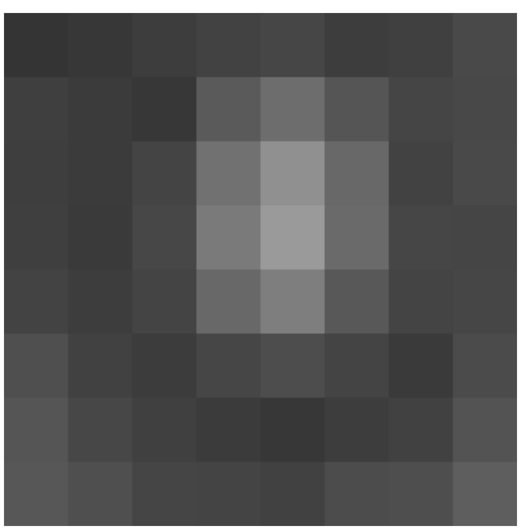
Digital Image Processing

Image Enhancement :
Gray-Level Transformations

Suyash P. Awate

Digital Image

- Assumptions
  - 8-bit per pixel
  - 256 intensity levels or gray levels
    - 0,1, ..., 255 : this is called the grayscale
    - 0 = black
    - 255 = white



$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

Enhancement

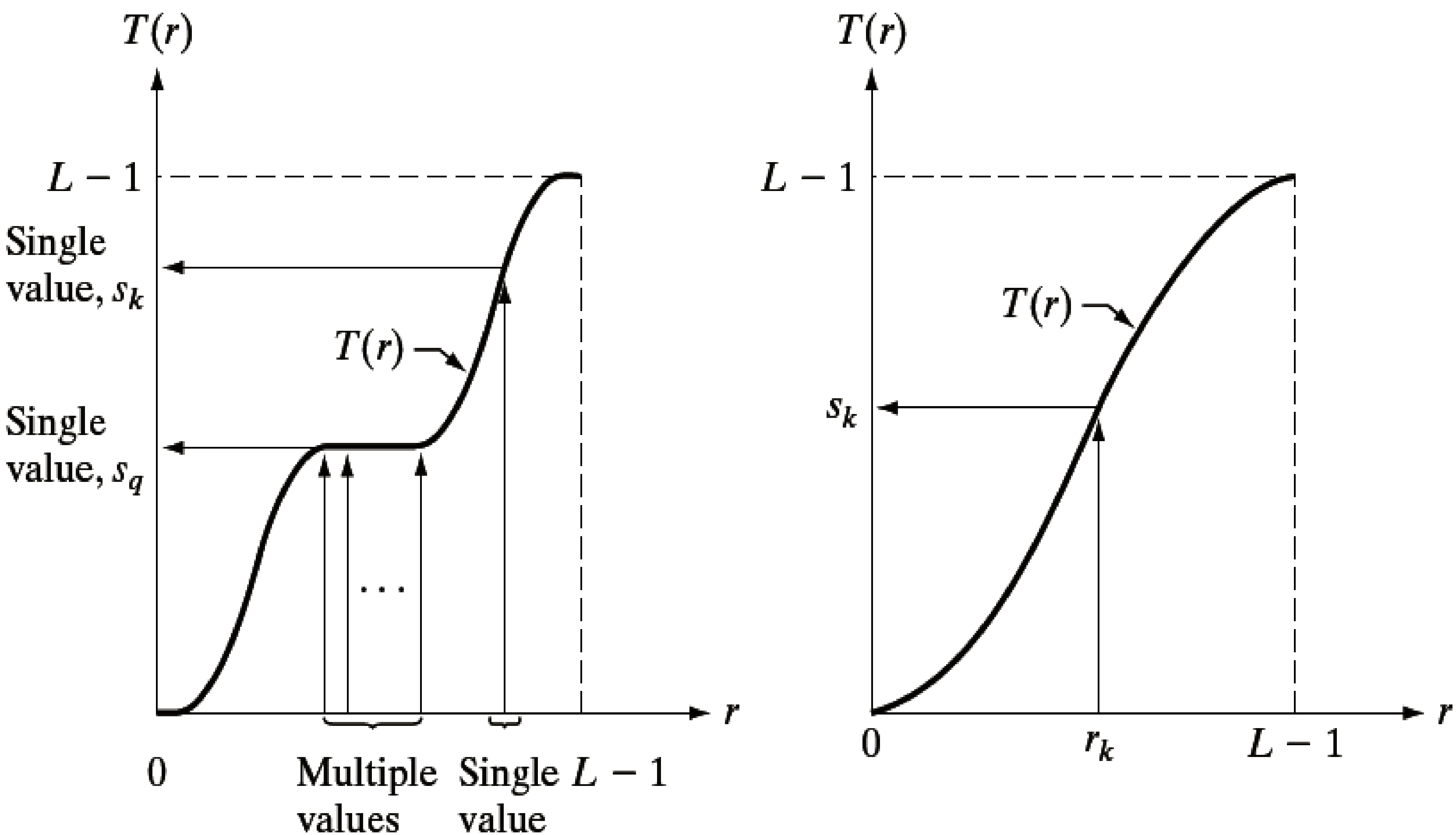
- What does it mean ?
  - Processing an image so that
    the result it more suitable (compared to the original image)
    for a **specific** application
- Evaluation
  - Visual, subjective
  - Quantitative

Gray-Level Transformations

- Design **functions** to transform gray levels within an image by directly operating on them
  - Example
    - Input Image : f (x,y)
    - Transformation : T (.) operates on individual intensities
    - Output Image : g (x,y) = T (f (x,y))
  - Another example
    - Transformation :
      U (.) operates on
      intensities in a neighborhood
    - Output Image :
      g (x,y) = U (f (x,y), f(x+1,y), f(x-1,y))

Gray-Level Transformations

- We’d like transformations to be **monotonic** functions
  - Monotonically increasing function: a > b → T(a) >= T(b)
  - Monotonically decreasing function: a > b → T(a) <= T(b)

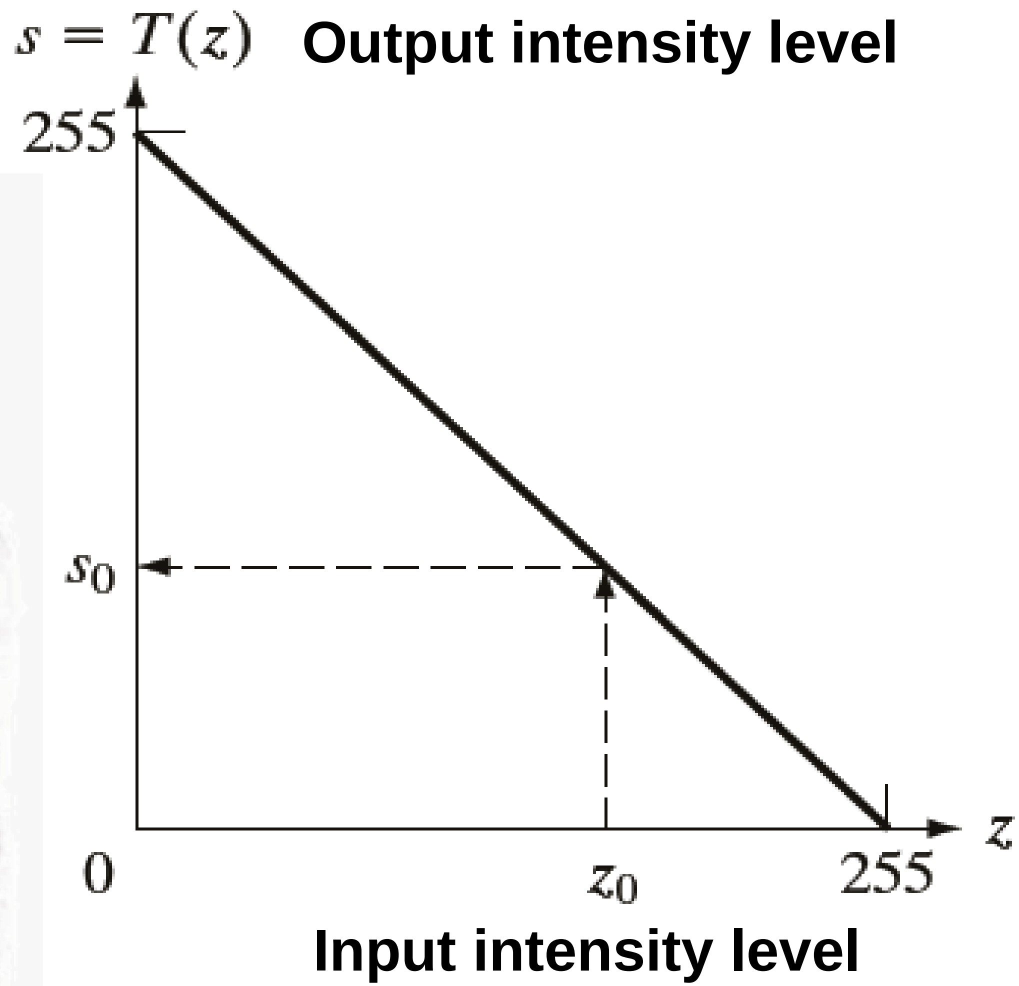


a b

FIGURE 3.17 (a) Monotonically increasing function, showing how multiple values can map to a single value. (b) Strictly monotonically increasing function. This is a one-to-one mapping, both ways.

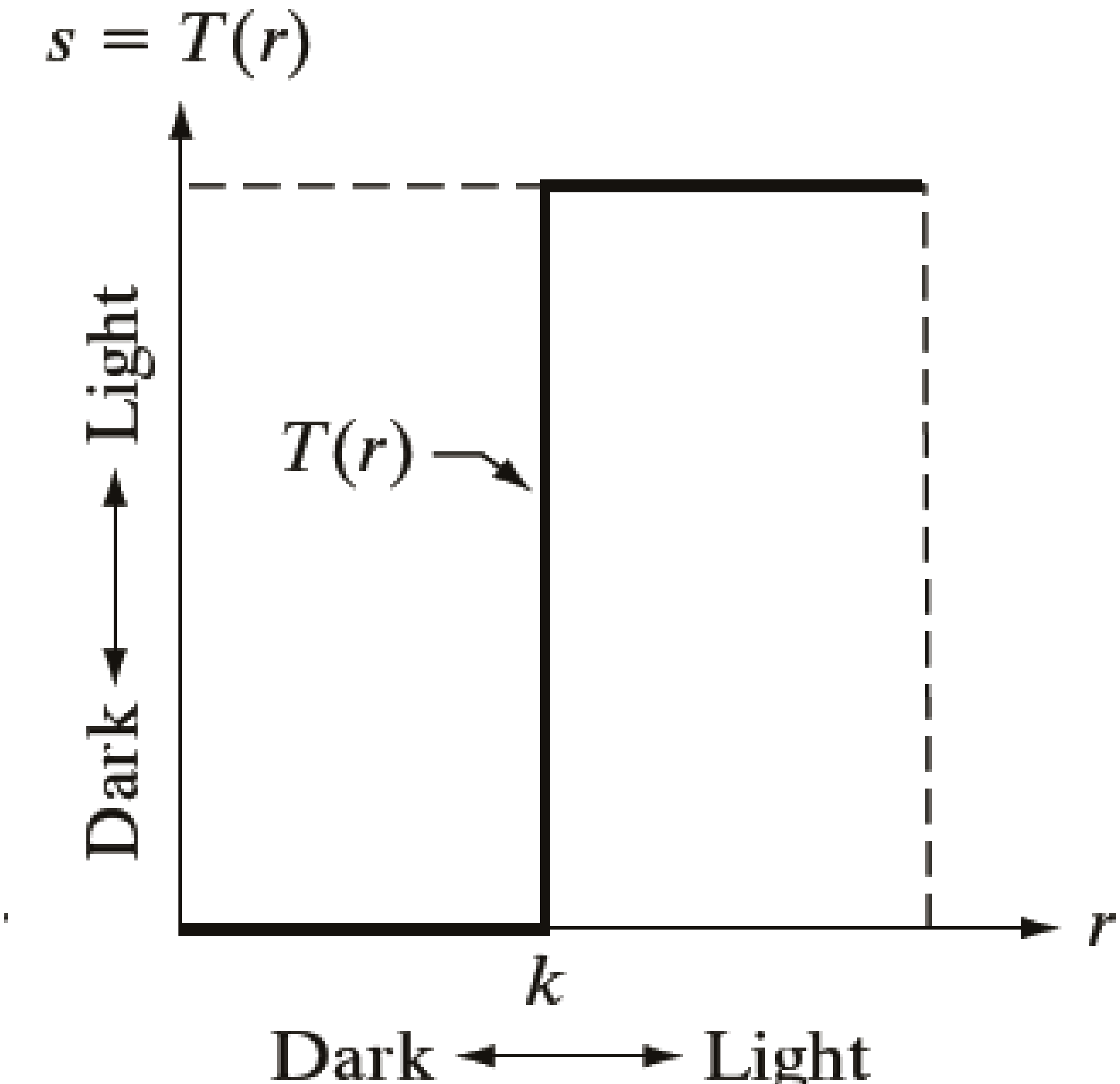
Gray-Level Transformations

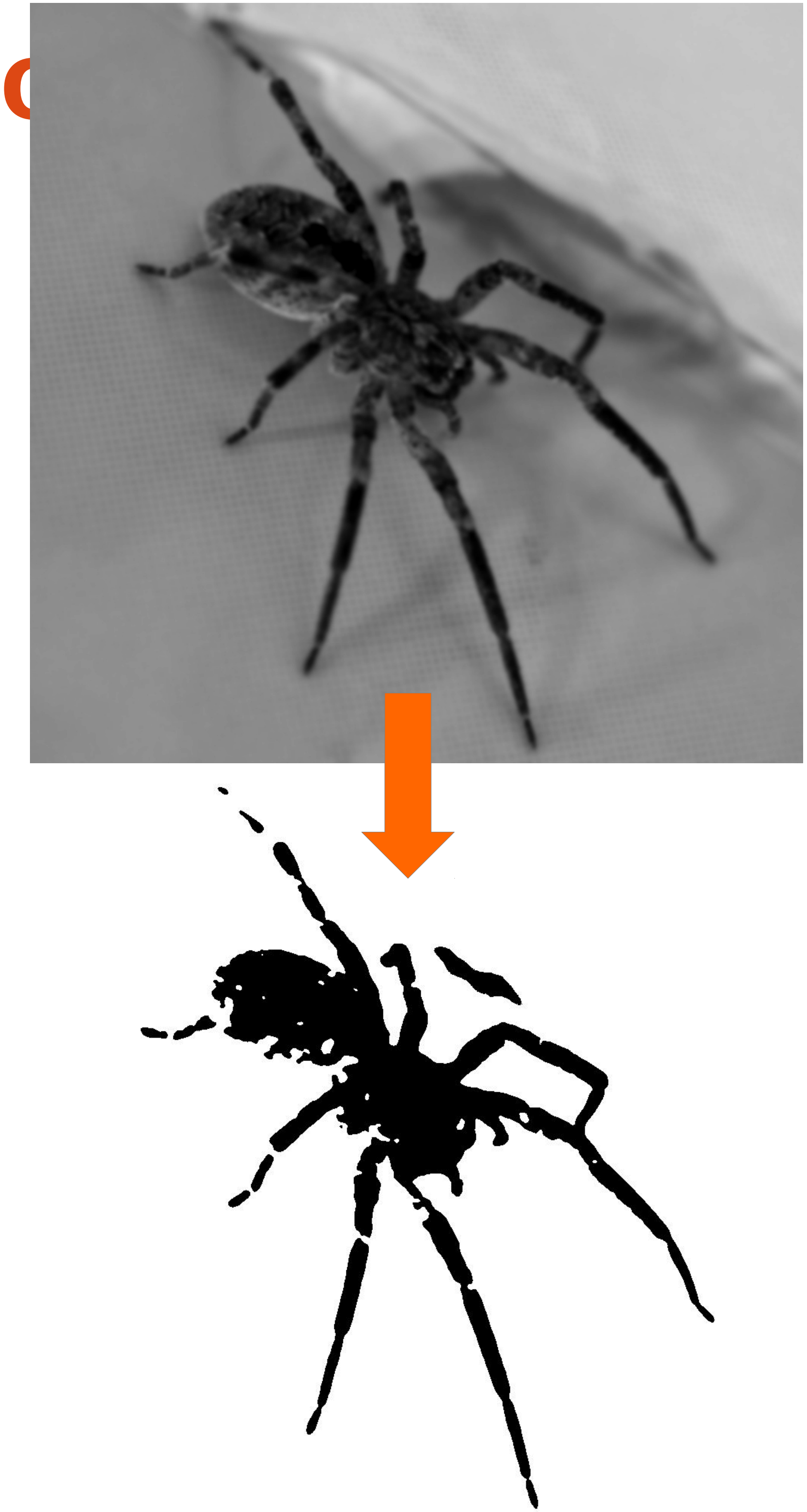
- "Negative"



Gray-Level Transfo

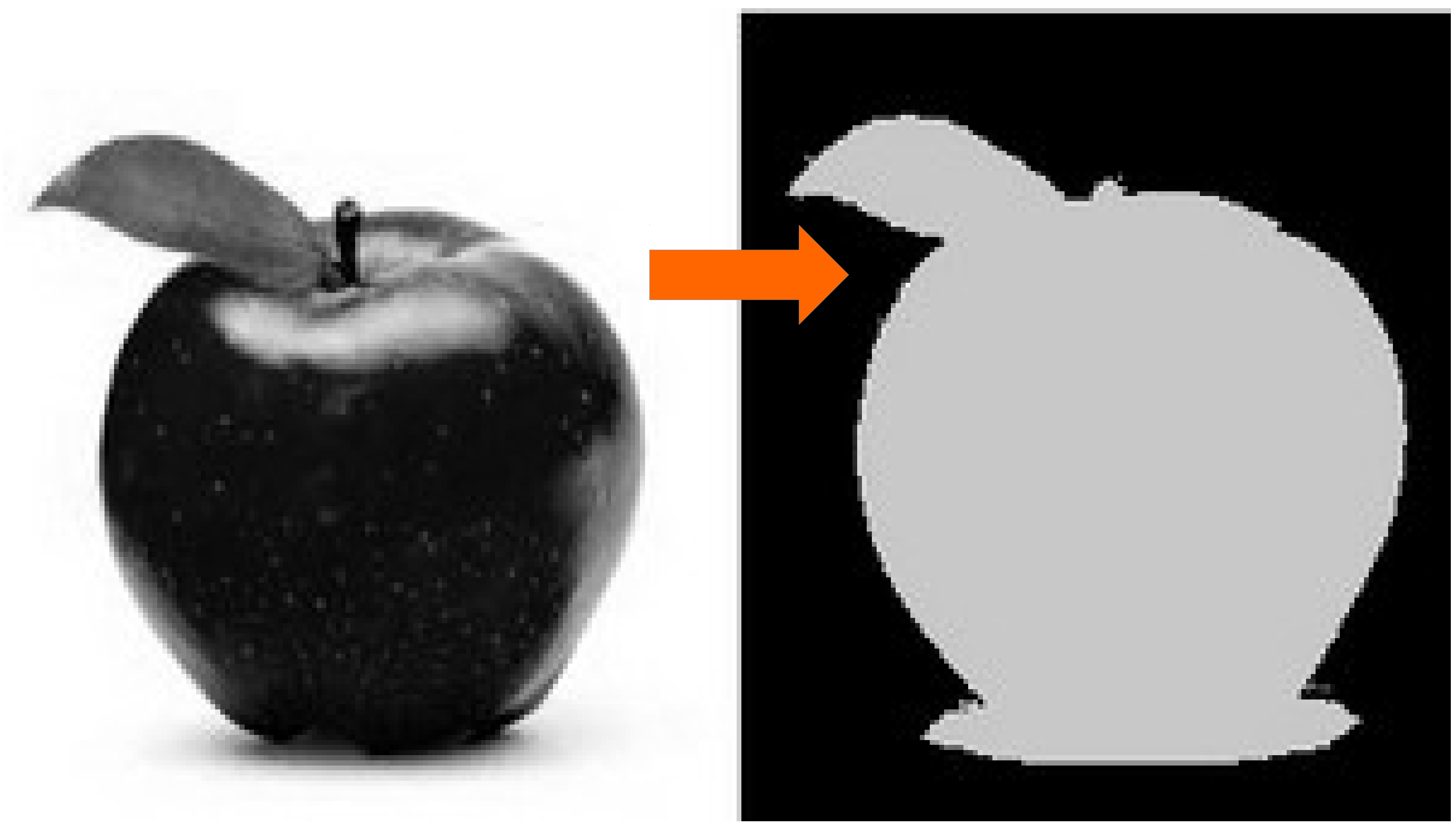
- Thresholding





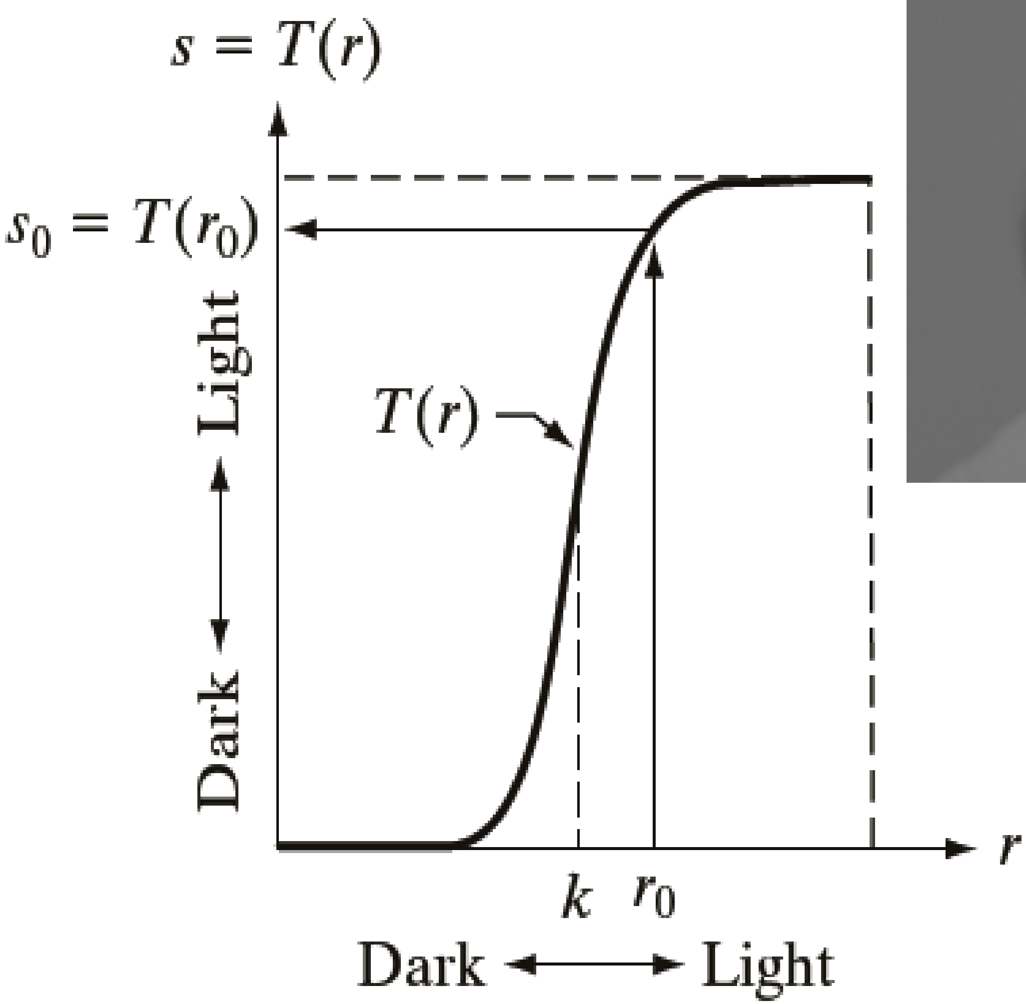
Gray-Level Transformations

- Thresholding
  - What is the transformation function for this ?



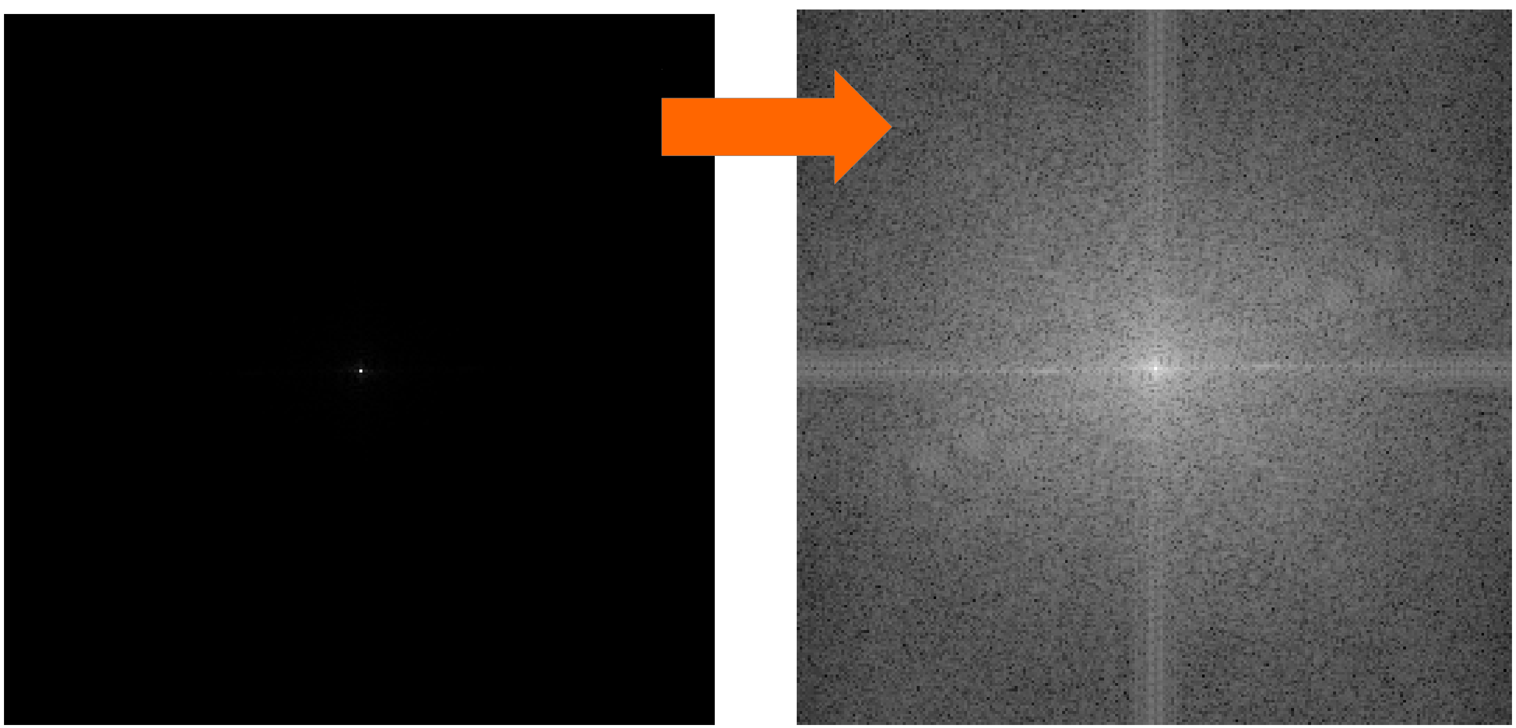
Gray-Level Transformations

- Contrast stretching



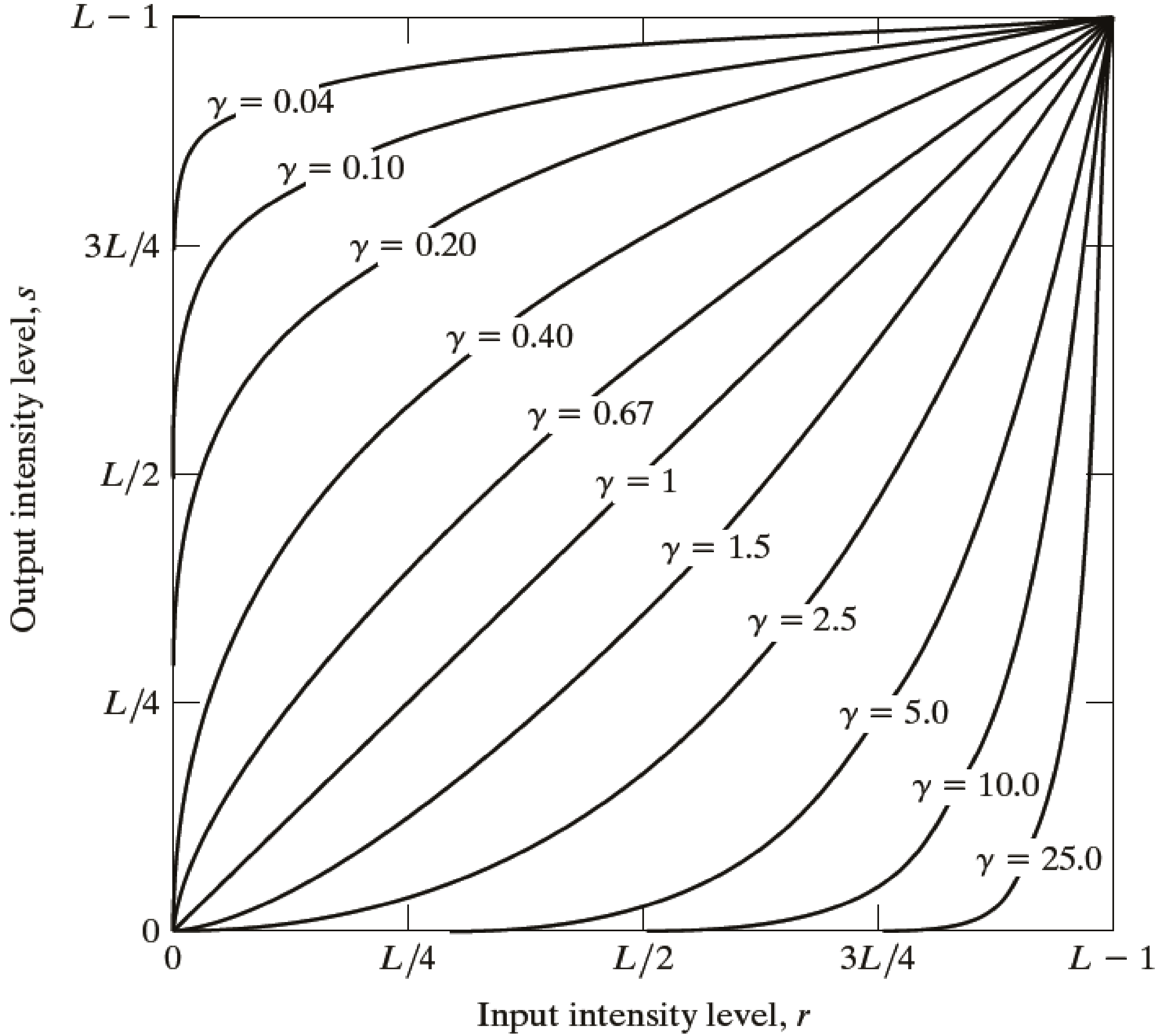
Gray-Level Transformations

- Logarithm
  - $g(x,y) = c \log [\, 1 + f(x,y)\,]$, where c = constant
  - Input image has some values very large
    - Remaining values map to few colors (left) → details lost
  - Log → compress large values, spread small values



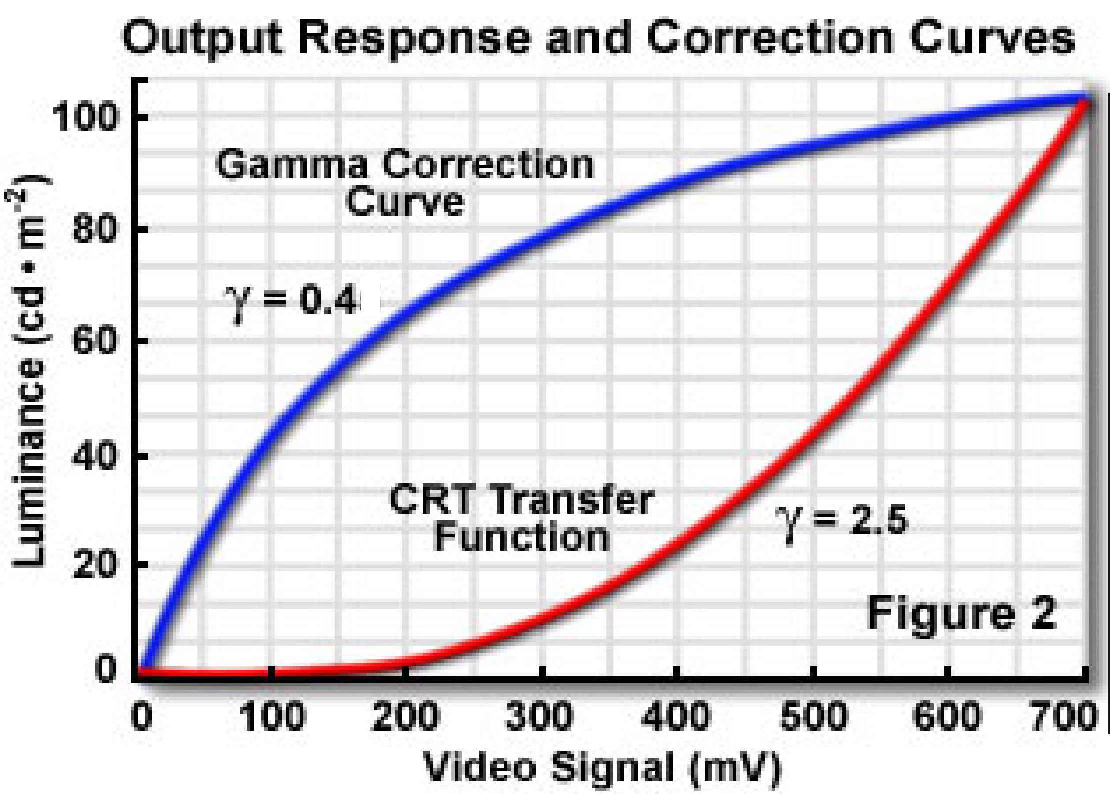
Gray-Level Transformations

- Power and root transformations
  - s(x,y) = d (r(x,y) / c)$^{\gamma}$
  - For graph →
    - r in [0,L-1]
    - c = L-1
    - d = L-1

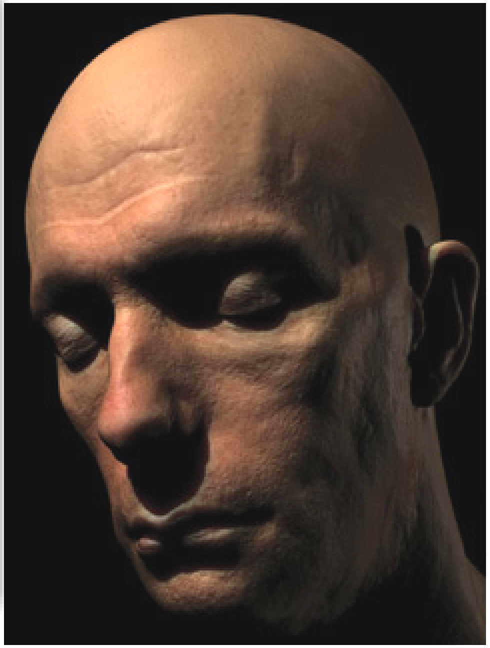


Gray-Level Transformations

- Gamma correction for display monitors
  - Mapping function from
    intensity (data) to voltage (screen appearance)
    is a power function
    - gamma within [1.8, 2.5]
  - Correction applies power transform: gamma' = 1 / gamma



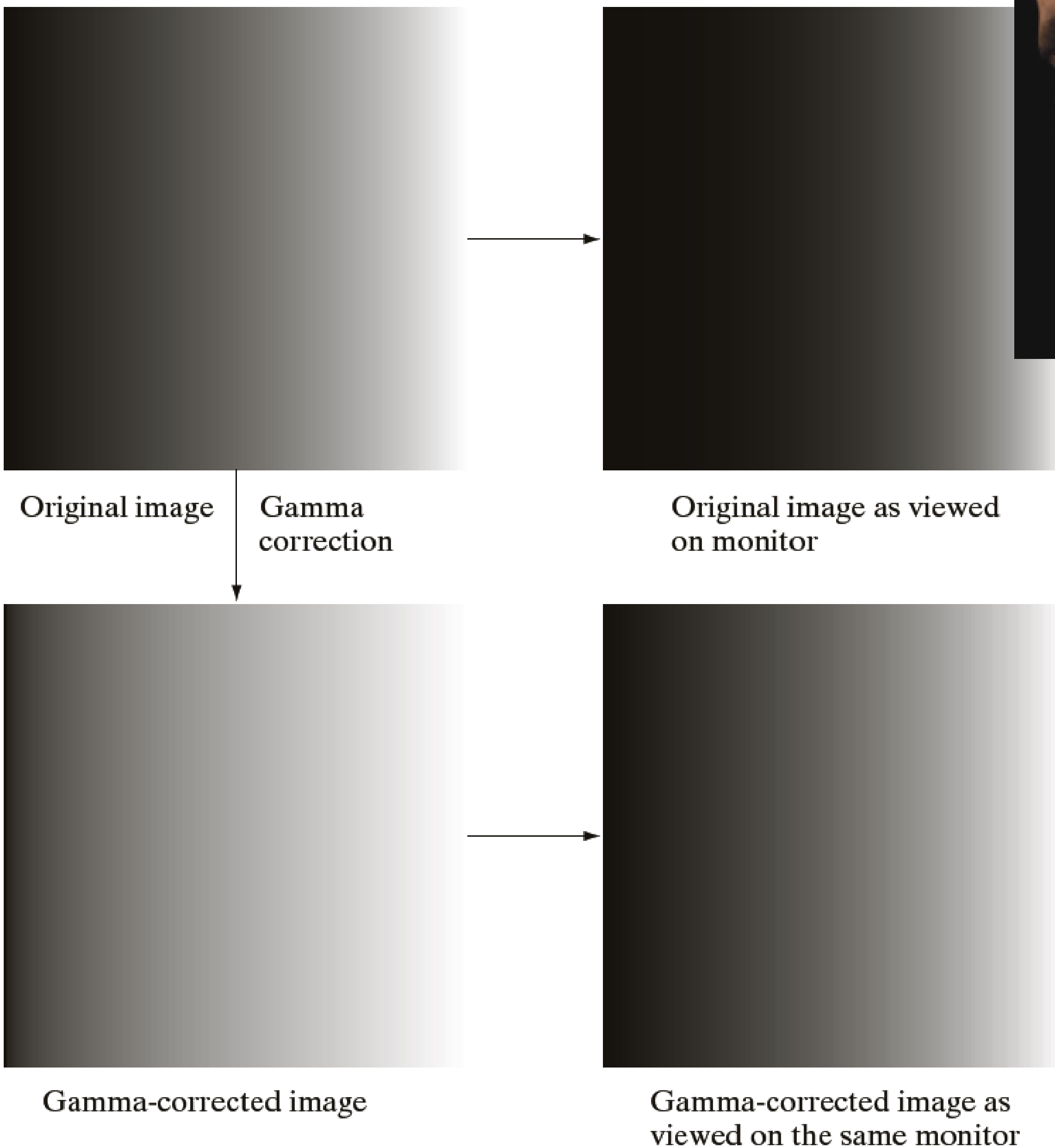
Without Correction



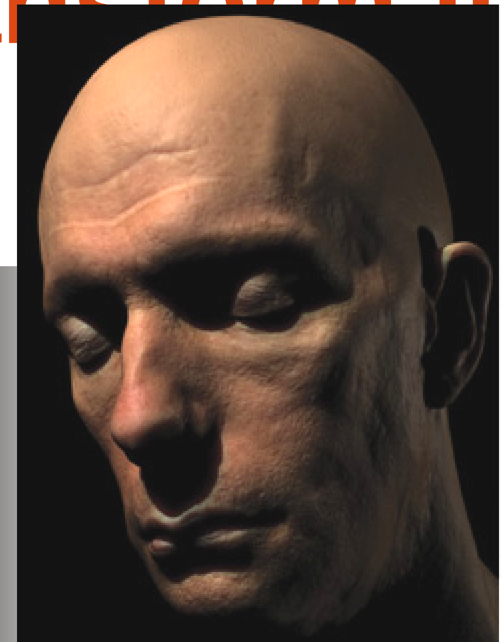
With Correction

Gray-Level Transformations

- Gamma correction



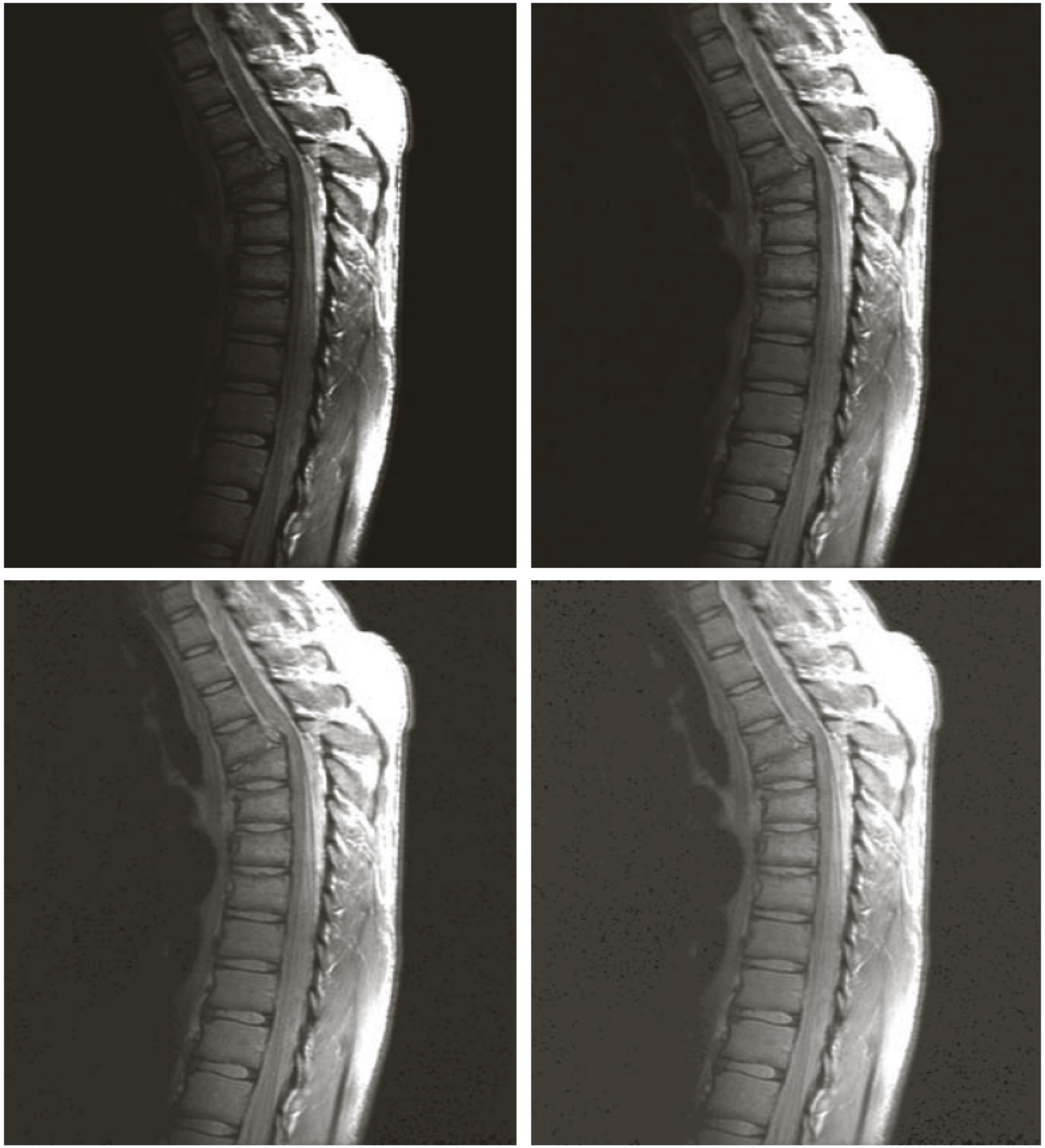
Without Correction



With Correction

Gray-Level Transformations

- Gamma correction in medical imaging (e.g., MRI)
  - gamma =
    1.0 , 0.6
    0.4 , 0.3

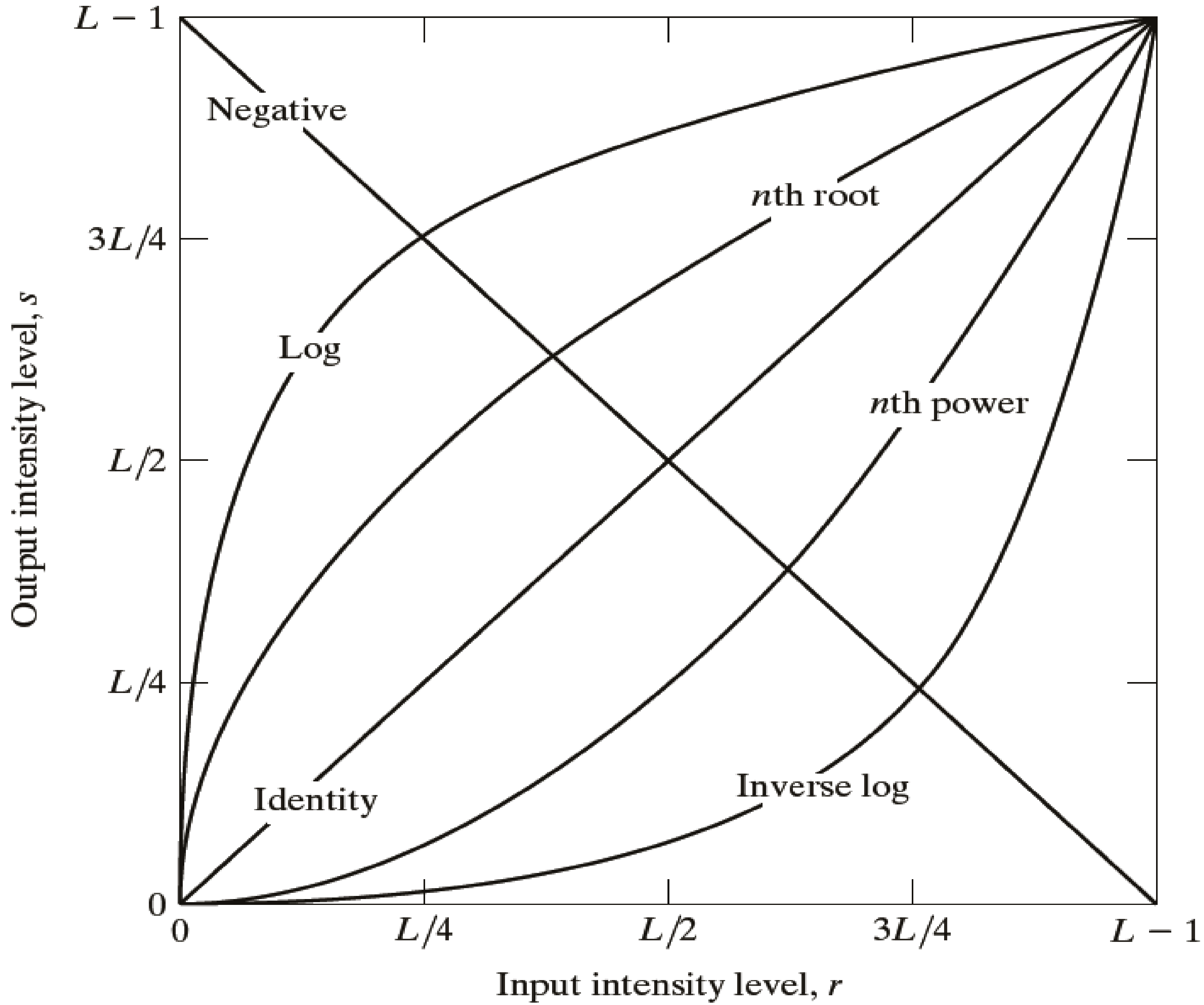


Gray-Level Transformations

- Gamma correction to improve washed-out images
  - Input image = top left
  - For other images, is gamma more than 1 or less than 1 ?

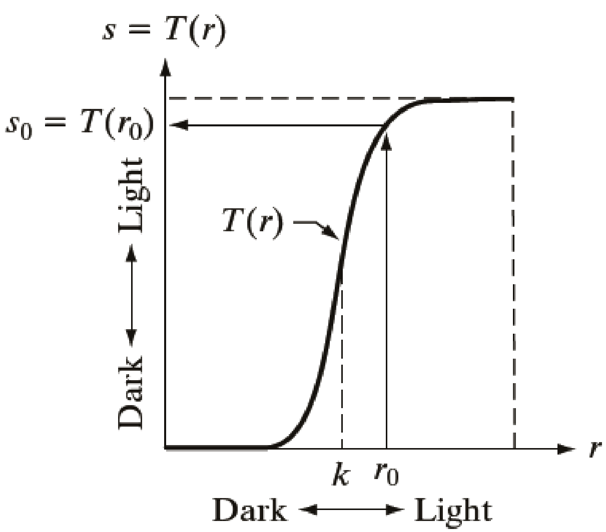
Gray-Level Transformations

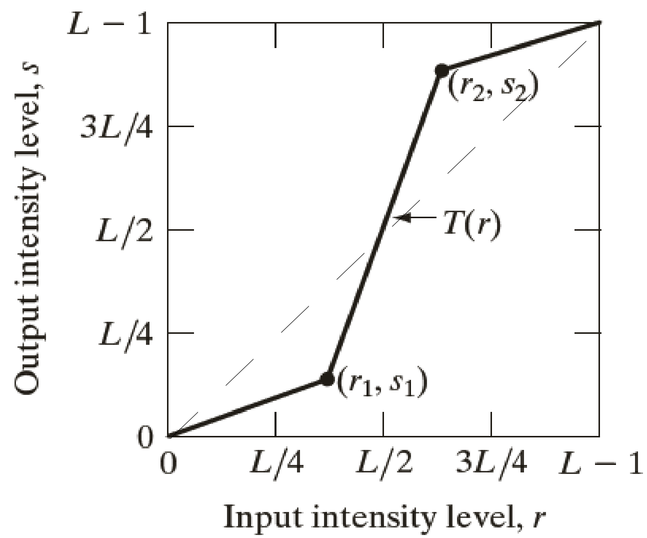
- Typical transformation functions
  - Log, exp
  - Power, root
    - n > 1



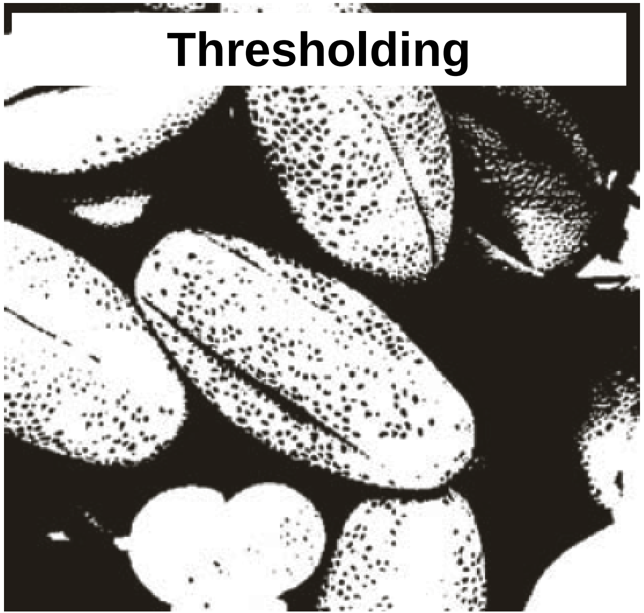
Gray-Level Transformations

- **Piecewise-linear** transformation functions
  - Easier to design than complex nonlinear functions





  - Can model complex transformations
  - Contrast stretching



Gray-Level Transformations

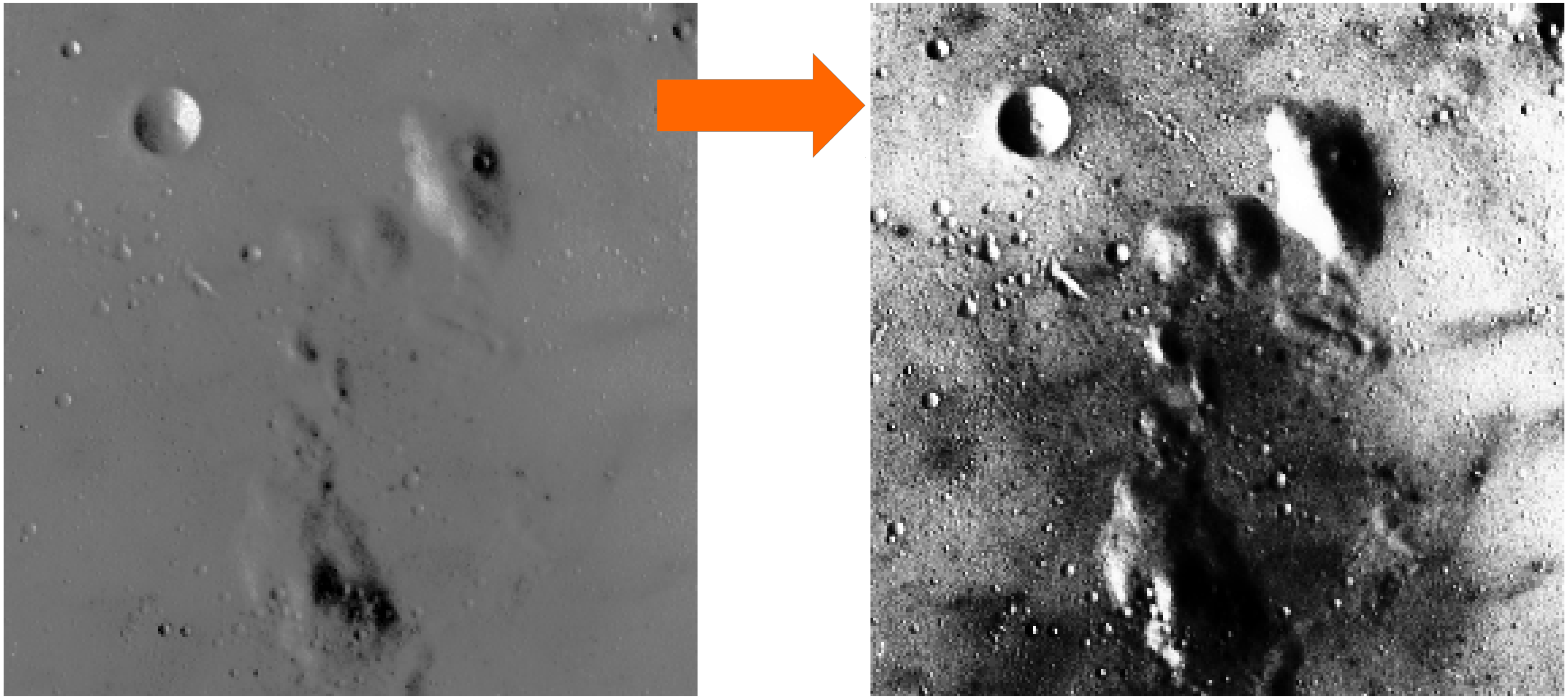
- Histogram processing
  - What is a histogram ?
    - Probability theory refresher
  - What can it achieve ?
    - Contrast enhancement
    - Automatic, without tuning parameters

Gray-Level Transformations

- Histogram processing
  - What can it achieve ?

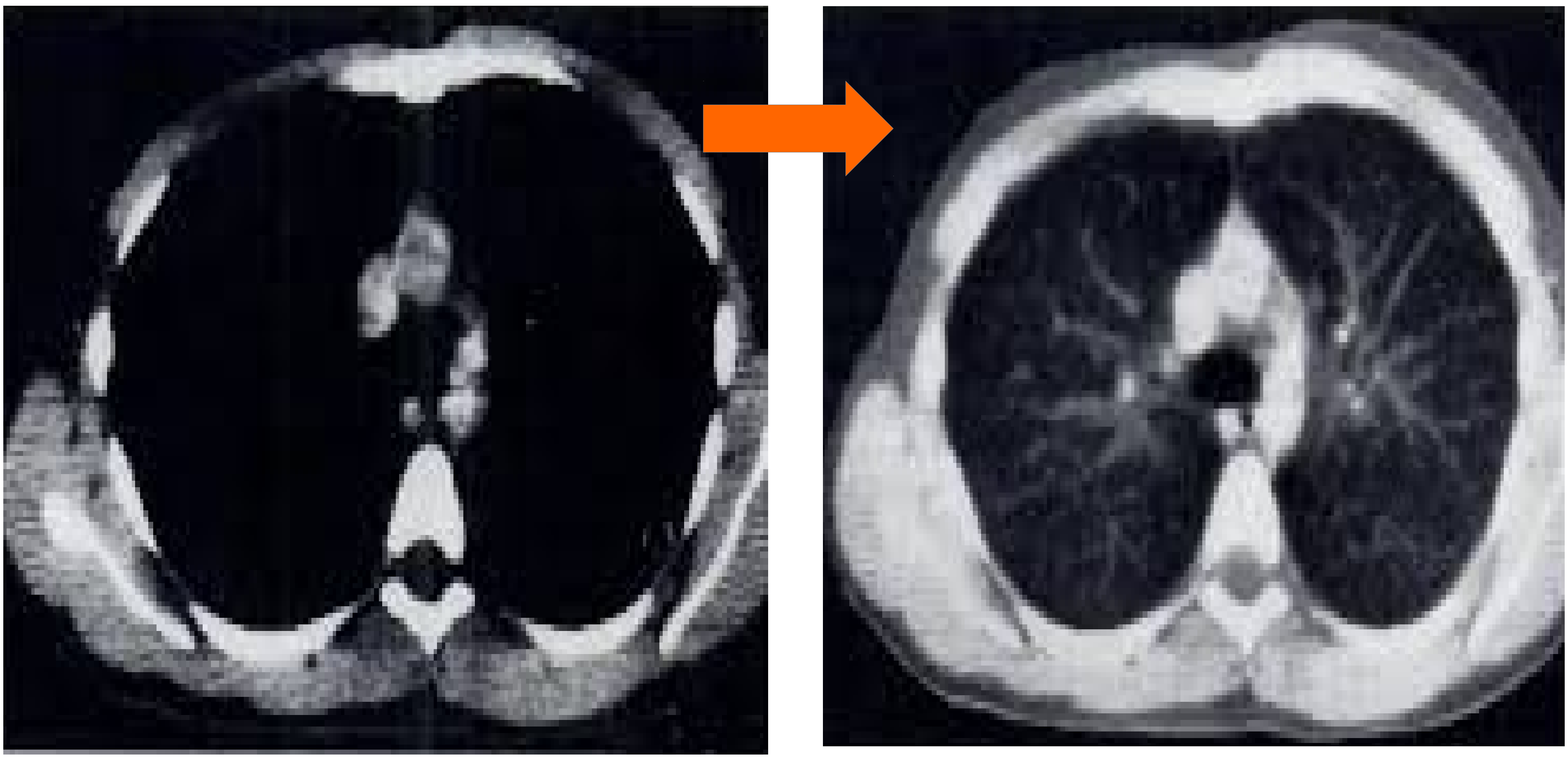
Gray-Level Transformations

- Histogram processing
  - What can it achieve ?



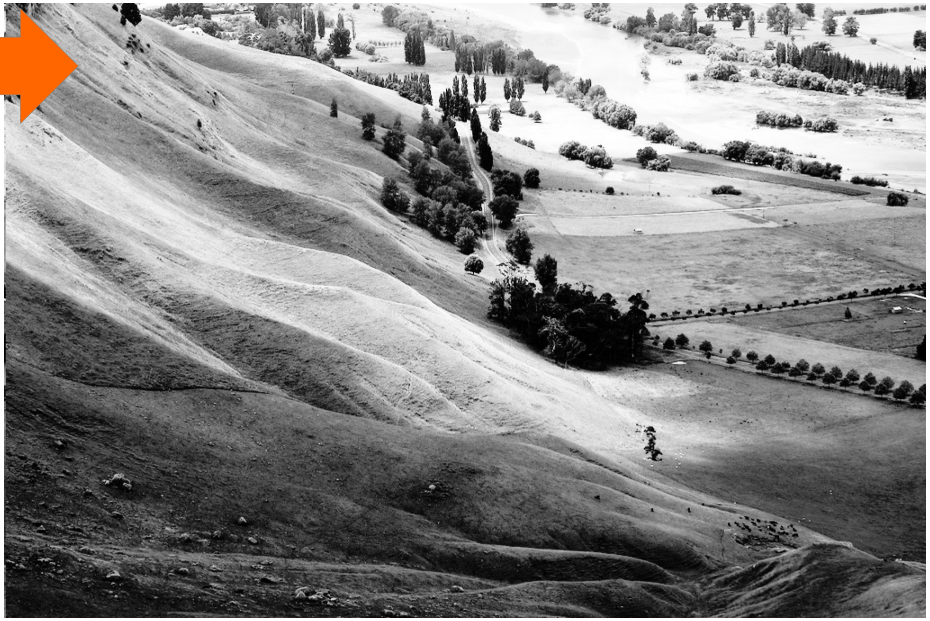
Gray-Level Transformations

- Histogram processing
  - What can it achieve ?

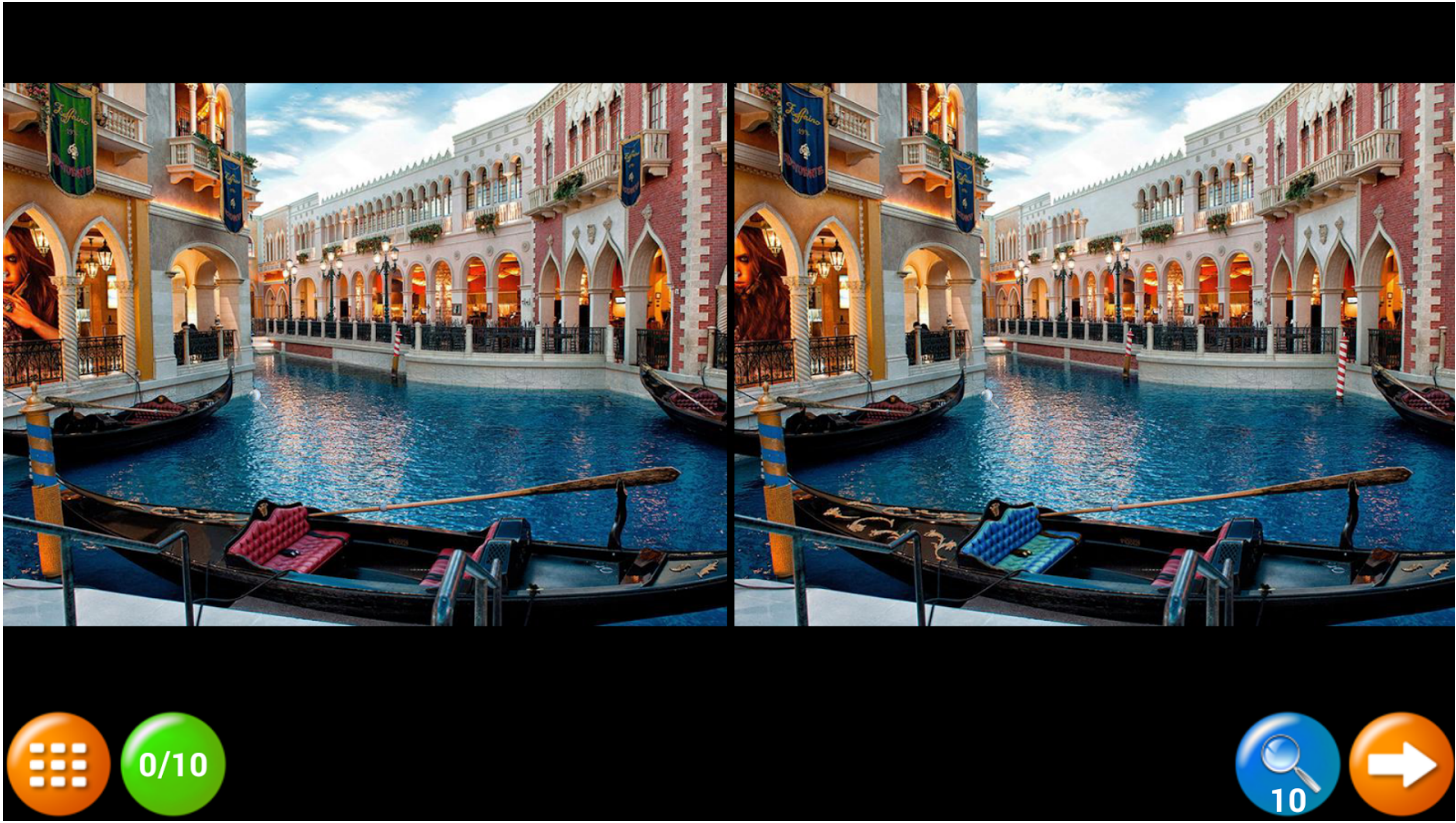


Gray-Level Transformations

- Histogram processing
  - What can it achieve ?

Find All Differences



Gray-Level Transformations

- Histogram processing
  - What is a histogram ?
    - Probability theory refresher (very brief)
      - Probability mass/density function
      - Random variables
      - Random experiments
      - Sample space, events, probability functions
      - Transformation of a random variable

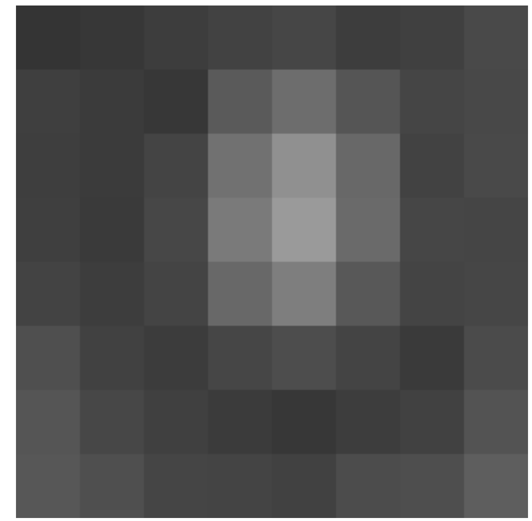
Gray-Level Transformations

- **Random experiment** = an experiment whose outcome isn't certain
  - Flip of a coin
  - Throw of a die

- **Sample space** = set of all possible outcomes of a random experiment
  - Coin flip : { head, tail }
  - Die throw : { 1, 2, 3, 4, 5, 6 }

Gray-Level Transformations

- Random experiment
  - Select a pixel location in an image
- Sample space
  - Set of all pixel locations
    { (1,1), (1,2), …, (1,8),
      (2,1), (2,2), …, (2,8),
      …
      (8,1), (8,2), …, (8,8) }



$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

Gray-Level Transformations

- **Event** is a subset of a sample space
  - Event of getting a head on a coin toss = { head }
  - Event of getting an even number on a die roll = ? = { 2, 4, 6 }
  - Event of getting a pixel in the left half of the image

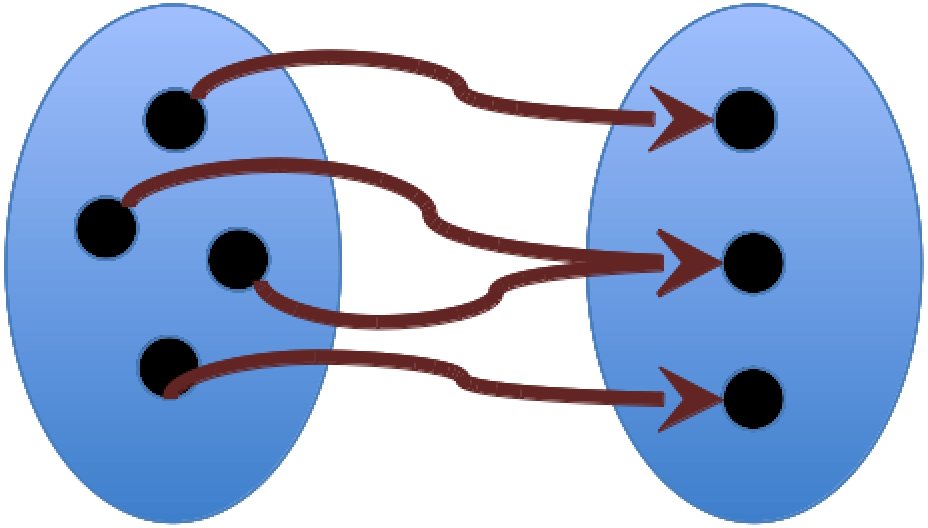
- **Event space** = set of all possible events

Gray-Level Transformations

- **Probability function P(.)**
  - A probability function on **sample space Ω assigns every event A** in Ω a **number** in [0,1] s.t.
    - $P(\Omega) = 1$
    - $P(A \cup B) = P(A) + P(B)$ when $A \cap B = 0$
  - P(A) is the **probability** that **event** A **occurs**
  - Examples
    - Probability function for coin toss = ?
    - Probability function for die roll = ?

Gray-Level Transformations

- **Random variable** = a function
  - $X : \Omega \rightarrow R$
  - Domain = sample space
  - Range = set of real numbers



Gray-Level Transformations

- Random variable is an **abstraction**
  - We don't care about outcomes (directly)
  - We only care about values mapped to outcomes
  - Die example
    - We don't care about die value, just even or odd
  - Image example
    - We don't care about pixel location, just pixel intensity

Gray-Level Transformations

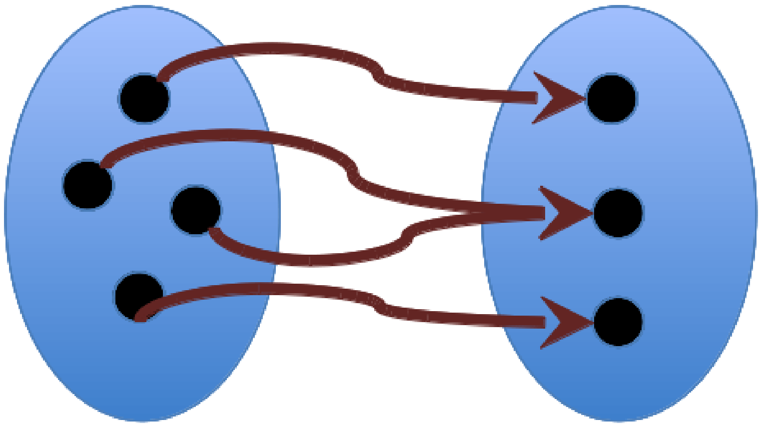
- Defining events using random variables
  - Example: Sum of a pair of dice
  - Sample Space: Ω = { (i,j) : 1 <= i,j <= 6 }
  - Probability function: P ((i,j)) = 1/36
  - Random variable: X ((i,j)) = i + j
  - Event { X=a } = { w \in Ω : X(w)=a }
    - P (X = 5)
      = P ({ (1,4), (2,3), (3,2), (4,1) })
      = P ((1,4)) + P ((2,3)) + P ((3,2)) + P ((4,1))
      = 4 / 36

Gray-Level Transformations

- Defining events using random variables
  - Example:
    Intensity at a randomly-chosen pixel in 8x8 image f(x,y)
  - Sample Space: $\Omega = \{ (x,y) : 1 <= x,y <= 8 \}$
  - Probability function: $P ((x,y)) = 1/64$
  - Random variable: $Z ((x,y)) = f (x,y)$
  - Event { Z > 100 } = { w \in Ω : Z(w) > 100 }

Gray-Level Transformations

- **Discrete** random variable (RV)
  - Maps outcomes to values in a countable set
  - **Probability mass function** $p : R \rightarrow [0,1]$
    - Probability that the RV takes a value 'a'
    - $p(a) = p(X=a) = \sum_{w : X(w) = a} P(w)$
  - **Cumulative distribution function** $F : R \rightarrow [0,1]$
    - Probability that the RV takes a value **less than** x
    - $F(a) = p(X \le a) = \sum_{w : X(w) \le a} P(w)$
    - $F(a) = p(X \le a) = \sum_{b : b \le a} p(X=b)$

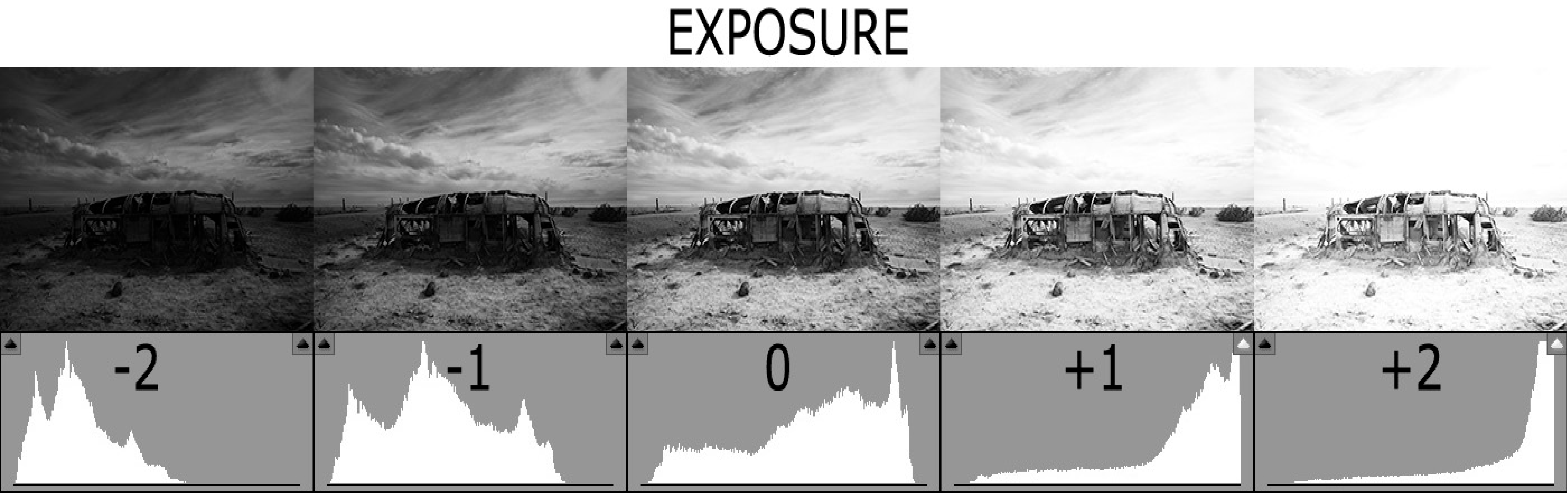


Gray-Level Transformations

- **Continuous** random variable (RV)
  - Maps outcomes to values in an uncountable set
    - Sample space itself is uncountable
      - e.g., continuous image function X(.) defined on [0,1]x[0,1]
  - **Cumulative distribution function** $F : R \rightarrow [0,1]$
    - Probability that the RV takes a value less than c
      - $F(c) = p(x < c) = \int_{w\,:\,X(w) <= c} P(w)\, dw$
      - $F(c) = p(x < c) = \int_{x<c} p(x)\, dx$
  - **Probability density function** $p : R \rightarrow R$
    - p (.) is derivative of F (.)
    - p (x) is NOT probability of event 'x' (p(x) can be > 1)
    - Events, of interest, are subsets of the form { x : a < x < b }
    - $p(a < x < b) = \int_{a<x<b} p(x)\, dx$

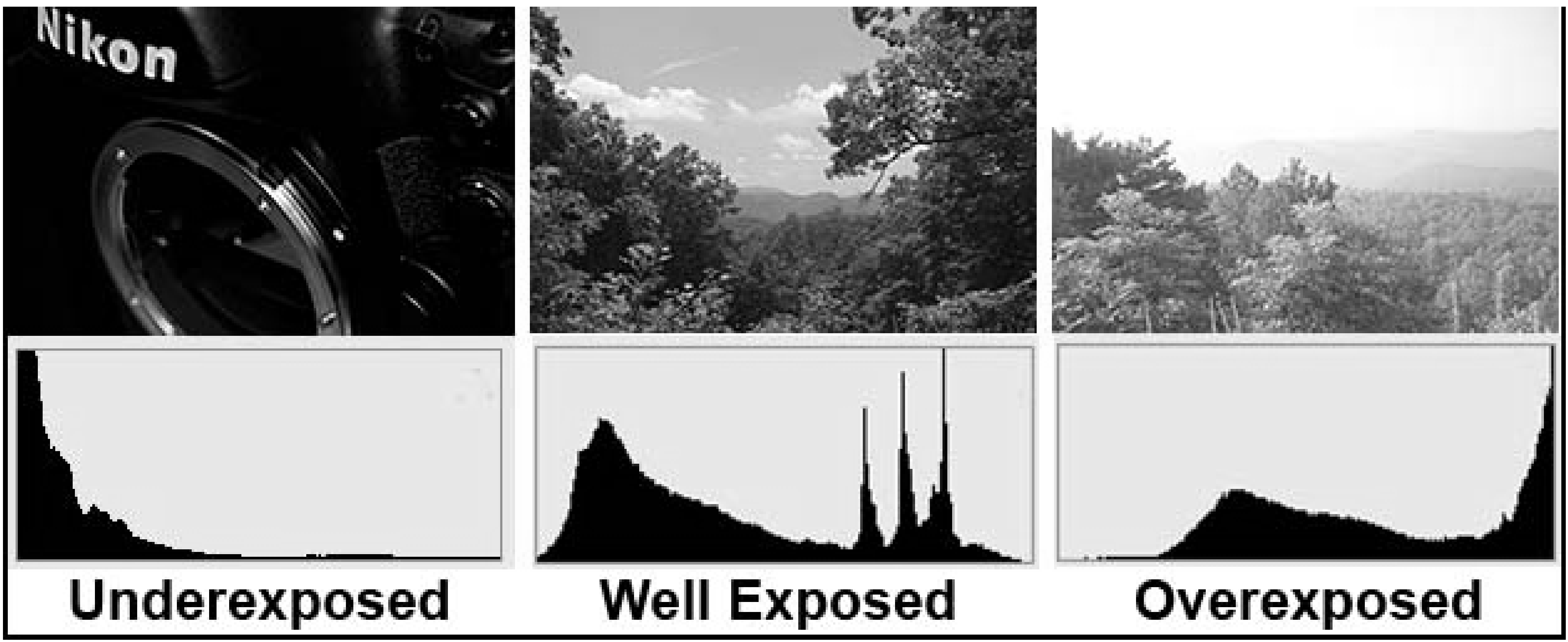
Gray-Level Transformations

- Histograms are related to photograph exposure
  - Aperture size (amount of light per unit time)
  - Shutter speed (time of exposure)



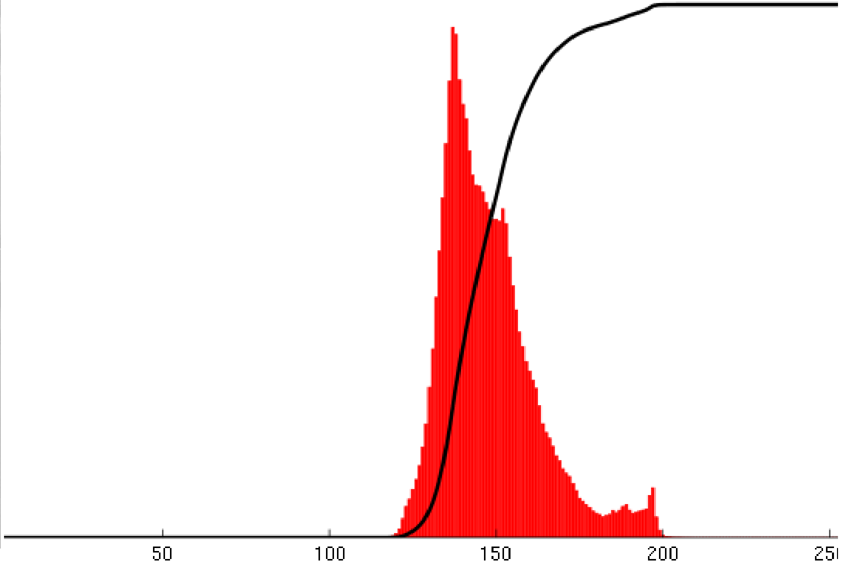
Gray-Level Transformations

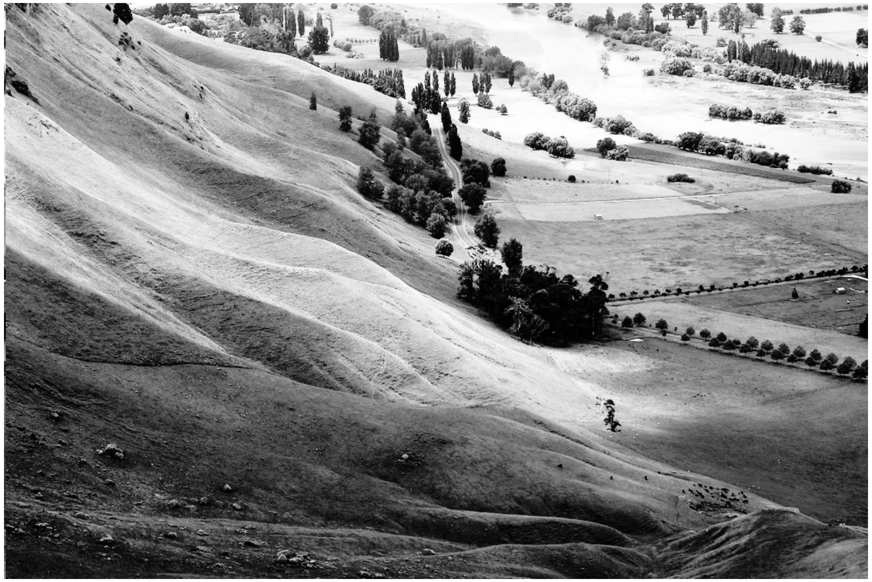
- Histograms are related to photograph exposure
  - Aperture size
  - Shutter speed



Gray-Level Transformations

- Histogram equalization : Motivation



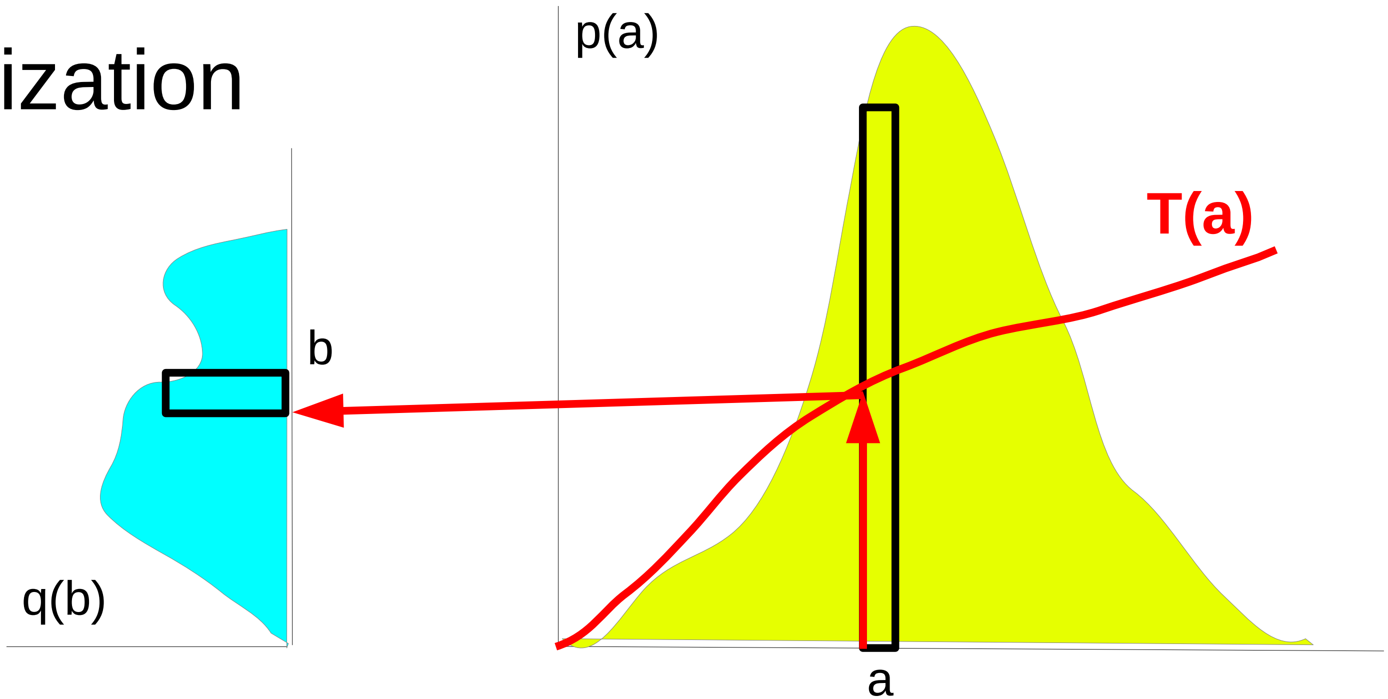


Gray-Level Transformations

- Histogram equalization
  - Assume **continuous** distributions (and image)
  - Given histogram p (a)
  - **Design function T (a) = b**
such that
q (b) = uniform distribution
  - Assume $0 < a < 1$, $0 < b < 1$
  - Then, we want $q(b) = 1$
  - Consider a **monotonically increasing** intensity-transformation function T(.)

Gray-Level Transformations

- Histogram equalization



  - Let b = T(a)
  - **Mass conservation** implies: p(a) da = q(b) db
  - Integrate left hand side from 0 to a'
  - Integrate right hand side from 0 = T (a') = b'
  - Left hand side = CDF_a (a')
  - Right hand side = CDF_b (b')

Gray-Level Transformations

- Histogram equalization
  - If we want q (b) = 1, then CDF_b (b') = b'
  - So, CDF_a (a') = CDF_b (b') = b'
  - So, transformation function T (.) =
    CDF of intensities in original image CDF_a (.)

Gray-Level Transformations

- Histogram equalization example (3-bit image)

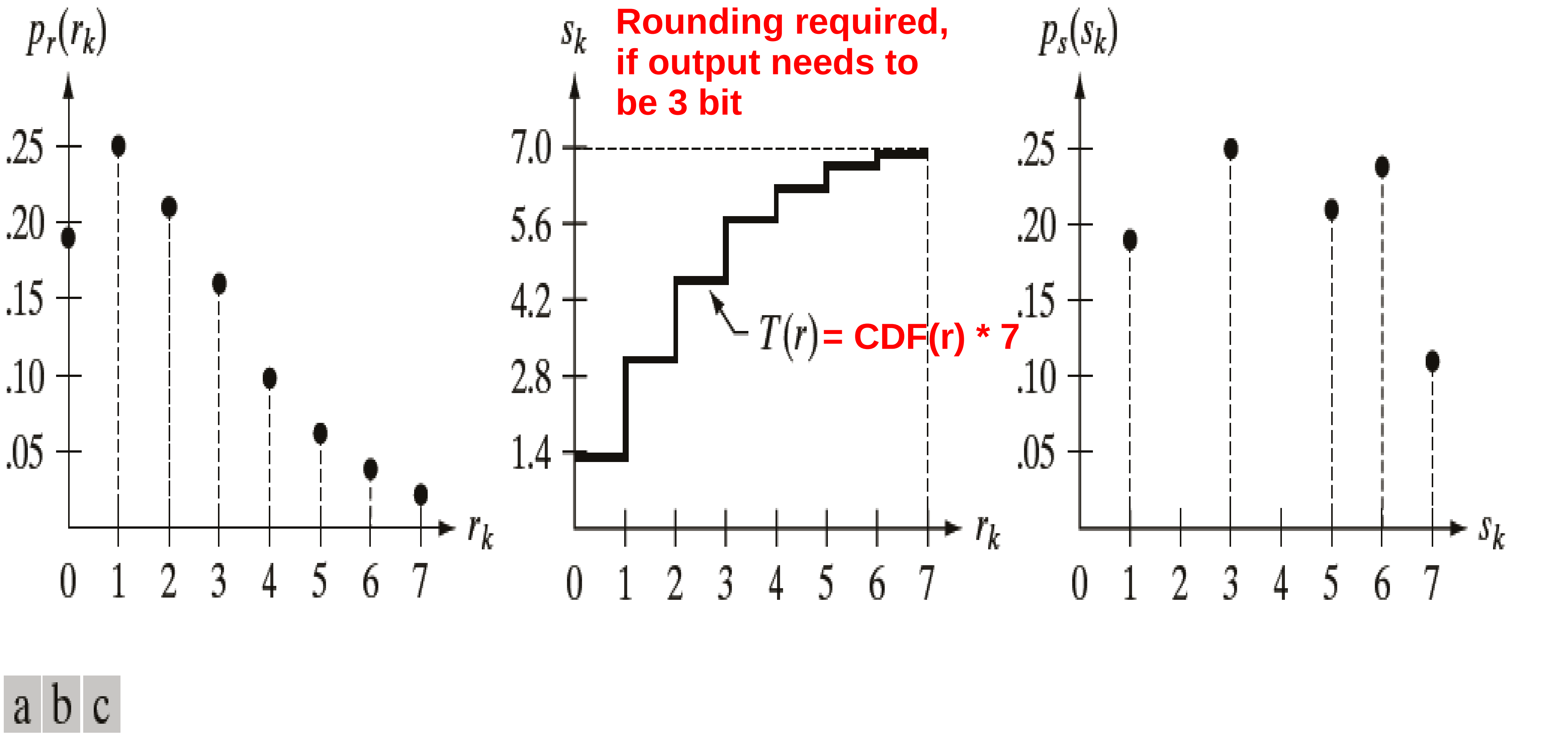
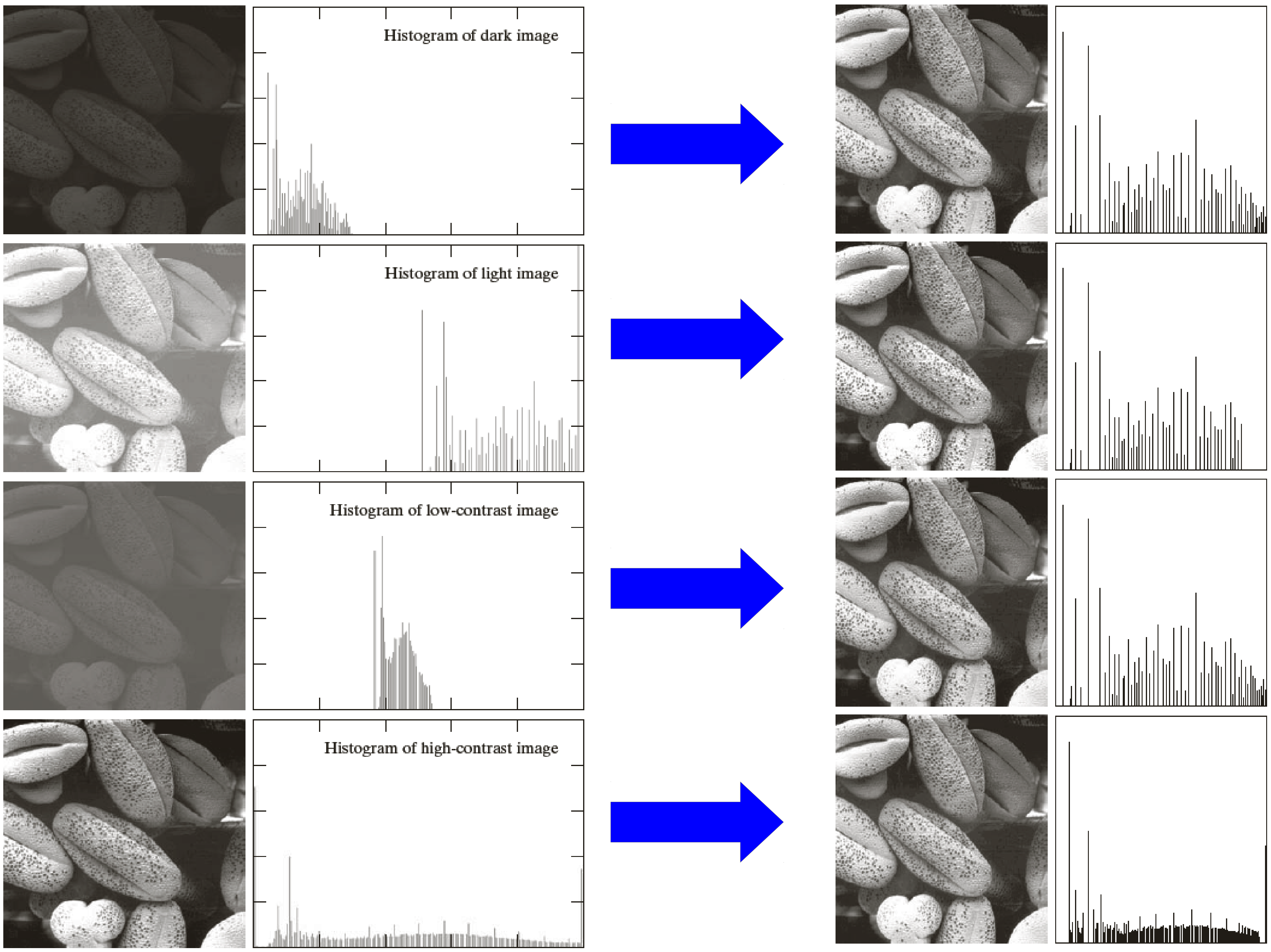
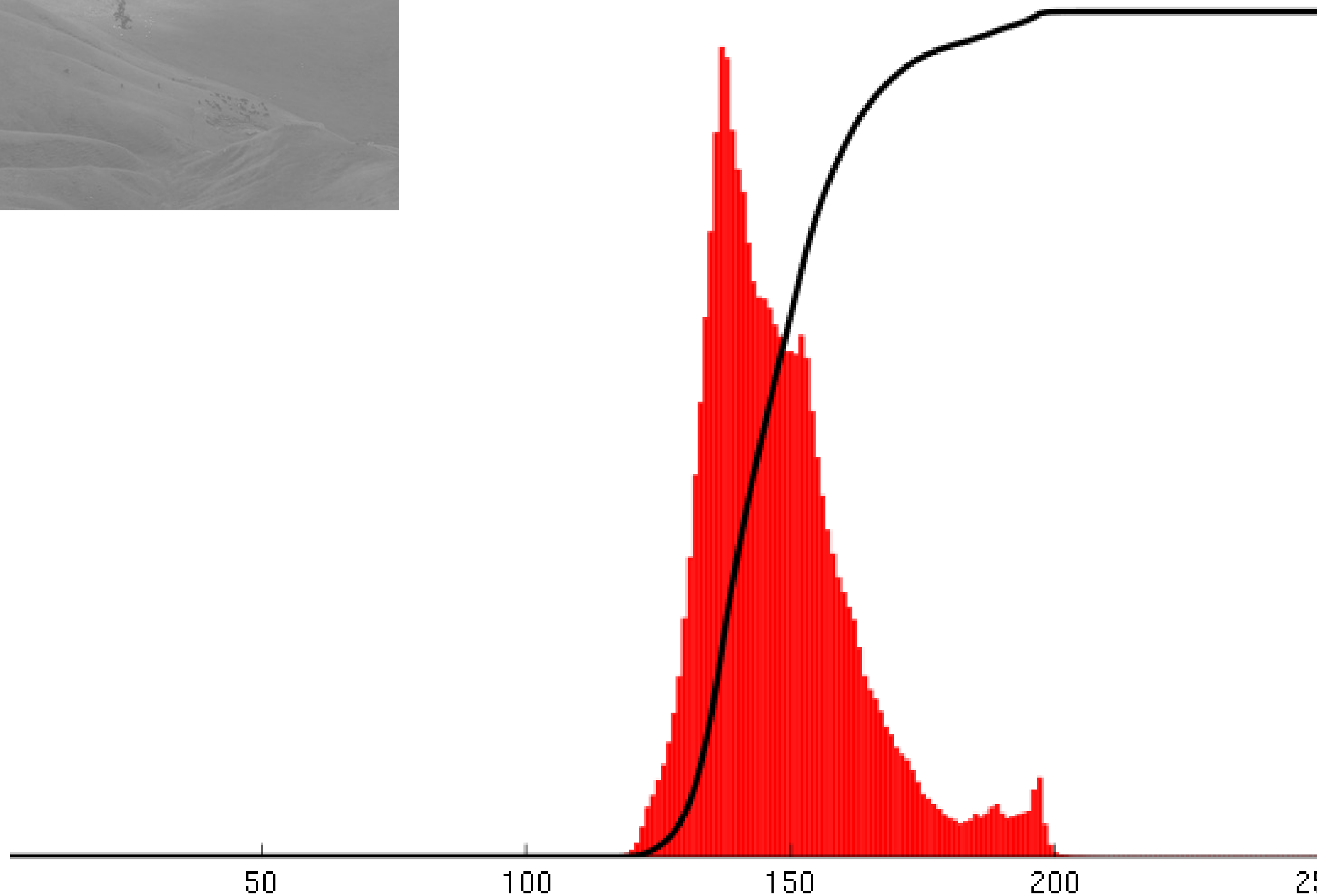


FIGURE 3.19 Illustration of histogram equalization of a 3-bit (8 intensity levels) image. (a) Original histogram. (b) Transformation function. (c) Equalized histogram.

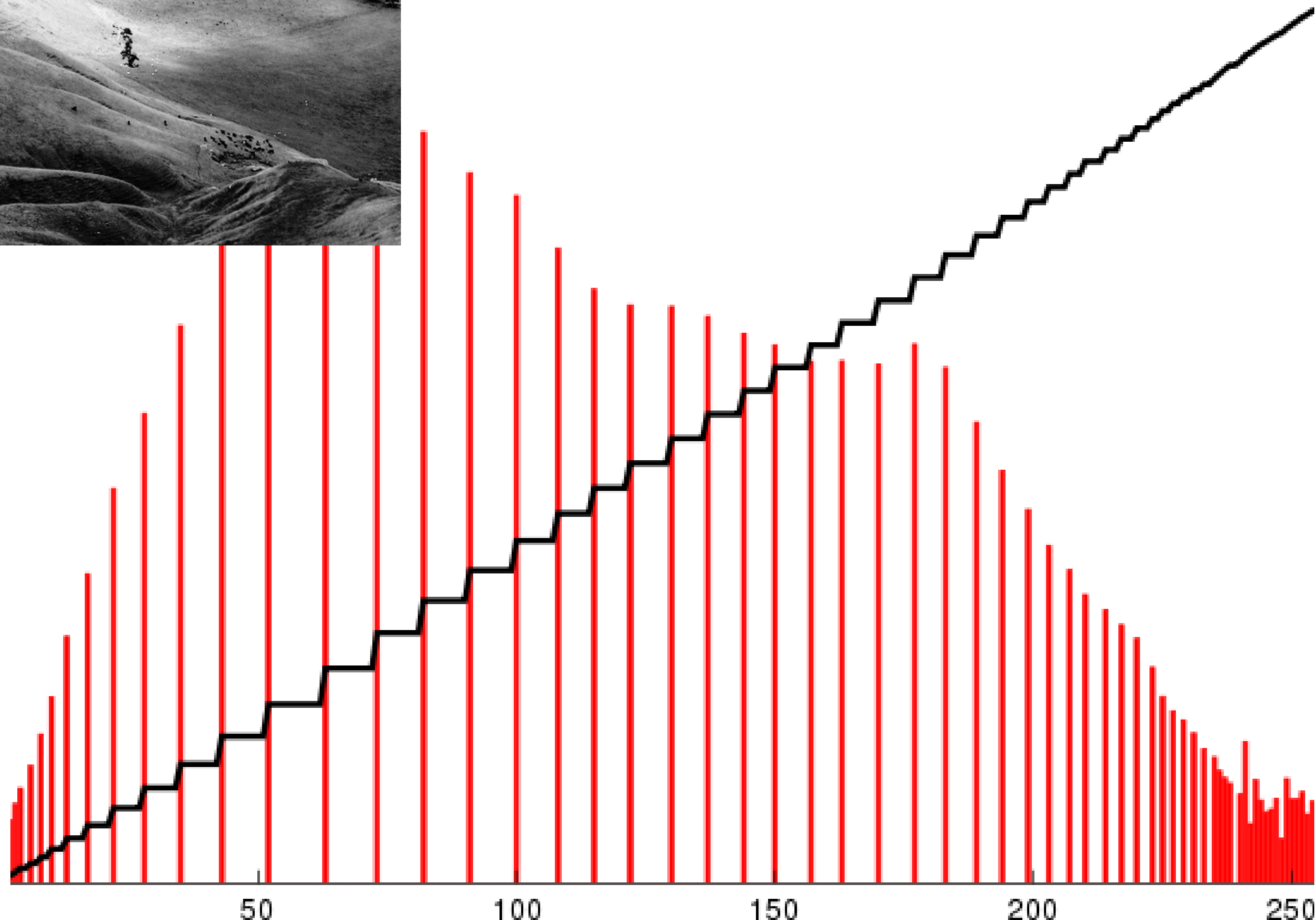


Histogram of dark image
Histogram of light image
Histogram of low-contrast image
Histogram of high-contrast image

ansformations



ansformations



Gray-Level Transformations

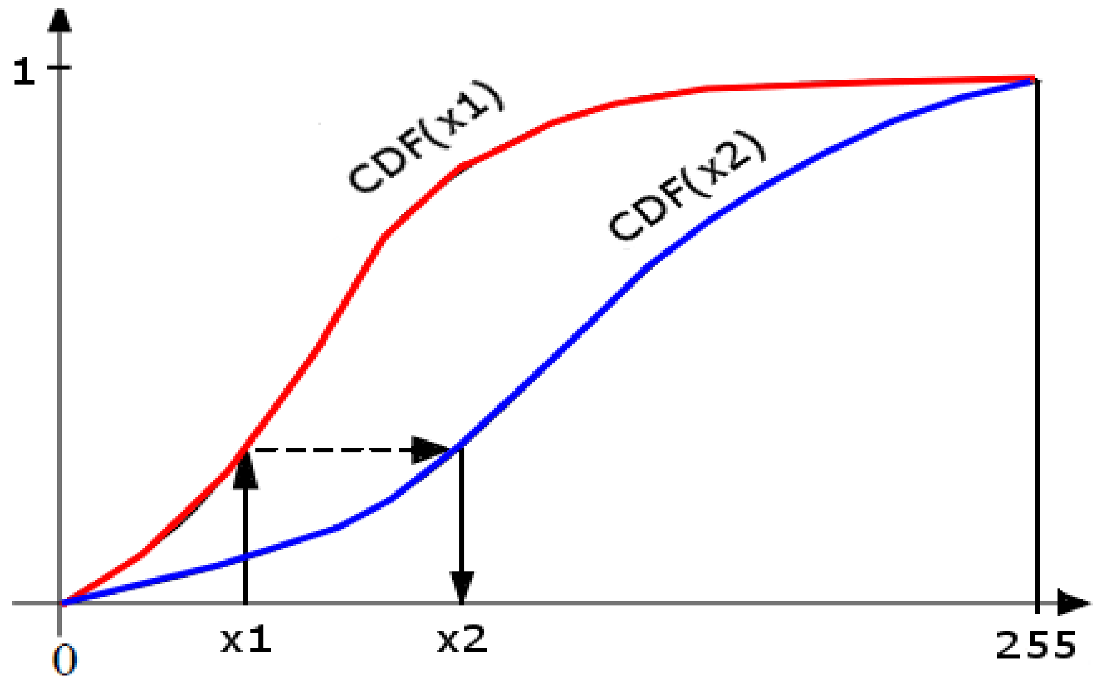
- Why is the output histogram NOT exactly uniform ?
  - Because of discretization of space and intensity
  - Theoretically
    - Contradicts the assumption of continuous distributions (i.e., continuous spatial domain, continuous intensities)
    - We used that to derive transformation for histogram equalization
  - Practical problem (1 of 2)
    - Imagine 8-bit images → 256 intensities (B=256 bins)
    - Imagine an input image containing only N=100 unique intensities (unique upto bin-width)
    - For output image, uniform histogram must contain **some** pixels for each intensity (bin)
    - How can we map N=100 values to B=256 bins ?
      - No systematic straightforward way to map N unique values to B (> N) bins

Gray-Level Transformations

- Why is the output histogram NOT exactly uniform ?
  - Because the set of intensities in digital images is discrete
  - Practical problem (2 of 2)
    - Imagine 8-bit images → 256 intensities (B = 256 bins)
    - Imagine an input image containing N=256 unique intensities (some mass exists in each bin) and M=2560 pixels
    - For output image, uniform histogram must contain **an equal number of (M/N=10)** pixels for each intensity
    - If some intensity “a” in input image has 20 pixels, then intensity b=T(a) in output image will have >= 20 pixels
    - How can we decide which 10 of those >=20 pixels should be assigned intensity “b” ?
      - No systematic and straightforward way to do that

Gray-Level Transformations

- Histogram **matching**
  - Sometimes, we don't want to equalize
  - But, we want to match the histogram of one image to another chosen histogram (NOT uniform)
    - e.g., for standardizing the intensity
  - In that case,
    $CDF_a\,(a) = CDF_b\,(b)$
    implies
    $b = CDF_b^{-1}\,(CDF_a\,(a))$



Gray-Level Transformations

- Histogram **matching**