

# COMP9517

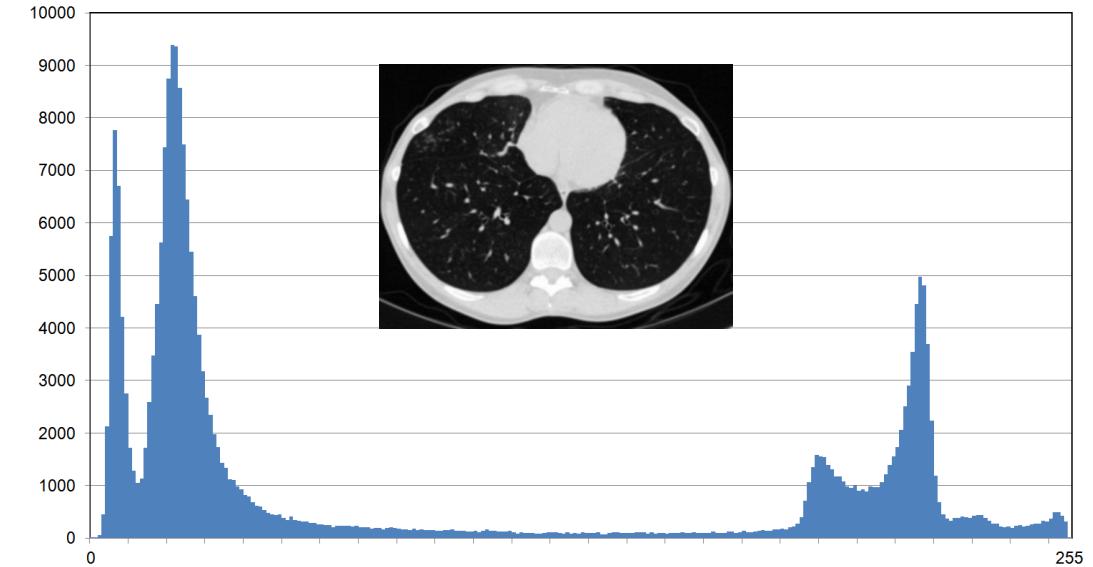
## Computer Vision

2024 Term 3 Week 1

Professor Erik Meijering



**UNSW**  
SYDNEY



## Image Processing

### Basics

# What is image processing?

- **Image processing** = image in > image out
- Aims to **suppress distortions** and **enhance relevant information**
- Used to **prepare images for further analysis** and interpretation
- **Image analysis** = image in > features/measurements out
- **Computer vision** = image in > interpretation out

# Types of image processing

- Two main types of image processing operations:
  - **Spatial domain operations** (in image space) Next week
  - **Transform domain operations** (mainly in Fourier space)
- Two main types of spatial domain operations: Today
  - **Point operations** (intensity transformations on individual pixels)
  - **Neighbourhood operations** (spatial filtering on groups of pixels)

# Topics and learning goals

- Describe the workings of **basic point operations**  
Contrast stretching, thresholding, inversion, log/power transformations
- Understand and use the **intensity histogram**  
Histogram specification, equalization, matching
- Define **arithmetic and logical operations**  
Summation, subtraction, AND/OR, averaging

# Spatial domain operations

- General form of spatial domain operations

$$g(x, y) = T[f(x, y)]$$

where

$f(x, y)$  is the input image

$g(x, y)$  is the processed image

$T[\cdot]$  is the operator applied at  $(x, y)$

# Spatial domain operations

- Point operations:  $T$  operates on individual pixels

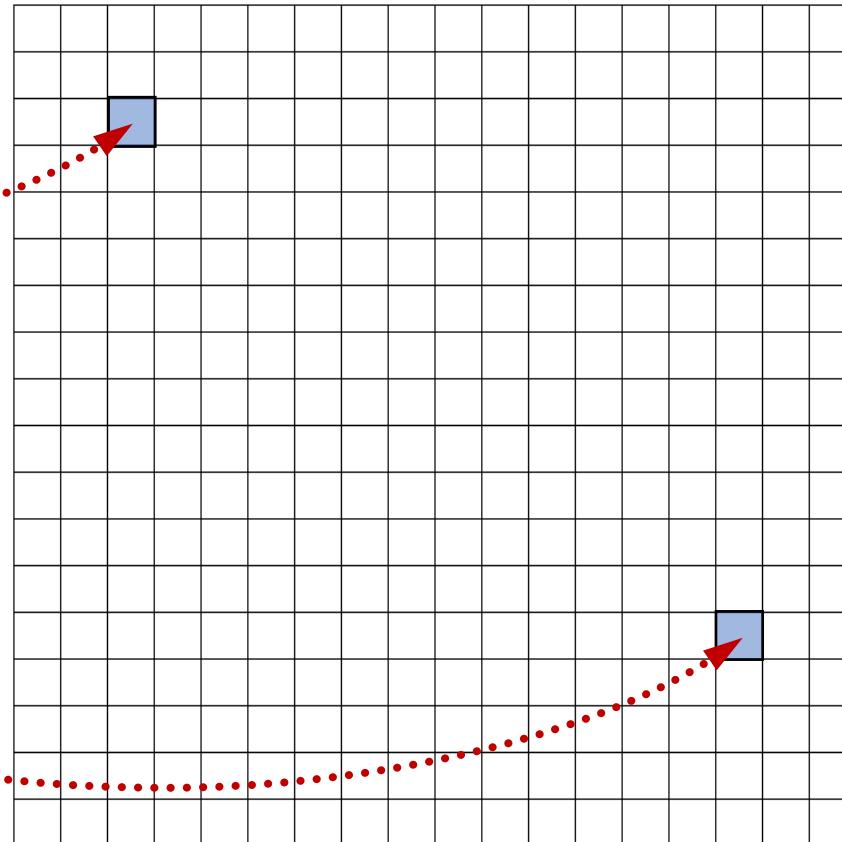
$$T: \mathbb{R} \rightarrow \mathbb{R} \quad g(x, y) = T(f(x, y))$$

- Neighbourhood operations:  $T$  operates on multiple pixels

$$T: \mathbb{R}^n \rightarrow \mathbb{R} \quad g(x, y) = T(f(x, y), f(x + 1, y), f(x - 1, y), \dots)$$

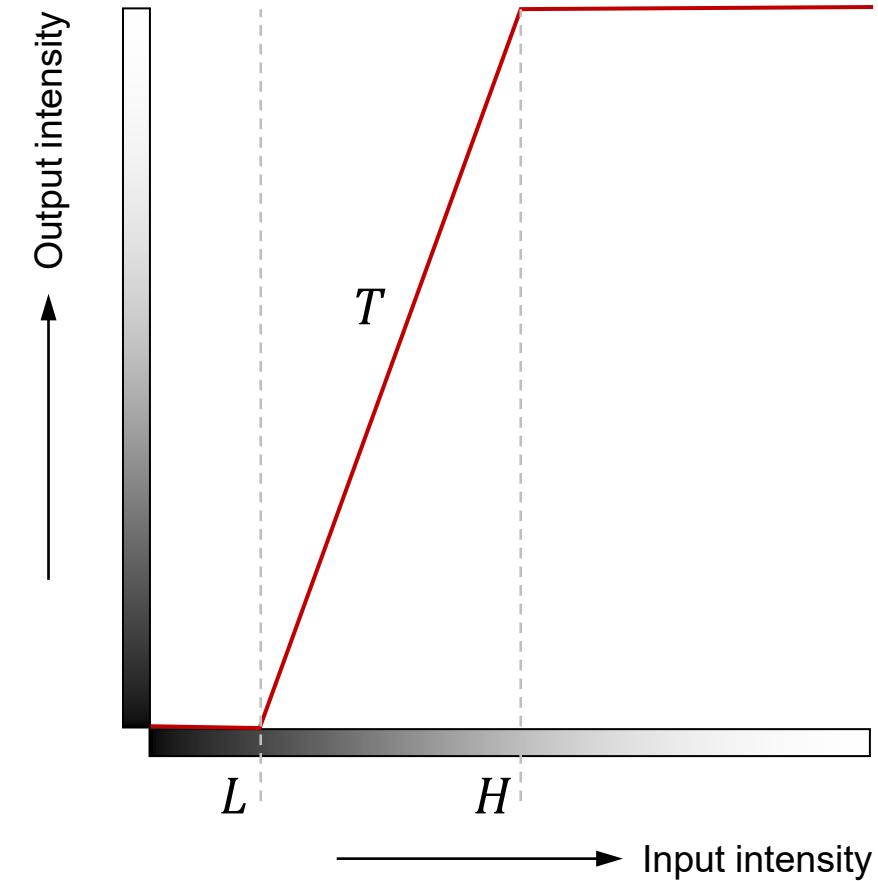
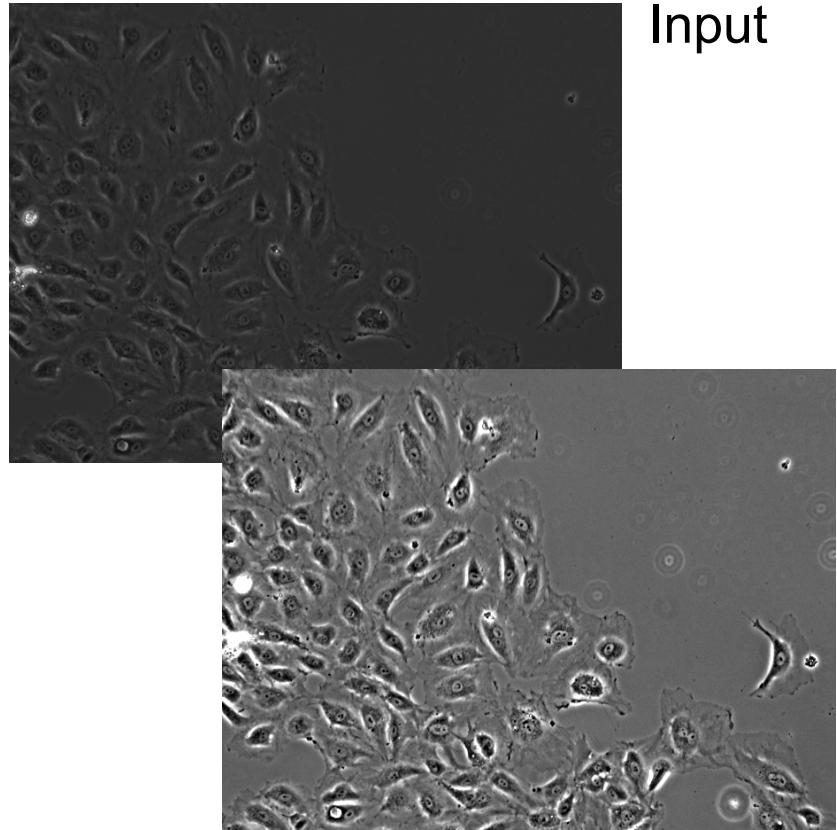
# Point operations

Input  
image



# Output image

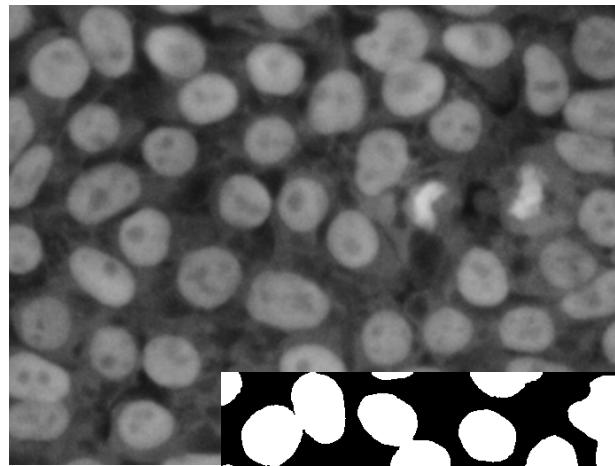
# Contrast stretching



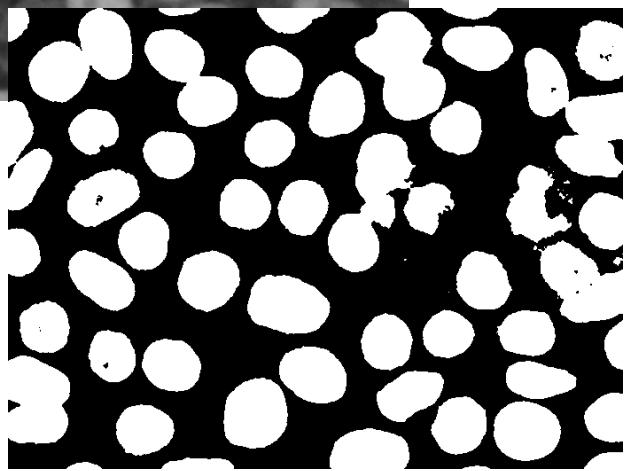
# Contrast stretching

- Produces images of higher contrast
- Puts values below  $L$  in the input to the minimum (black) in the output
- Puts values above  $H$  in the input to the maximum (white) in the output
- Linearly scales values between  $L$  and  $H$  (inclusive) in the input to between the minimum (black) and the maximum (white) in the output

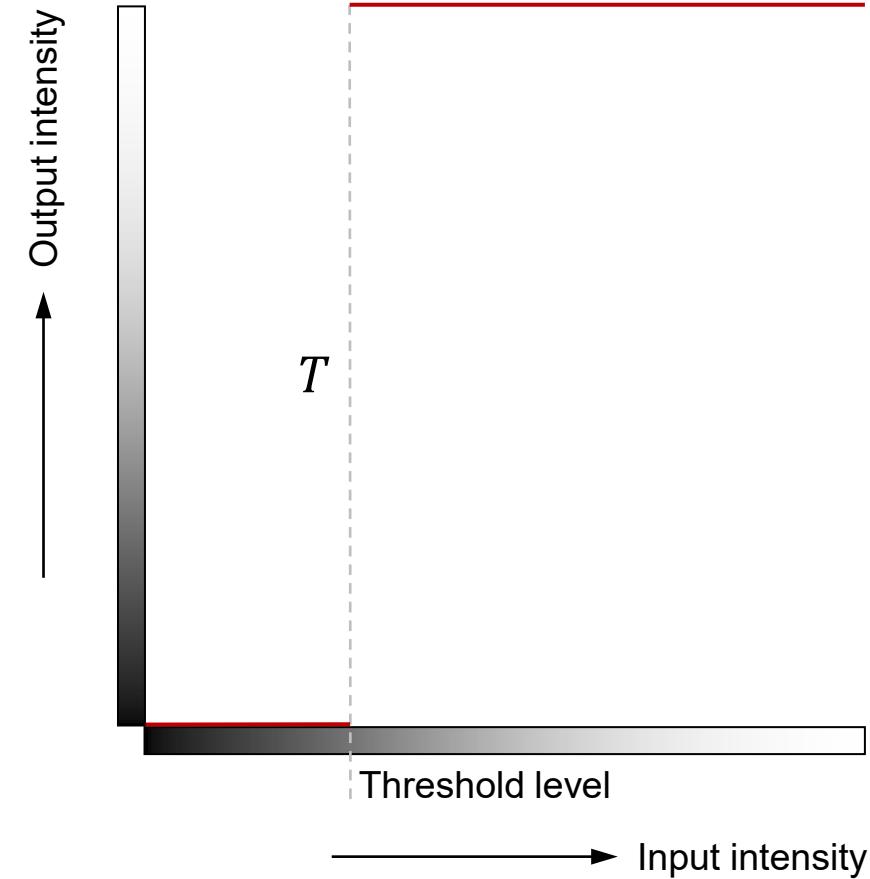
# Intensity thresholding



Input



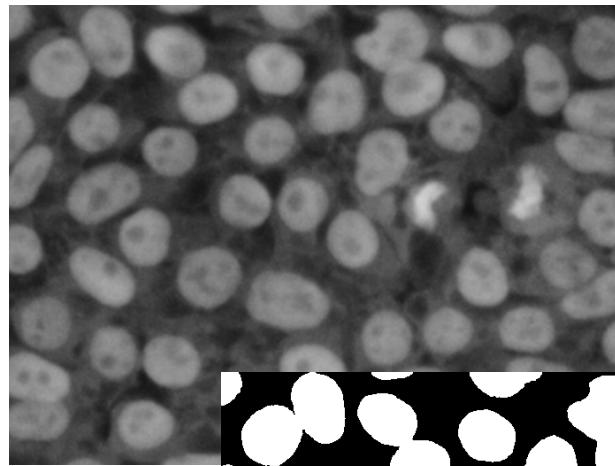
Output



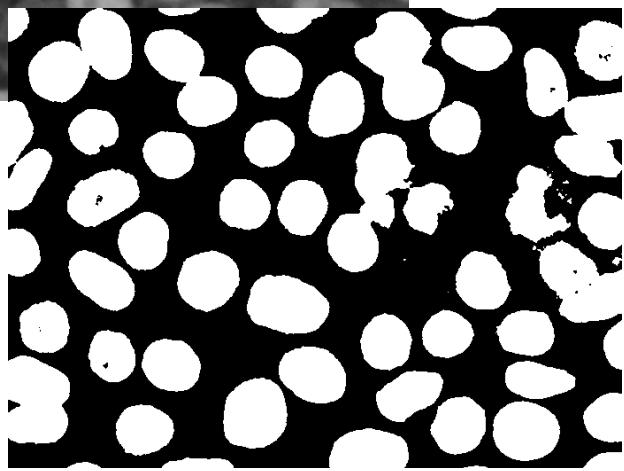
# Intensity thresholding

- Limiting case of contrast stretching
- Produces binary images of gray-scale images
- Puts values below the threshold to black in the output
- Puts values equal/above the threshold to white in the output
- Popular method for image segmentation (discussed later)
- Useful only if object and background intensities are very different
- Result depends strongly on the threshold level (user parameter)

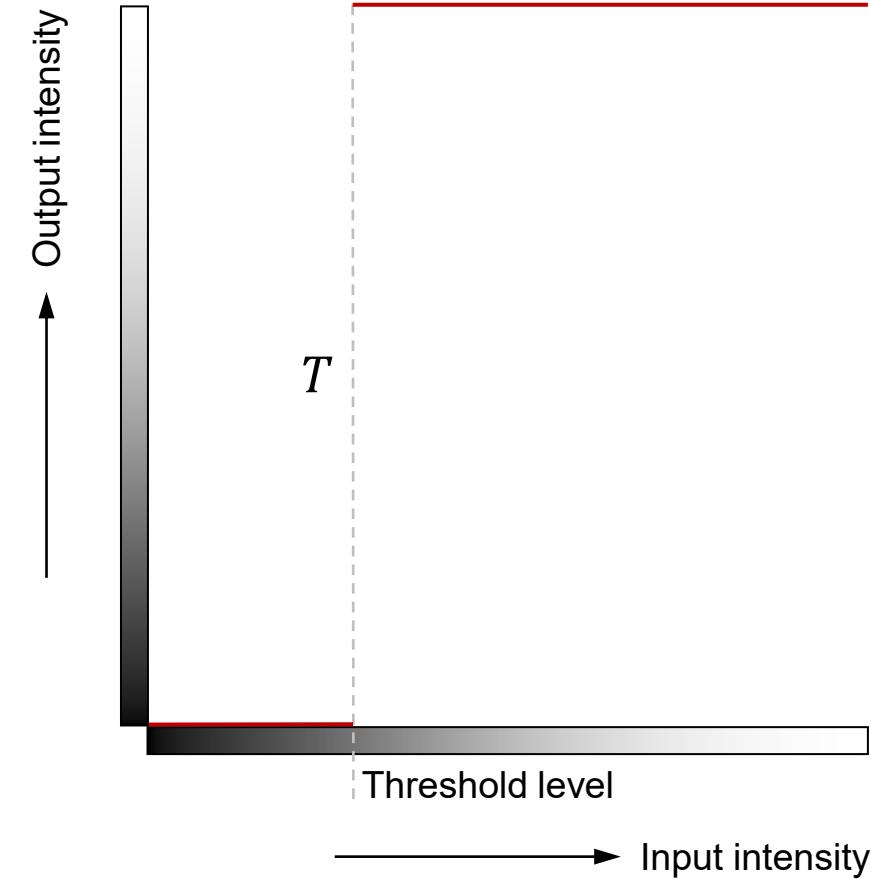
# Intensity thresholding



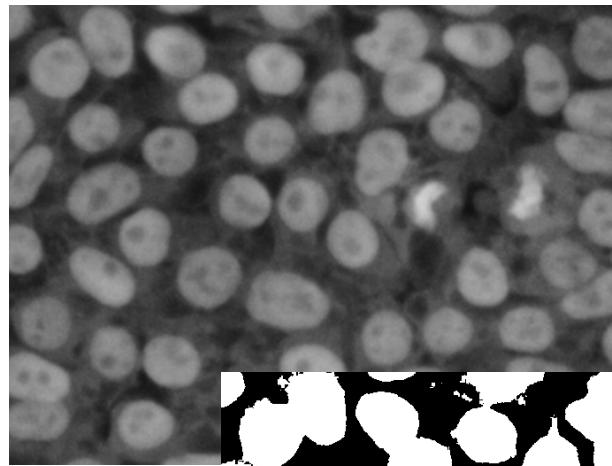
Input



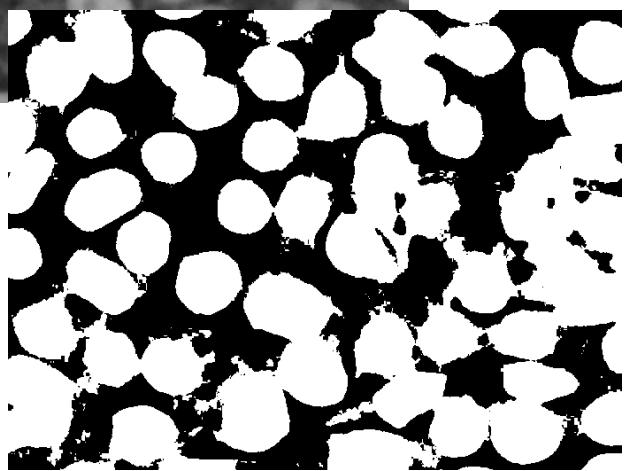
Output



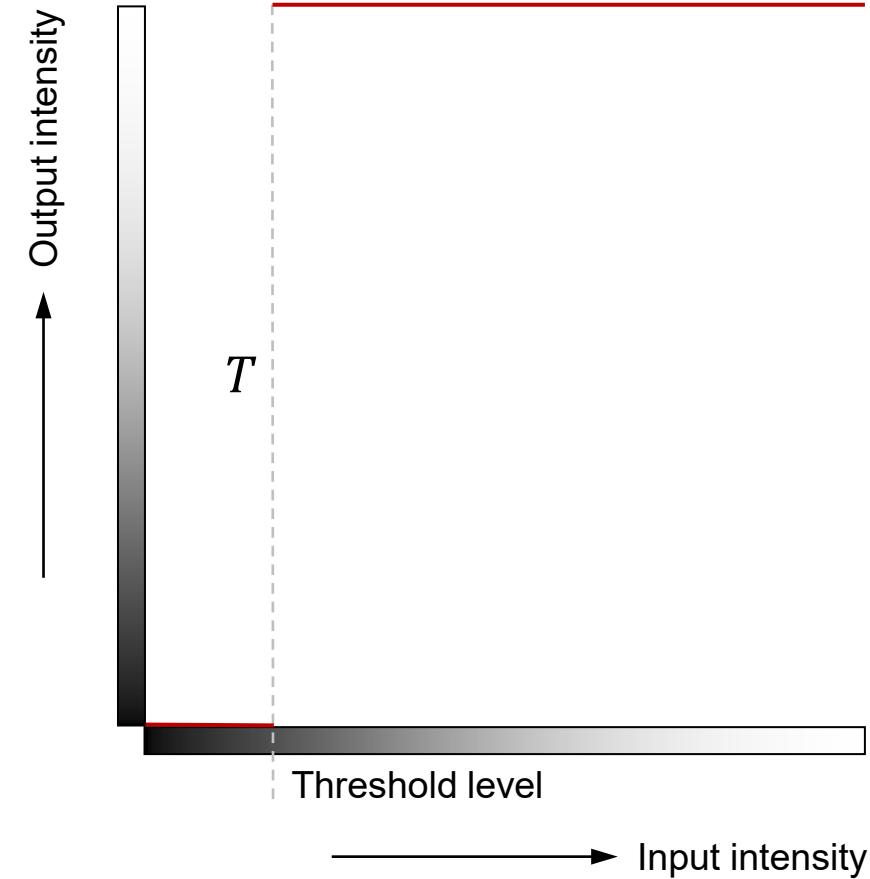
# Intensity thresholding



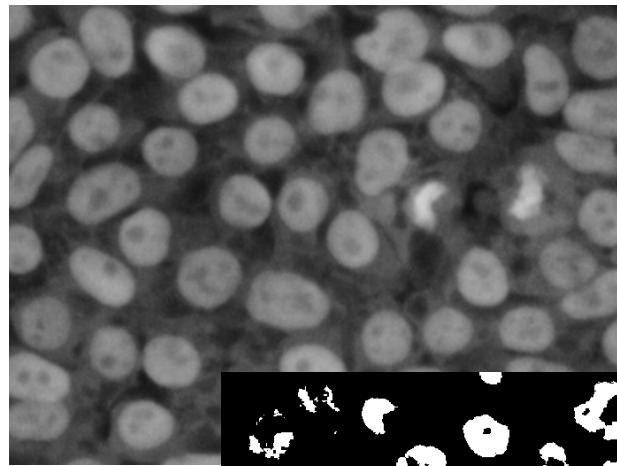
Input



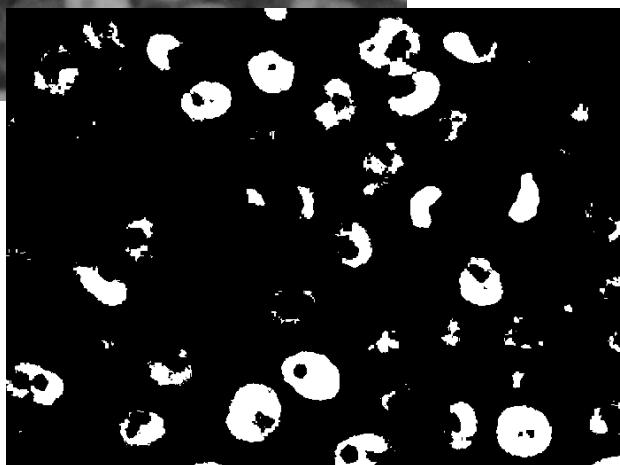
Output



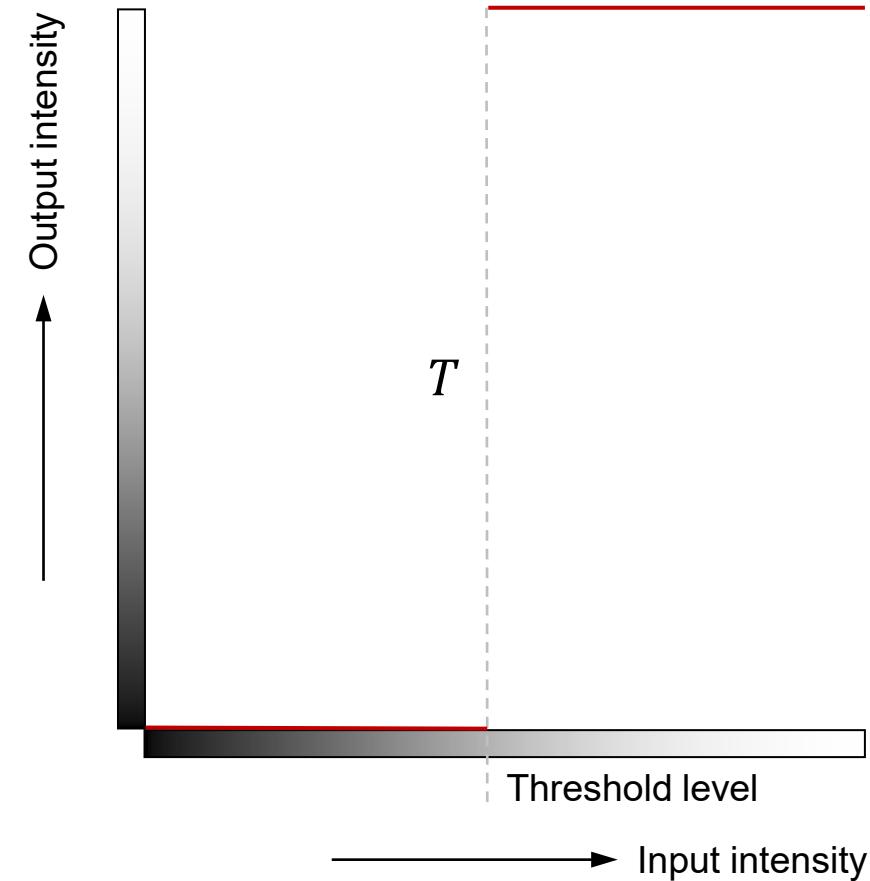
# Intensity thresholding



Input



Output



# Automatic intensity thresholding

- Otsu's method for computing the threshold automatically <https://doi.org/10.1109/TSMC.1979.4310076>  
Exhaustively searches for the threshold **minimising the intra-class variance**

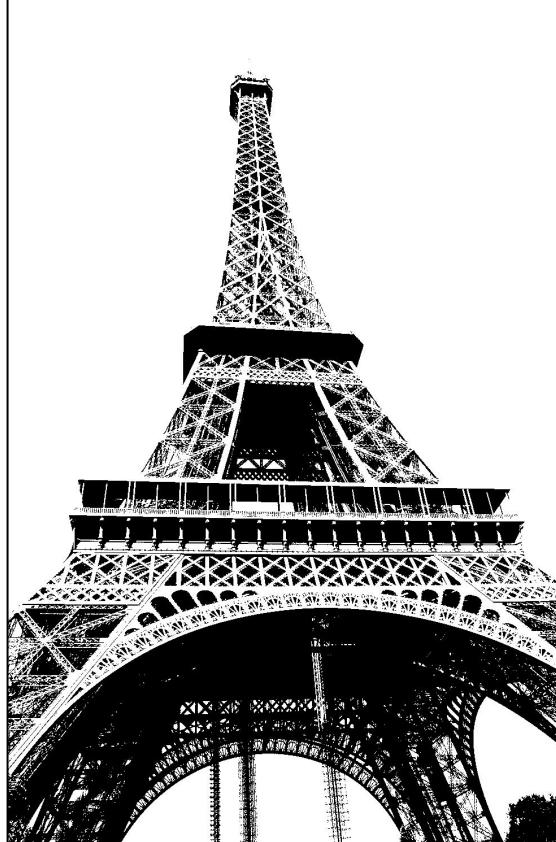
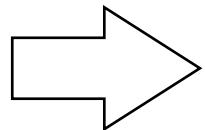
$$\sigma_W^2 = p_0 \sigma_0^2 + p_1 \sigma_1^2$$

Equivalent to **maximising the inter-class variance** (much faster to compute)

$$\sigma_B^2 = p_0 p_1 (\mu_0 - \mu_1)^2$$

Here,  $p_0$  is the fraction of pixels below the threshold (class 0),  $p_1$  is the fraction of pixels equal to or above the threshold (class 1),  $\mu_0$  and  $\mu_1$  are the mean intensities of pixels in class 0 and class 1,  $\sigma_0^2$  and  $\sigma_1^2$  are the intensity variances, and  $p_0 + p_1 = 1$  and  $\sigma_0^2 + \sigma_1^2 = \sigma^2$

# Otsu thresholding example

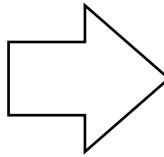


# Automatic intensity thresholding

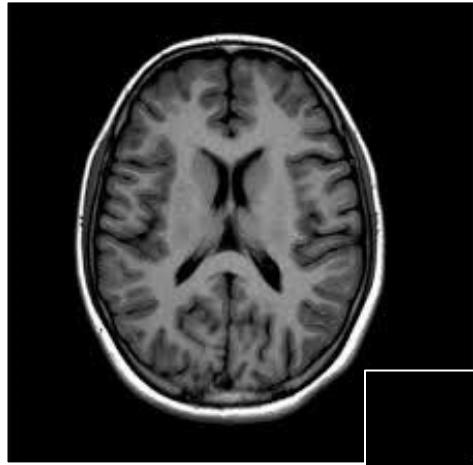
- IsoData method for computing the threshold automatically
  1. Select an arbitrary initial threshold  $t$
  2. Compute  $\mu_0$  and  $\mu_1$  with respect to the threshold
  3. Update the threshold to the mean of the means:  $t = (\mu_0 + \mu_1)/2$
  4. If the threshold changed in Step 3, go to Step 2

Upon convergence, the threshold is midway between the two class means

# Isodata thresholding example



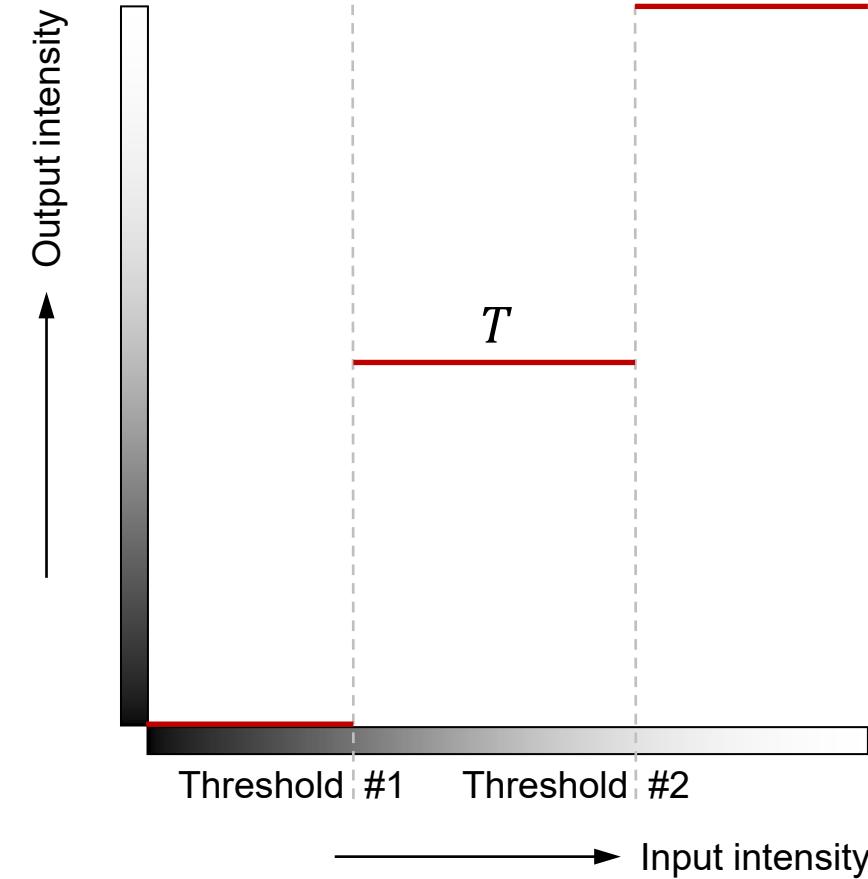
# Multilevel thresholding



Input



Output



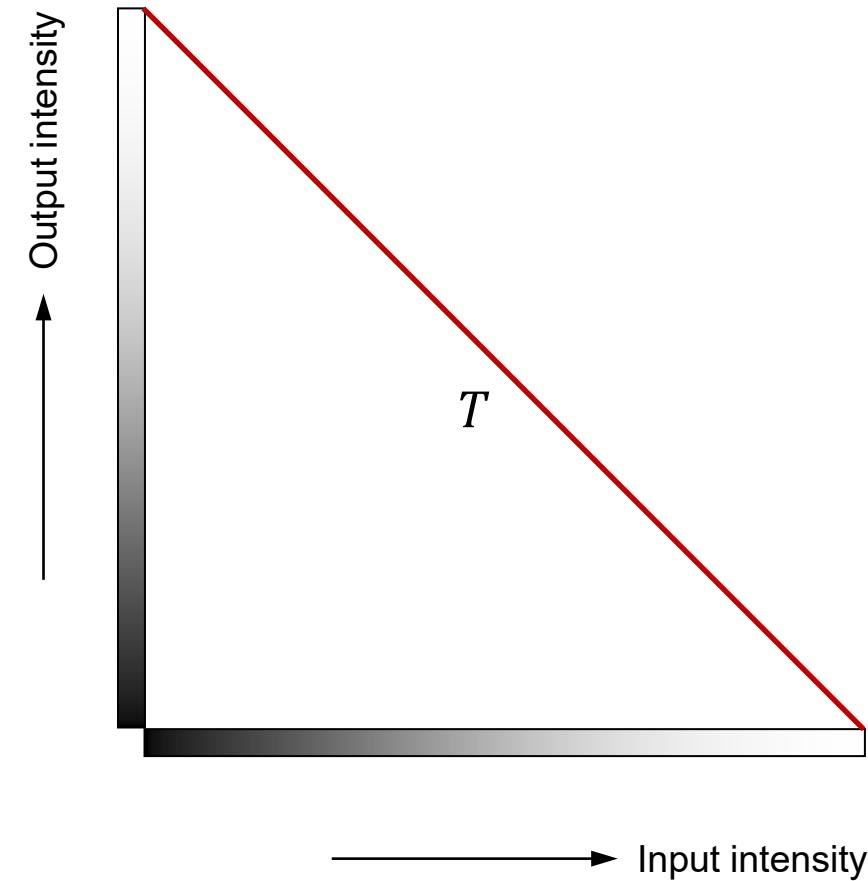
# Intensity inversion



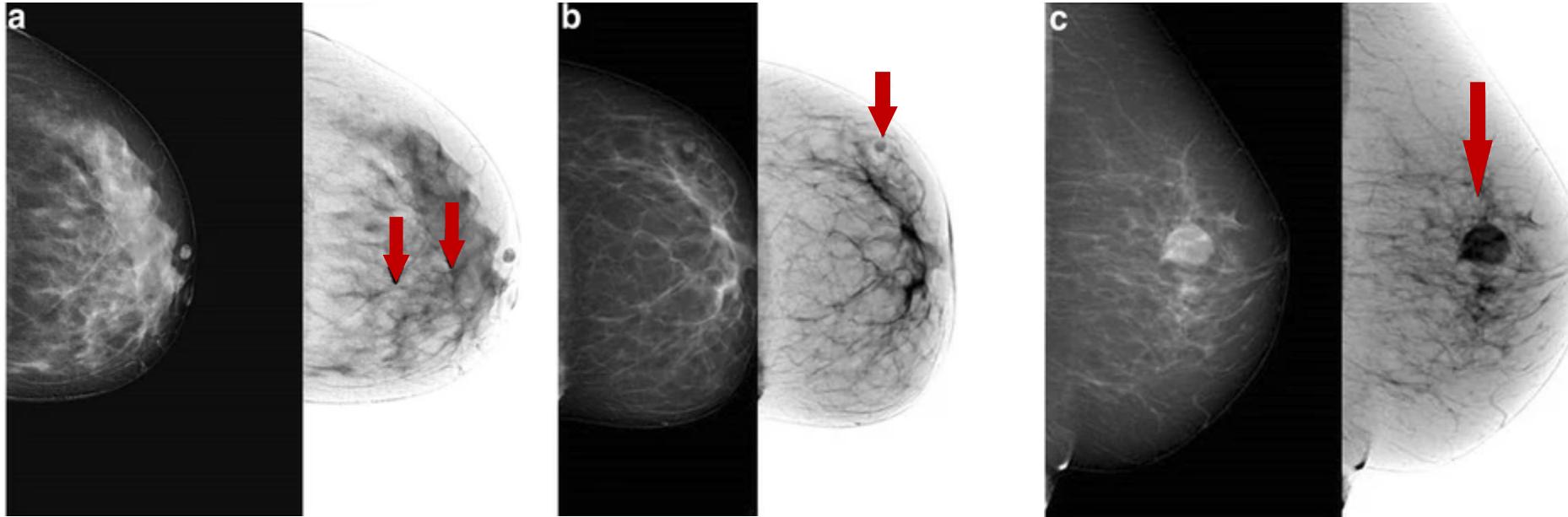
Input



Output



# Intensity inversion examples



“Assessment of grayscale inverted images in addition to standard images facilitates the detection of microcalcification.” <https://doi.org/10.1186/s12880-017-0196-6>

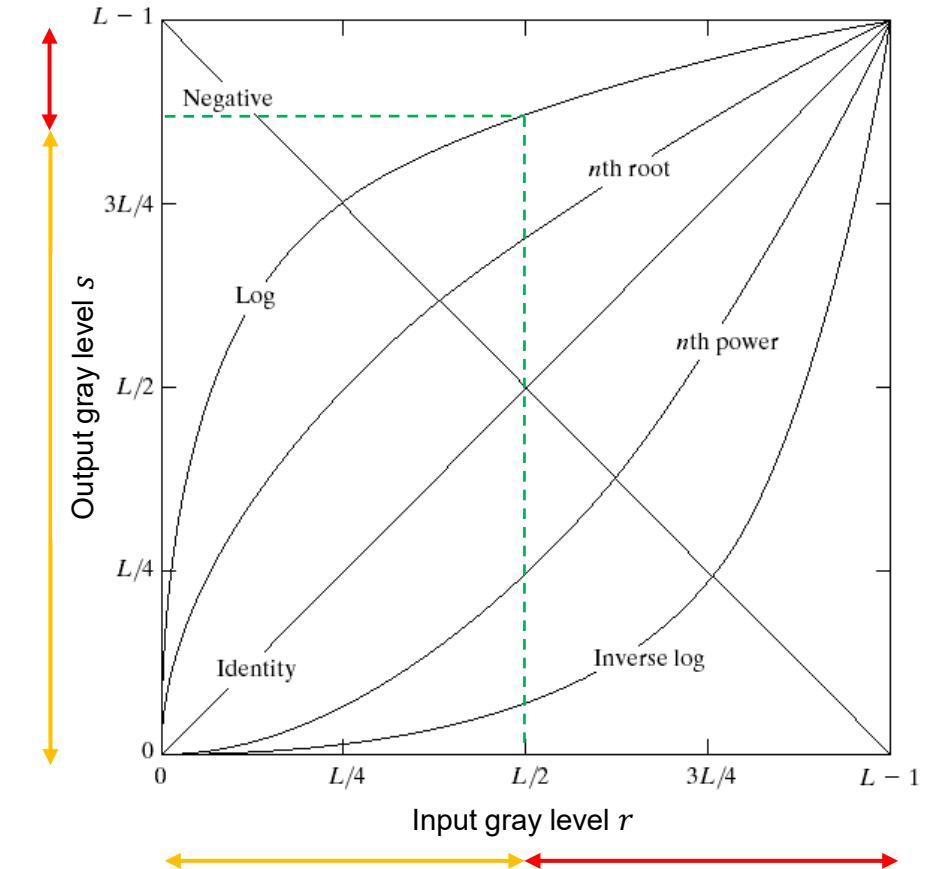
# Log transformation

- Definition of log transformation

$$s = c \log(1 + r)$$

where  $r$  is the input intensity,  $s$  is the output intensity, and  $c$  is a constant

- Maps a narrow input range of low gray-level values into a wider range of output values, and vice versa for higher gray-level values
- Also compresses the dynamic range of images with large variations in pixel values (such as Fourier spectra, to be discussed later)



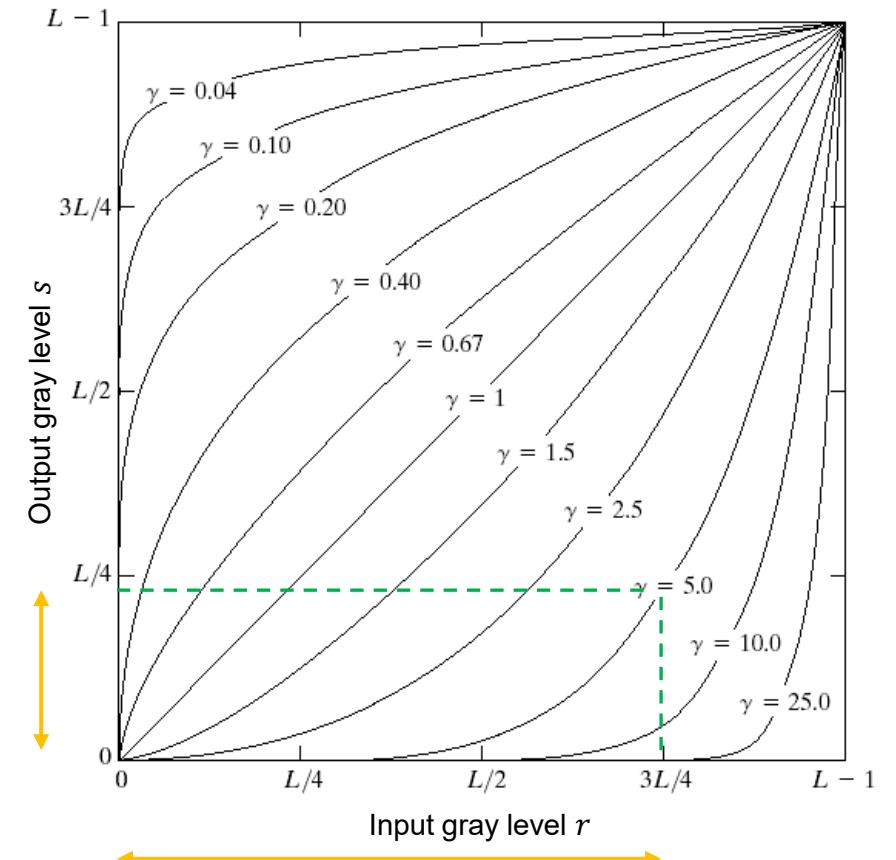
# Power transformation

- Definition of power transformation

$$s = c r^\gamma$$

where  $c$  and  $\gamma$  are constants

- Similar to (inverse) log transformation
- Represents a family of transformations by varying  $\gamma$
- Many devices respond according to a power law
- Example power transformation: gamma correction
- Useful for general-purpose contrast manipulation



# Power transformation examples

$$c = 1$$

Input



$$\gamma = 3$$



$$\gamma = 4$$

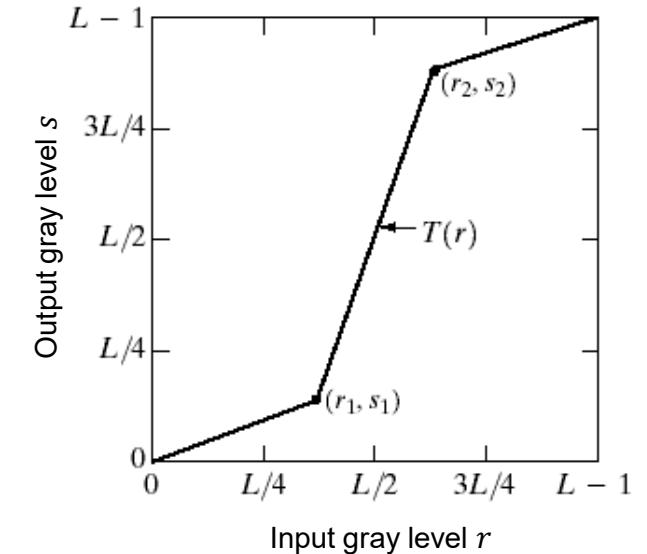


$$\gamma = 5$$



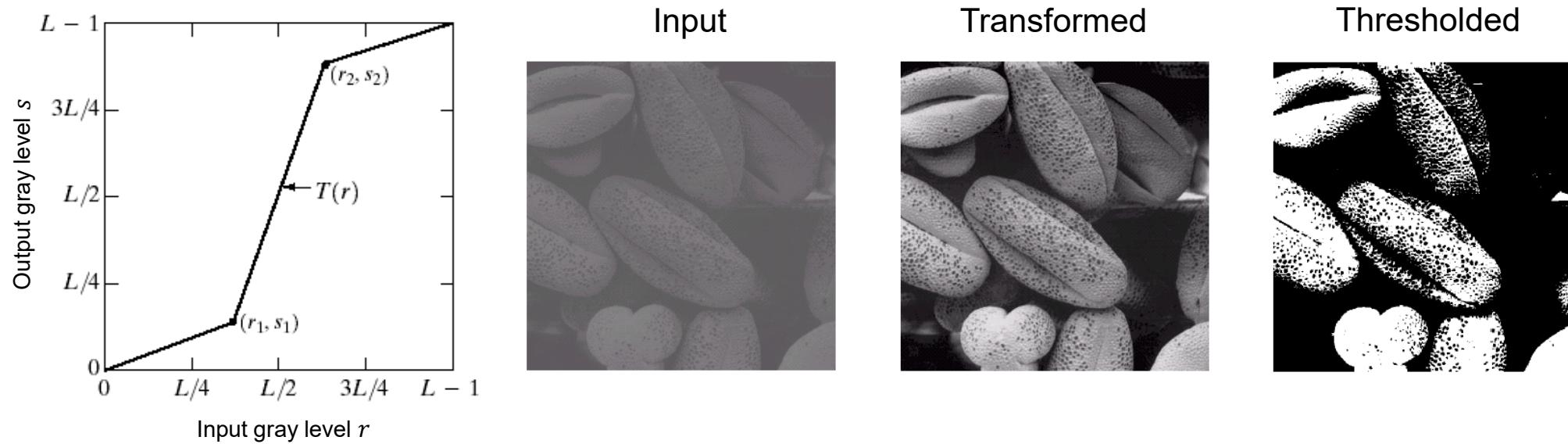
# Piecewise linear transformations

- Complementary to other transformation methods
- Enable more fine-tuned design of transformations
- Can have very complex shapes
- Requires more user input



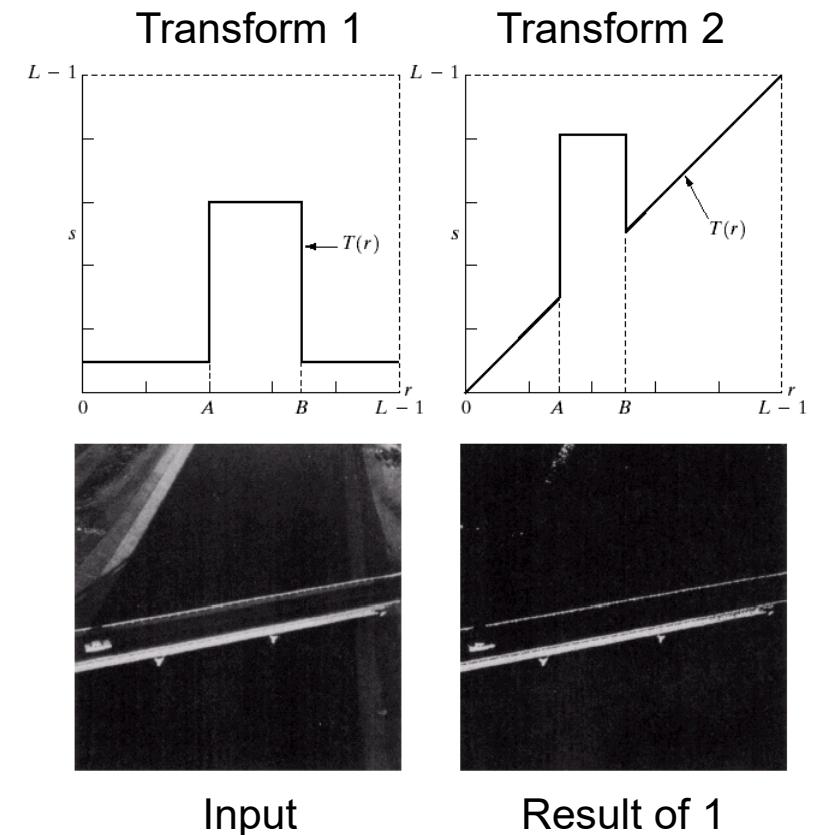
# Piecewise contrast stretching

- One of the simplest piecewise linear transformations
- Increases the dynamic range of gray levels in images
- Used in display devices or recording media to span full range



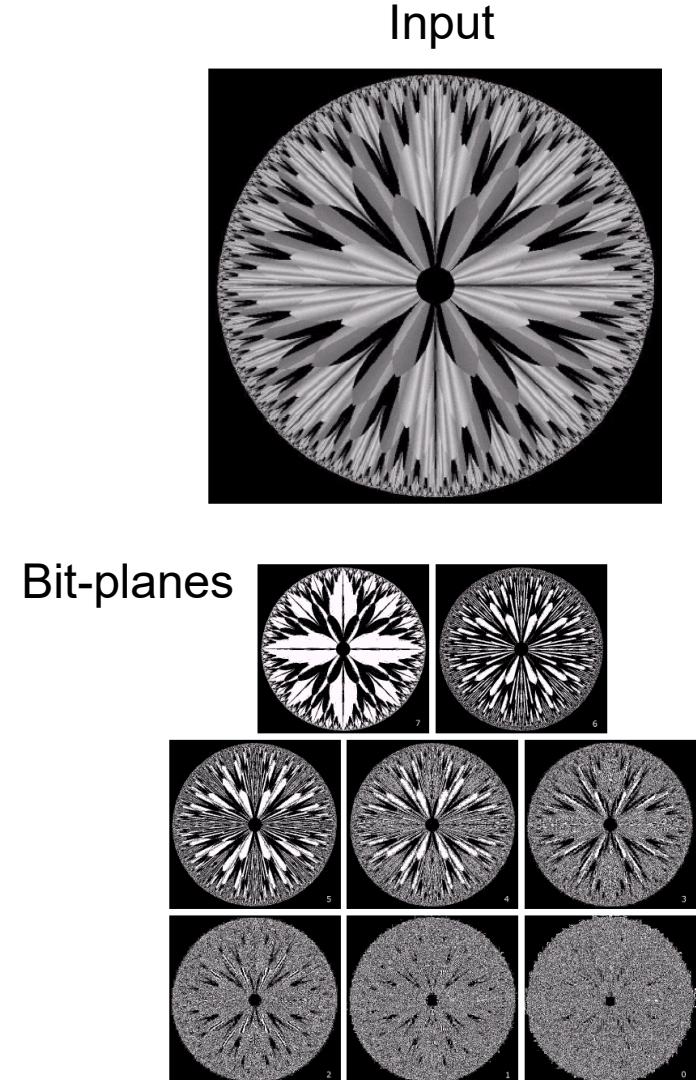
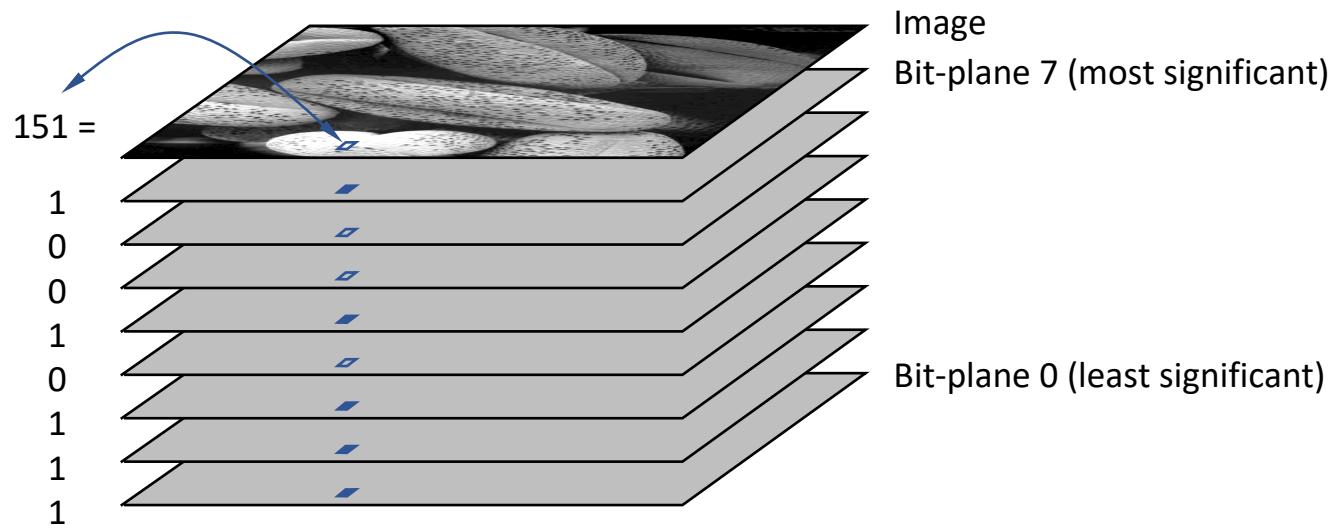
# Gray-level slicing

- Used to highlight a specific range of gray levels
- Two different slicing approaches:
  - 1) High value for all gray levels in a range of interest and low value for all others (produces a binary image)
  - 2) Brighten a desired range of gray levels while preserving background and other gray-scale tones of the image



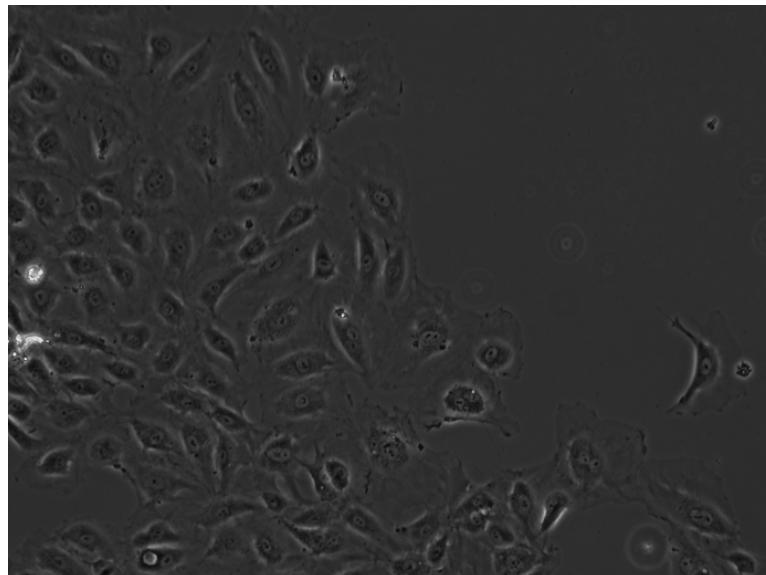
# Bit-plane slicing

- Highlights contribution to total image by specific bits
- An image with  $n$  bits/pixel has  $n$  bit-planes
- Can be useful for image compression

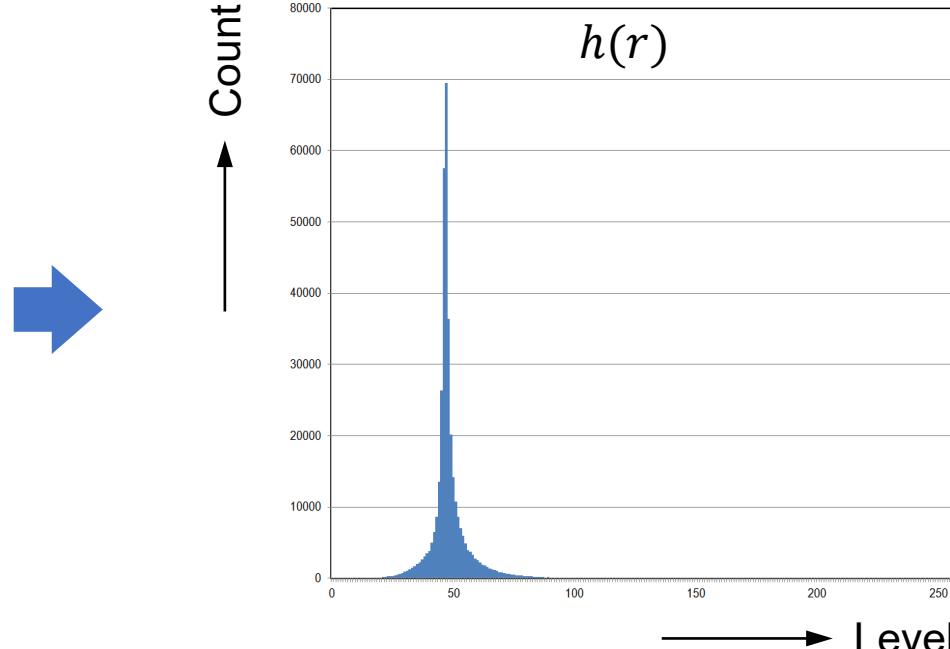


# Histogram of pixel values

- For every possible gray-level value, count the number of pixels having that value, and plot the pixel counts as a function of gray level



8-bit image



$$L = 2^8 = 256$$

$$N = \#\text{pixels}$$

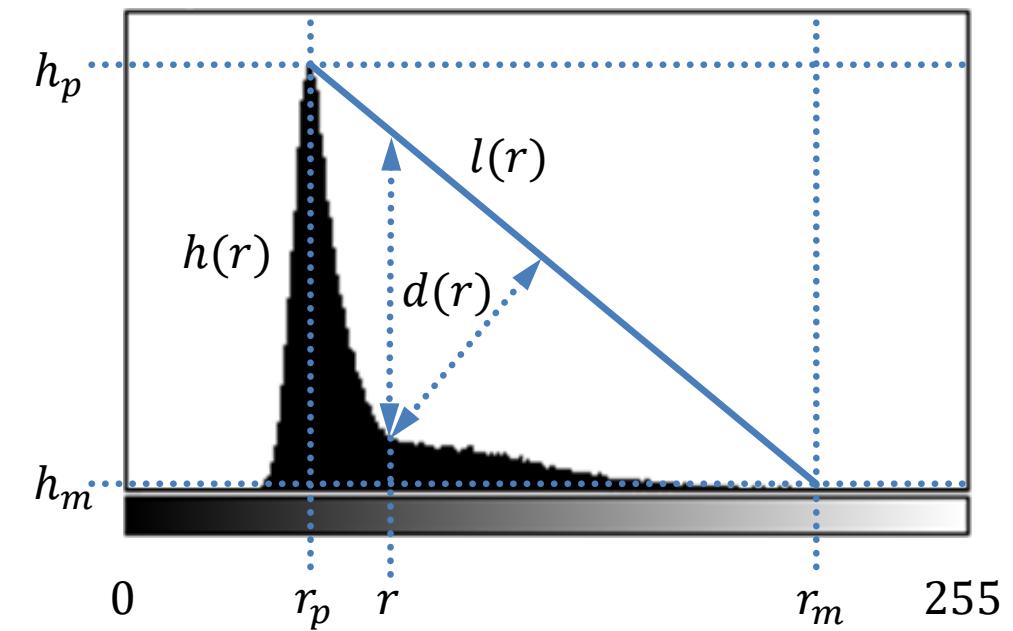
$$\sum_{r=0}^{L-1} h(r) = N$$

Normalized histogram  
= probability function

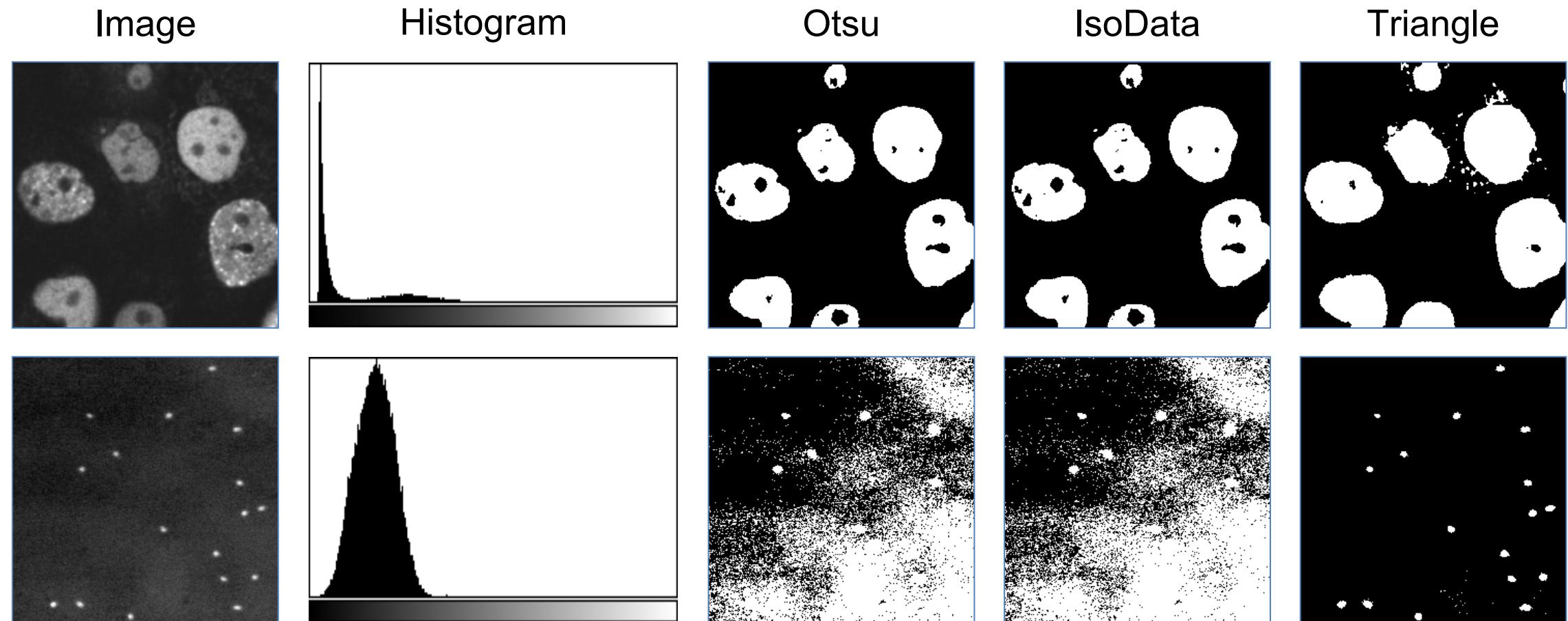
$$\frac{1}{N} h(r) = p(r)$$

# Histogram based thresholding

- Triangle method for computing the threshold automatically
  1. Find the histogram peak  $(r_p, h_p)$  and the highest gray level point  $(r_m, h_m)$
  2. Construct a straight line  $l(r)$  from the peak to the highest gray level point
  3. Find the gray level  $r$  for which the distance  $\|l(r) - h(r)\|$  is the largest



# Comparison of thresholding methods



# Histogram processing

- **Histogram equalization**

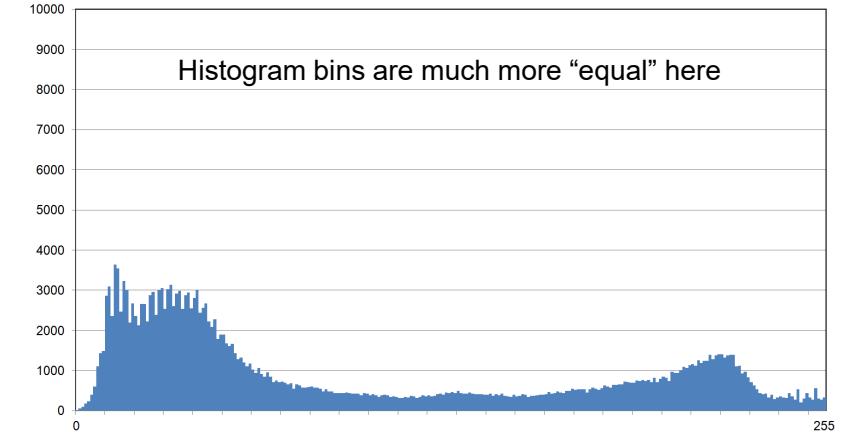
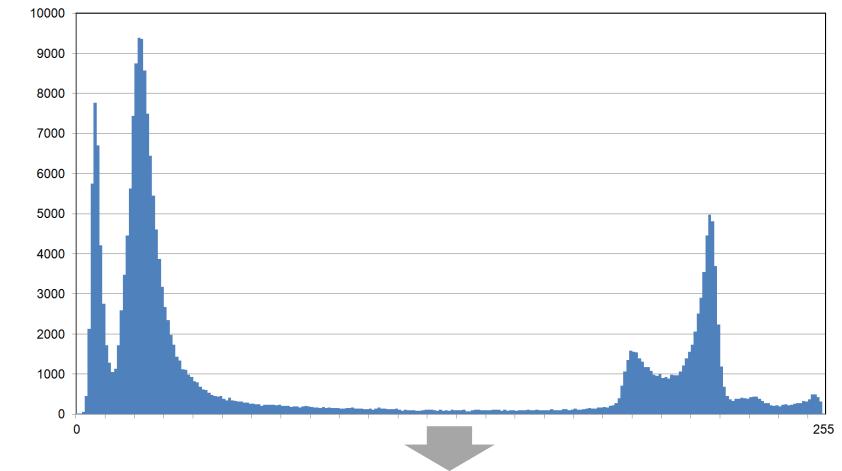
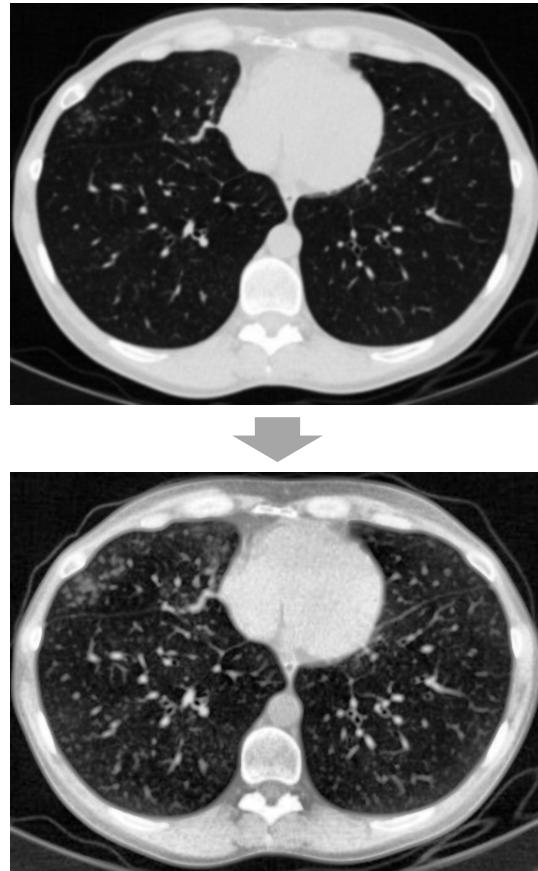
Aim: To get an image with equally distributed intensity levels over the full intensity range

- **Histogram specification** (also called **histogram matching**)

Aim: To get an image with a specified intensity distribution, determined by the shape of the histogram

# Histogram equalization

Enhances contrast for intensity values near histogram maxima and decreases contrast near histogram minima



# Histogram equalization

- Let  $r \in [0, L - 1]$  represent pixel values (intensities, gray levels)  
 $r = 0$  represents black and  $r = L - 1$  represents white
- Consider transformations  $s = T(r)$ ,  $0 \leq r \leq L - 1$ , satisfying
  - 1)  $T(r)$  is single-valued and monotonically increasing in  $0 \leq r \leq L - 1$   
This guarantees that the inverse transformation  $T^{-1}(s)$  exists
  - 2)  $0 \leq T(r) \leq L - 1$  for  $0 \leq r \leq L - 1$   
This guarantees that the input and output ranges will be the same

# Histogram equalization (continuous case)

Consider  $r$  and  $s$  as continuous random variables over  $[0, L - 1]$  with PDFs  $p_r(r)$  and  $p_s(s)$

If  $p_r(r)$  and  $T(r)$  are known and  $T^{-1}(s)$  satisfies monotonicity, then, from probability theory

$$p_s(s) = p_r(r) \left| \frac{dr}{ds} \right| \quad \text{<https://www.cl.cam.ac.uk/teaching/2003/Probability/prob11.pdf>}$$

Let us choose:  $s = T(r) = (L - 1) \int_0^r p_r(\xi) d\xi$

This is the CDF (cumulative distribution function) of  $r$  which satisfies conditions (1) and (2)

Now:  $\frac{ds}{dr} = \frac{dT(r)}{dr} = (L - 1) \frac{d}{dr} \left[ \int_0^r p_r(\xi) d\xi \right] = (L - 1)p_r(r)$

Therefore:  $p_s(s) = p_r(r) \left| \frac{1}{(L-1)p_r(r)} \right| = \frac{1}{L-1}$  for  $0 \leq s \leq L - 1$  (uniform distribution)

# Histogram equalization (discrete case)

For discrete values we get probabilities and summations instead of PDFs and integrals:

$$p_r(r_k) = n_k / MN \text{ for } k = 0, 1, \dots, L - 1$$

where  $MN$  is the total number of pixels in image,  $n_k$  is the number of pixels with gray level  $r_k$ , and  $L$  is the total number of gray levels in the image

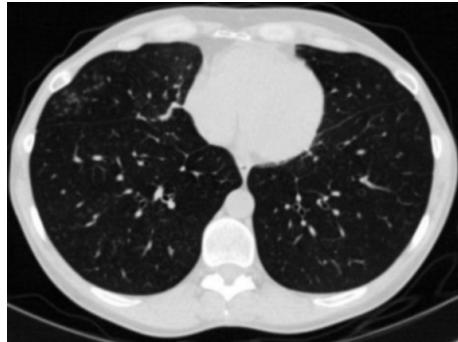
$$\text{Thus: } s_k = T(r_k) = (L - 1) \sum_{j=0}^k p_r(r_j) = \frac{L-1}{MN} \sum_{j=0}^k n_j \text{ for } k = 0, 1, \dots, L - 1$$

This transformation is called *histogram equalization*

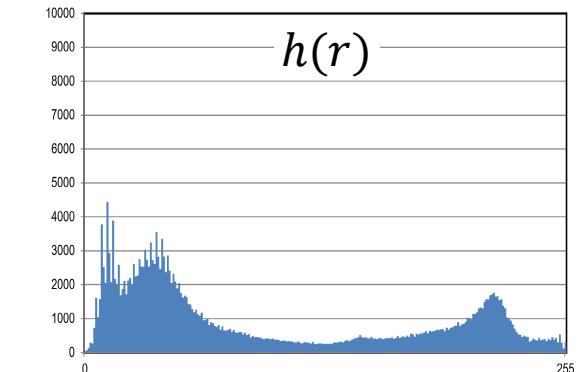
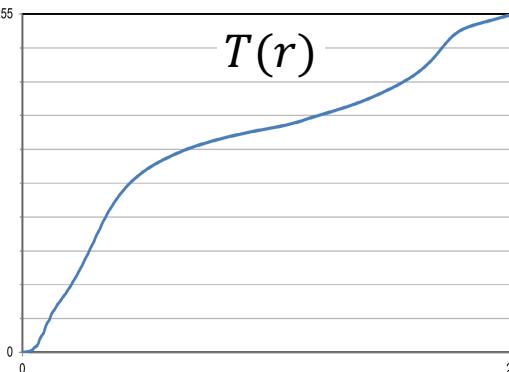
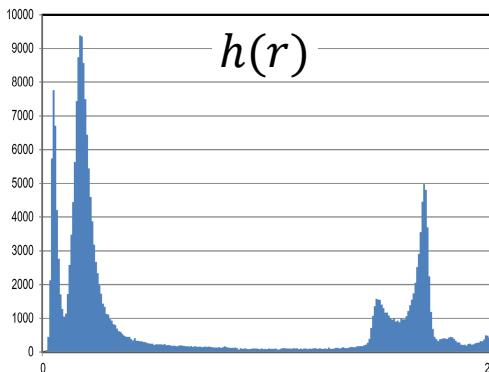
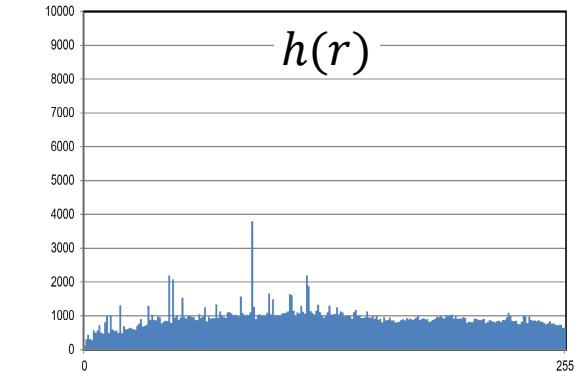
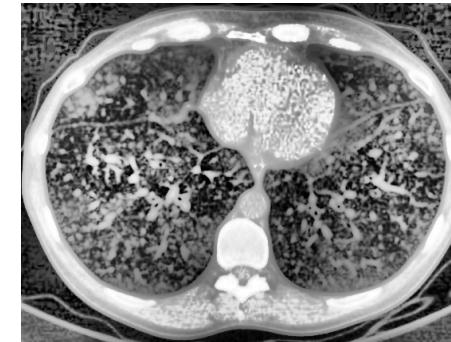
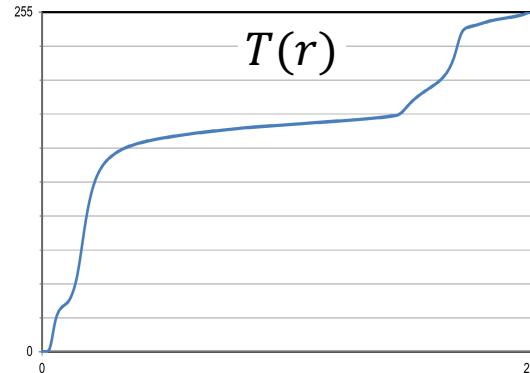
However, for discrete images, applying a single mapping function does not give a truly uniform distribution, and adaptive approaches (multiple mapping functions) are needed

# Constrained histogram equalization

Input



Full histogram equalization (slope of  $T(r)$  is unconstrained)



Constrained histogram equalization (slope of  $T(r)$  is constrained)

# Histogram matching (continuous case)

Assume that  $r$  and  $s$  are continuous and  $p_z(z)$  is the target distribution for the output image

From our previous analysis we know the following transformation results in a uniform distribution:

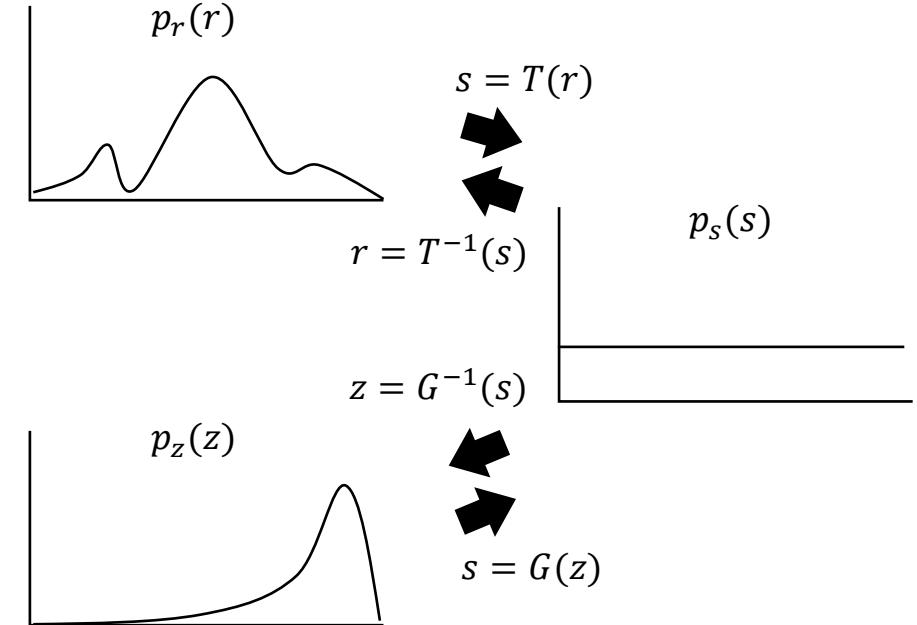
$$s = T(r) = (L - 1) \int_0^r p_r(\xi) d\xi$$

Now we can define a function  $G(z)$  as:

$$G(z) = (L - 1) \int_0^z p_z(\xi) d\xi = s$$

Therefore:

$$z = G^{-1}(s) = G^{-1}[T(r)]$$



# Histogram matching (discrete case)

For discrete image values we can write:

$$s_k = T(r_k) = (L - 1) \sum_{j=0}^k p_r(r_j) = \frac{L-1}{MN} \sum_{j=0}^k n_j$$

$$k = 0, 1, \dots, L - 1$$

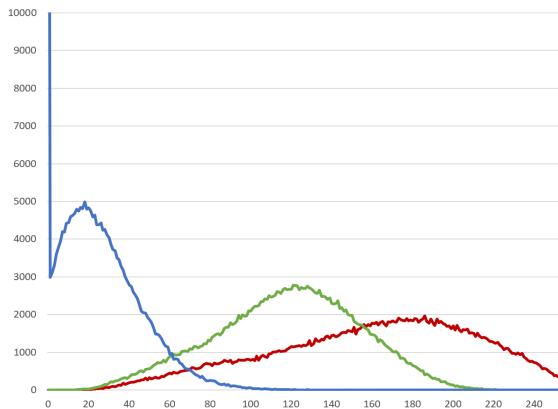
And:  $G(z_q) = (L - 1) \sum_{i=0}^q p_z(z_i)$

Therefore:  $z_q = G^{-1}(s_k)$

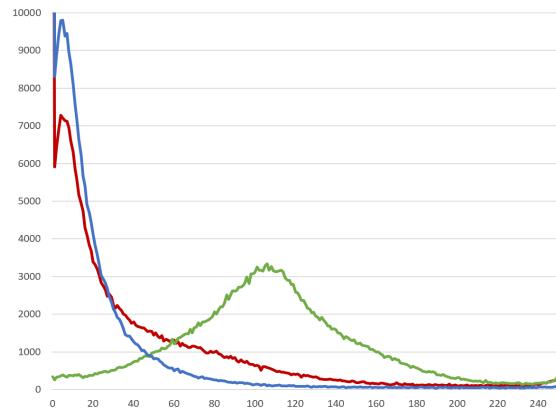
# Histogram matching example

<https://automaticaddison.com/tag/image-processing/page/3/>

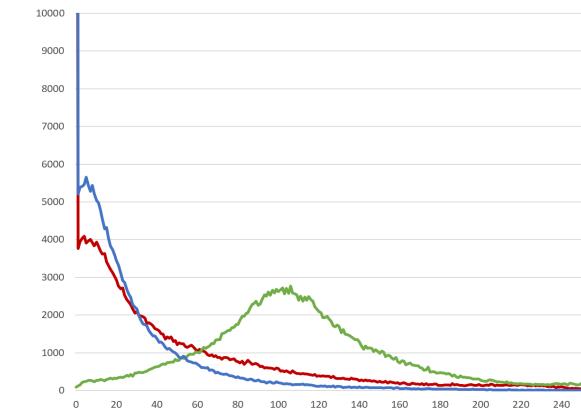
Input



Target



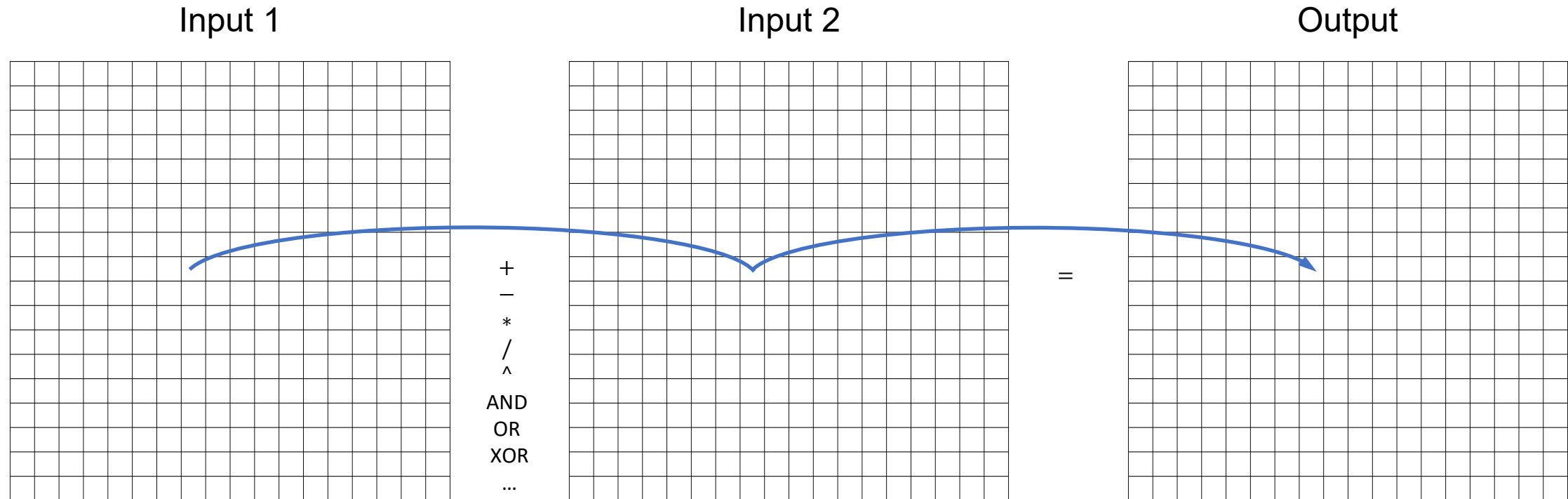
Output



Matching  
done for  
each colour  
channel  
(R,G,B)

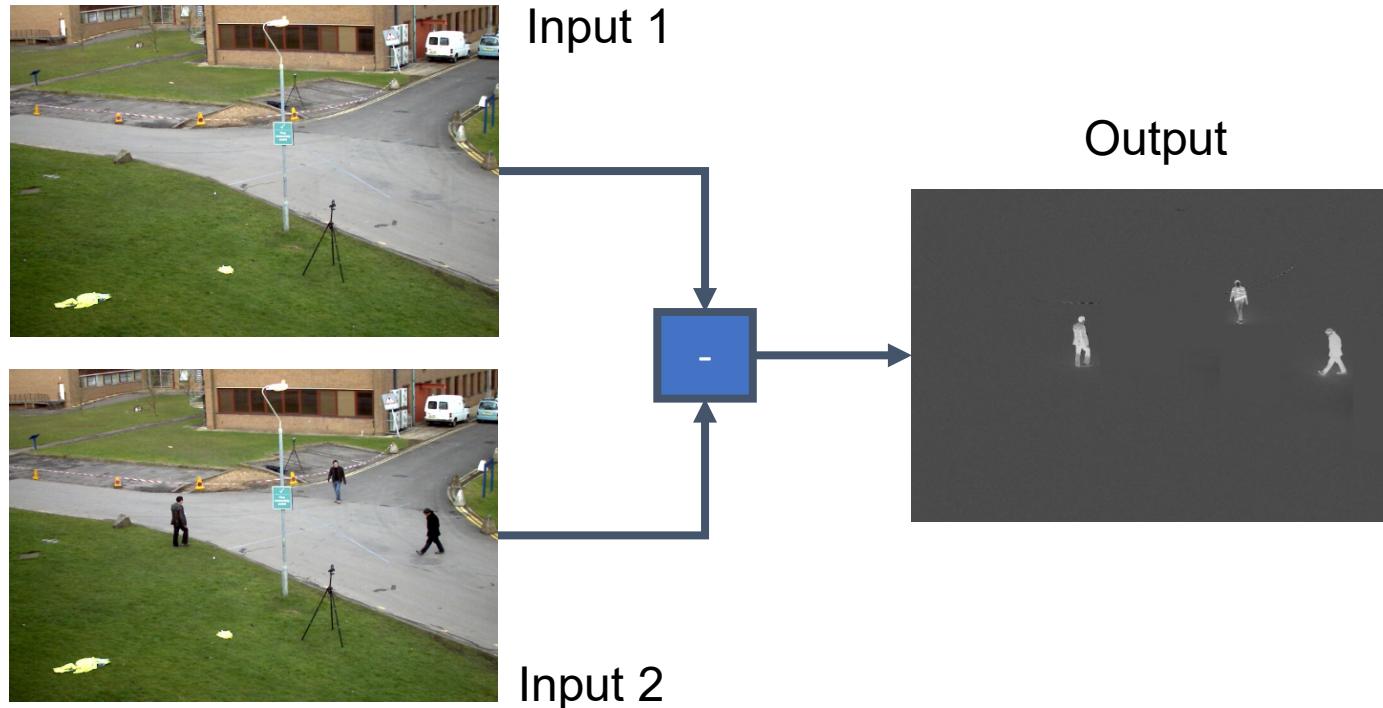
# Arithmetic and logical operations

- Defined on a pixel-by-pixel basis between two images



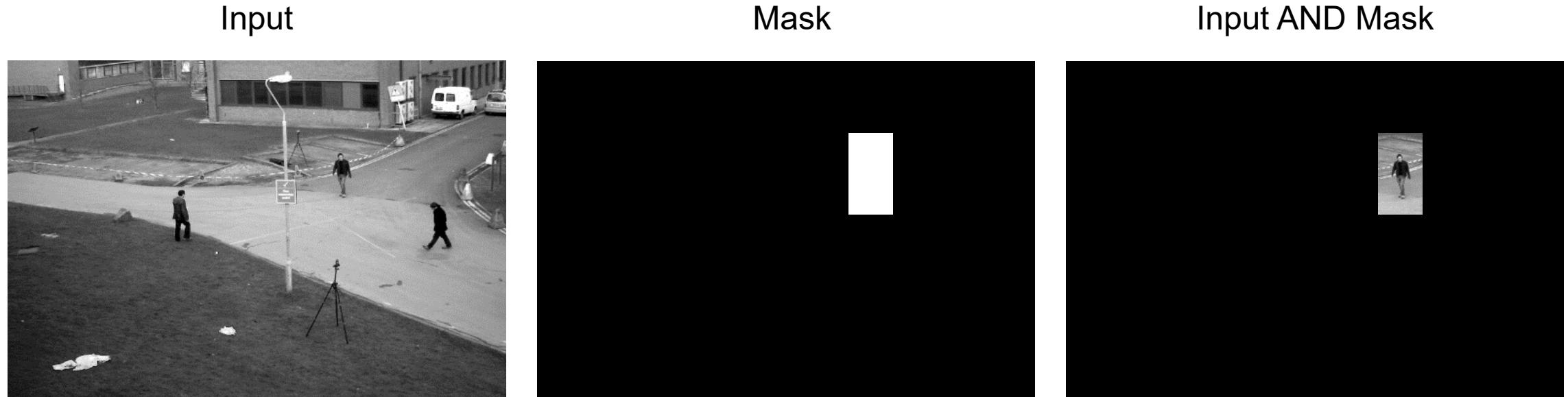
# Arithmetic and logical operations

- Useful arithmetic operations include addition and subtraction



# Arithmetic and logical operations

- Useful logical operations include bitwise AND and OR



# Arithmetic and logical operations

- Useful logical operations include bitwise AND and OR



# Averaging

- Useful for example to reduce noise in images

Assume the true noise-free image is  $g(x, y)$  and the actual observed images are  $f_i(x, y) = g(x, y) + n_i(x, y)$  for  $i = 1, \dots, N$ , where the  $n_i$  are zero-mean, independent and identically distributed (i.i.d.) noise images, then we have  $E[f_i(x, y)] = g(x, y)$  and  $\text{VAR}[f_i(x, y)] = \text{VAR}[n_i(x, y)] = \sigma^2(x, y)$

$$\rightarrow \bar{f}(x, y) = \frac{1}{N} \sum_{i=1}^N f_i(x, y) = \frac{1}{N} \sum_{i=1}^N [g(x, y) + n_i(x, y)] = g(x, y) + \frac{1}{N} \sum_{i=1}^N n_i(x, y)$$

$$\rightarrow \text{VAR}\left[\frac{1}{N} \sum_{i=1}^N n_i(x, y)\right] = \frac{1}{N^2} \sum_{i=1}^N \text{VAR}[n_i(x, y)] = \frac{1}{N^2} N \sigma^2(x, y) = \frac{\sigma^2(x, y)}{N}$$

# Averaging

- Useful for example to reduce noise in images



$N = 1$

$\sigma$

$N = 8$

$\sigma/2.8$

$N = 16$

$\sigma/4$

$N = 64$

$\sigma/8$

$N = 128$

$\sigma/11.3$

# Further reading on discussed topics

- Sections 3.1-3.3 of Szeliski
- Chapter 3 of Gonzalez and Woods 2002

## Acknowledgement

- Some images drawn from the mentioned resources

# Example exam question

Which one of the following statements about intensity transformations is incorrect?

- A. Contrast stretching linearly maps intensities between two values to the full output range.
- B. Log transformation maps a narrow range of high intensities to a wider range of output values.
- C. Power transformation can map intensities similar to log and inverse log transformations.
- D. Piecewise linear transformations can achieve contrast stretching and intensity slicing.