max output value
- a - min output value

Point operations example





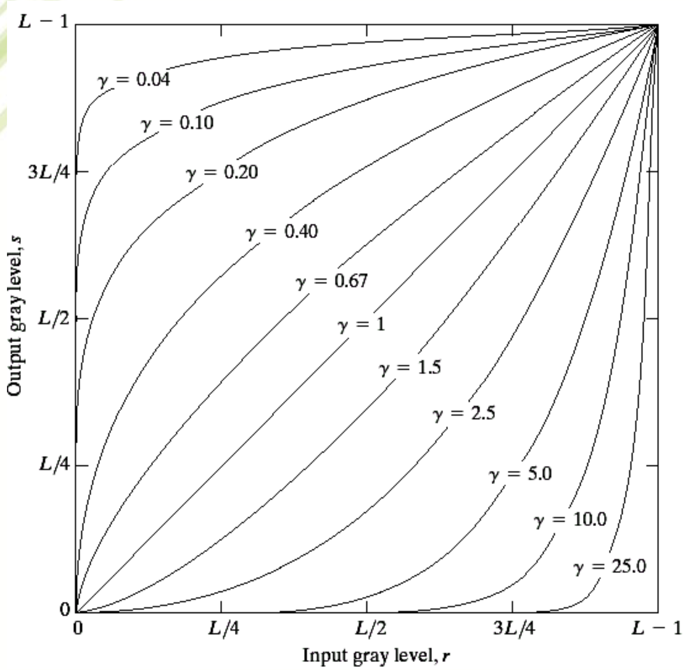
$$I_{out} = (I_{in} - c)\frac{b-a}{d-c} + a$$

- Matlab function: imadjust()
  - Note: imadjust works for grayscale and color images.
  - But you can apply imadjust on each RGB channel individually

Gamma correction

- Originates from nonlinearity of CRT displays.
  - the relationship between screen luminance and the applied electric signal is exponential.
  - The lightness/brightness sensitivity of human eye is not linear as well.

Gama compression



Gama expansion

$$I_{out}(i,j) = c \cdot I_{in}^{\gamma}(i,j)$$

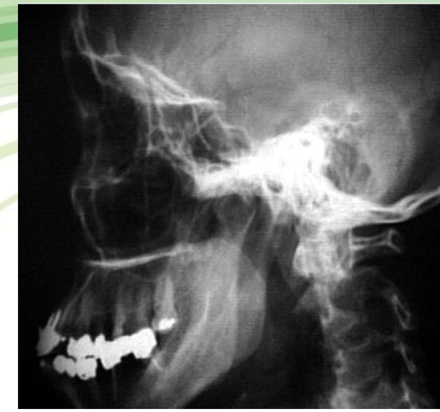
Gamma correction

$$I_{out}(i, j) = c \cdot I_{in}^{\gamma}(i, j)$$

- Transforms a narrow dark/bright band
  - to a wider band (thus whitens) or
  - the opposite (darkens) the image.
- Parameter is γ
  - The constant c is chosen such that functions start and end in bottom left/upper right corners.
  - Conventional value for gamma is 2.2 (for CRTs), or 1/2.2 (for camera devices)
  - Sometimes, gamma correction is used (abused)
    - to nonlinearly adjust brightness levels
    - even when there is no CRT or known source of nonlinearity

Gamma adjustment example

$\gamma = 1{,}5$



$\gamma = 0{,}5$

Matlab function:

```
imadjust( Image,  stretchlim(Image), [0;1], Gamma);
```

LUT for point operations

- LUT = look-up table

Input Image → **Point operation with Look-Up-Table (LUT)** → Output Image

Possible, when pixel values are quantized (e.g. with 8 bit grayscale image, we need a LUT with 256 entries)

Significant speedup for operations.

0	255
1	254
2	253
	252
	….
255	0

LUT entries are calculated in advance
This example shows Inversion operation (negative)

Image arithmetics

- Image addition:
  - mostly to improve S/N ratio
  - Example: Σ Sequence of dark images of a stationary object

- Image subtraction:
  - Change detection (object detection & tracking)
  - Background subtraction (non-uniform lighting)
  - to evaluate how much and where two images differ.

Image addition

- Example
  - We have *N* images of a stationary scene
  - Images are affected by noise
  - Noise is uncorrelated, zero mean Gaussian noise
- Calculate new image I as a mean of all images:

$$I = \frac{I_1 + I_2 + ... + I_N}{N}$$

- The noise variance decreases by the factor N:
  - And the noise standard deviation by the factor of: $\sqrt{N}$

$$\sigma_I = \sigma / \sqrt{N}$$

Image arithmetics - examples

- Image summation

- Image subtraction:

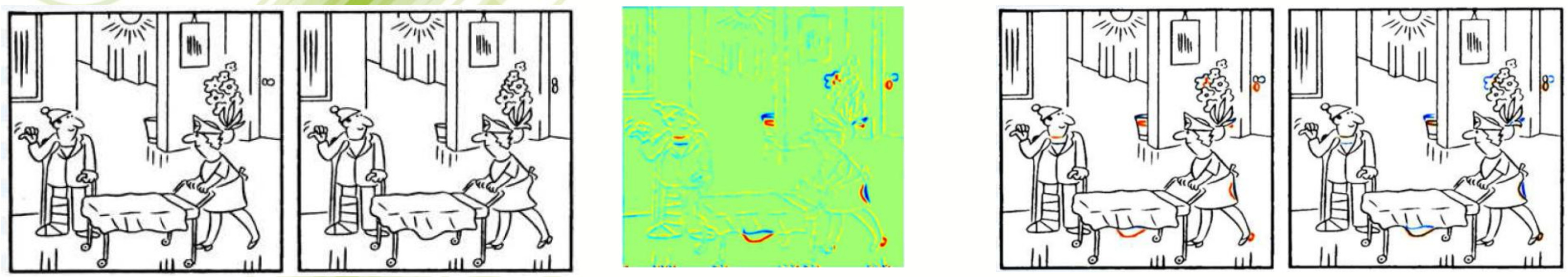


Image arithmetics in Matlab

- One can manipulate images as matrices
  - E.g. C=A+B - Preferred way for labs, requires no toolboxes
- But there are specialized functions as well:

• imabsdiff	Absolute difference of two images
• imadd	Add two images or add constant to image
• imapplymatrix	Linear combination of color channels
• imcomplement	Complement image
• imdivide	Divide one image with another or divide image by constant
• imlincomb	Linear combination of images
• immultiply	Multiply two images or multiply image by constant
• imsubtract	Subtract one image from another or subtract constant from image

Questions?

Histogramming

Histogramming

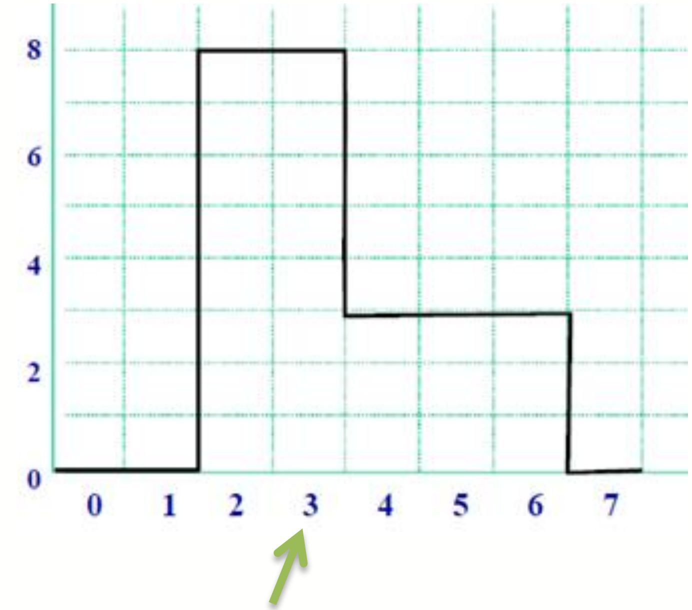
- We treat the pixel value as a realization of random variable, defined by its
  - range [0..255], $I=i_1,i_2, \ldots\ i_k \ldots\ i_N$
  - and probability distribution $P(I)=P(i=i_k)=(p_1,p_2, \ldots\ p_k \ldots\ p_N)$, $\sum_{k=1}^{N} p_k = 1$
- We scan the image
  - and count the number of occurrences of each pixel value, for all pixel values
- Finally
  - We plot a number of occurrences of each pixel value as a function of pixel value. This is *histogram* $H(i_k)$, $k=1,2 \ldots N$
  - If normalized, $\sum_{k=1}^{N} H_k = 1$ the histogram is *approximation* (derived empirically) of a probability distribution

Histogram example

- Let us assume a small grayvalue image with pixel values within [0..7]
- We compute a grayvalue histogram, and this is what we get.

Drawing conclusions:

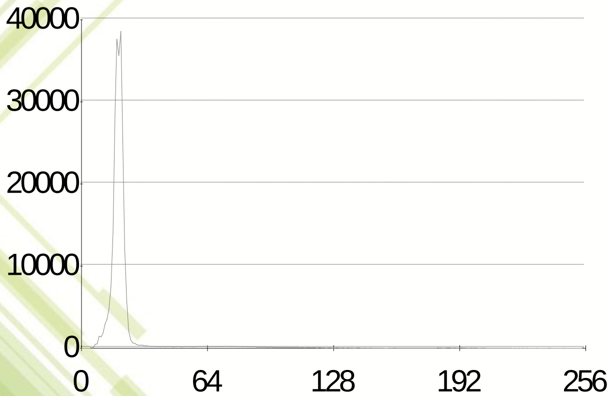
- There are no pixels with values 0,1, and 7.
- There are 8 pixels of value 2, or 3.
- There are 25 pixels in the image (5x5 perhaps)
- Histogram is not normalized.
- How to normalize?
- Compute mean, E(I) =
- Compute variance, $E[I - E(I)^2]$=

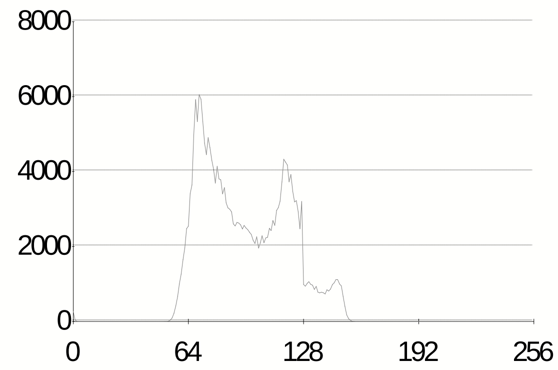


Histogram cell or “bin”

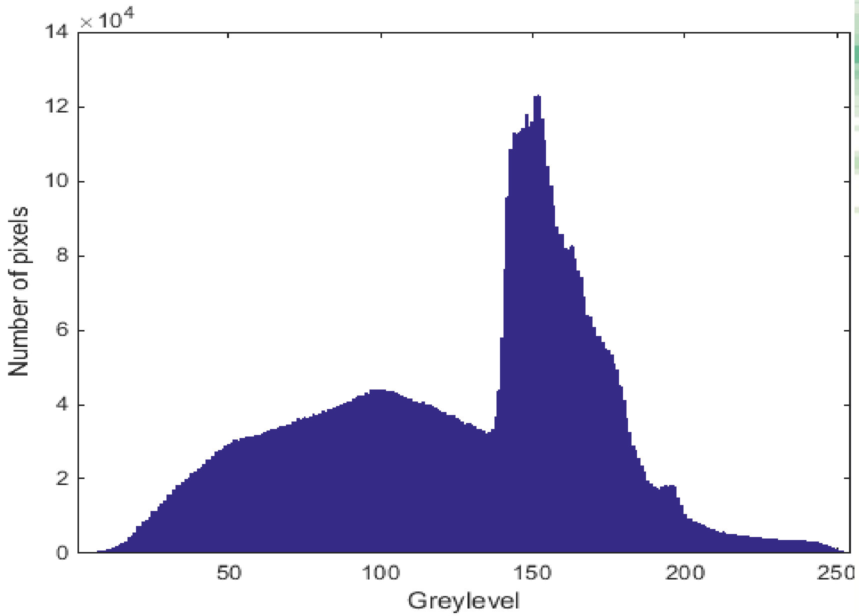
Histogramming

- There are many things that we can say about an image, based just on a grey-value histogram.
  - dynamic range, either well covered, or not,
  - image brightness, too dark, too bright, appropriate, perfect,
  - image contrast, low, high,
  - one, or more grey values is/are dominating the image.
- What can we say about images from these histograms?





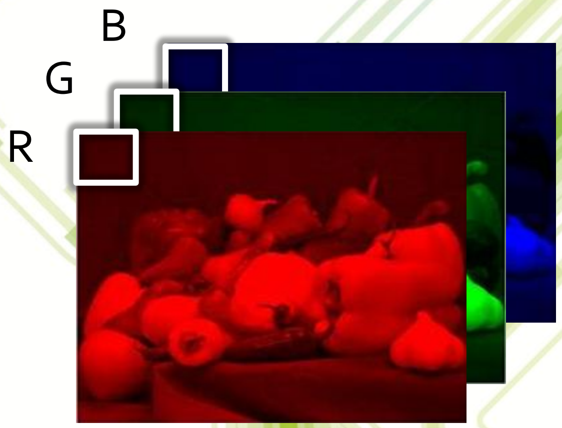
Histogram example



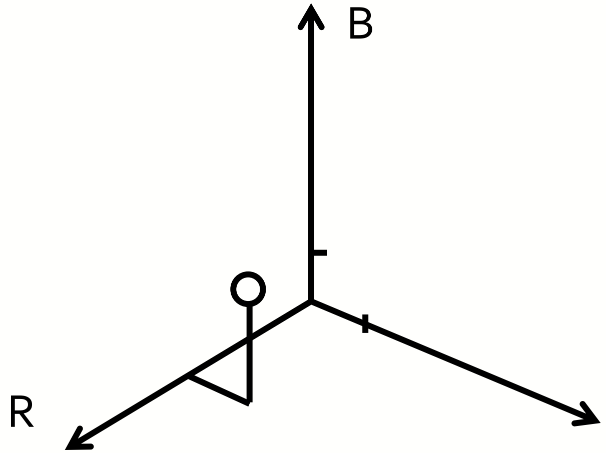
- Histogram of grayscale image, not normalized
- Gray values between 140 and 170 dominate (perhaps sky region?)
- Note: Histograms are best displayed as bar graphs
- Matlab function: *imhist()* or *hist()*

Color histogram

- Grayvalue histogram is 1D
- Color histogram is 3D

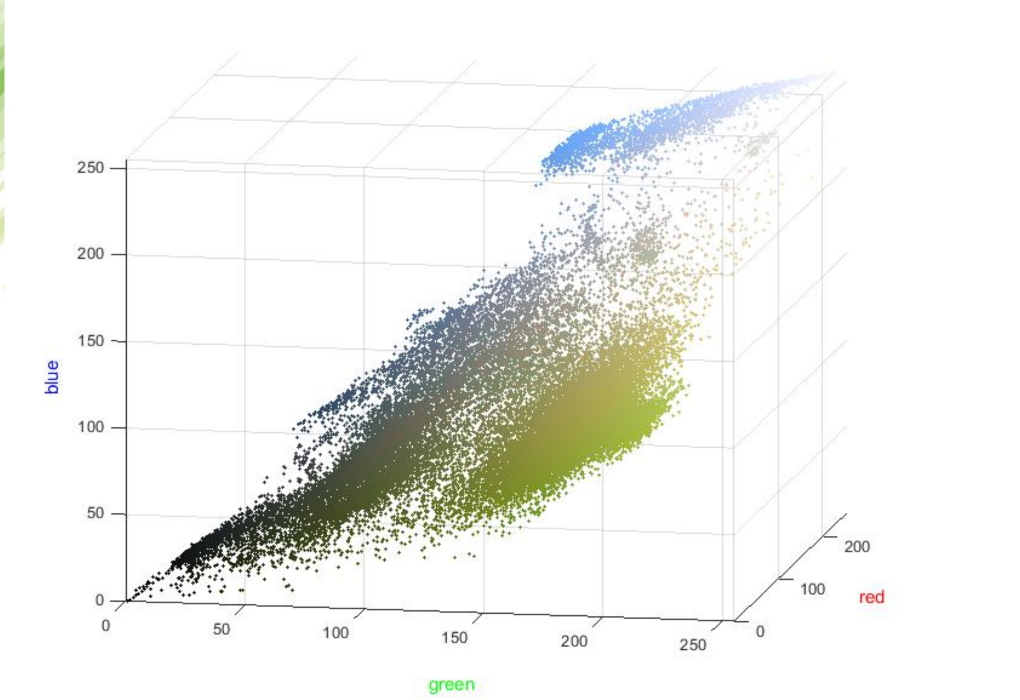


i=[20,10,10]



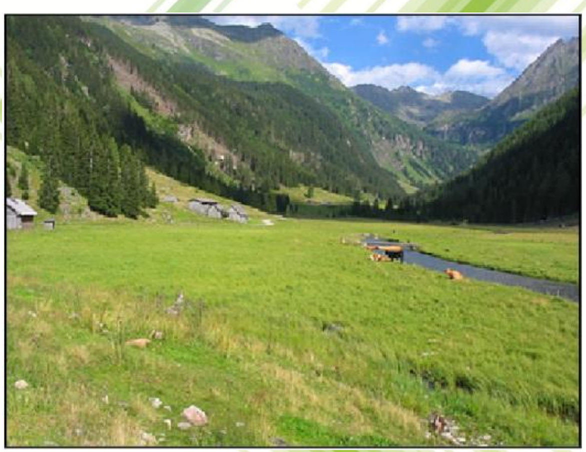
3D Color histogram

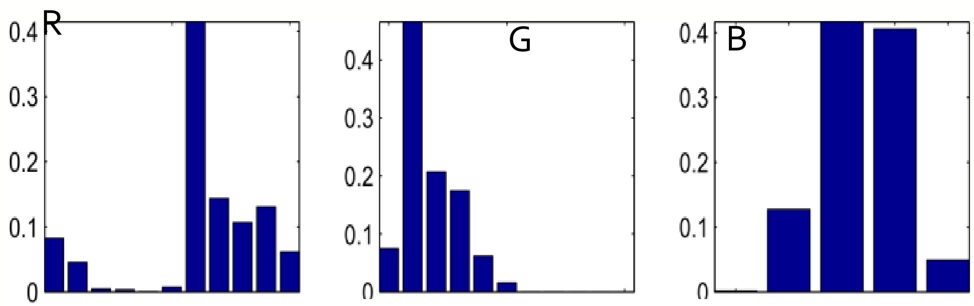
- Example of 3D color histogram



3x 1D color histograms

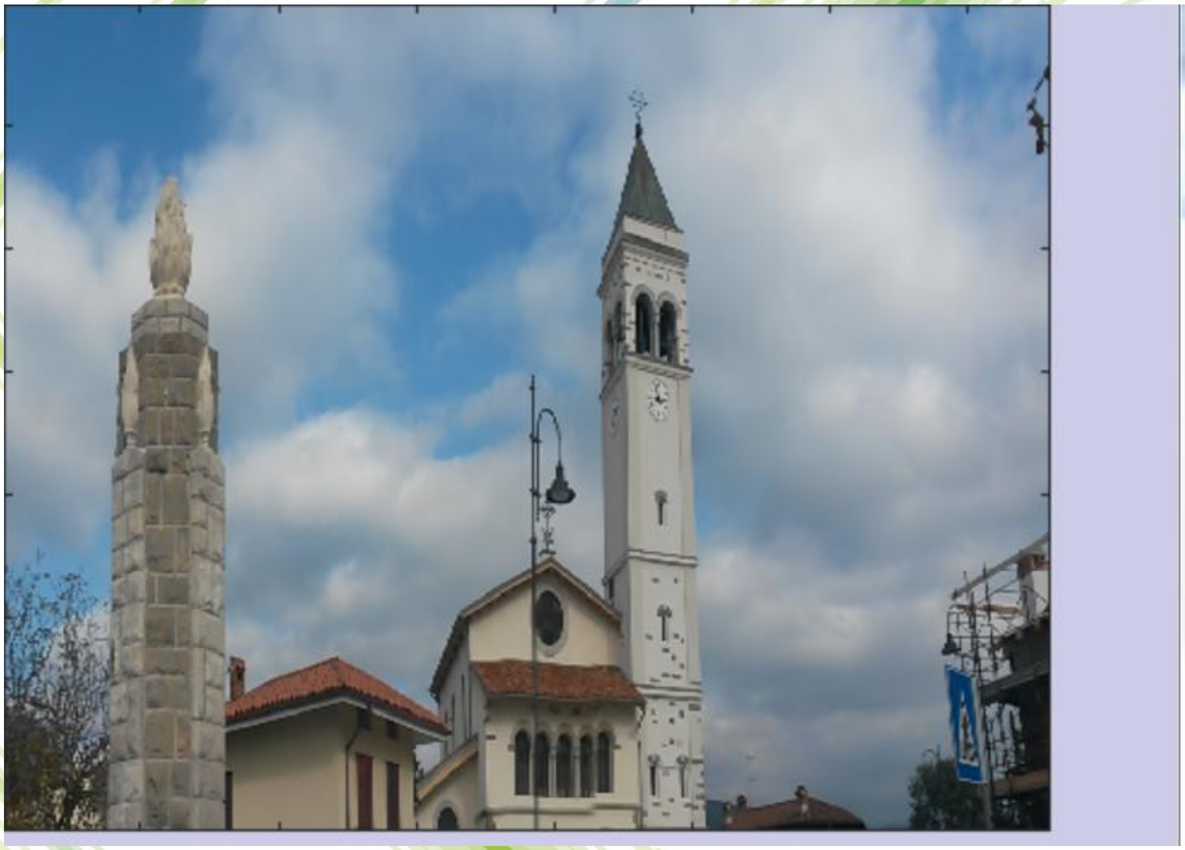
- Alternatively, we can represent color image with 3 x 1D color histograms
- 1 histogram per R,G,B channel
- This is NOT the same as 3D color histogram

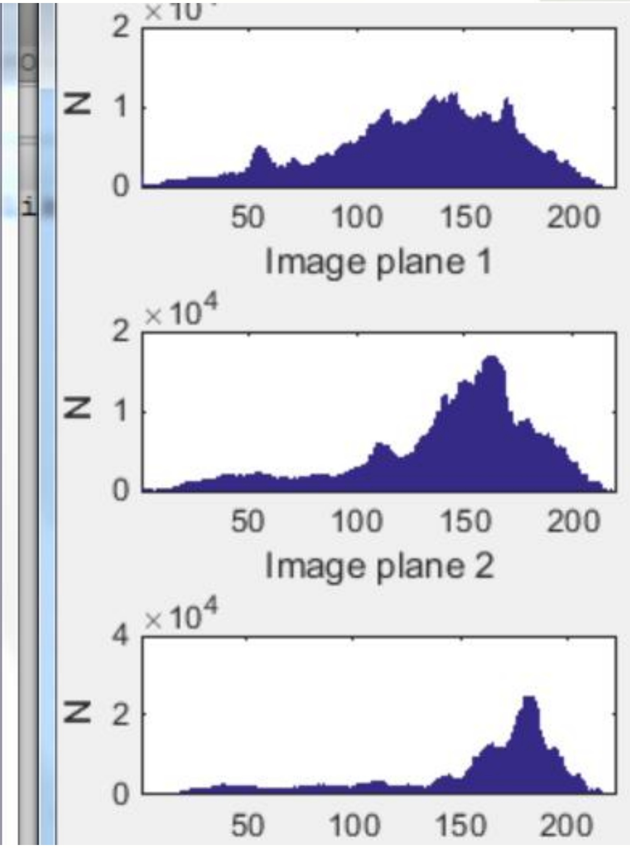




3x 1D color histograms example

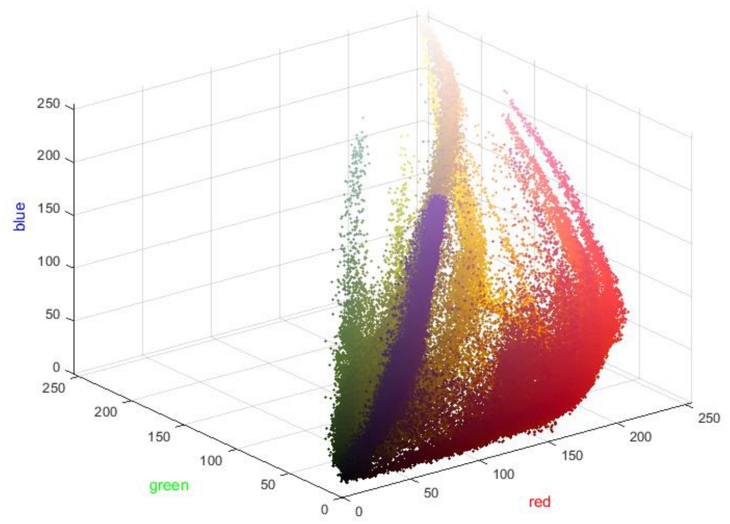
- 3 histograms (R,G,B) of a color RGB image





3D color histogram example

- 3D histogram of a color RGB image



3D color histogram and binning

- True color image, 8-bits per color channel.
- 3D color histogram: 256 x 256 x 256 array!
  - Pretty large array, about 16 M x (no. of bytes for one value)
  - An 1920 x 1080 color image: about 2 megapixels.
  - This means, 3D histograms are large *and sparse.*
  - Therefore, we usually apply ‚binning' of pixel values into bins.
  - For example, bins of 16 values wide result in 8 x 8 x 8 = 512 3D color histogram.
  - Binning is not specific to images, one can use it whenever the histogram would be too sparse!

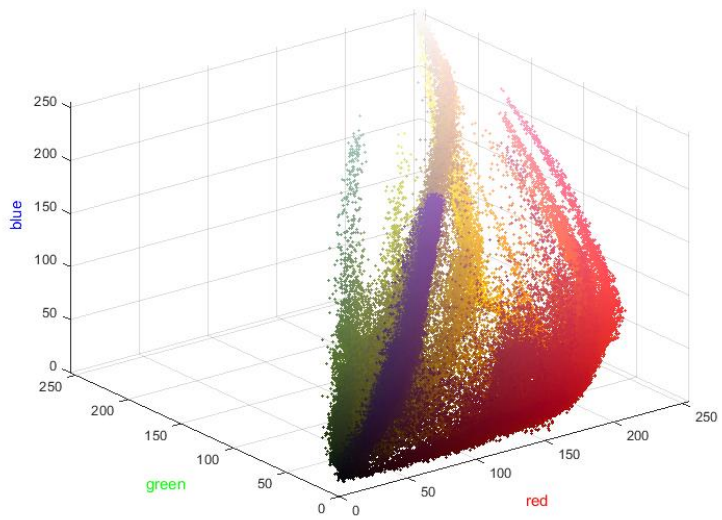
0..15 : bin 0
16..31 : bin 1
32..47: bin 2
46..63: bin 3
e.t.c.

Histogram uses

- Brightness / contrast correction
- Thresholding (*binarization)* to produce two valued (logical) image)
- Histogram equalization
  - for an arbitrary image with grey-value distribution
  - Make distribution of the new image uniform.
  - The idea: more frequently used values should get larger range.
- Histogram specification:
  - Starting with an arbitrary image, produce a new image with a specified histogram.

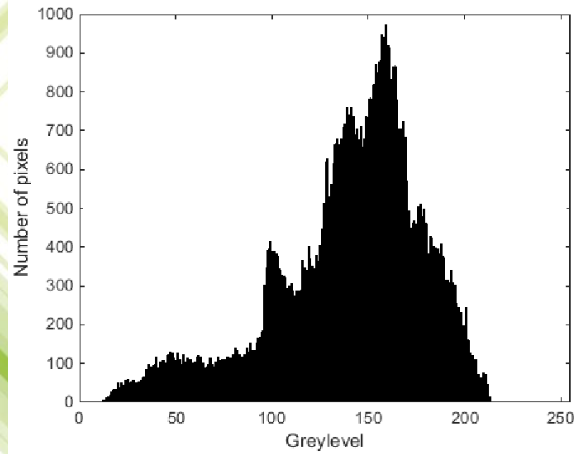
Histogram uses

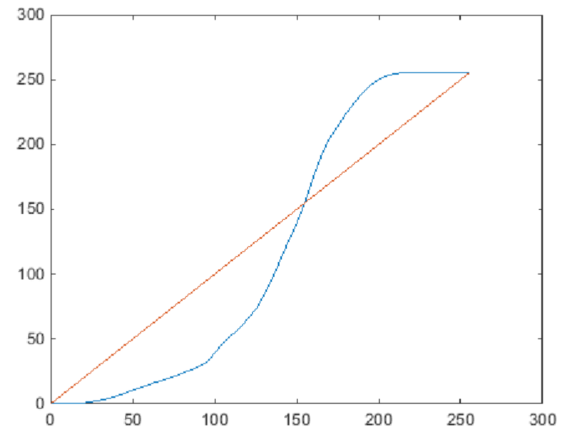
- Histogram can be used as image descriptor!
  - Either 3D histogram with proper binning
  - Or descriptor, formed by concatenating 3 x 1D histograms!
  - What does such descriptor describe?



Histogram equalization

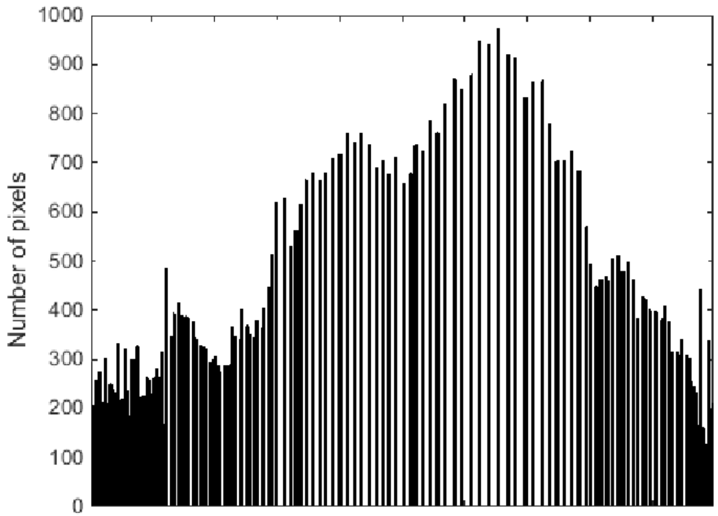
- Finding a transformation of image, that would yield equalized histogram (all bins the same)
  1. Compute histogram with 256 cells (bins) from I, I--> h.
  2. Compute cumulative histogram h_c
  3. Normalize cumulative histogram, h_nc = h_c / max(h_c)
  4. Multiply h_nc with maximum (output) value, e.g. 255, -> h_mnc
  5. Use h_mnc as LUT, R(i,j) = LUT(I(i,j))

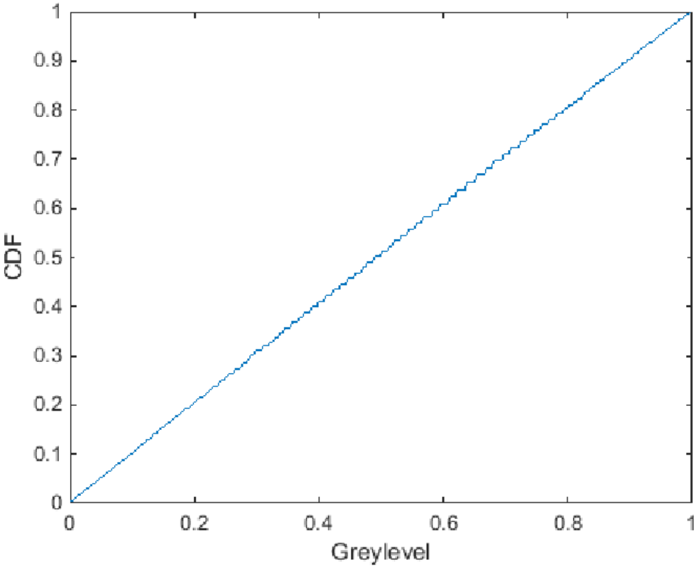




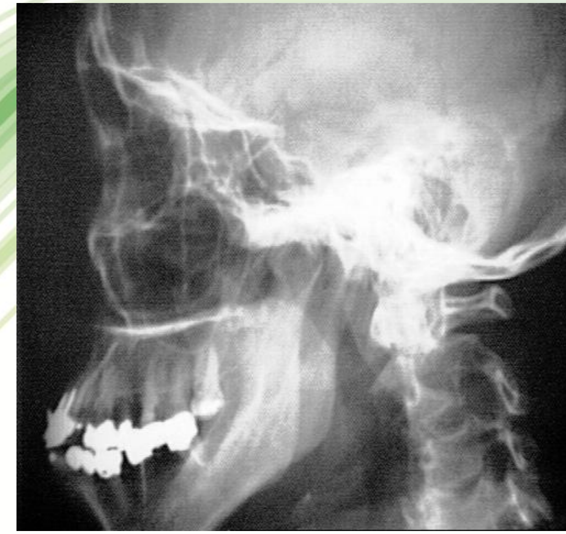
Histogram equalization

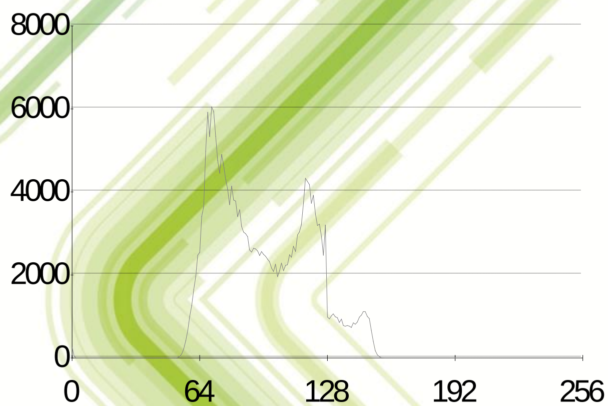
- Result:
  - Not exactly uniform histogram
  - But better than original
  - Cumulative histogram is almost ideal

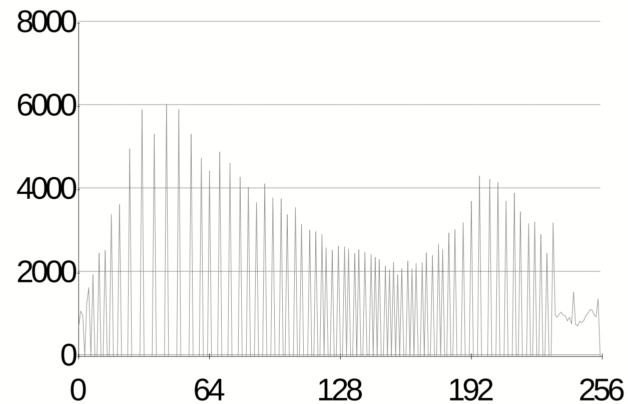




Histogram equalization - example





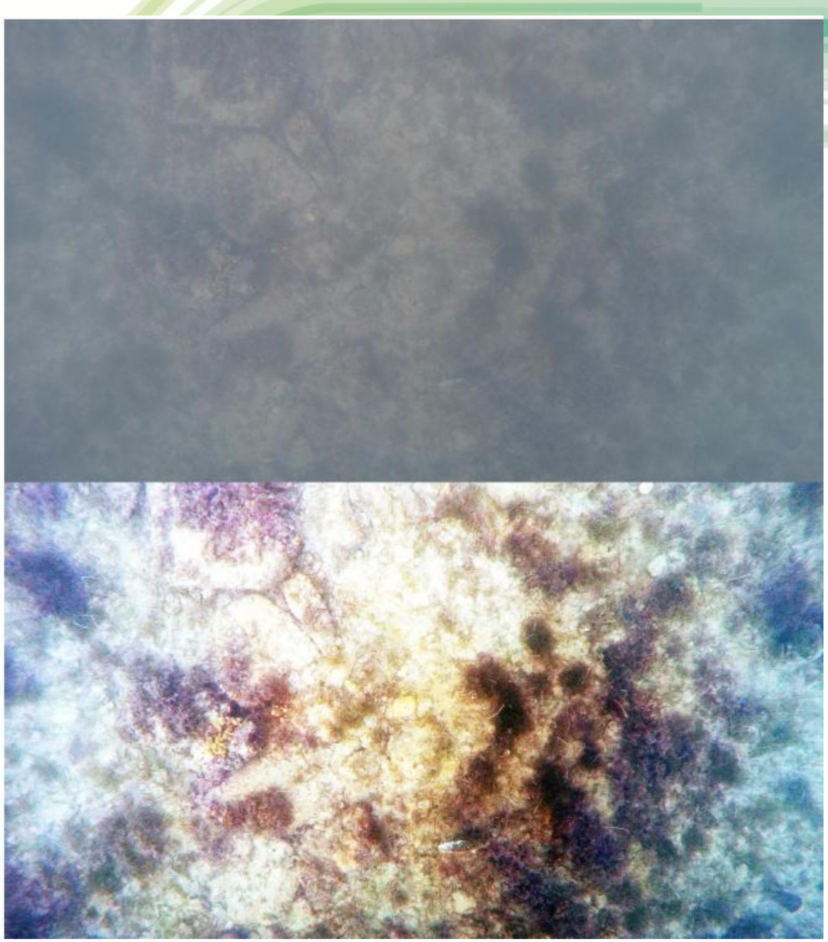


Matlab function: *histeq()*

Histogram equalization - example

- Underwater image

Matlab function: *histeq()* applied to each of the R,G,B channels separately

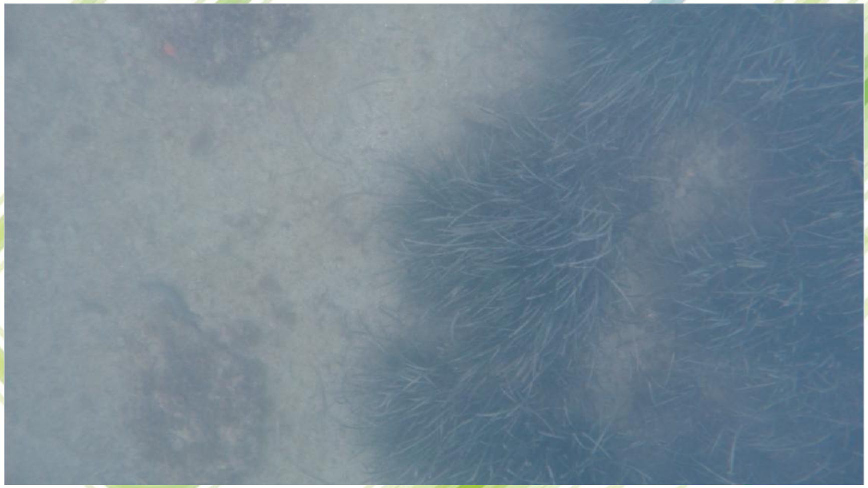


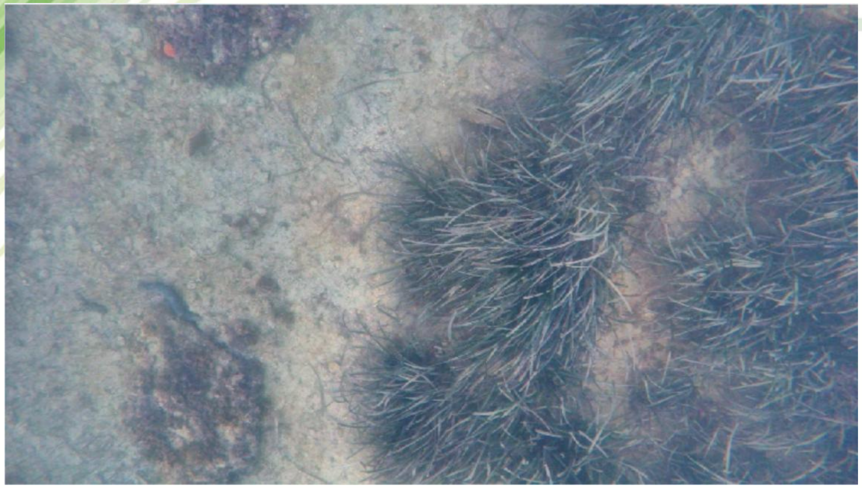
Adaptive histogram equalization

- Procedure
  - Divide the image into (possibly overlapping) regions (subimages).
  - Run histogram equalization on each individual region.
  - Combine the results into the final histogram equalized image.

- Note:
  - for RGB image you can equalize each color channel individualy. There are other options. One option would be to convert RGB to HSV, equalize V only, then convert back to RGB.

Adaptive histogram equalization - example





Matlab function: *adapthisteq()*

Histogram specification

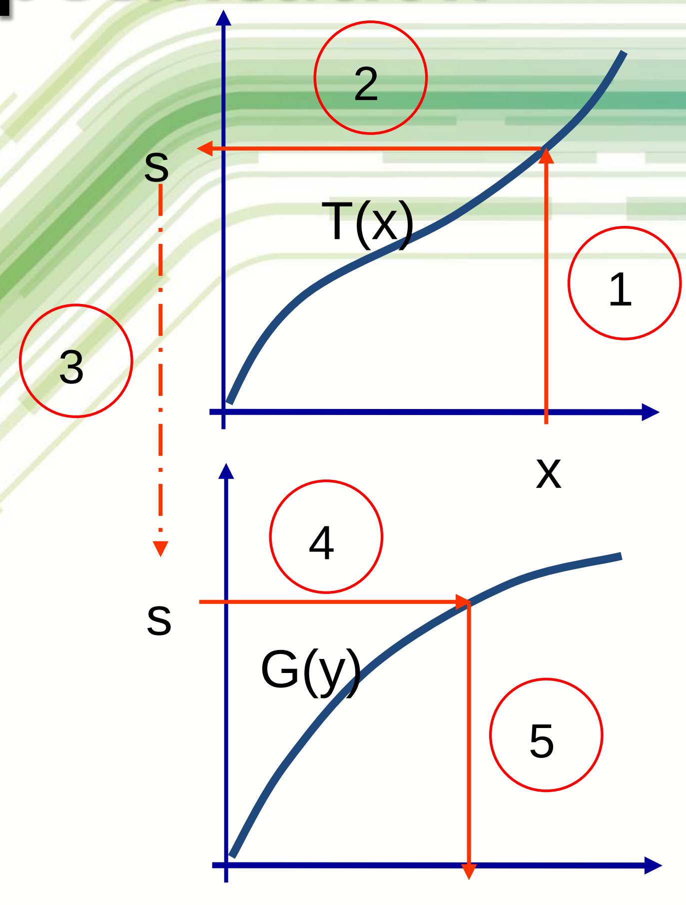
$$X \mapsto Y$$

$$\mathrm{p_x(x),\ p_y(y)}$$

$$\mathrm{s = T(x)} = \int_0^x p_x(x)dx$$

$$G(y) = \int_0^y p_y(y)dy = s$$

$$y = G^{-1}(s) = G^{-1}(\mathrm{T(x)})$$

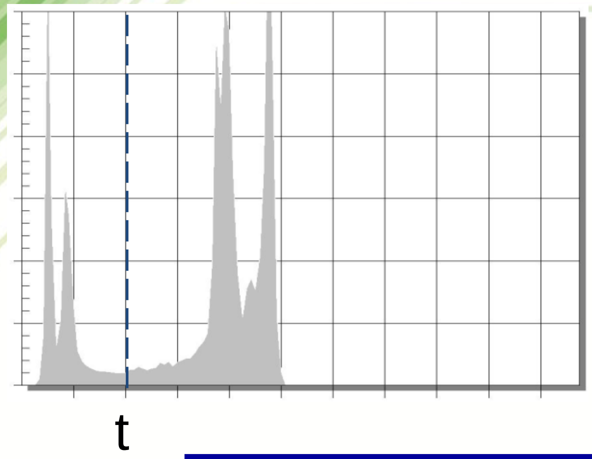


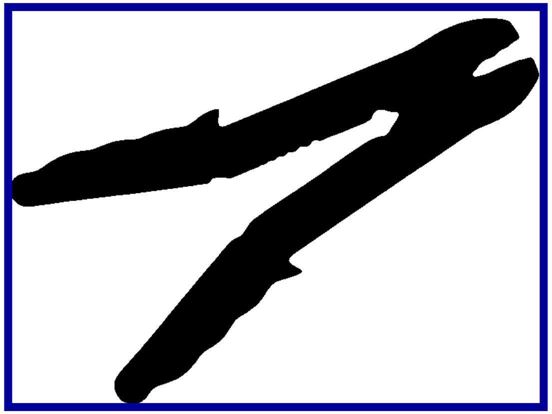
Can be efficiently implemented using LUTS

Function in Matlab *imhistmatch()*

Thresholding

We can use image histogram to select reasonable threshold!

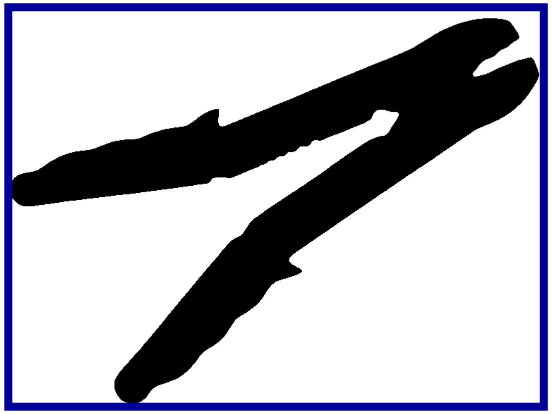




Where to we put t?
– Between two largest peaks
– In the widest valley
– Etc.

Why thresholding?

- Thresholding separates objects from background.
  - A binary image can then be used for further processing.
    - object counting' (connected components analysis)
    - ‚contour tracing / following'
    - topology
    - area, dimensions, orientation, ...
    - Shape analysis
  - Matlab function *regionprops()*



Questions?