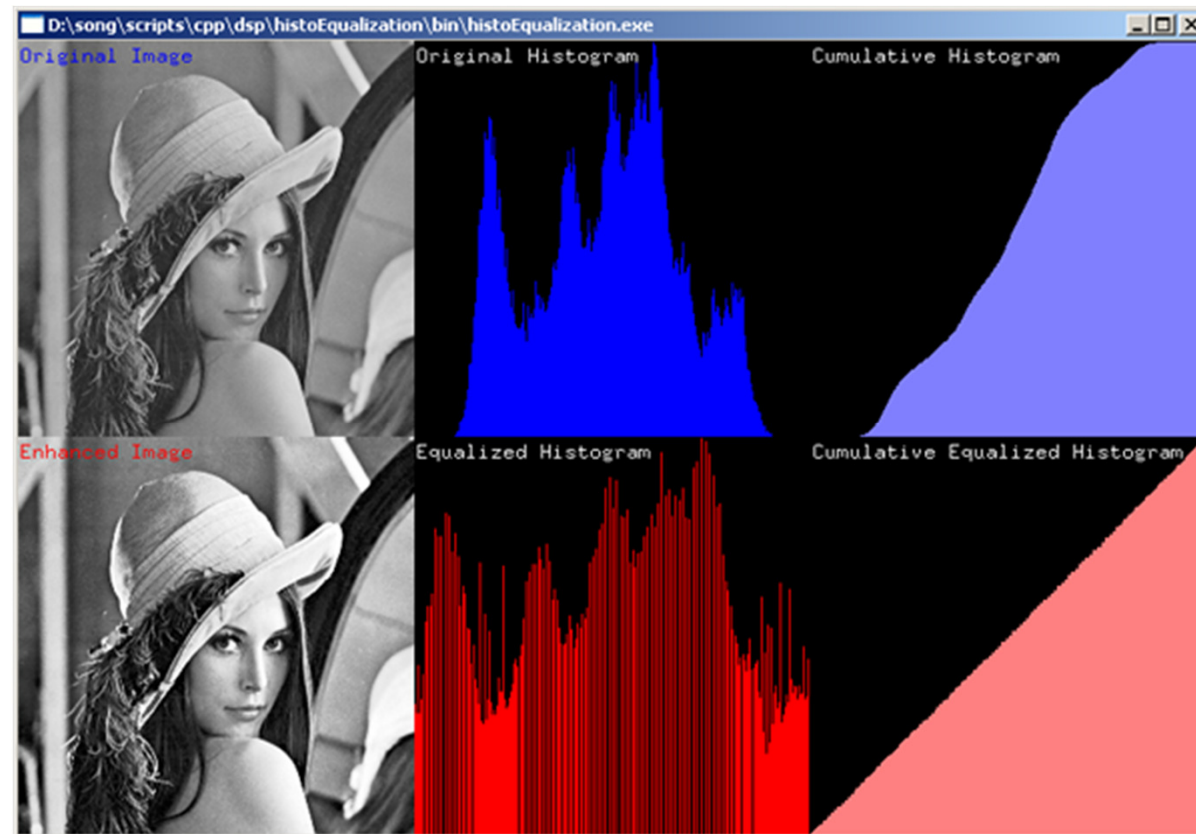


# Point Operations



# General Image Operations

- Three type of image operations
  1. Point operations
  2. Geometric operations
  3. Spatial operations

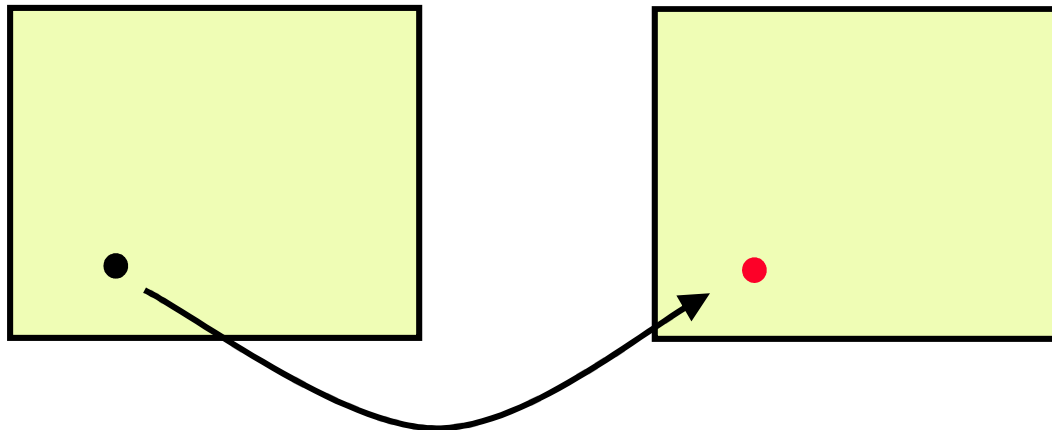


# Point Operations

---



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



# Point Operations

---

- Operation depends on Pixel's value.
- Context free (memory-less).
- Operation can be performed on the Histogram.
- Example:

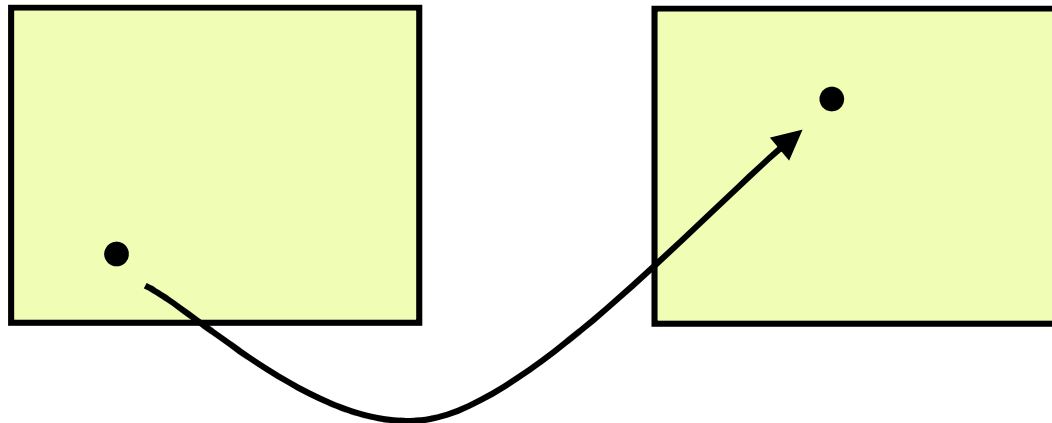
$$g(x, y) = \alpha \cdot f(x, y) + \beta$$

# Geometric Operations

---



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



# Geometric Operations

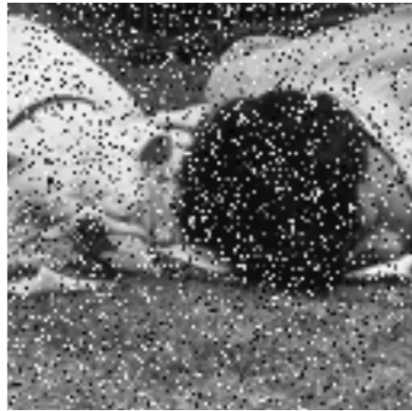
---

- Operation depend on pixel's coordinates.
- Context free.
- Independent of pixels value.
- Example:

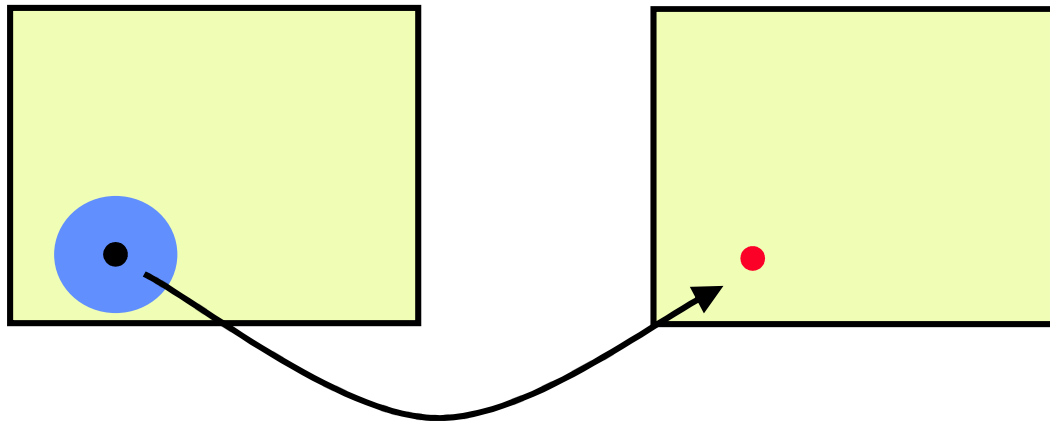
$$g(x, y) = f(x + a, y + b)$$

# Spatial Operations

---



$$g(x, y) = M(\{f(i, j) | (i, j) \in N(x, y)\})$$



# Spatial Operations

---

- Operation depends on Pixel's value and coordinates.
- Context dependant.
- Spatial v.s. Frequency domain.
- Example:

$$g(x, y) = \sum_{i, j \in N(x, y)} f(i, j) / n$$



# Point Operations

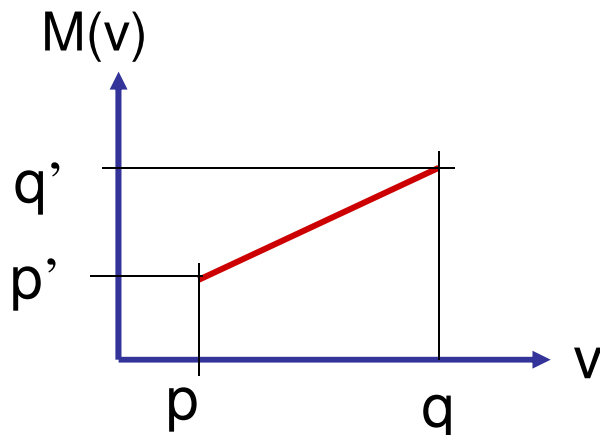
---

- A point operation can be defined as a mapping function:

$$v_{new} = M(v_{old})$$

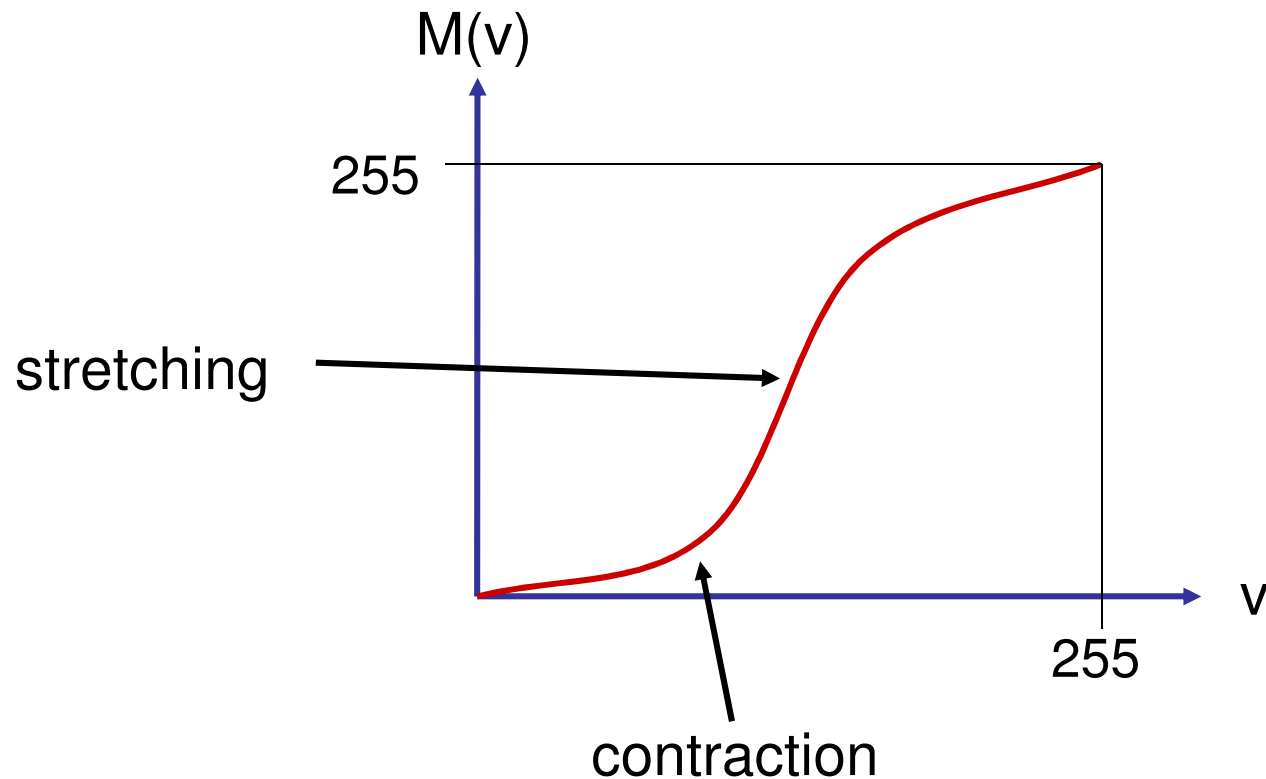
where  $v$  stands for gray values.

- $M(v)$  takes any value  $v$  in the source image into  $v_{new}$  in the destination image.
- Simplest case - Linear Mapping:  $M(v) = \alpha v + \beta$



$$\alpha = \frac{q' - p'}{q - p} \quad ; \quad \beta = p' - \alpha p$$

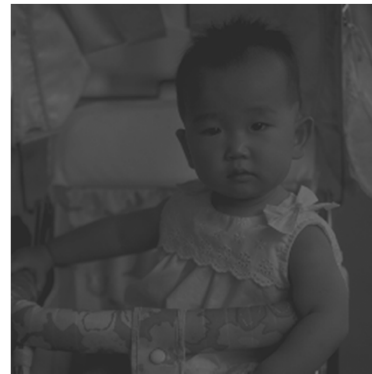
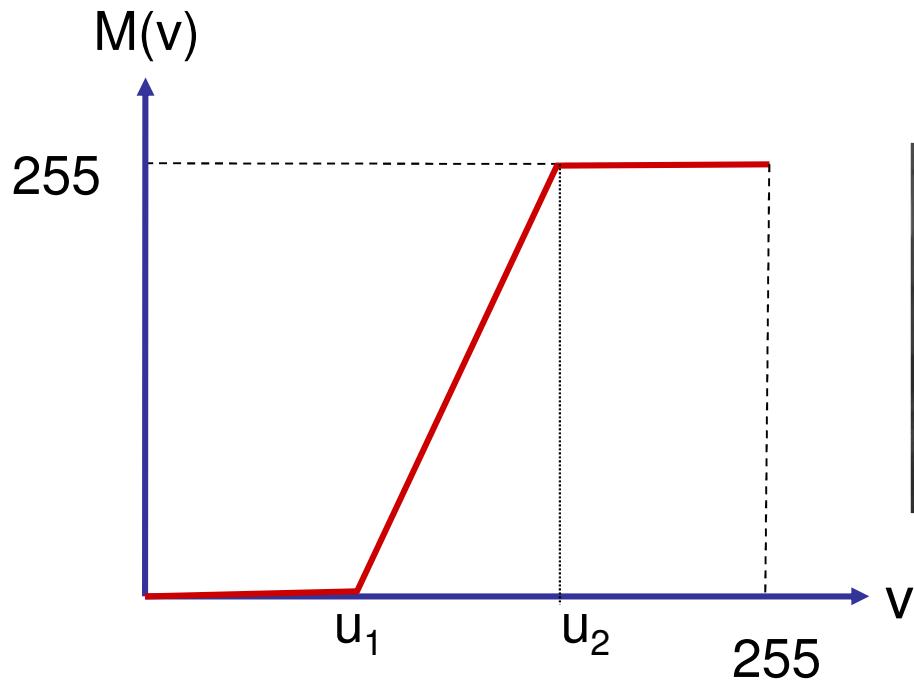
- If it is required to map the full gray-level range (256 values) to its full range while keeping the gray-level order - a **non-decreasing monotonic mapping function** is needed:



# Contrast Enhancement

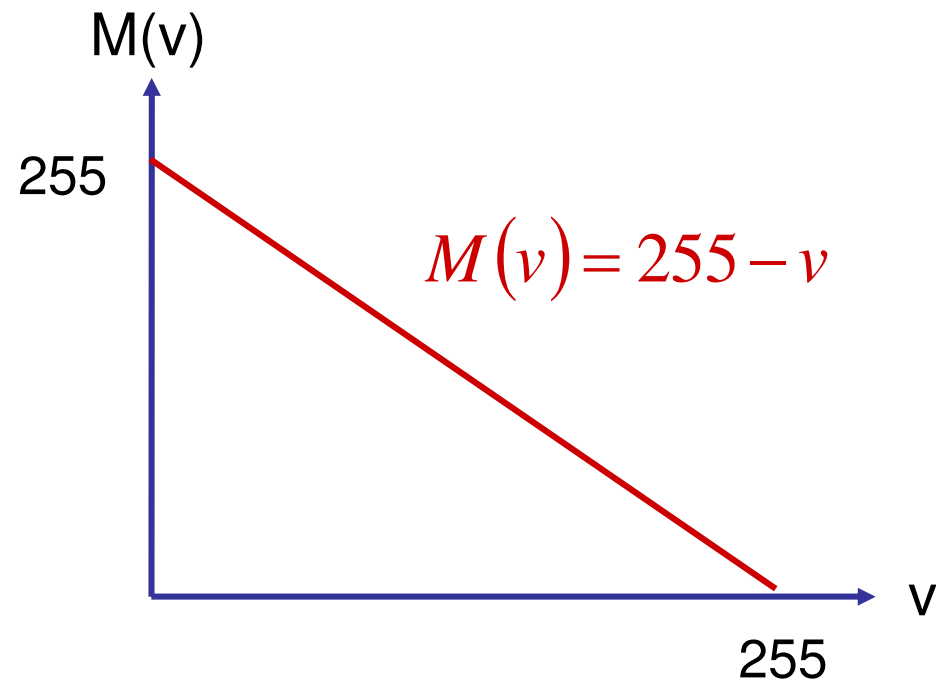
---

- If most of the gray-levels in the image are in  $[u_1, u_2]$ , the following mapping increases the image contrast.
- The values  $u_1$  and  $u_2$  can be found by using the image's accumulated histogram.



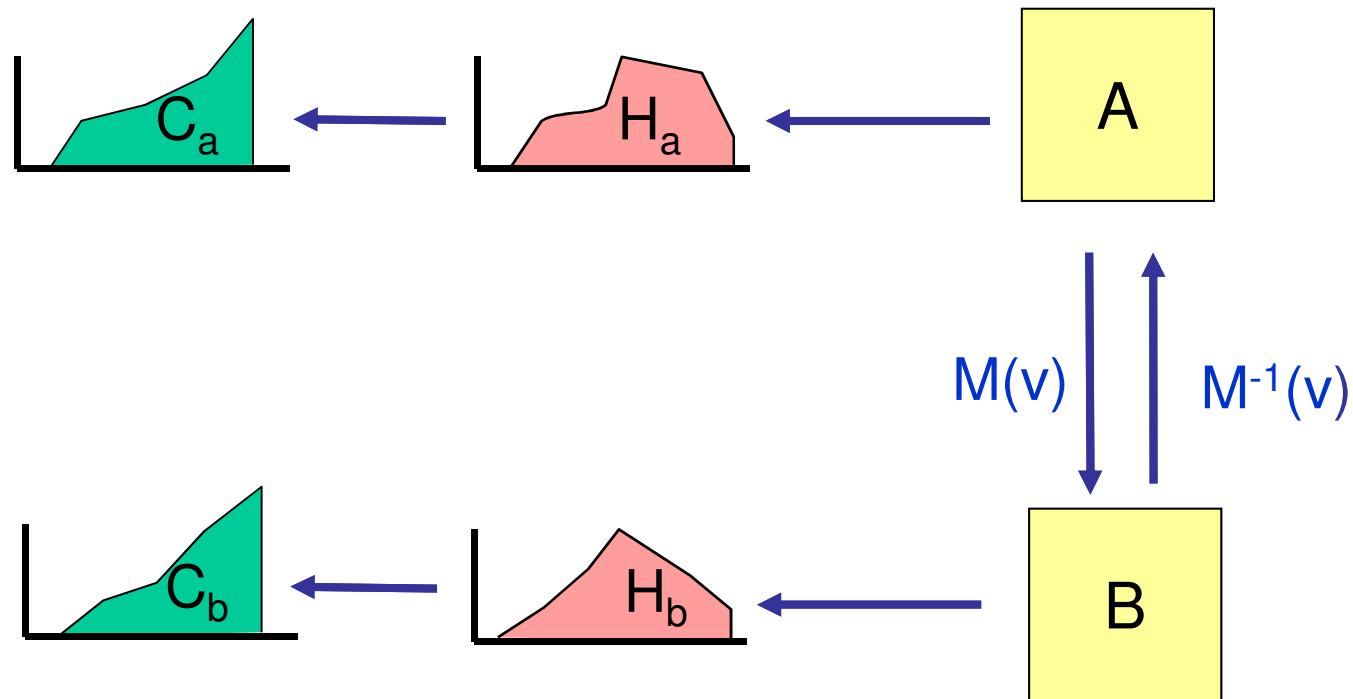
# The Negative Mapping

---



# Point Operations and the Histogram

Given a point operation:  $v_b = M(v_a)$



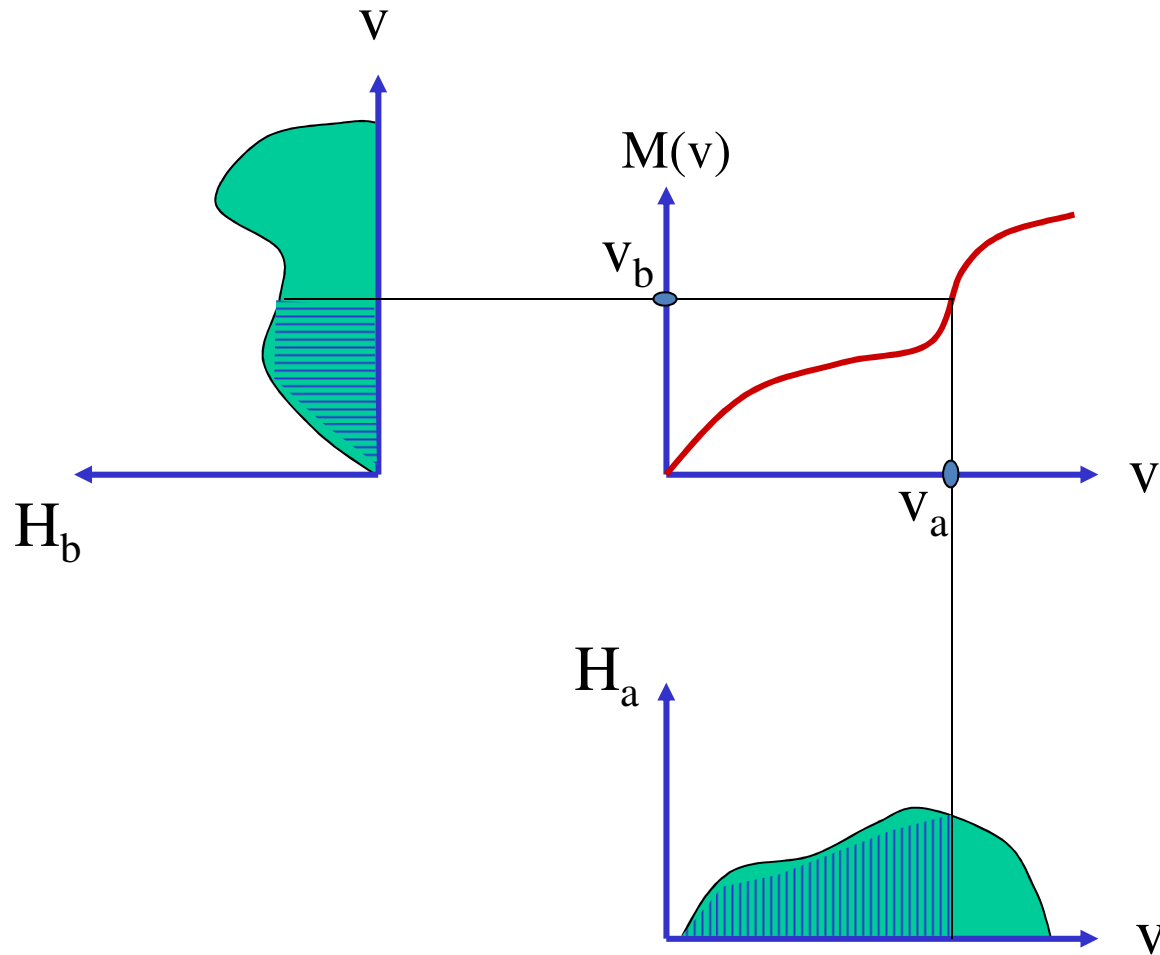
# Point Operations and the Histogram

- Given a point operation:

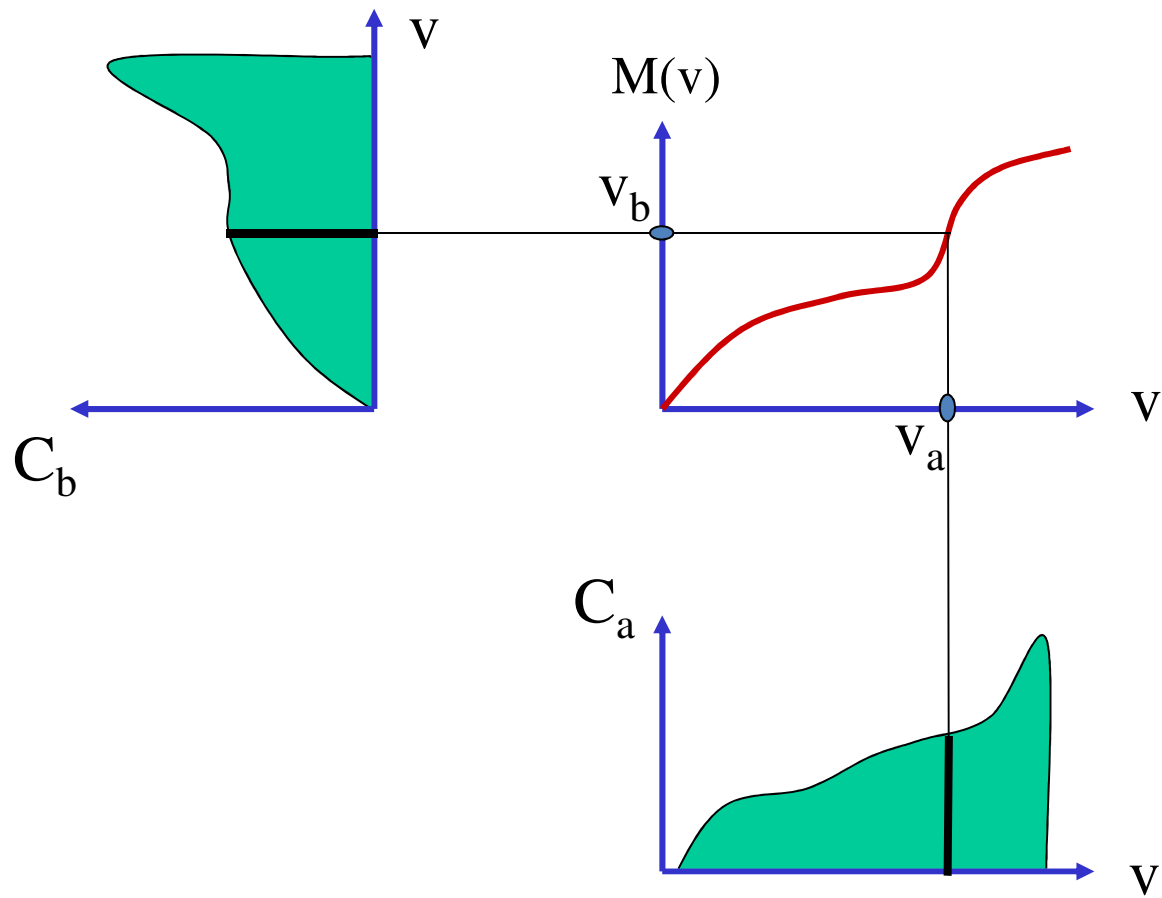
$$v_b = M(v_a)$$

- $M(v_a)$  takes any value  $v_a$  in image A and maps it into  $v_b$  in image B.
- **Requirement:** the mapping M is a monotonic increasing function ( $M^{-1}$  exists).
- In this case, the area under  $H_a$  between 0 and  $v_a$  is equal to the area under  $H_b$  between 0 and  $v_b$

- The area under  $H_a$  between 0 and  $v_a$  is equal to the area under  $H_b$  between 0 and  $v_b=M(v_a)$

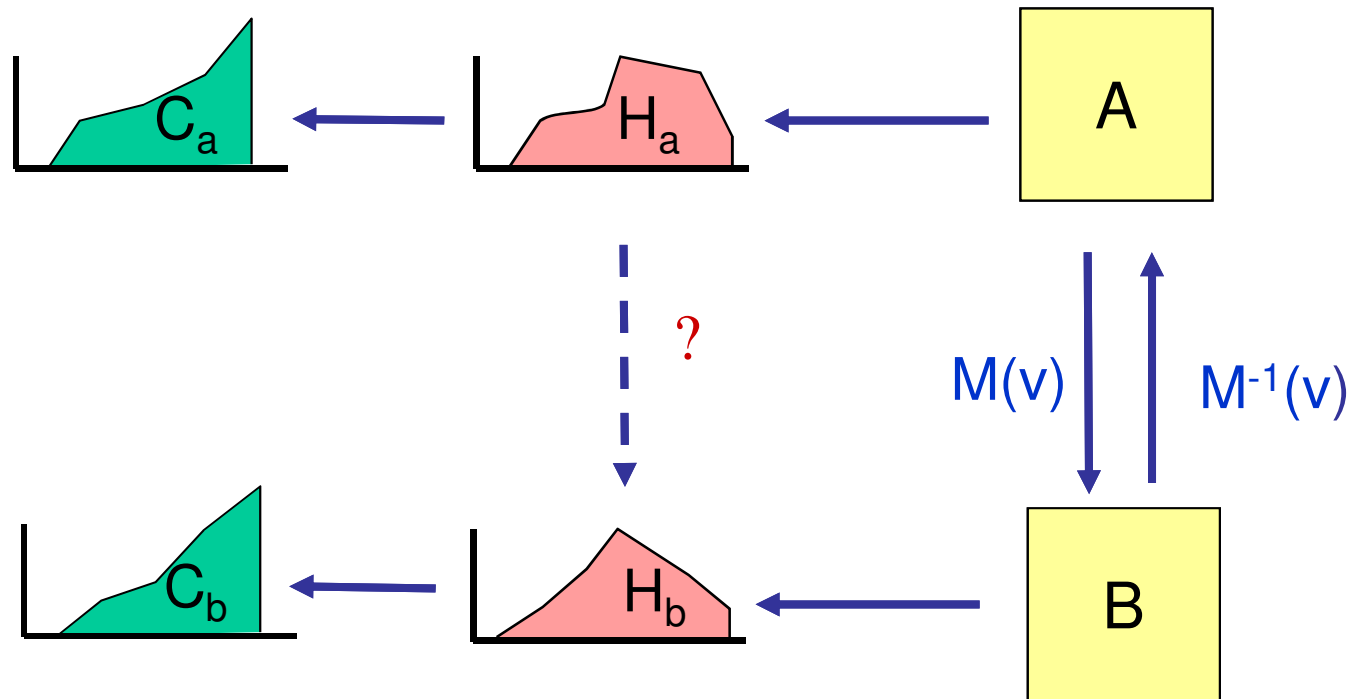


- The value of  $C_a$  at  $v_a$  is equal to the value of  $C_b$  at  $v_b$ .



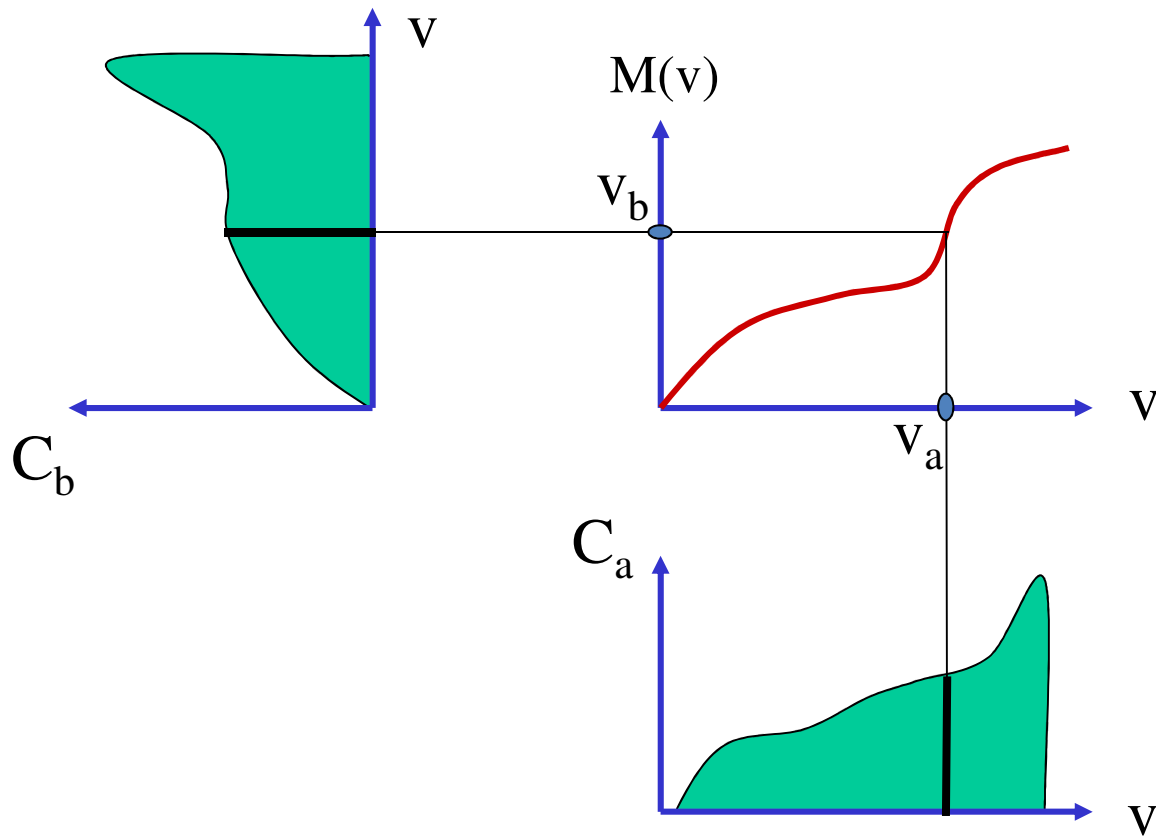


Q: Is it possible to obtain  $H_b$  directly from  $H_a$  and  $M(v)$ ?



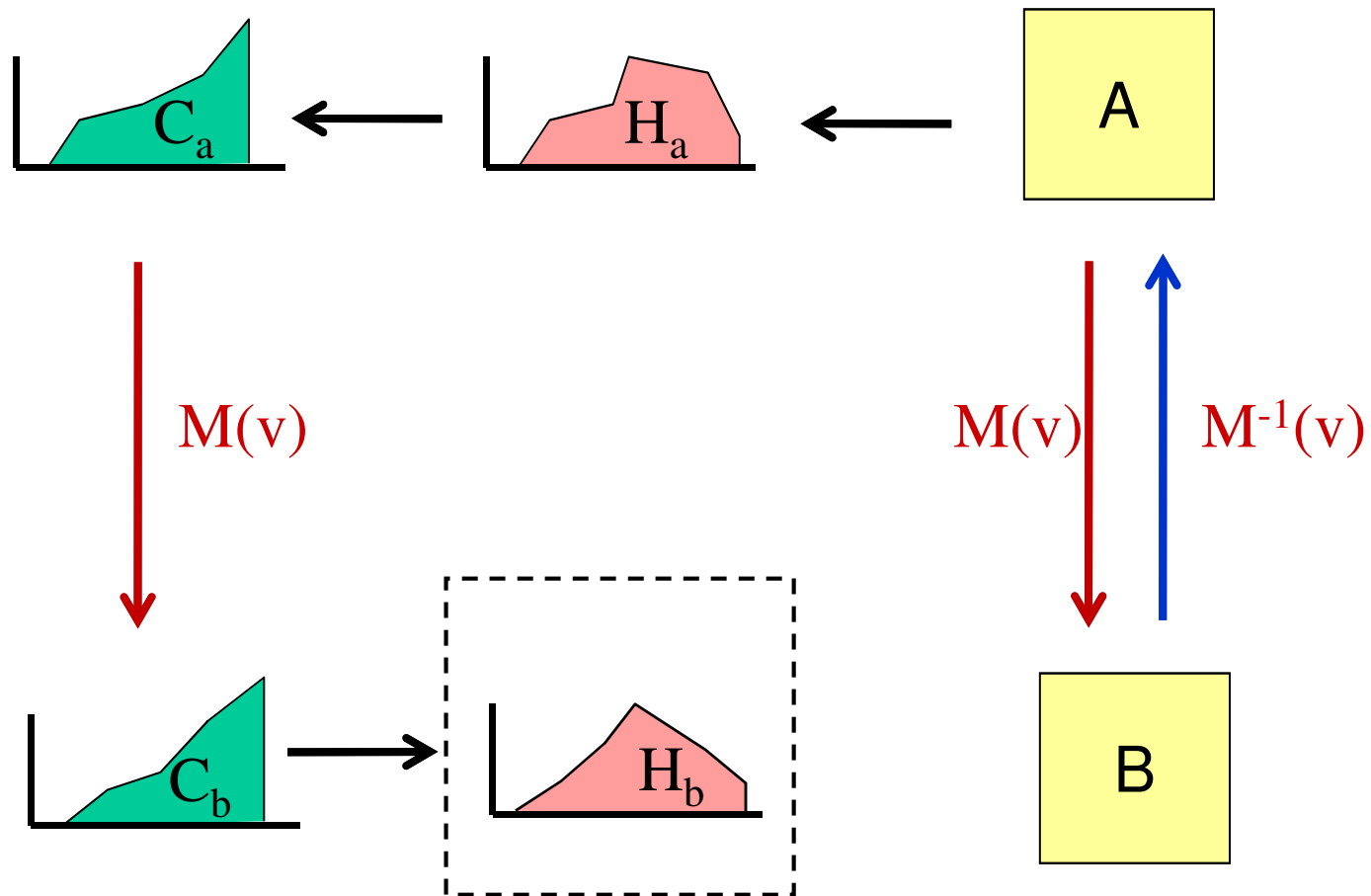
- **A:** Since  $M(v)$  is monotonic,  $C_a(v_a) = C_b(v_b)$  therefore:

$$C_b(v_b) = C_a(M^{-1}(v_b))$$

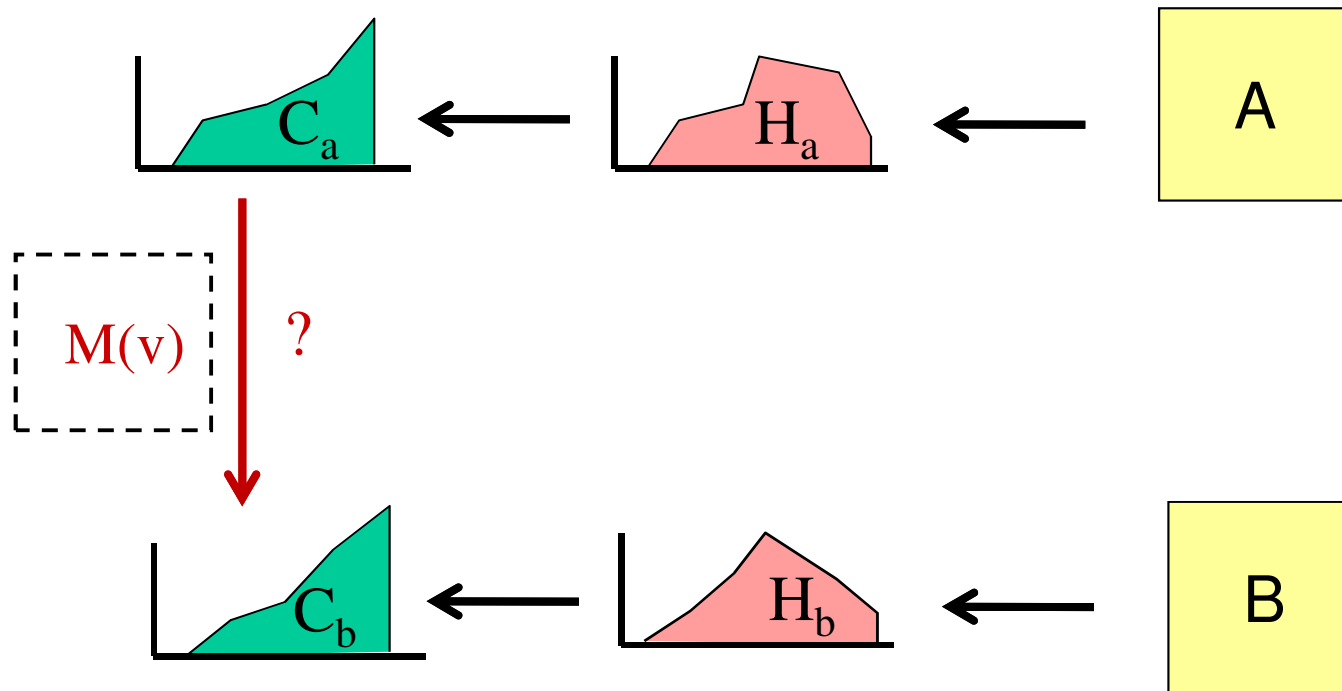


- The histogram of image B can be calculated:

$$H_b(v) = C_b(v) - C_b(v-1)$$



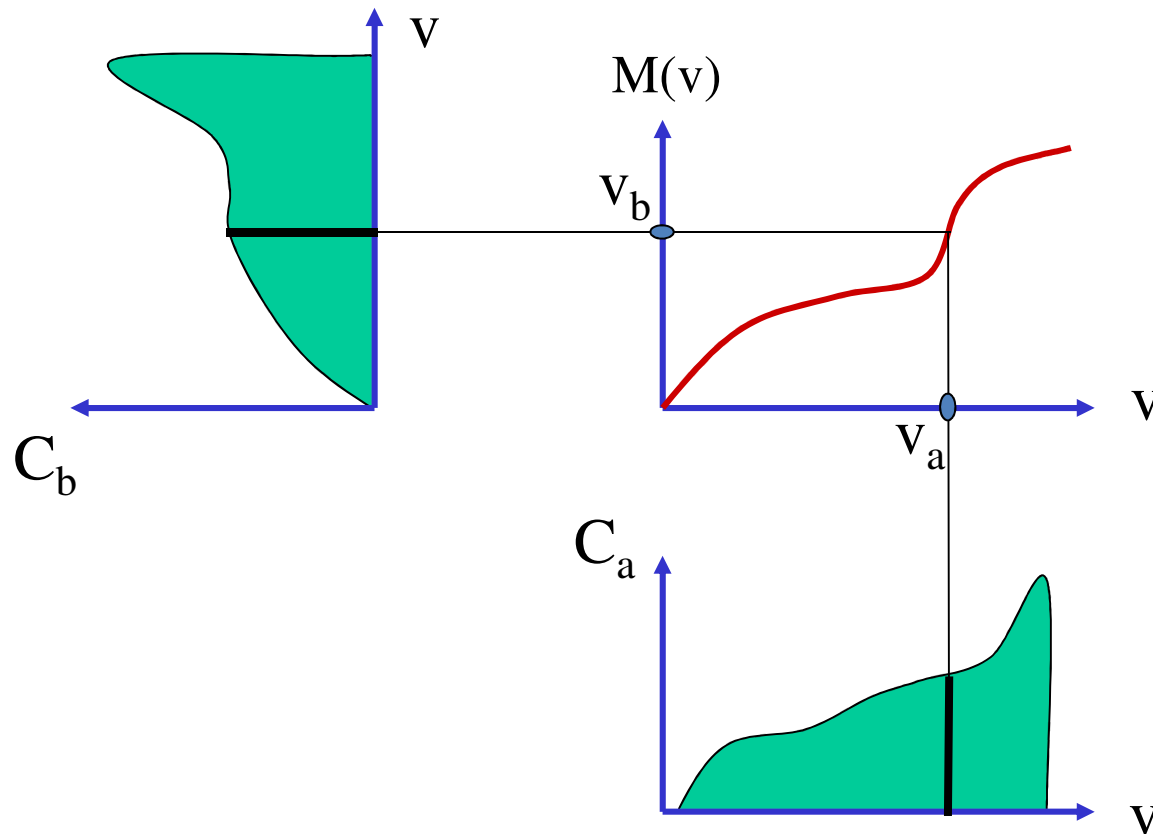
Q: Is it possible to obtain  $M(v)$  directly from  $H_a$  and  $H_b$  ?



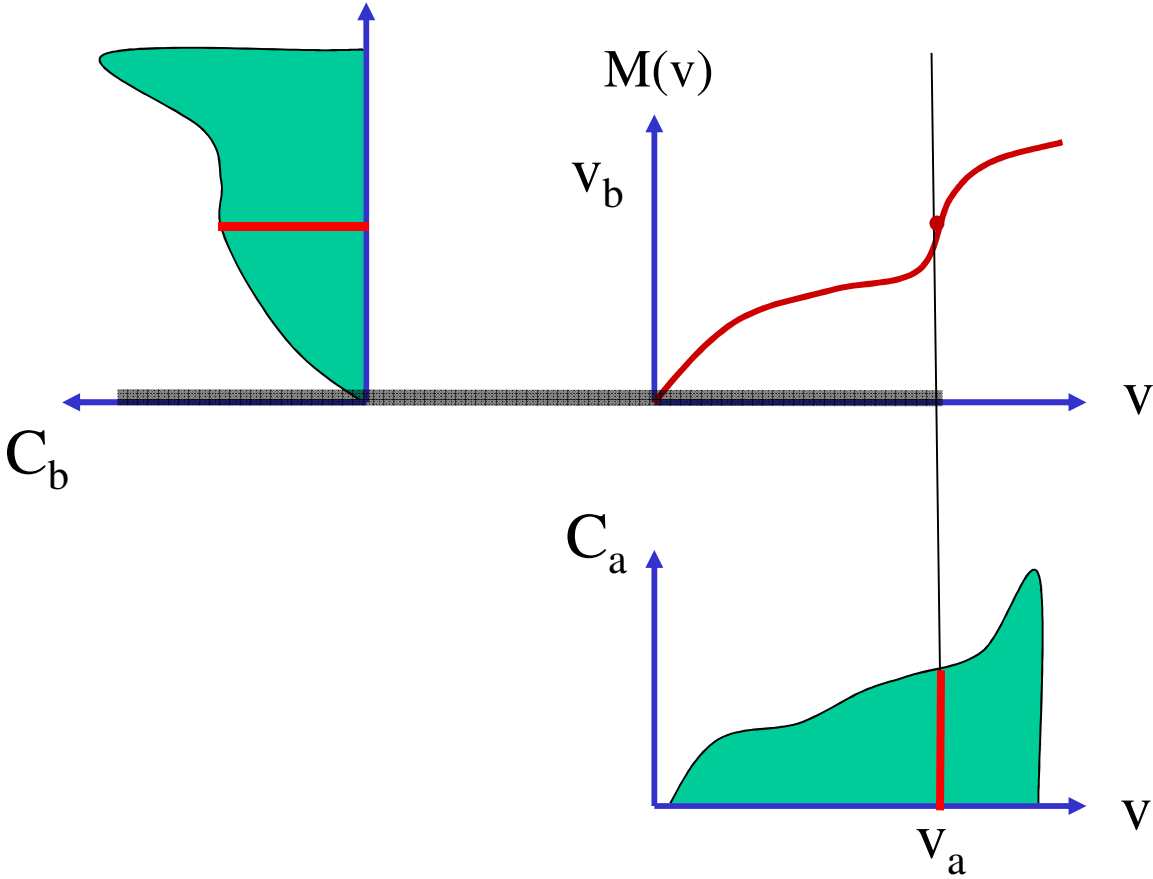
- **A:** Since  $C_b$  is monotonic and  $C_b(v_b)=C_a(v_a)$

$$v_b = C_b^{-1} C_a(v_a) = M(v_a)$$

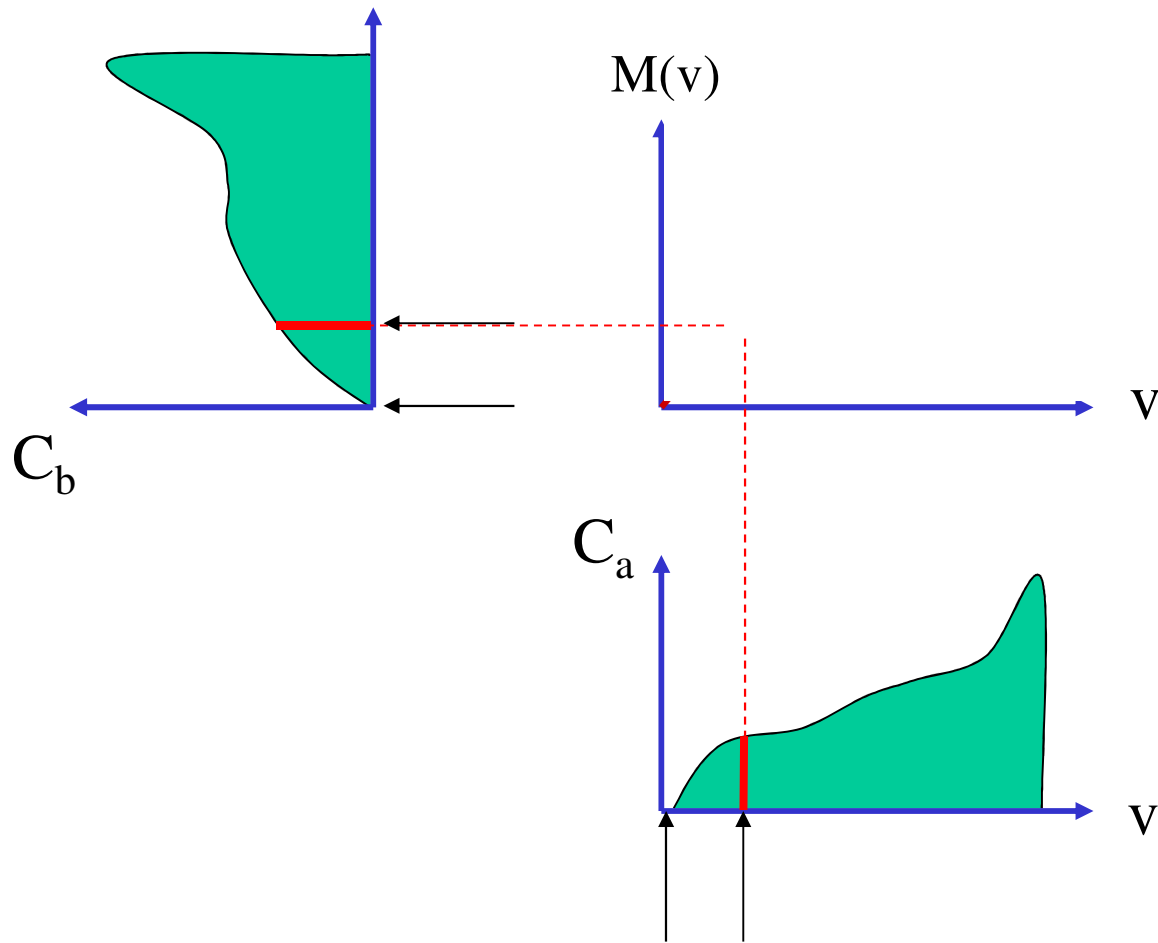
- Alternatively  $M(v_a)=v_b$  if  $C_a(v_a)=C_b(v_b)$



Calculating  $M(v)$  from  $C_a$  and  $C_b$ :



# Calculating $M(v)$ from $C_a$ and $C_b$ : 2-pointer Algorithm

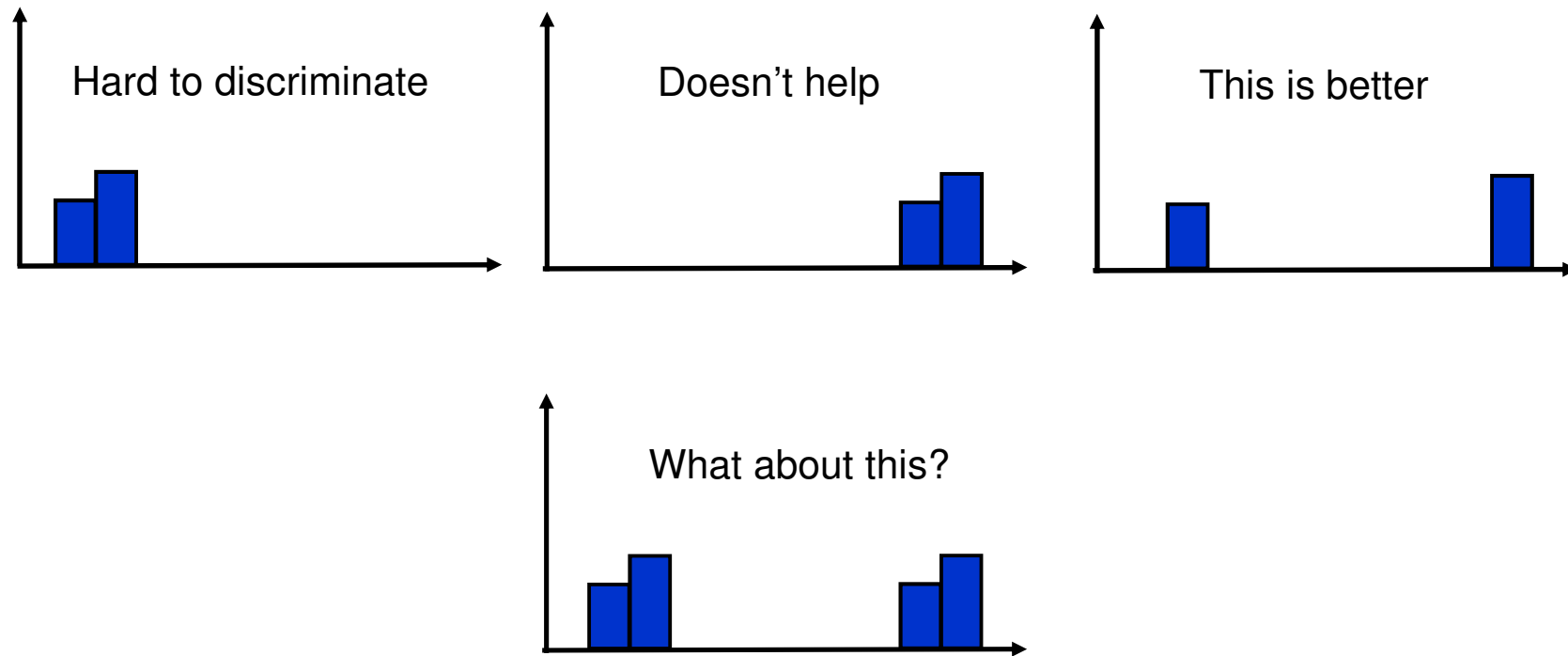


Is this always possible?

# Histogram Equalization

---

- Visual discrimination between objects depends on the their gray-level separation.
- How can we improve discrimination **after** image has been acquired?



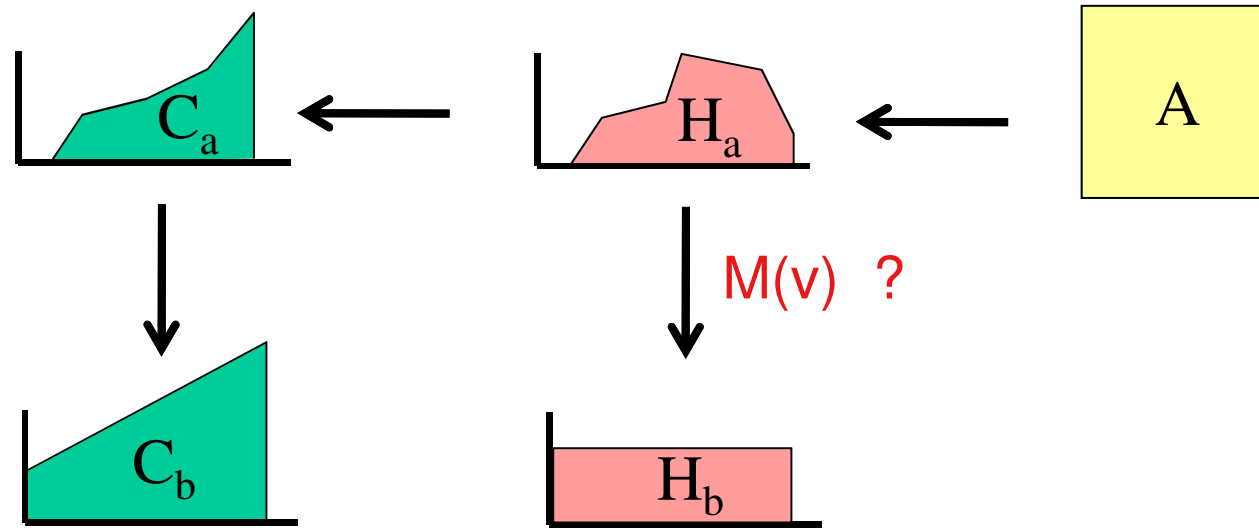


# Histogram Equalization

---

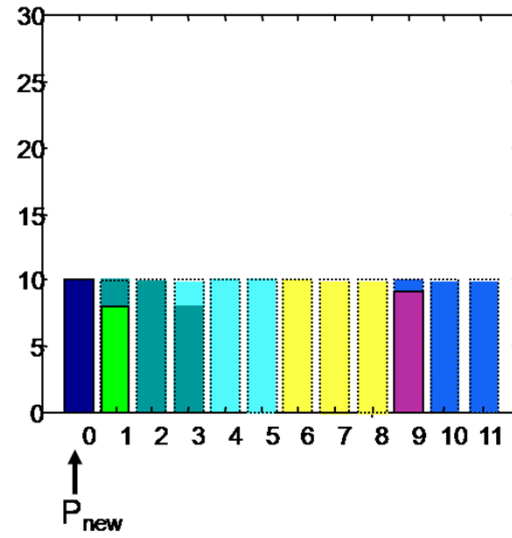
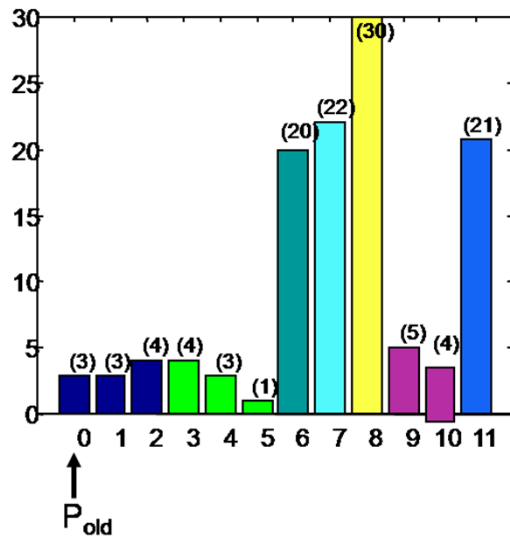
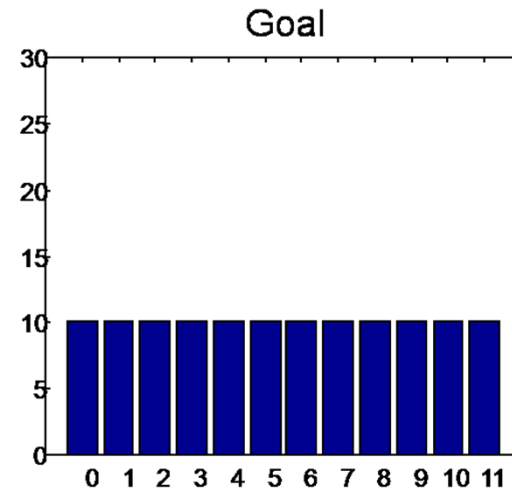
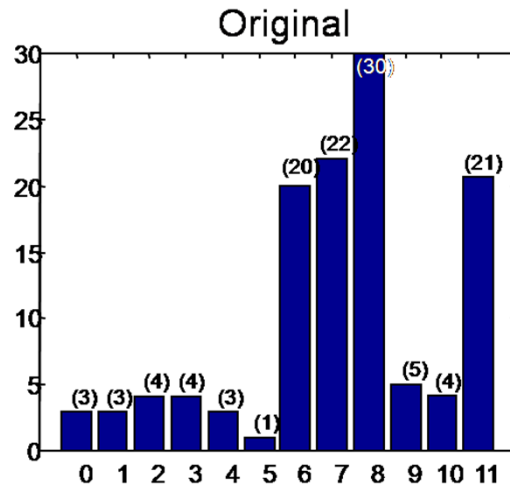
- For a better visual discrimination of an image we would like to re-assign gray-levels so that gray-level resource will be optimally assigned.
- **Our goal:** finding a gray-level transformation  $M(v)$  such that:
  - The histogram  $H_b$  is as flat as possible.
  - The order of gray-levels is maintained.
  - The histogram bars are not fragmented.

# Histogram Equalization: Algorithm



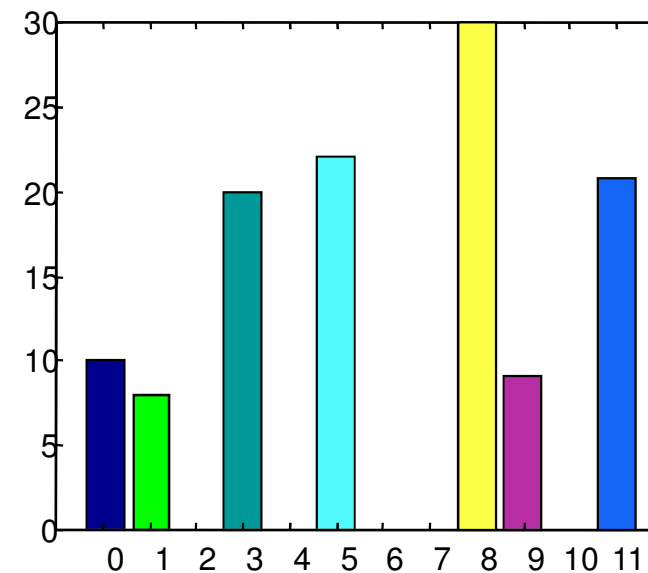
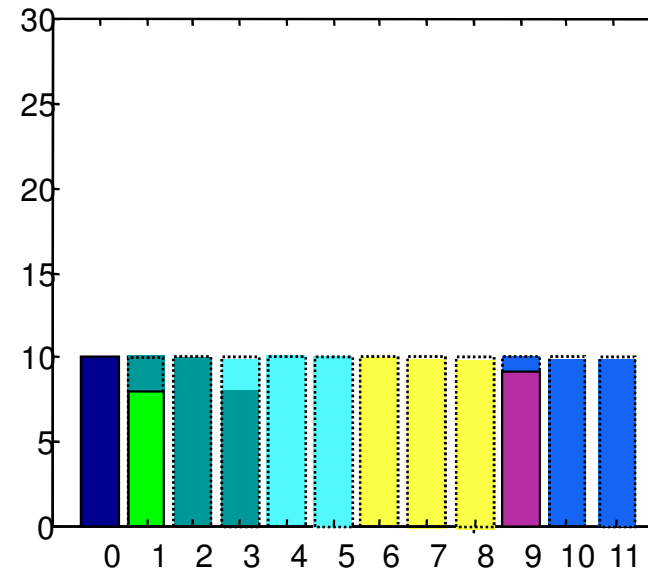
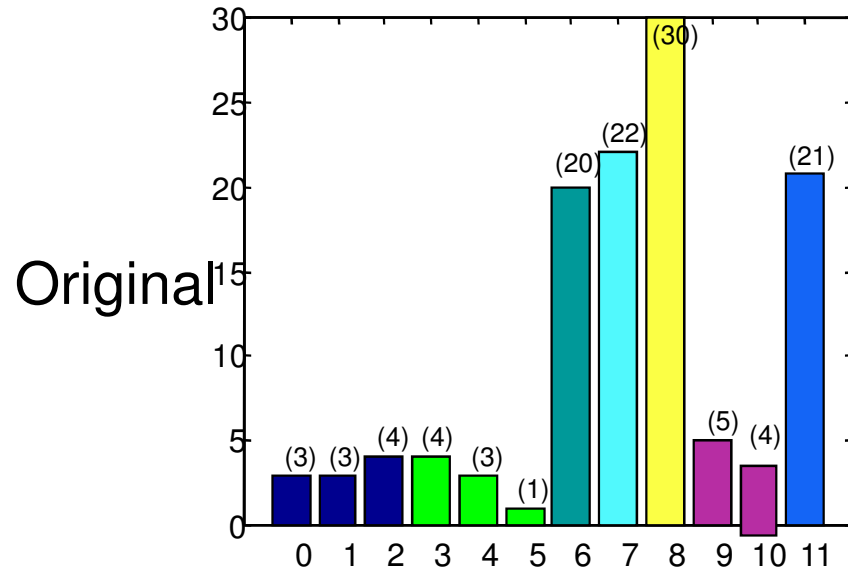
- Define: 
$$C_b(v) = v * \frac{(\# \text{ pixels})}{\# \text{ grayValues}}$$
- Assign: 
$$v_b = C_b^{-1}(C_a(v_a)) = M(v_a)$$


# Hist. Eq.: The two pointers algorithm



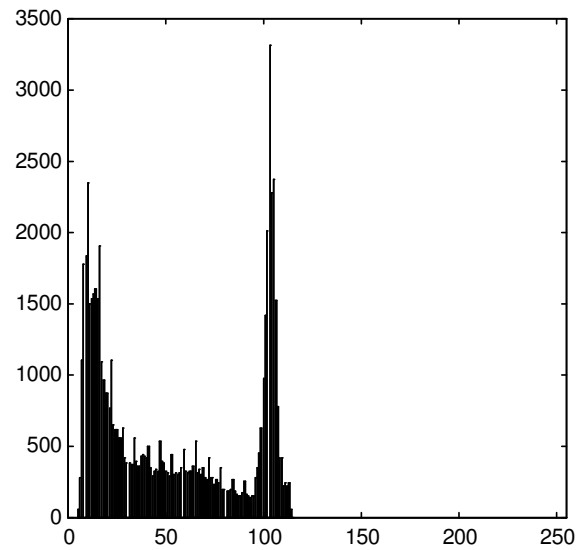
↓ 0 1 2 3 4 5 6 7 8 9 10 11    old  
    0 0 0 1 1 1 3 5 8 9 9 11    new

# Hist. Eq.: The two pointers algorithm

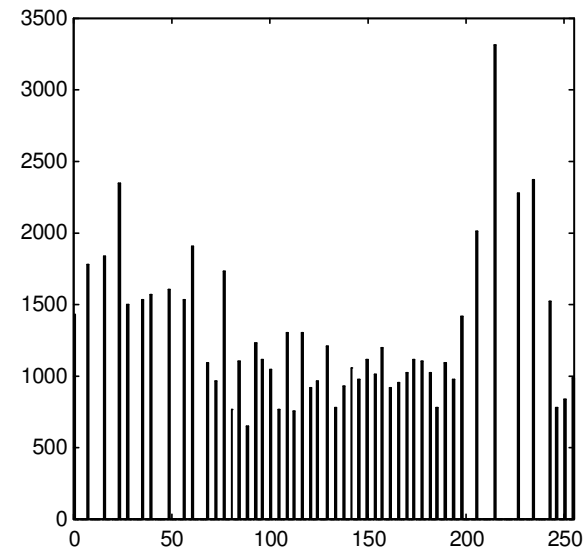



 0 1 2 3 4 5 6 7 8 9 10 11    old  
 0 0 0 1 1 1 3 5 8 9 9 11    new

# Hist. Eq.: Example 1

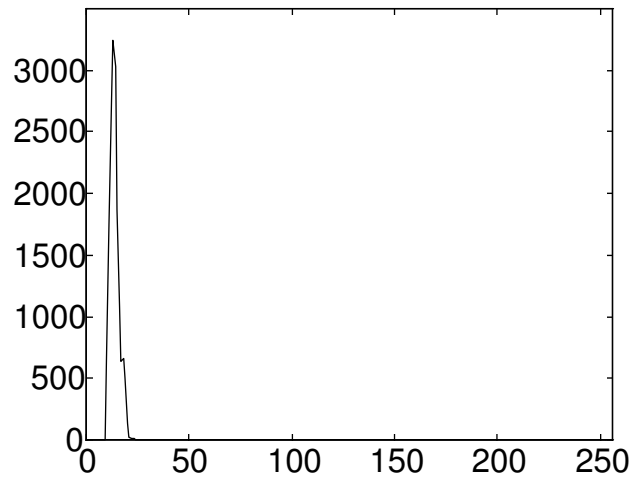


Original

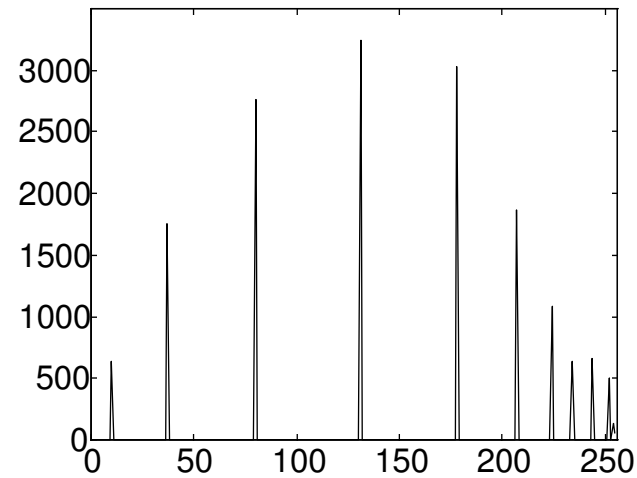


Equalized

# Hist. Eq.: Example 2



Original



Equalized

# Hist. Eq.: Example 3



# Adaptive Histogram Equalization

---



Original image



Global  
histogram  
equalization



Adaptive histogram  
equalization  
8x8 tiles

From B. Girod, IP



# Histogram matching

---

- Transforms an image A so that its histogram will match that of another image B.
- Usage: before comparing two images of the same scene (change detection) acquired under different lighting conditions or different camera parameters.



(a)

source

(b)

target

(c)

mapped source

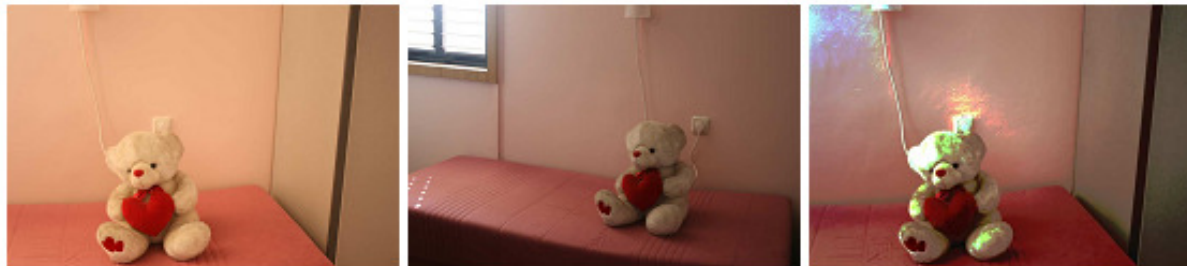
- However, in cases where corresponding colors between images are not “consistent” this mapping may fail:



(a)

(b)

(c)



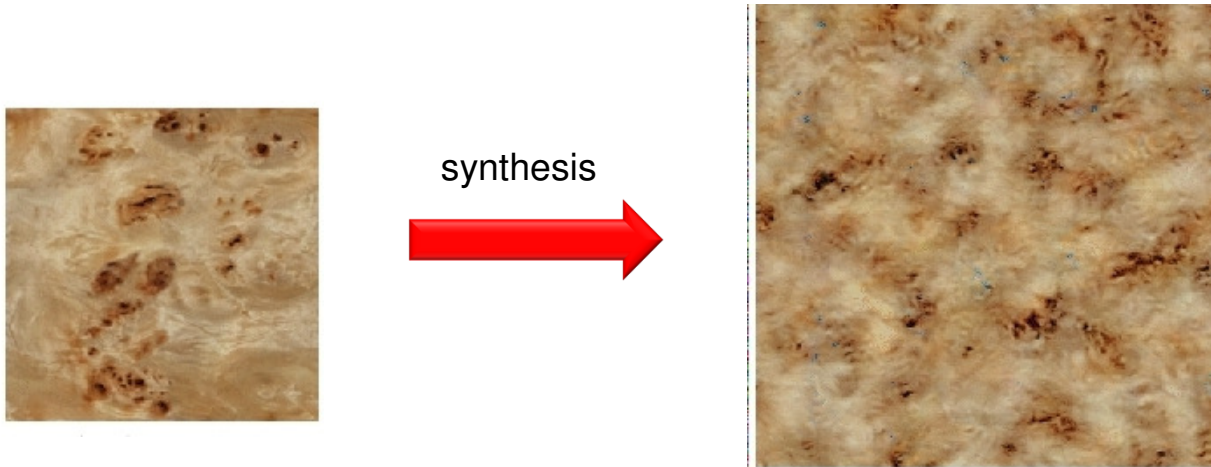
(a)

(b)

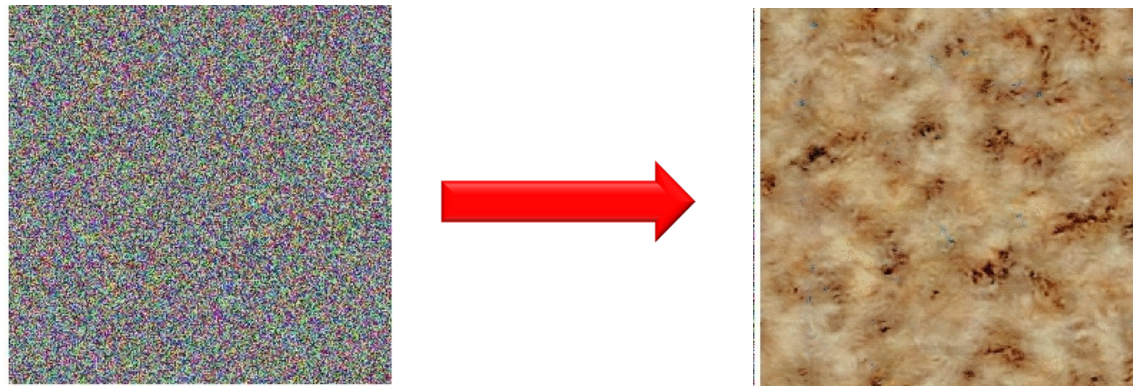
(c)

from: S. Kagarlitsky: M.Sc. thesis 2010.

- Surprising usage: Texture Synthesis (Heeger & Bergen 1995).



- Start with a random image.
- **Step1**: multi-resolution decomposition
- **Step2**: histogram matching for each resolution
- **Iterate**



- More results: (Heeger & Bergen 1995):



## Discussion:

- Histogram matching produces the optimal **monotonic** mapping so that the resulting histogram will be as **close** as possible to the target histogram.
- This does not necessarily imply similar images.

Example:

Original



Destination



hist. match



# Pattern Matching by Tone Mapping (MTM)

(ICCV 2011)

Find the BEST tone mapping between Pattern and Window

Pattern



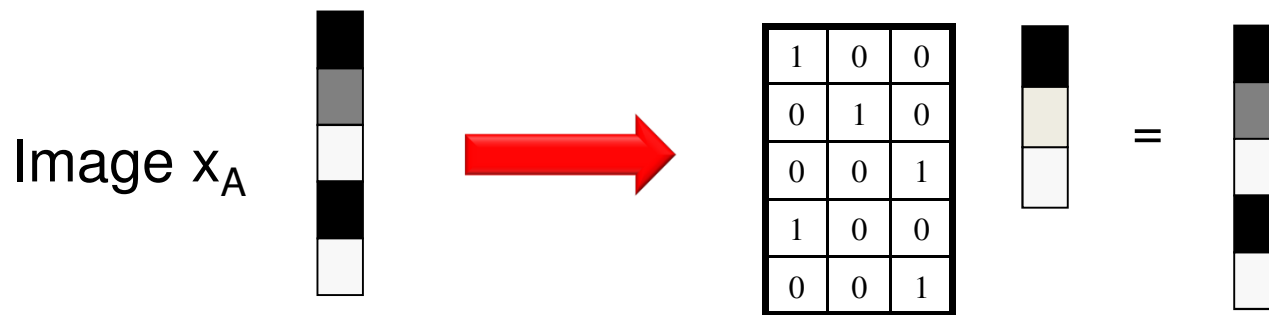
Window



# Pattern Matching by Tone Mapping (MTM)

(ICCV 2011)

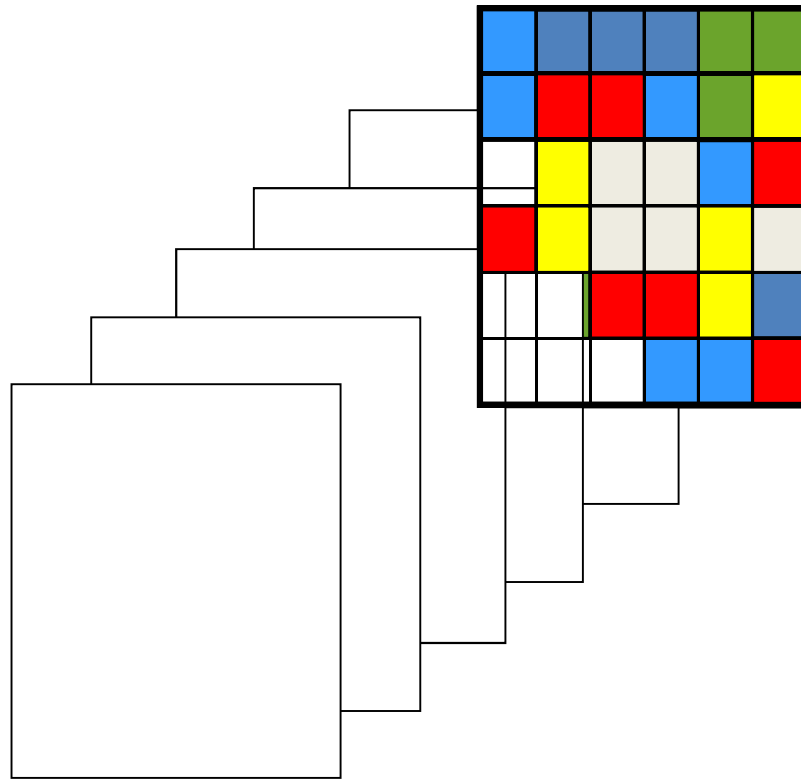
Use the Slice Transform (SLT) to represent image as a linear sum of “slice” signals:



$$S_p(x_A)p = x_A$$

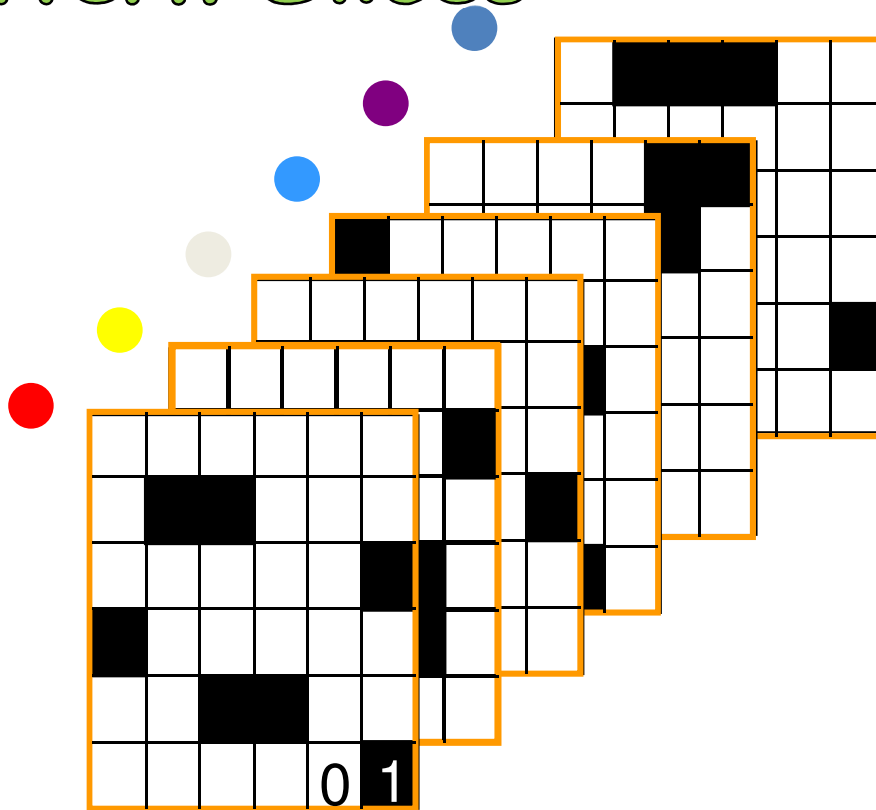


# Pattern Slices



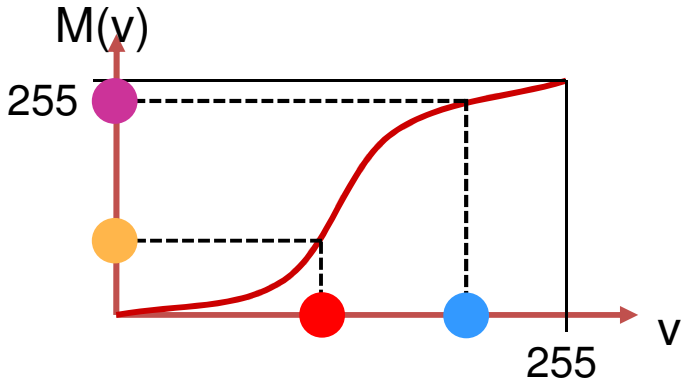
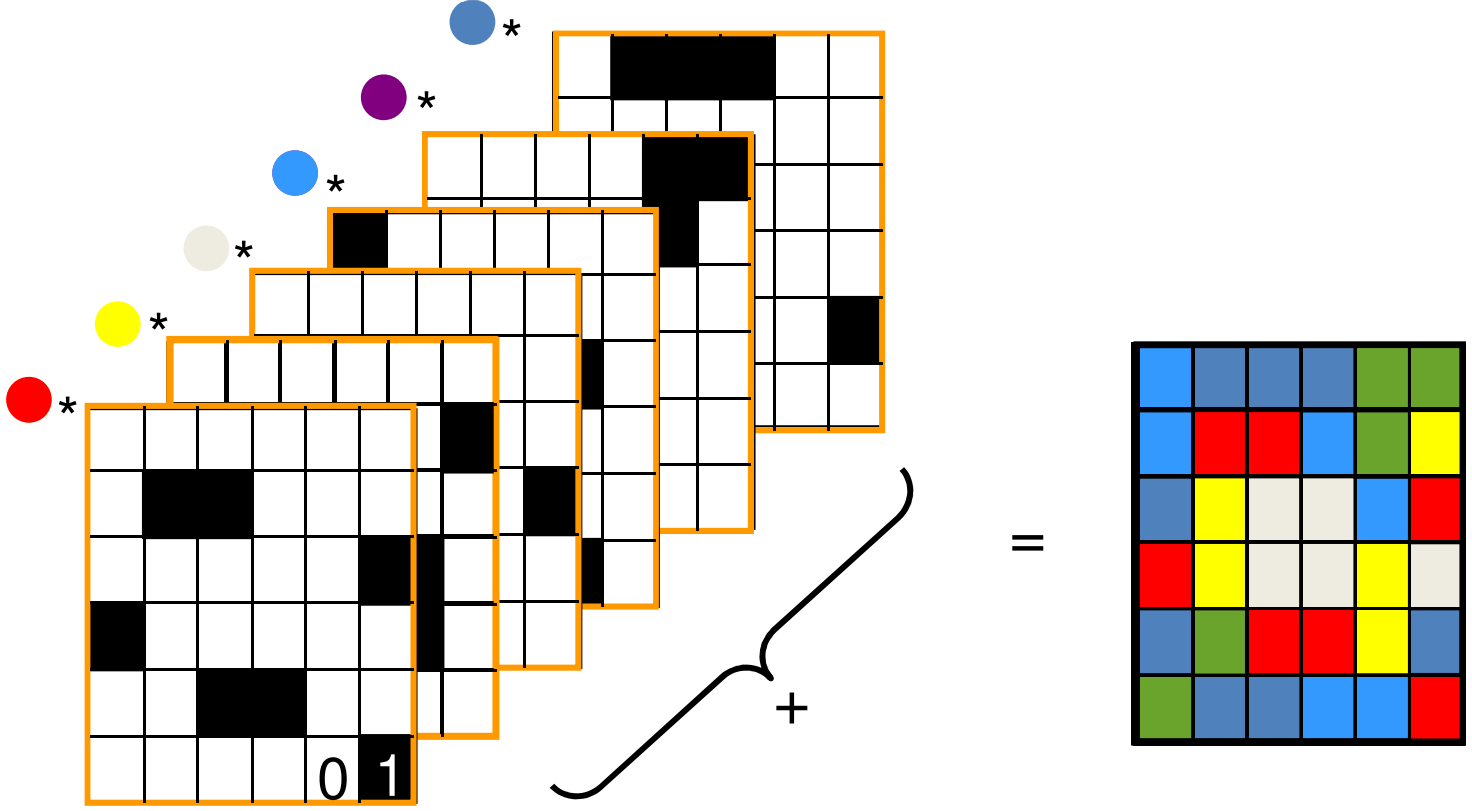
- We define a pattern slice  $p^j(i) = \begin{cases} 1 & \text{if } B(p(i)) = j \\ 0 & \text{otherwise} \end{cases}$

# Pattern Slices

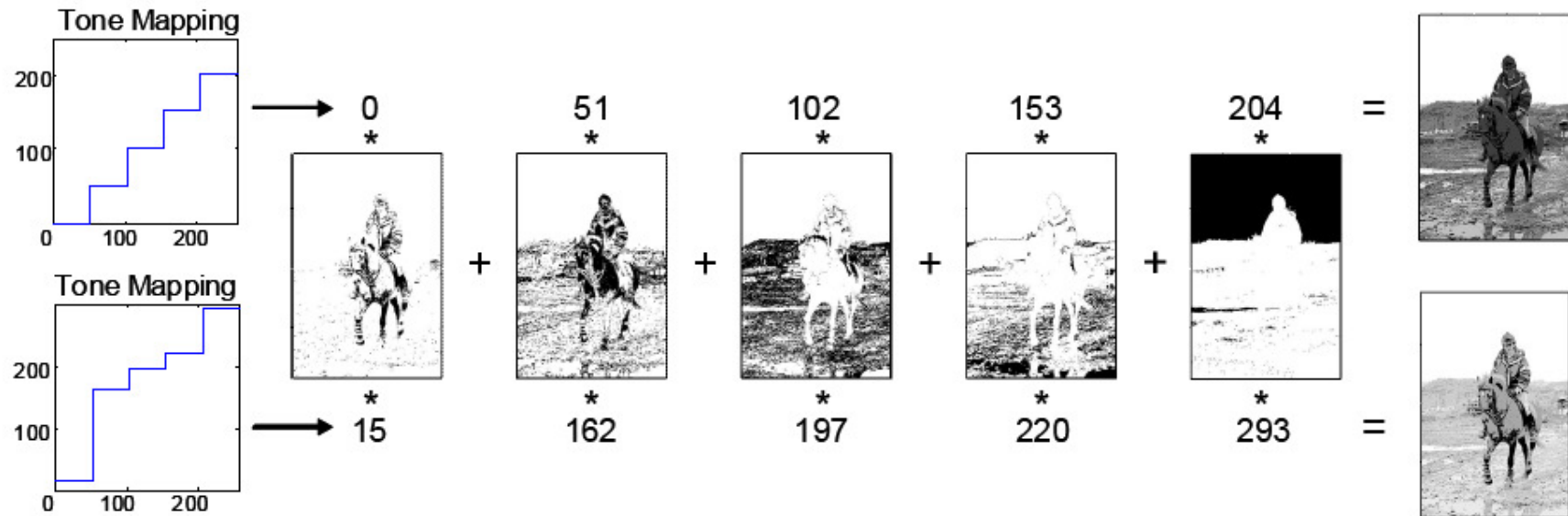


- We define a pattern slice  $p^j(i) = \begin{cases} 1 & \text{if } B(p(i)) = j \\ 0 & \text{otherwise} \end{cases}$

# Pattern Slices



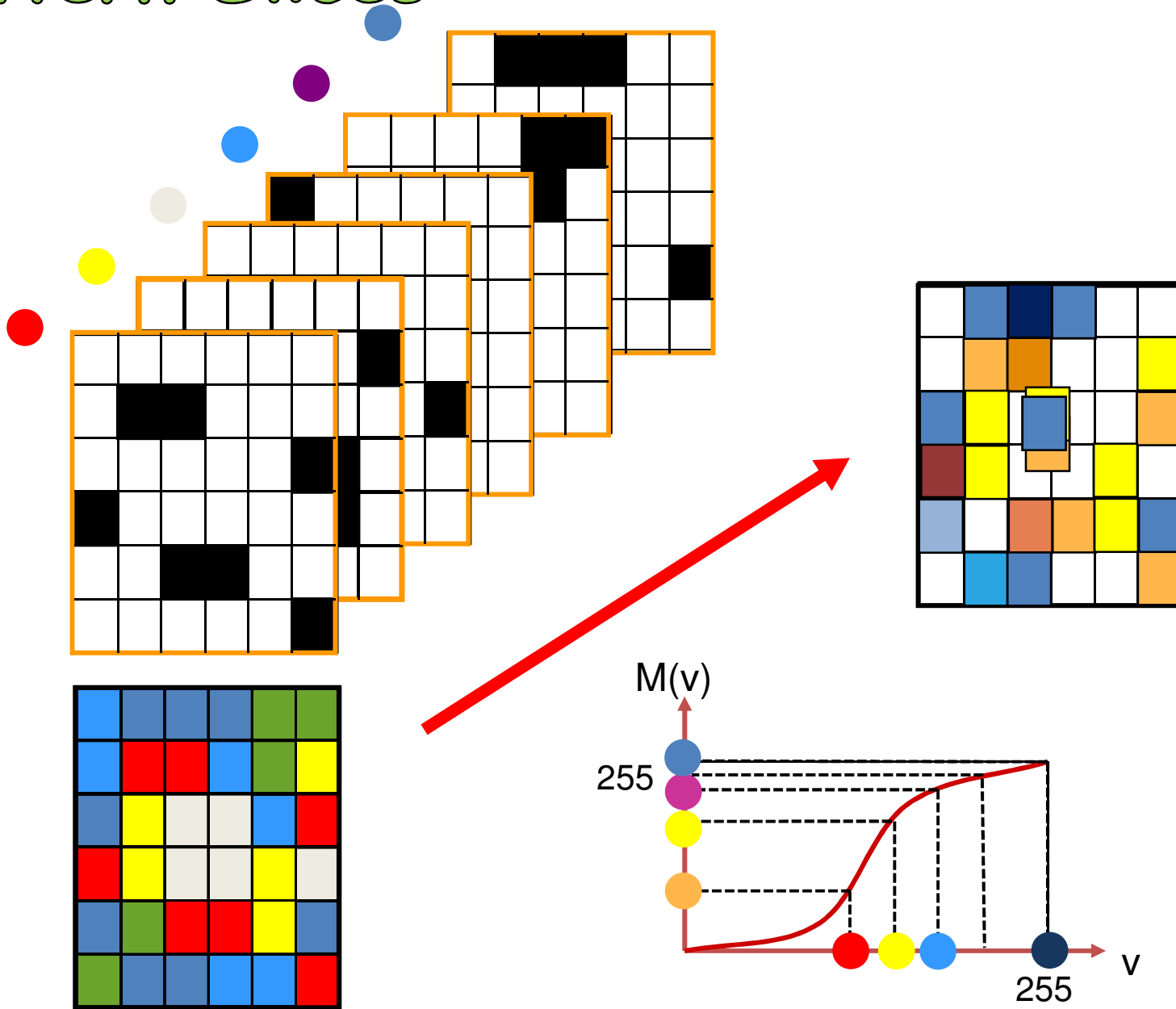
# Linear Combination of Image Slices



5 bins were used  $\alpha = [0, 51, 102, 153, 204, 256]$

Slices are shown inverted (1 = white, 0 = black)

# Pattern Slices



Example:

Original



Destination



results



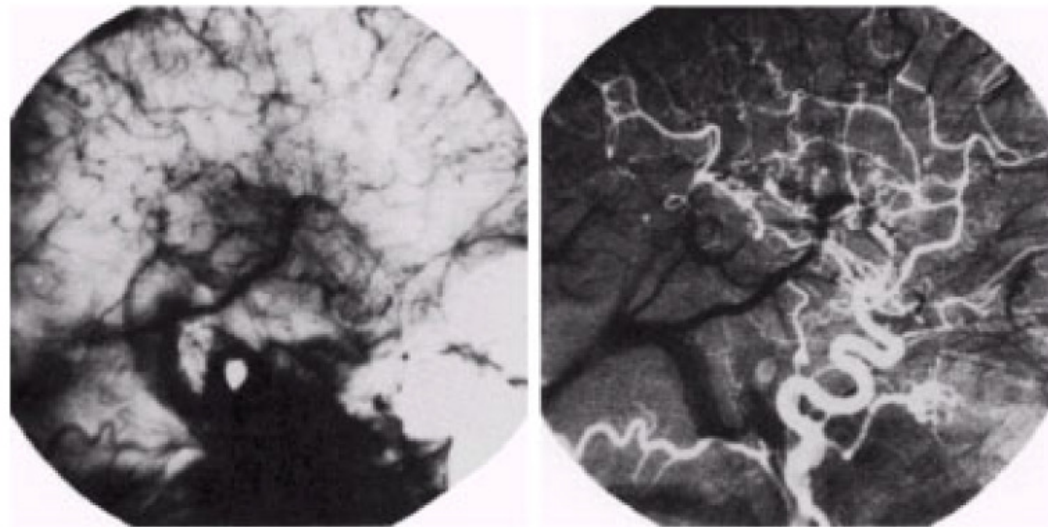
SLT



Hist. match

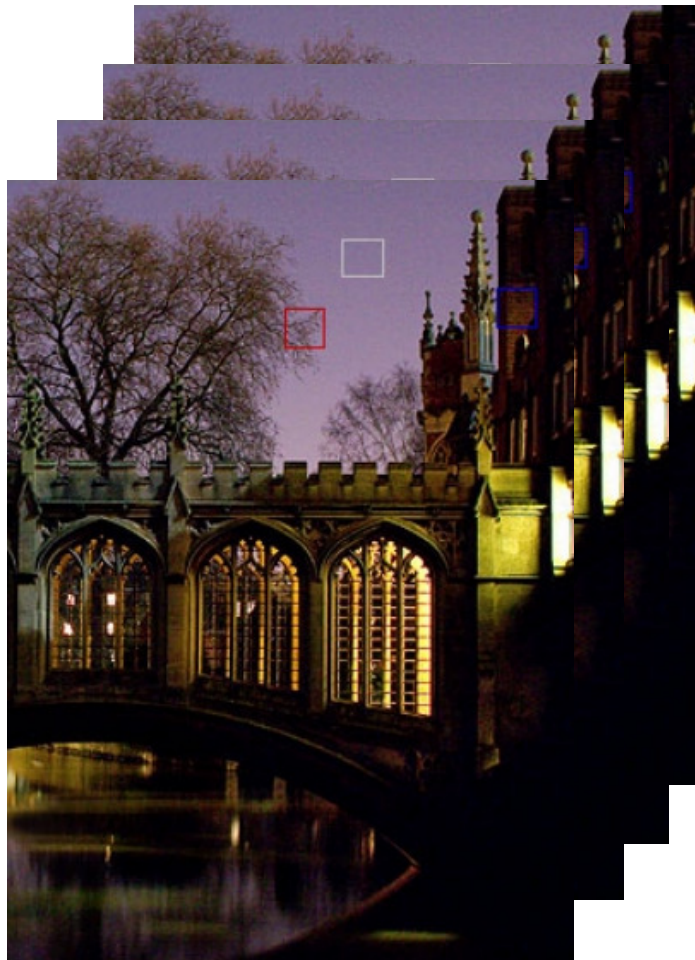
# Point Operation on Two+ Images

---



# Noise reduction by Averaging

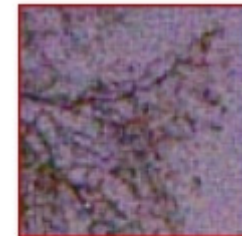
---



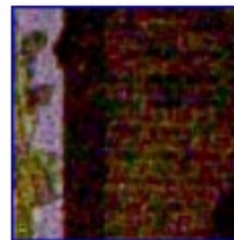
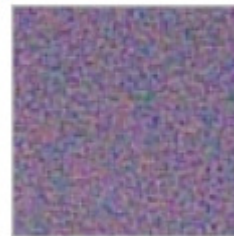
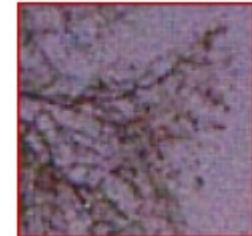
1 image



2 images



4 images



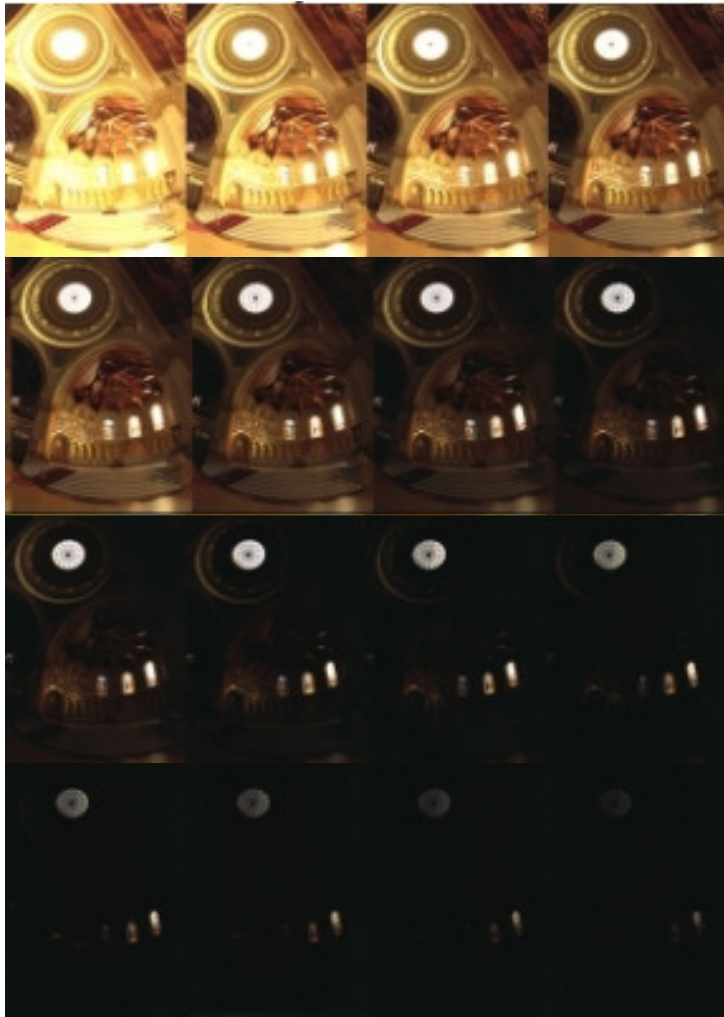
<http://www.cambridgeincolour.com/tutorials>



# High Dynamic Range Images

---

Different Exposures



Combined Image

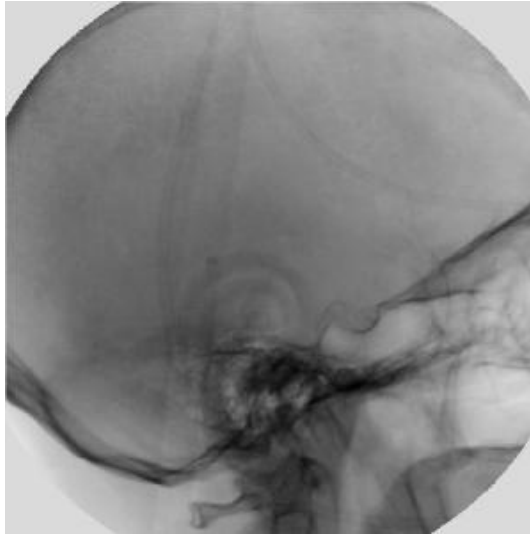


Malik 1997

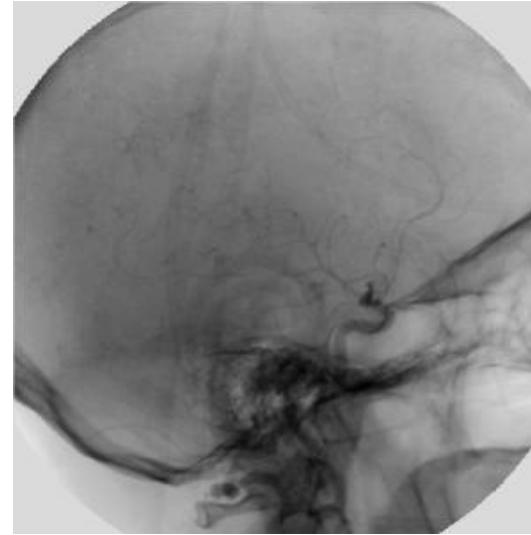
# Image Subtraction

---

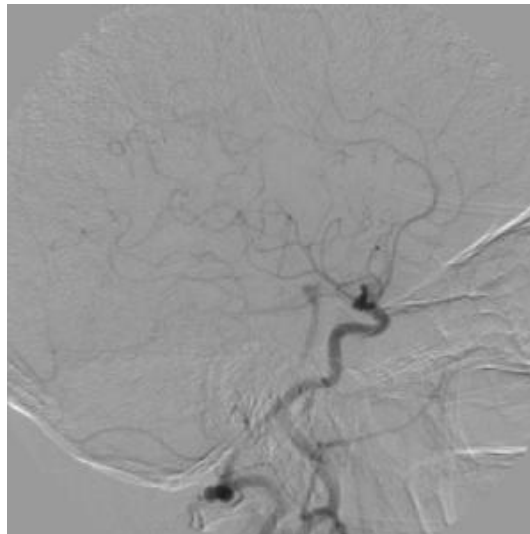
Mask  
Image



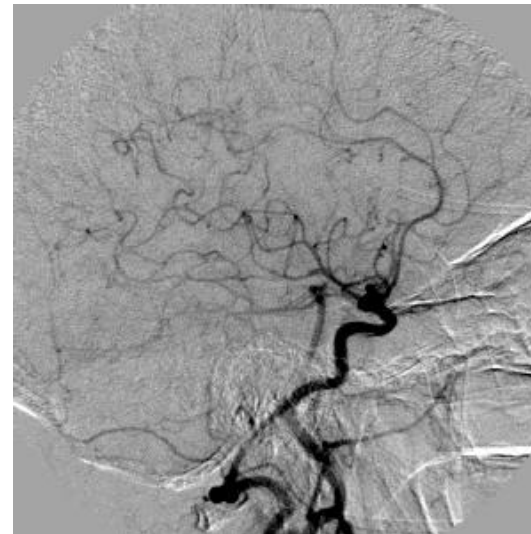
New  
Image



Difference  
Image



Contrast  
Enhanced  
Image



# Image Subtraction

---

Image  
Proto

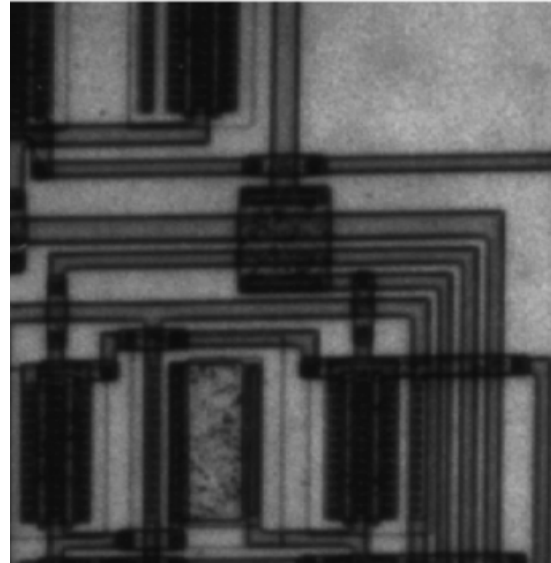
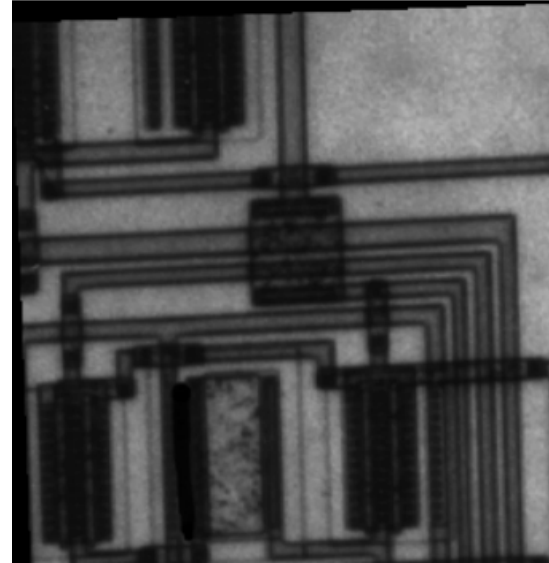
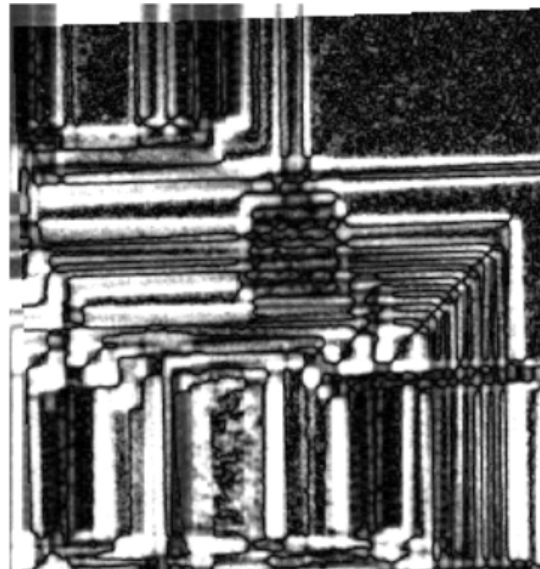


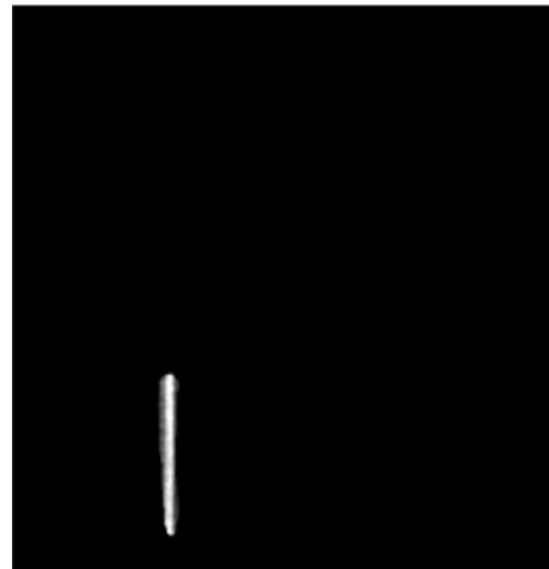
Image  
Defect



Difference  
Image  
No  
Alignment



Difference  
Image  
With  
Alignment



# Image Subtraction – Foreground Detection

---

Frame 1



Frame 2



Difference Image



Real Difference Image

