Image Processing

Lecture 07: Color Image Processing - II

(Ch6 Color Image Processing)

Zhiguo Zhang

zhiguozhang@hit.edu.cn

Contents of This Lecture

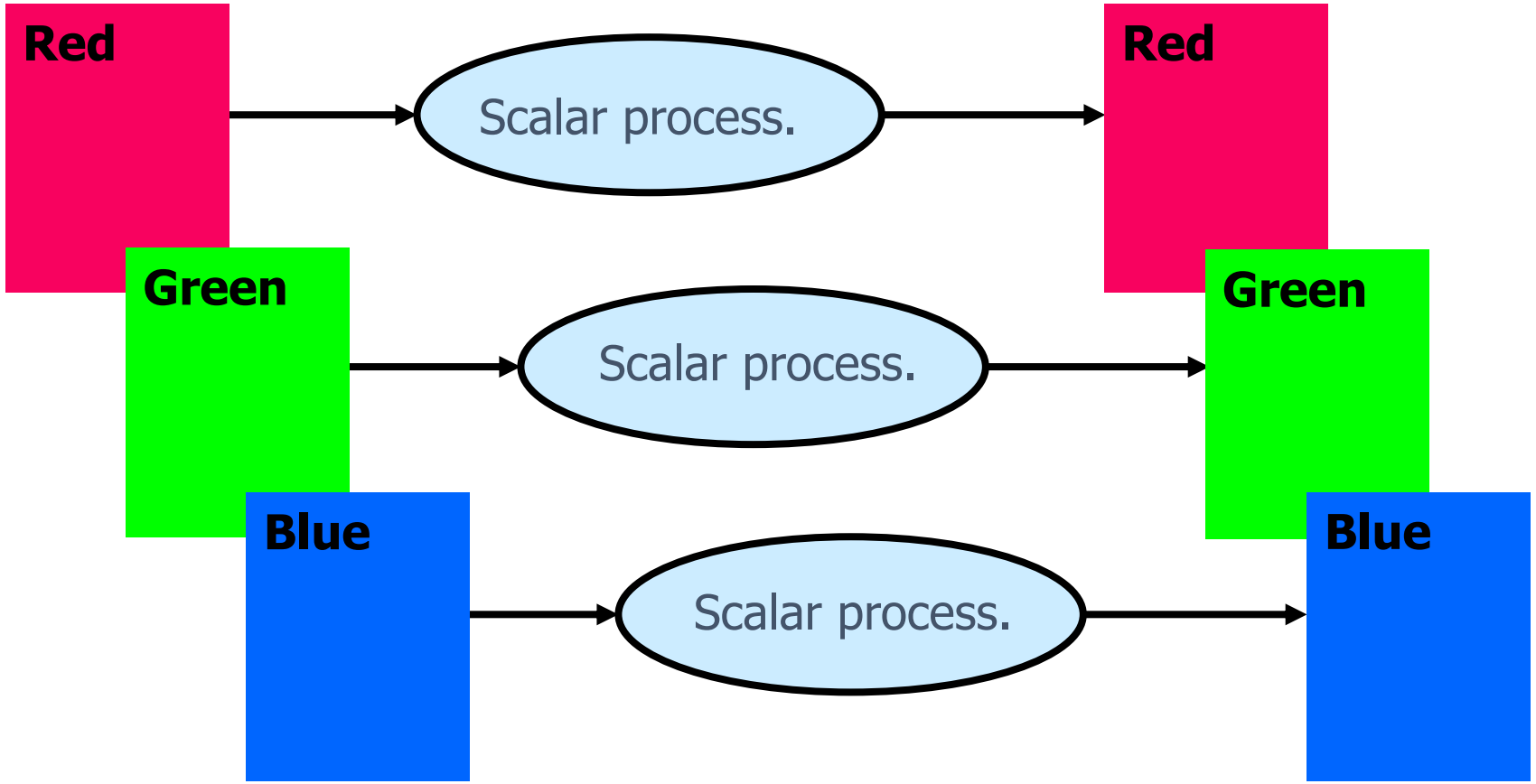
- Full-color processing
  - ✓ Color transformations
  - ✓ Smoothing and sharpening
  - ✓ Color segmentation
  - Noise in color images

Full Color Image Processing

- A full color image has a vector at each pixel. For colour images, these vectors each have 3 or 4 components.
- There are two categories to process vectorial images:
  - Category I: Per-color-component processing
    - ➢Process each component image individually
    - ➢Form a composite processed color image
  - Category II: Vector-based processing
    - ➢Work with color pixel directly
    - ➢Color pixels are vectors

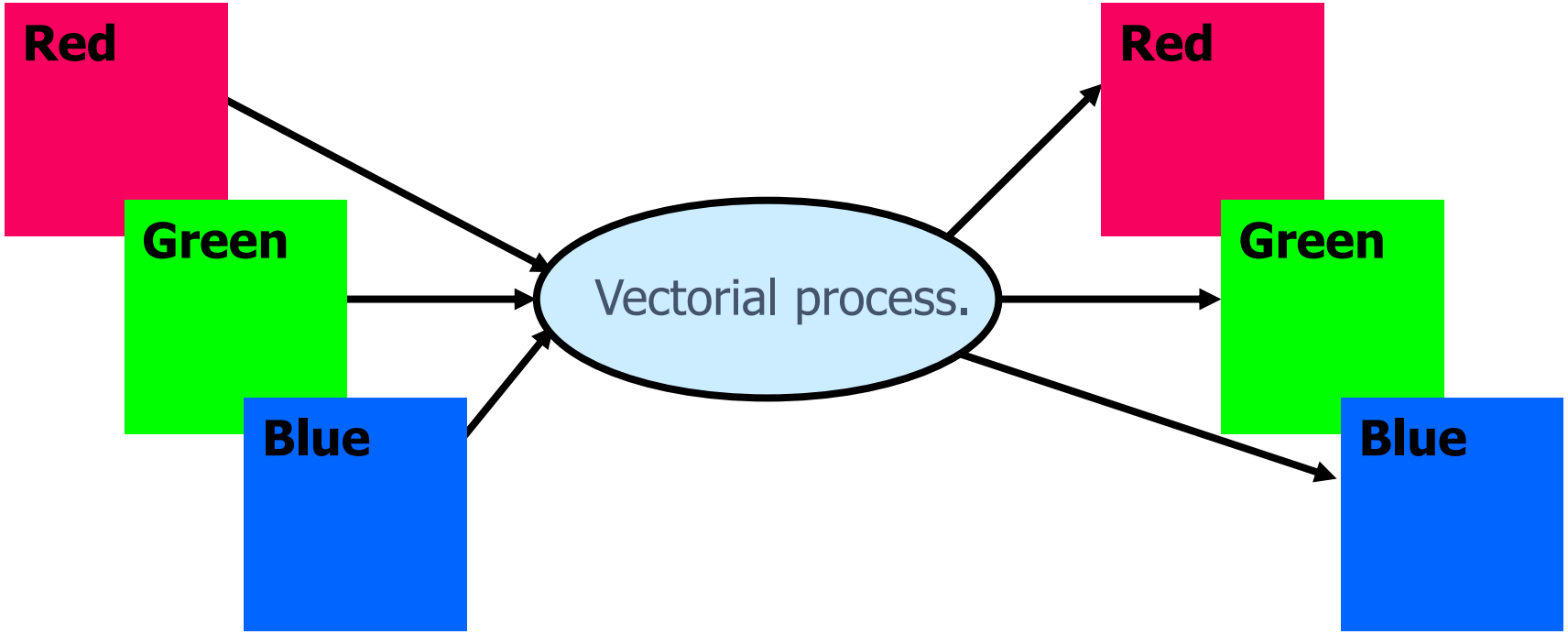
Full Color Image Processing

- Per-color-component processing: each channel is processed separately



Full Color Image Processing

- Vector-based processing: the colour triplets are processed as single units



Full Color Image Processing

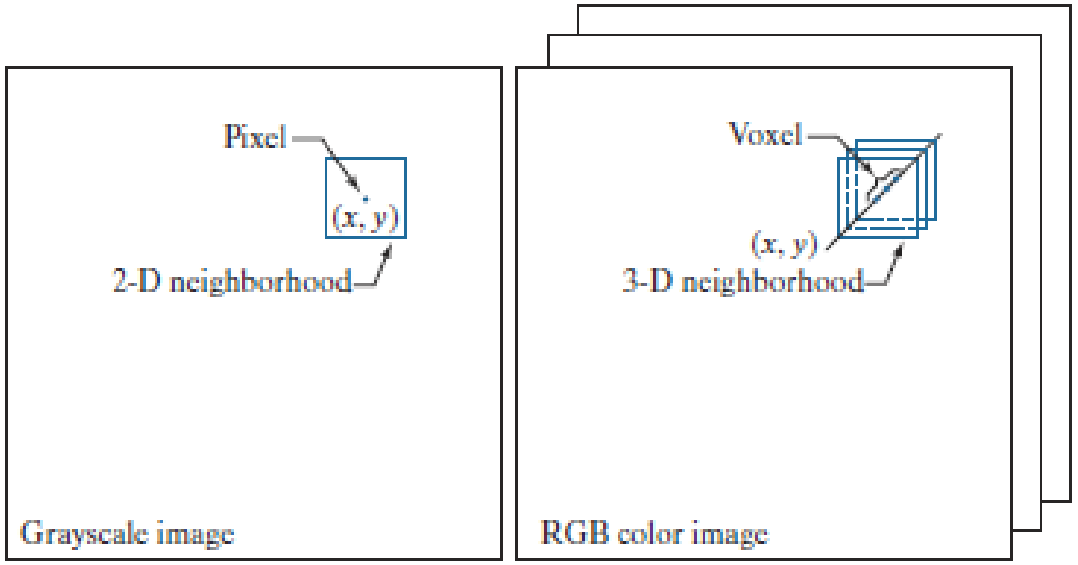
- Per-color-component and vector-based processing could be equivalent if:
  1. the process is applicable to both vectors and scalars
  2. the operation on each component of a vector is independent of other components.

For example:

Neighborhood averaging

Per-color-component processing =

Vector-based processing



Color Transformation

- Gray-scale image transformation: $g(x, y) = T[\, f(x, y)]$
- Different with gray level image, pixel values in color images are triplets or quarte

$$s_i = T_i(r_1, r_2, ..., r_n), \quad \text{for } i = 1,2,..., n$$

where r_i and s_i denote the color components of $f(x, y)$ and $g(x, y)$ at any point (x, y), n denotes the number of color components, $\{T_1, T_1, \ldots, T_n\}$ denote a set of transformation or color mapping functions.

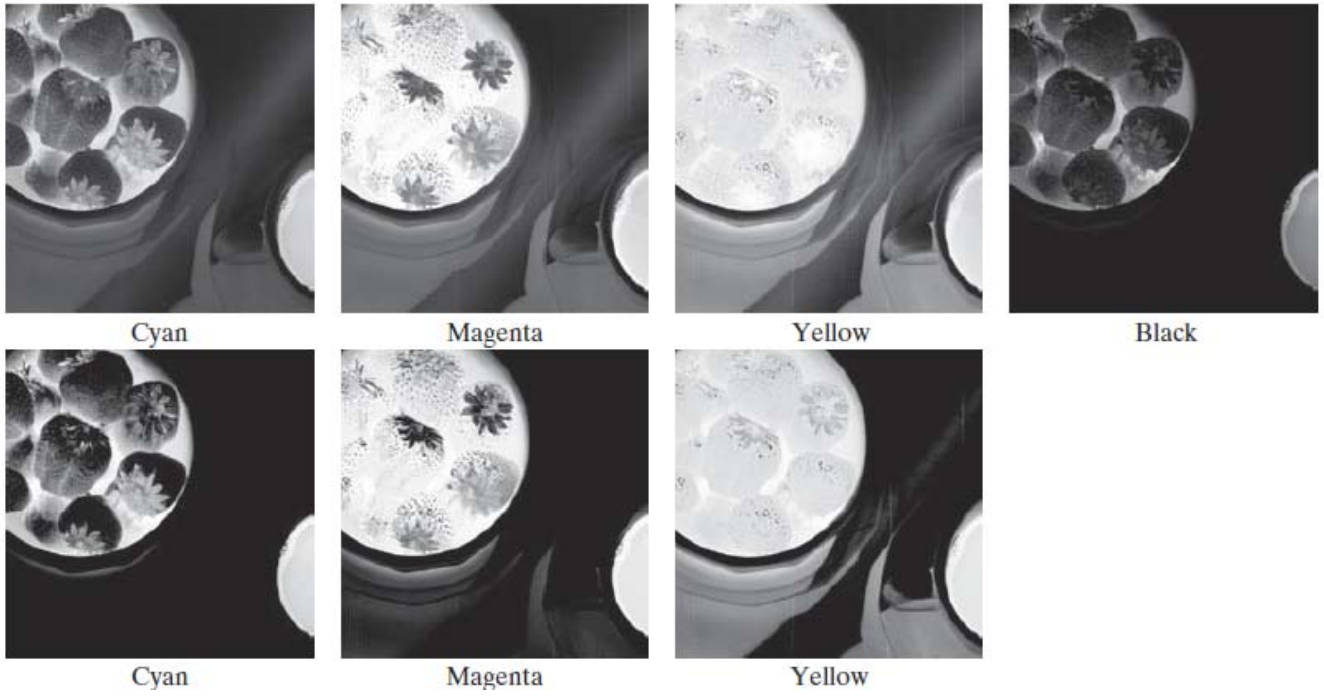
Color Transformation

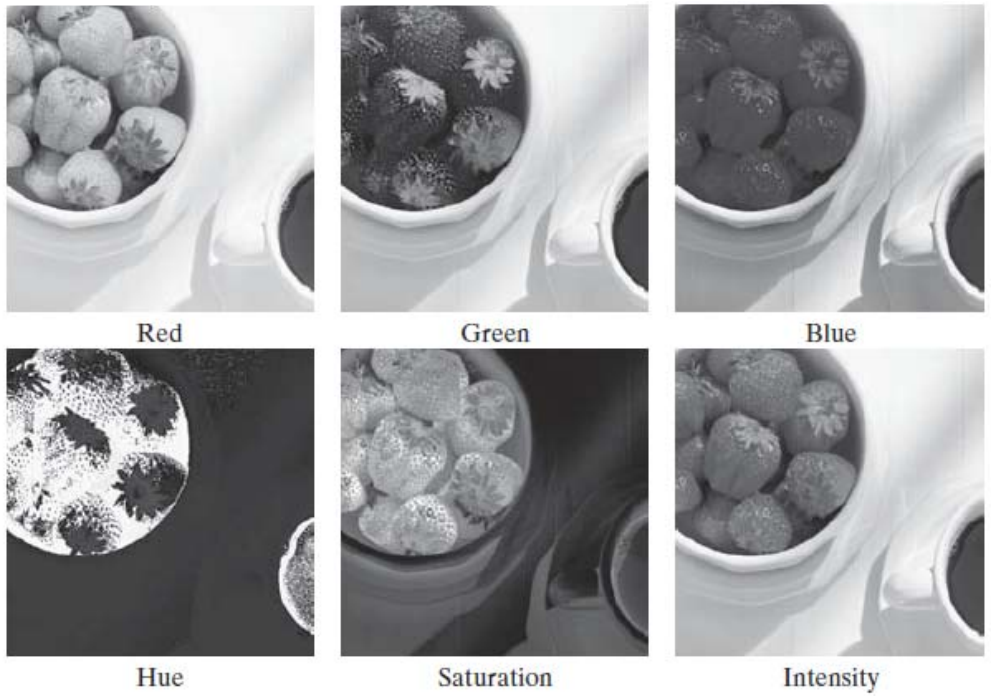
- For example:
  - RGB color space
    - ✓ $n = 3$, r_1, r_2, r_3 denote the red, green, blue
  - CMYK or HSI space are chosen
    - ✓ $n = 4$ or $n = 3$

Color Transformation

Full color image

A full-color image and its various color-space components.





Color Transformation

- Modify the intensity of the image in previous slide using:

$$g(x, y) = kf(x, y), \quad \text{where} \quad 0 < k < 1$$

- In the HSI color space, simple transformation:

$$s_3 = kr_3, \quad \text{while} \quad s_1 = r_1 \text{ and } s_2 = r_2$$

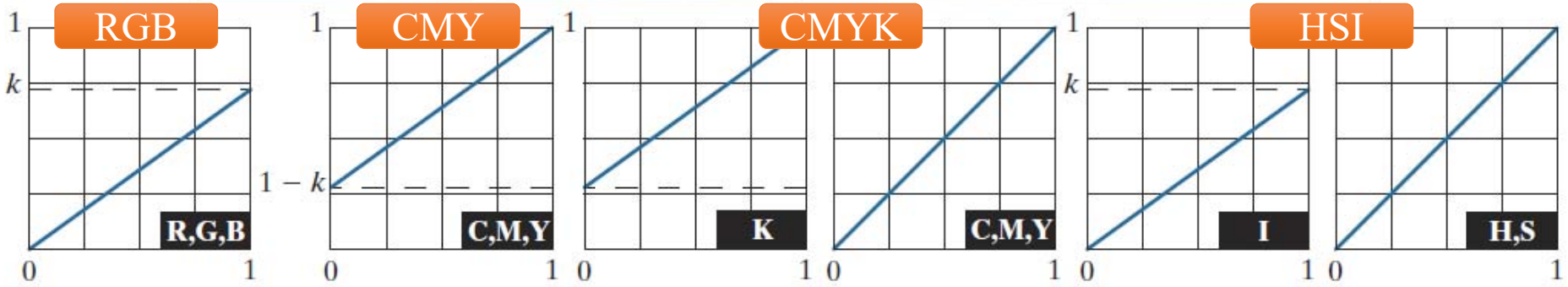
- In the RGB color space, three components must be transformed:

$$s_i = kr_i, \quad \text{for } i = 1, 2, 3$$

- In the CMY space requires a similar set of linear transformations:

$$s_i = kr_i + (1 - k), \quad \text{for } i = 1, 2, 3$$

Color Transformation



$s_i = kr_i$ $\quad s_i = kr_i + (1-k)$ $\quad s_3 = kr_3$

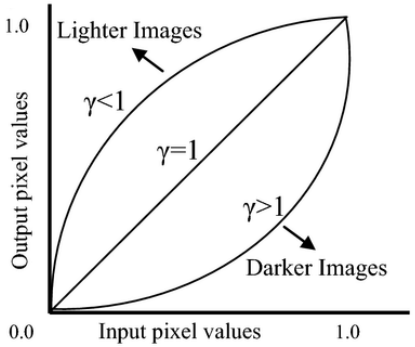
Color Transformation

- Brightness Enhancement: Gamma Correction

  HSI Space, I Component:

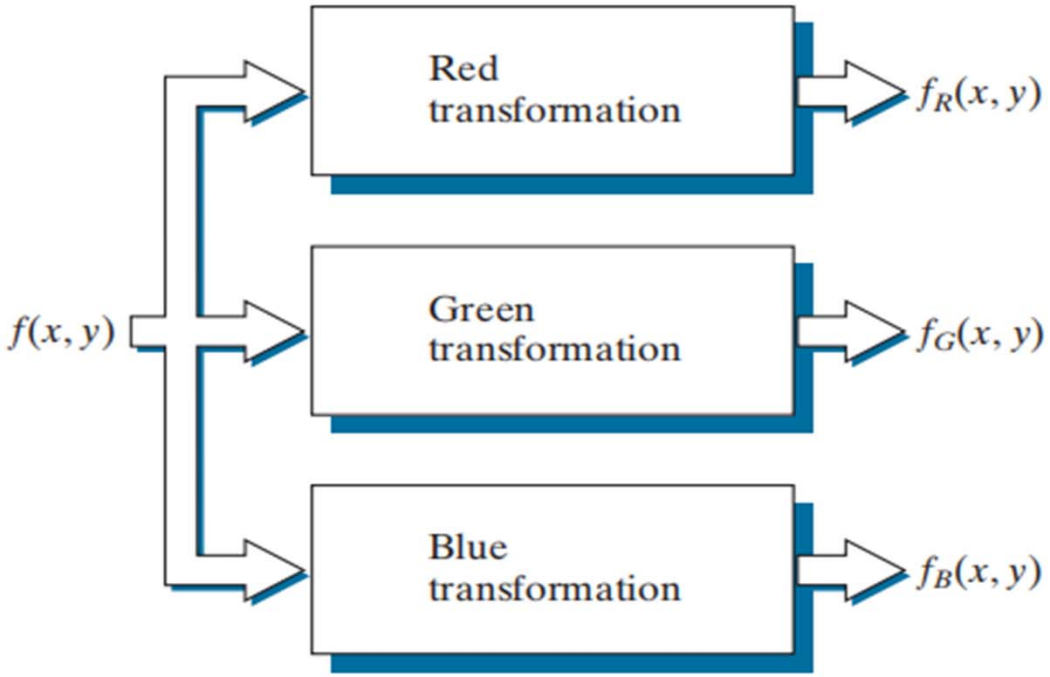
$$s = r^{\gamma} \quad r \in [0,1]$$

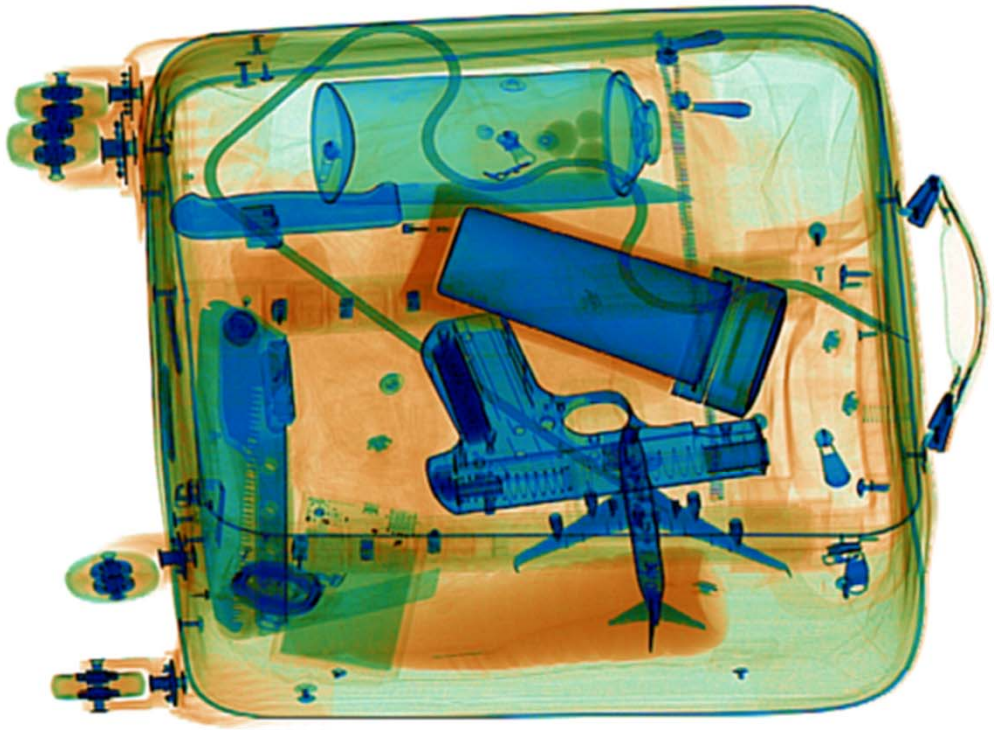
$\gamma > 1$



$\gamma < 1$

Color Transformation





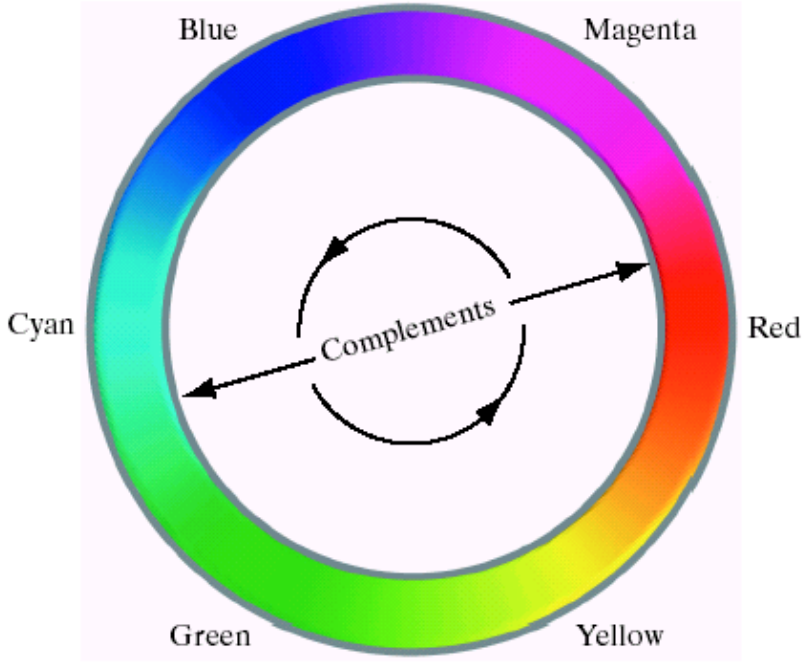
Color Transformation

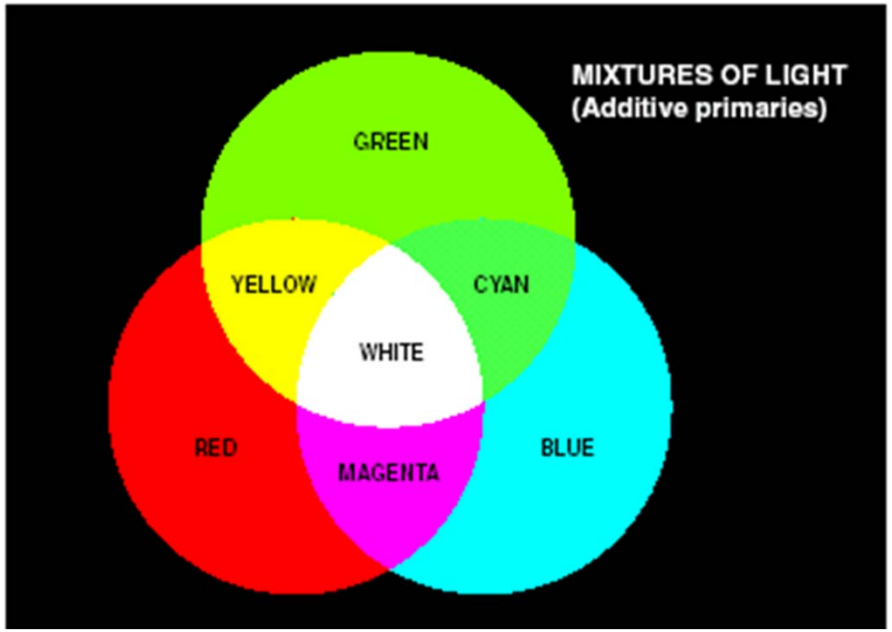
Remarks

- Some transformations can be performed in any color model, but actually, some operations are better suited to specific models.
    - HSI transformation involves the fewest number of operations.
    - But the computations required to convert an RGB or CMYK image to HSI space more than offsets the advantages of the simpler transformation.

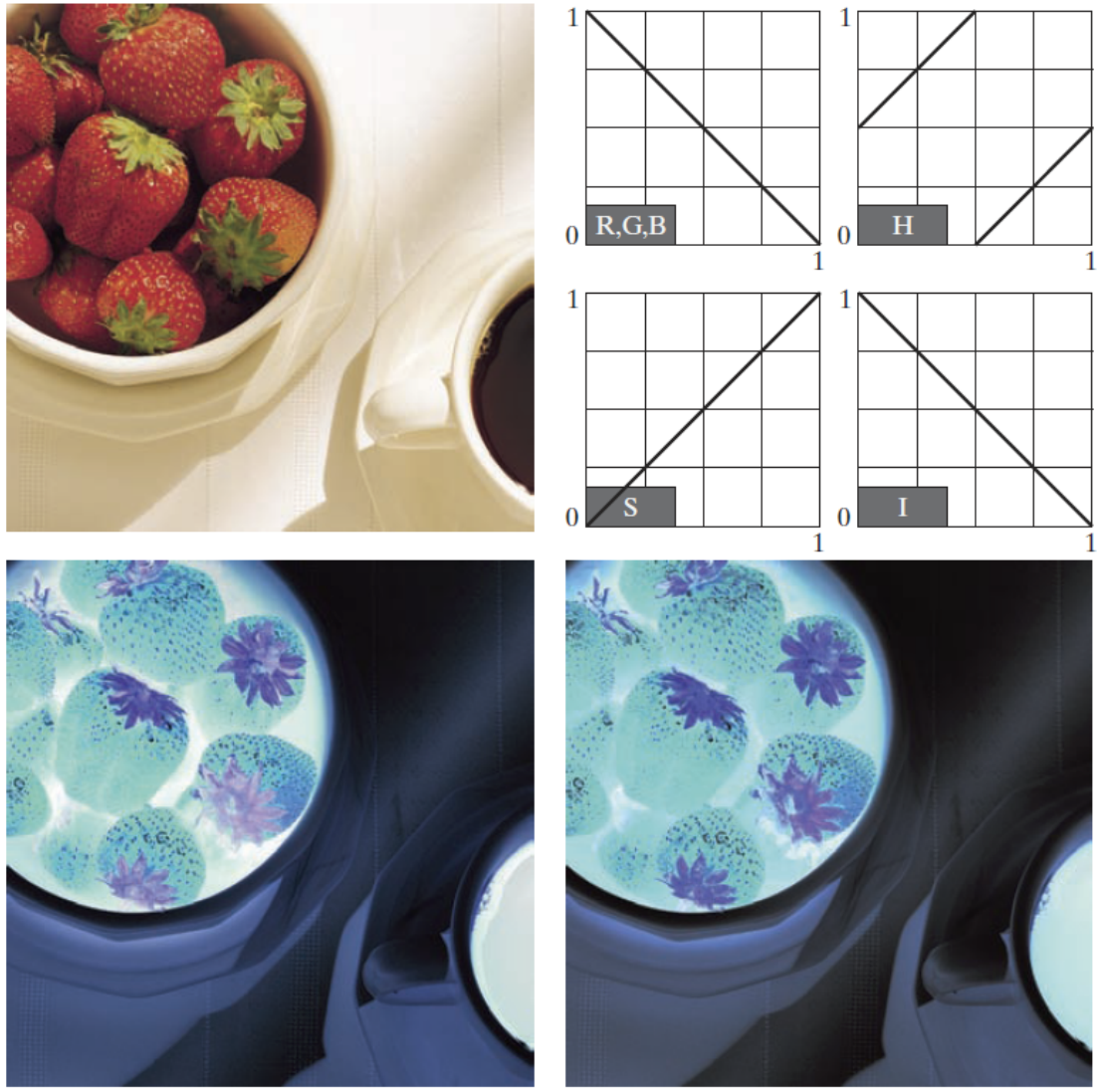
Color Complements

- Color complements are the same as gray level negative.
- Color complements are useful for enhancing detail that is embedded in dark regions of a color image.





Color Complements



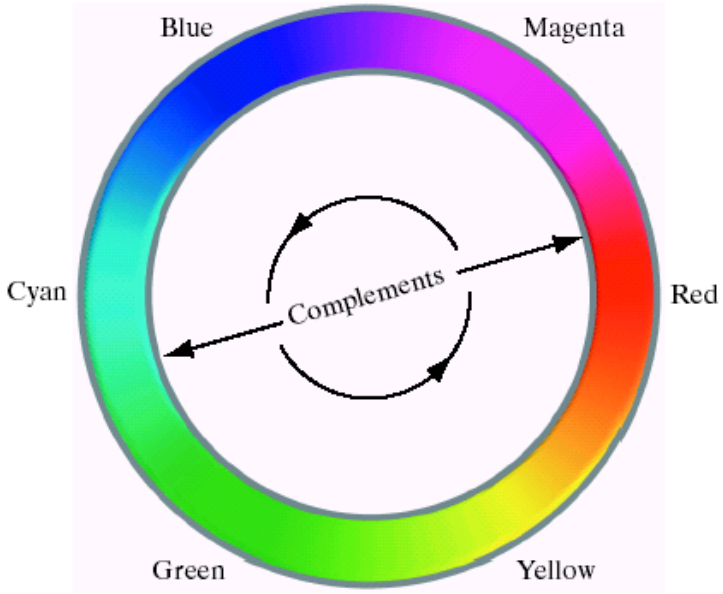
a b
c d

Color complement transformations.
(a) Original image. (b) Complement transformation functions.
(c) Complement of (a) based on the RGB mapping functions. (d) An approximation of the RGB complement using HSI transformations.

Note: RGB complement transformation functions used in this example do not have a straightforward HSI equivalent.

Color Complements

- Reducing the red color plane moves the color balance towards cyan.
- Reducing the green color plane moves the color balance towards magenta.
- Reducing the blue color plane moves the color balance towards yellow.



Color Slicing

- Color slicing can highlight a specific range of colors.
  - To separate objects from their surrounds.
- Basic ideas:
  - To display the colors of interest.
  - To use the regions defined by the colors as a mask for further processing.
- More complicated than gray-level counterparts.
- Require each pixel's transformed color components to be a function of all *n* original pixel's color components.

Color Slicing

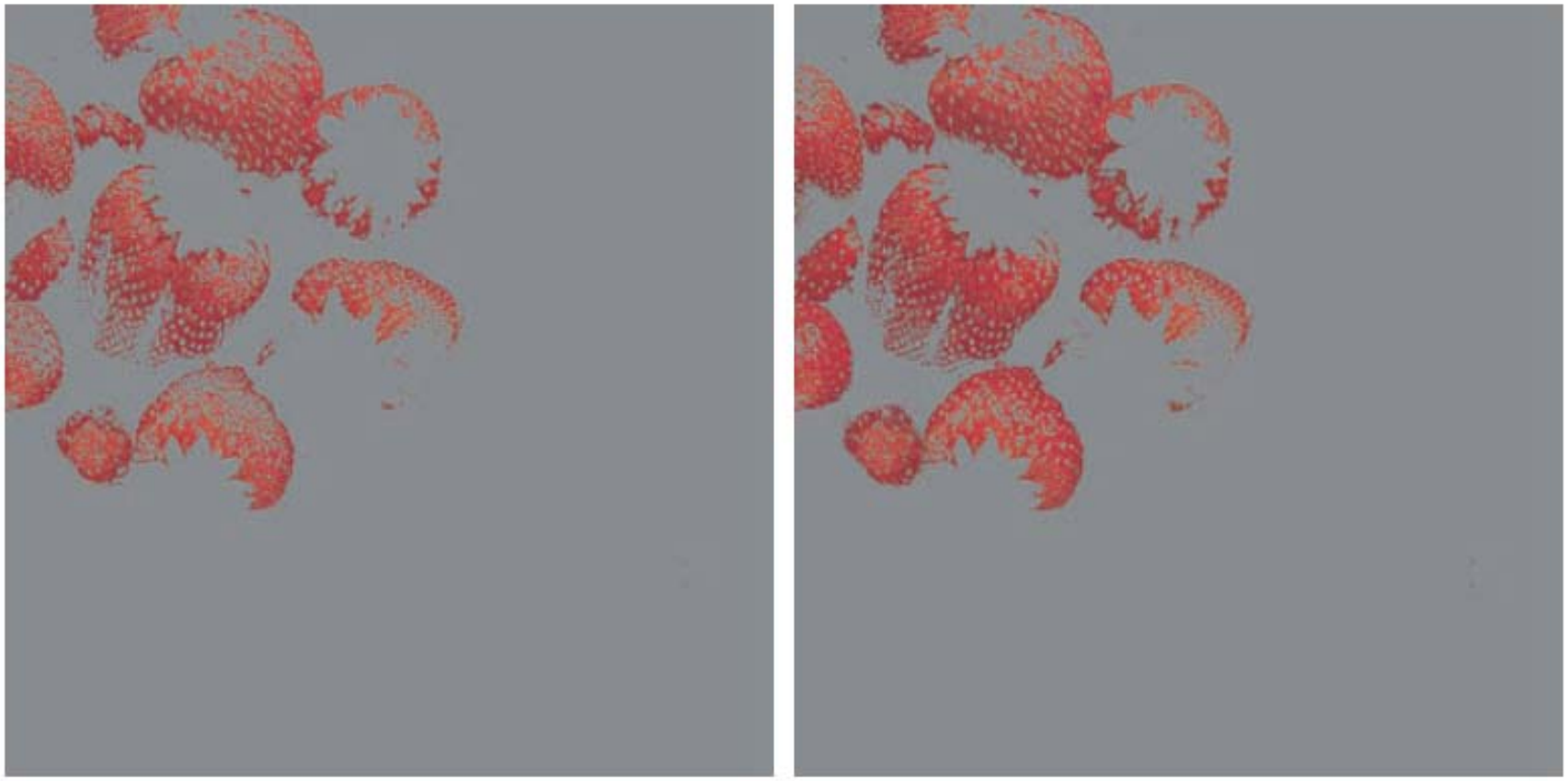
- The simplest way: Map the colors outside some range of interest to a non-prominent neutral color.
- If the colors of interest are enclosed by a cube of width W and centered at a prototypical color with components $(a_1, a_2, \ldots, a_n)$, the necessary set of transformation is:

$$s_i = \begin{cases} 0.5, \text{if } |r_i - a_j| > W/2 \text{ for any } 1 \le j \le n, i = 1,2,...,n \\ r_i, \text{ otherwise} \end{cases} \quad (1)$$

- If a sphere is used to specify the colors of interest, becomes:

$$s_i = \begin{cases} 0.5, \text{if } \sum_{j=1}^{n} (r_i - a_j)^2 > R_0^2 \text{ for } i = 1,2,...,n \\ r_i, \text{ otherwise} \end{cases} \quad (1)$$

Color Slicing

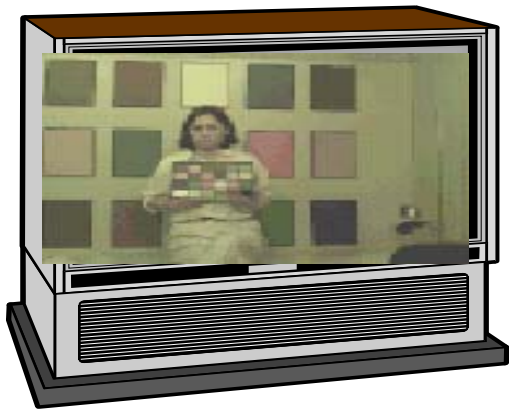


Color-slicing transformations that detect (*left*) reds within an RGB cube of width $W = 0.2549$ centered at (0.6863, 0.1608, 0.1922), and (*right*) reds within an RGB sphere of radius 0.1765 centered at the same point. Pixels outside the cube and sphere were replaced by color (0.5, 0.5, 0.5).

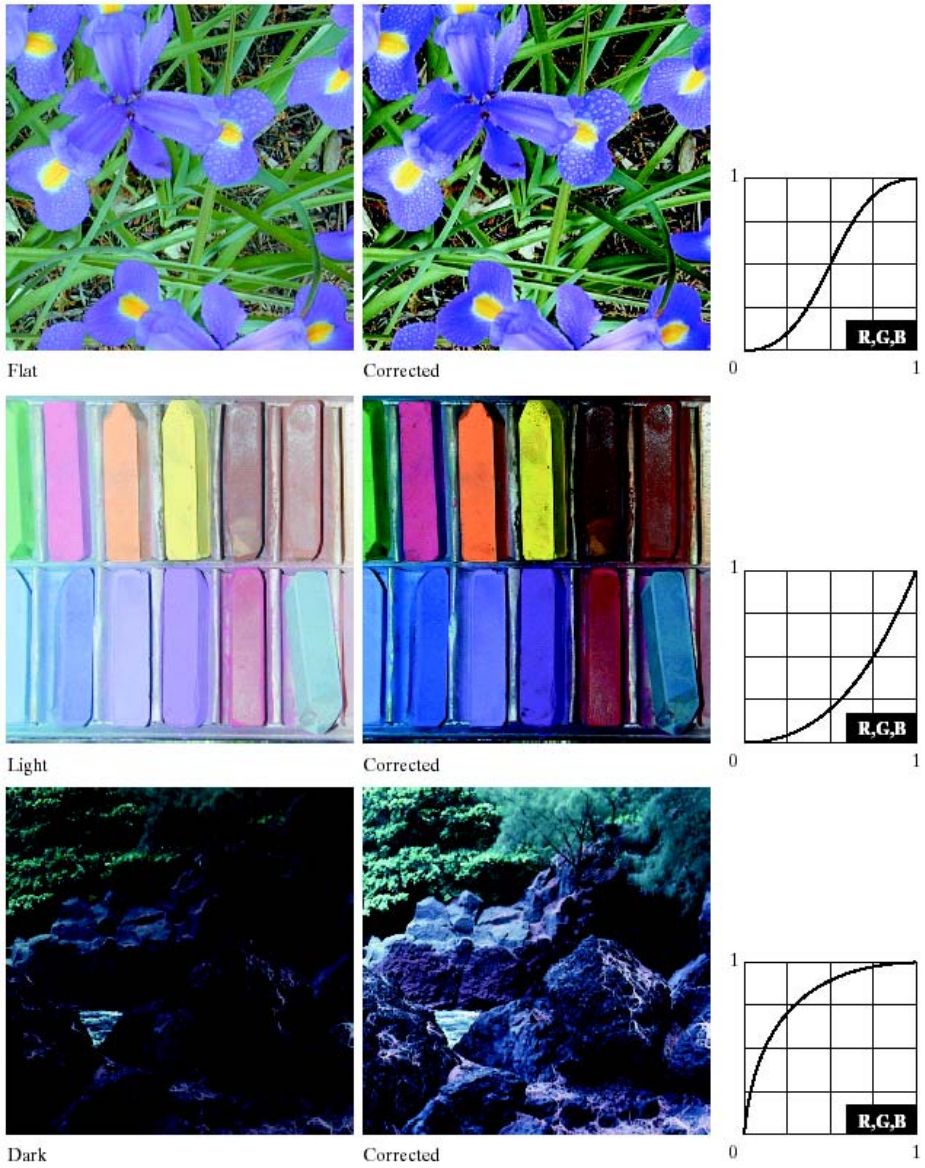
Color Slicing

Tone and Color Corrections

- Most Common Uses
  - Photo enhancement
  - Color reproduction
- Consistent: since these transformations are developed, refined, and evaluated on monitors, it is necessary to maintain high color consistency between monitor and output devices.

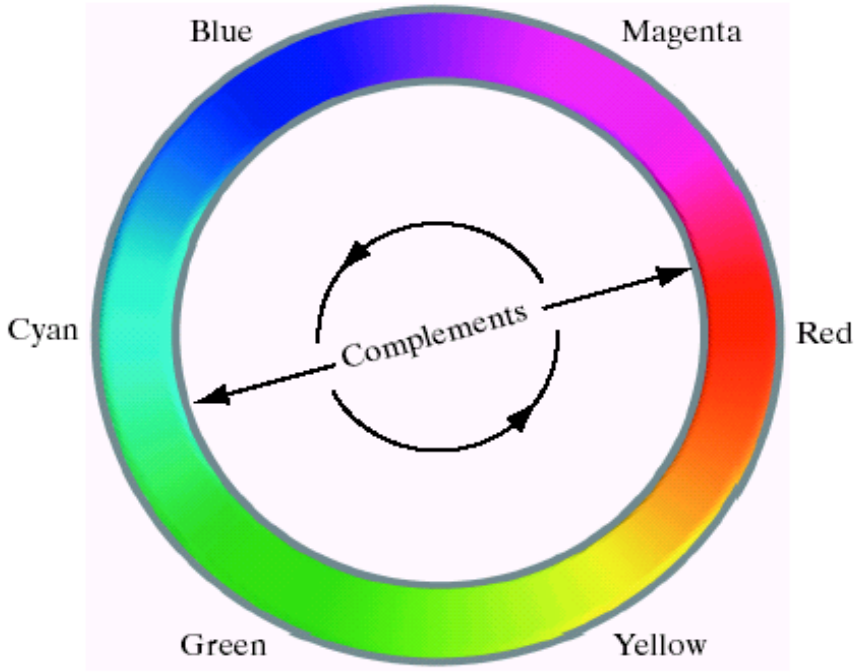


Tone and Color Corrections



- Tonal corrections for flat, light (high key), and dark (low key) color images. Adjusting the red, green, and blue components equally does not always alter the image hues significantly.
- The principle of calibrated imaging systems is that they allow intensity and color imbalances to be corrected interactively and independently.
- That is, in two sequential operations.

Color Corrections

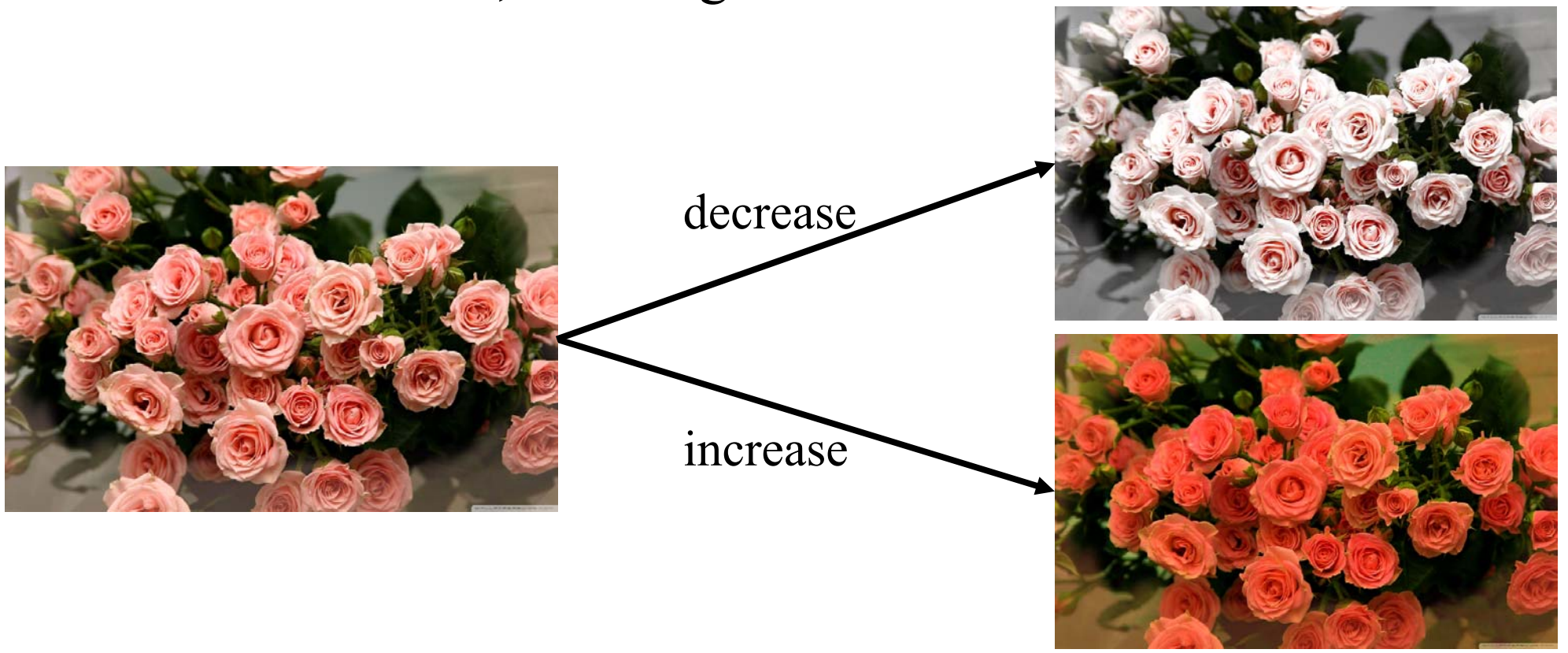


- Note: Every action affects the overall color balance of the image, because the perception of one color is affected by its surrounding colors.

Color balancing a CMYK image

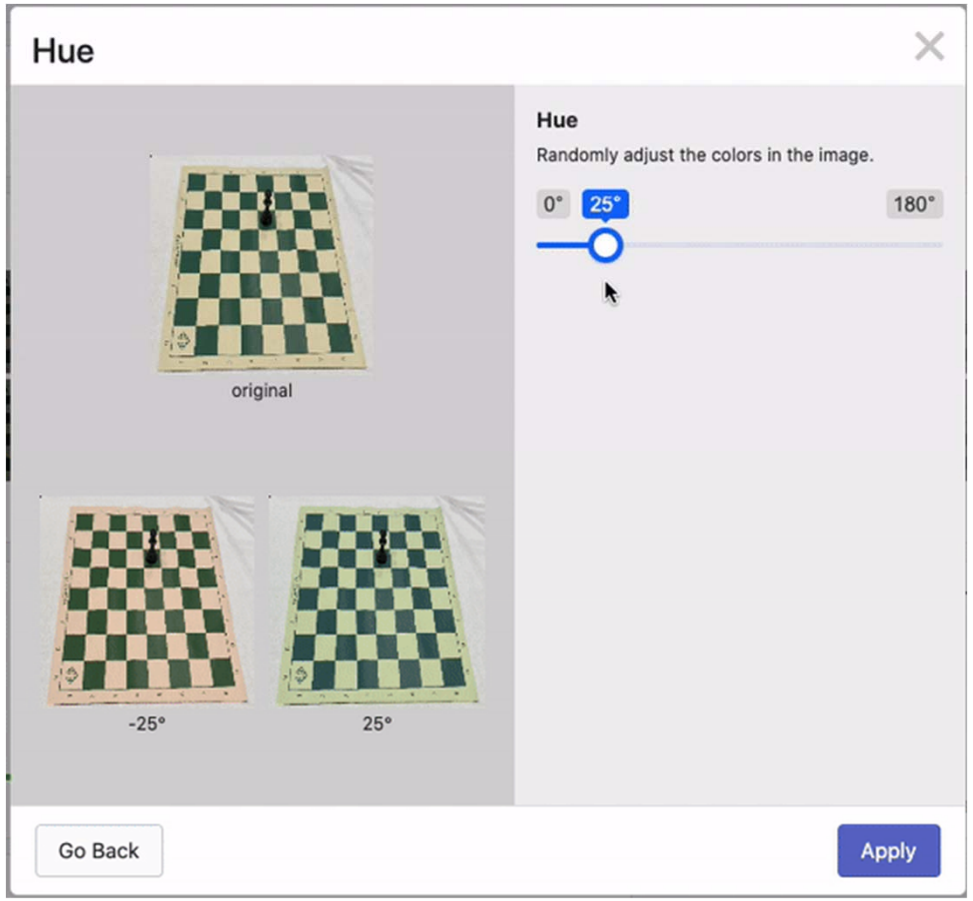
Saturation Enhancement

- Decrease the saturation, the image color becomes lighter, the original lighter area has become gray.
- Increase the saturation, the image becomes more colorful.



Hue Enhancement

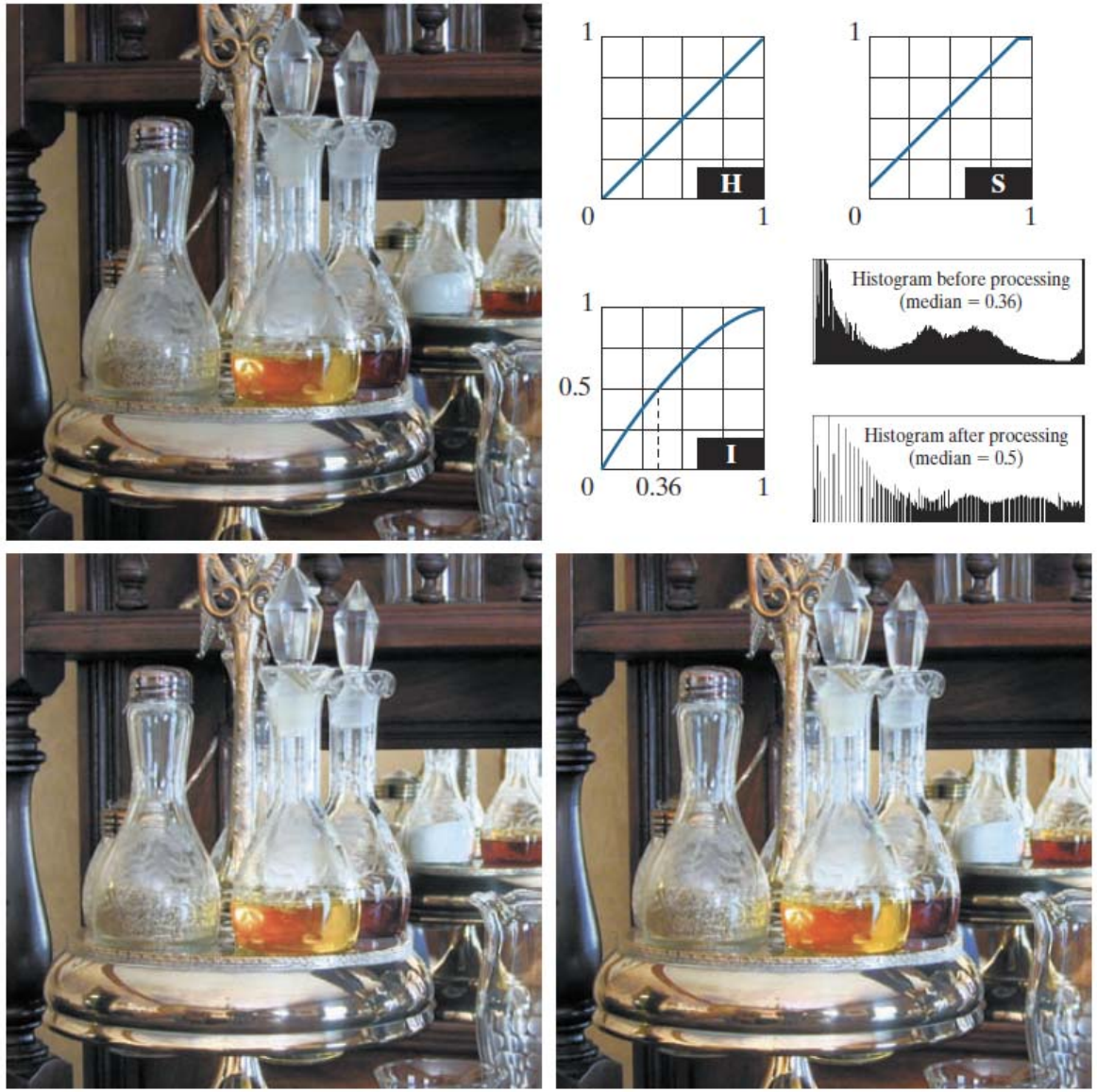
- Hue enhancement is special compared to brightness and saturation enhancement.
- In HSI space, hues correspond to angles and are cyclic.
- If the hue as a whole adds or subtracts a number, the color moves across the color spectrum. When the number is small, the hue becomes cooler or warmer; when the number is large, the hue changes dramatically.



Histogram Processing

- Automated way, and very efficient in gray image processing.
- Unwise to histogram equalize the components of a color image independently (lead to erroneous color).
- A more logical approach
  - Spread the color intensities uniformly.
  - Leaving the colors (e.g., hue) unchanged.

Histogram Processing



Histogram equalization (followed by saturation adjustment) in the HSI color space.

Histogram Processing

Original image

Histogram equalization in RGB space

Histogram equalization in HSI space

Smoothing and Sharpening

- Modify value based on the characteristics of the surrounding pixels.
  - Smoothing
  - Sharpening

- Gray image: Each pixel is replaced by the average of the pixels in the neighborhood defined by the mask.
- Color image smoothing: Deal with component vectors.

Color Image Smoothing

- Let S_{xy} denote the set of coordinates defining a neighborhood centered at (x, y) in an RGB color images.
- The average of the RGB component vectors in this neighborhood is

$$\bar{c}(x,y) = \frac{1}{K}\sum_{(x,y)\in S_{xy}} c(x,y)$$

- The properties of vector addition that

$$\bar{c}(x,y) = \frac{1}{K}\begin{bmatrix} \sum_{(x,y)\in S_{xy}} R(x,y) \\ \sum_{(x,y)\in S_{xy}} G(x,y) \\ \sum_{(x,y)\in S_{xy}} B(x,y) \end{bmatrix}$$

Note: The results are the same when smoothing each plane of starting RBG image using conventional gray scale neighborhood processing.

Color Image Smoothing

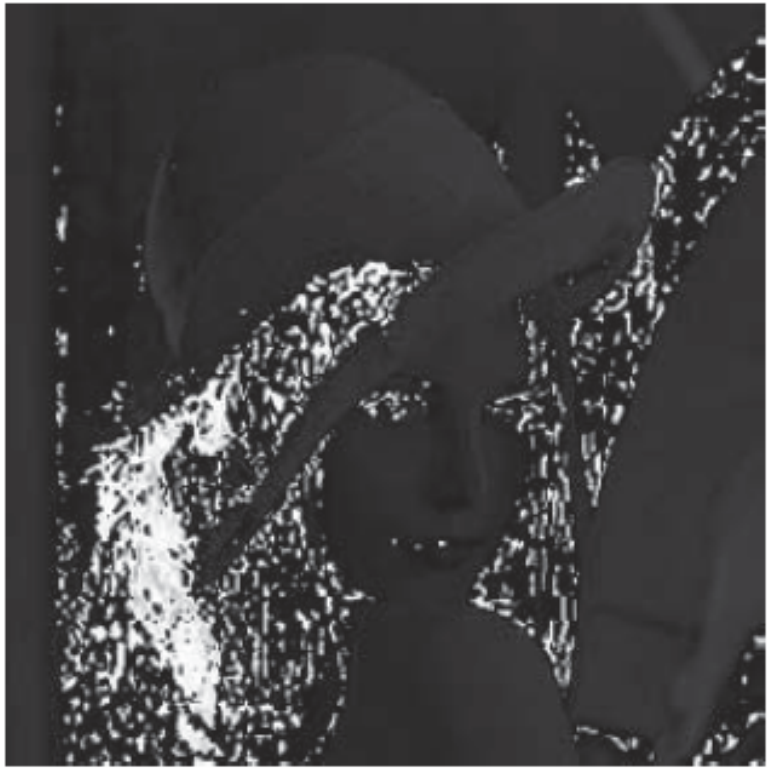
RGB image

Red component

Green component

Blue component

Color Image Smoothing



H

S

I

HSI components of the RGB color image

Color Image Smoothing

Original Image

Result of smoothing each RGB component with a 5 × 5 averaging kernel

Result of smoothing the intensity component of the HSI image with a 5 × 5 averaging kernel and converting to RGB

Difference between the two results

Color Image Smoothing

RGB smoothing

I smoothing

HSI smoothing

Wrong!

Color Image Smoothing

Color Image Sharpening

- In the RGB color system, the Laplacian of vector **c** is

$$\nabla^2[\mathbf{c}(x,y)] = \begin{bmatrix} \nabla^2[R(x,y)] \\ \nabla^2[G(x,y)] \\ \nabla^2[B(x,y)] \end{bmatrix}$$

- We can compute the Laplacian of a full-color image by computing the Laplacian of each component image separately.

Color Image Sharpening

Original Image

Result of sharpening each RGB component using the Laplacian

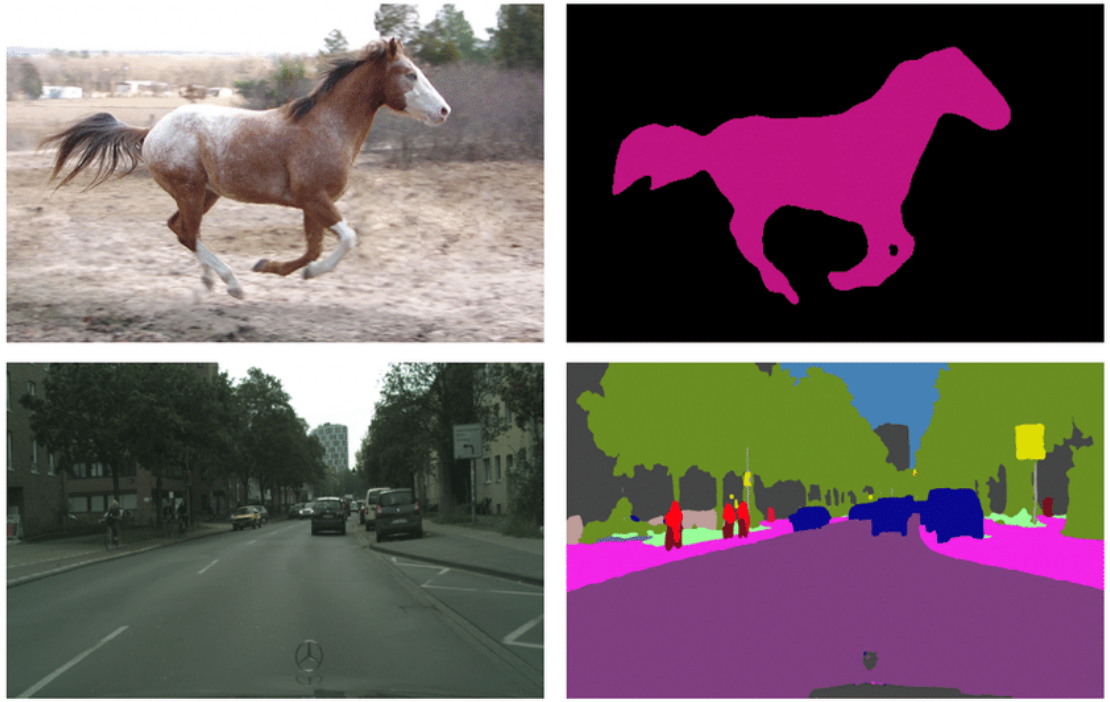
0	-1	0
-1	5	-1
0	-1	0

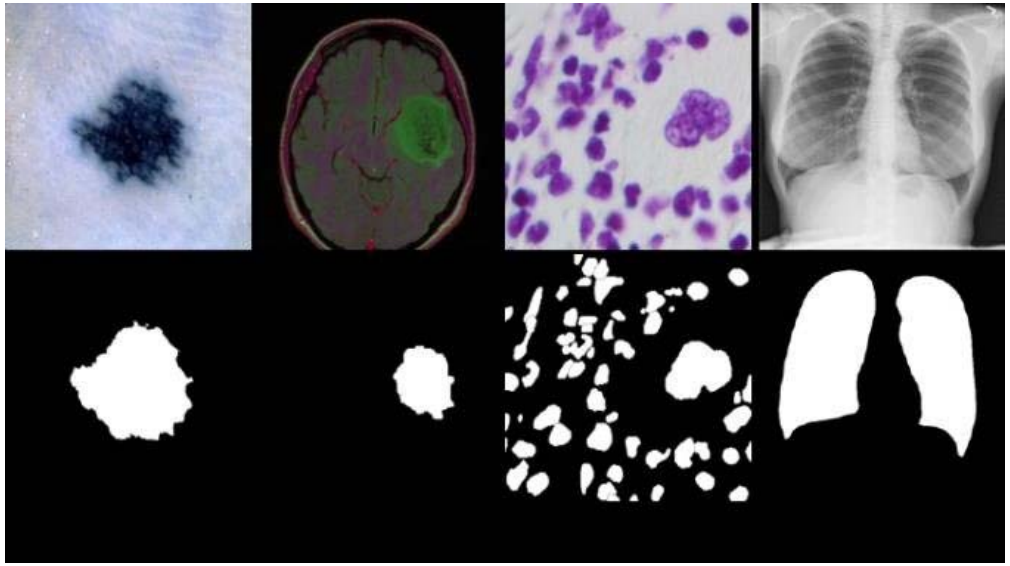
Result of sharpening the intensity component of the HSI image using the Laplacian and converting to RGB

Difference between the two results

Color Image Segmentation

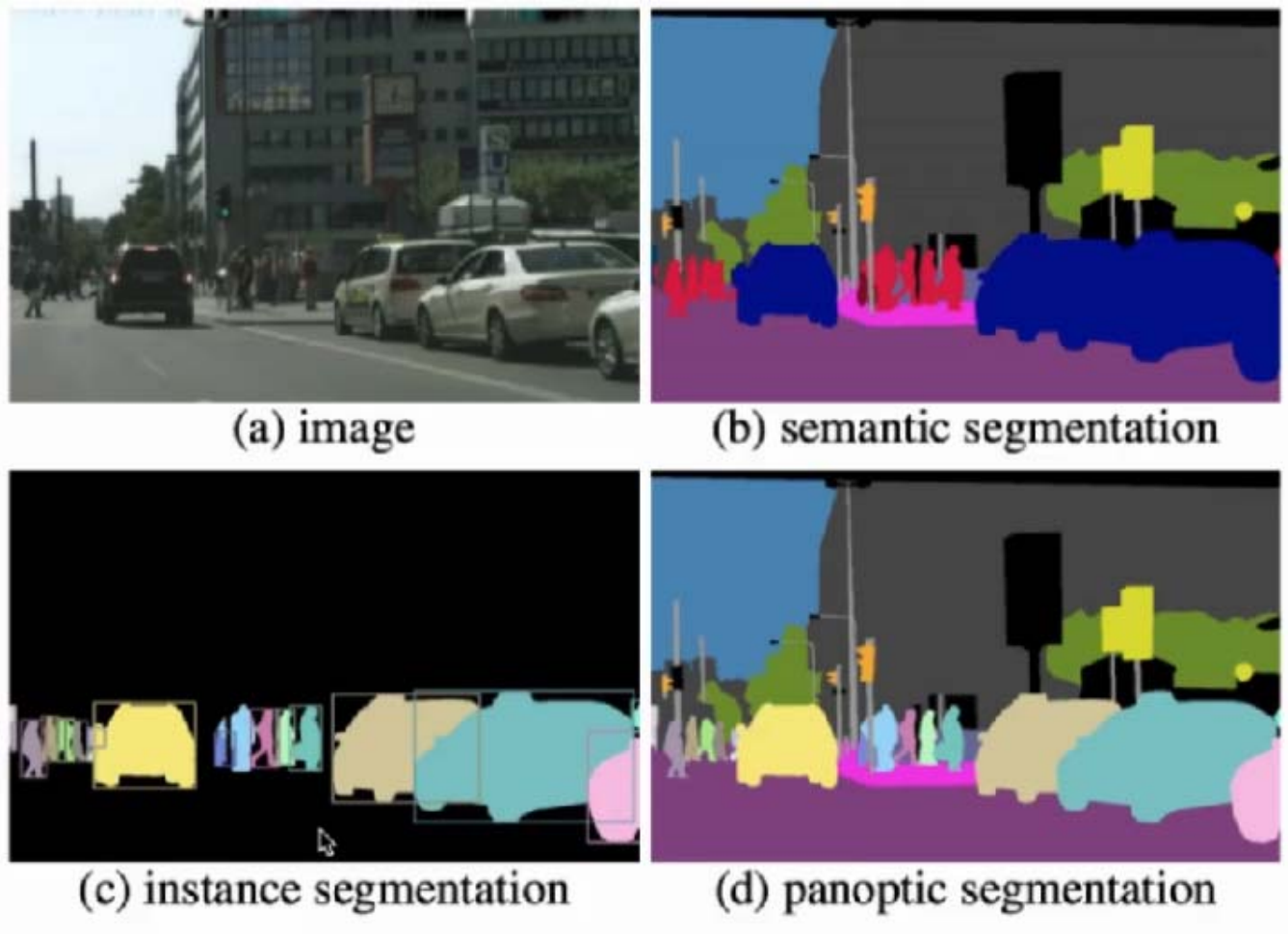
- Segmentation: Partition an image into regions
- Image segmentation is a classical problem and also one of the most difficult problems in image processing.





Color Image Segmentation

- Image segmentation: pixel-level description of an image, including
  - semantic segmentation
  - instance segmentation
  - panoptic segmentation

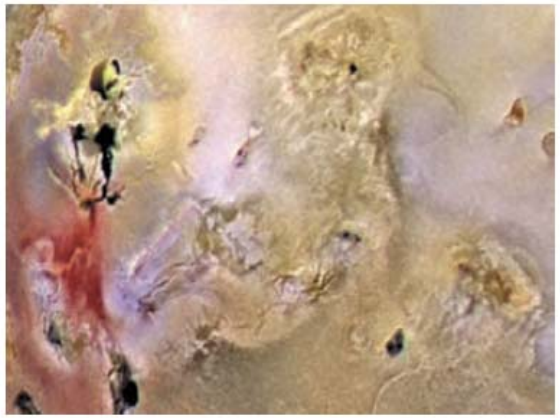


Segmentation in HSI Space

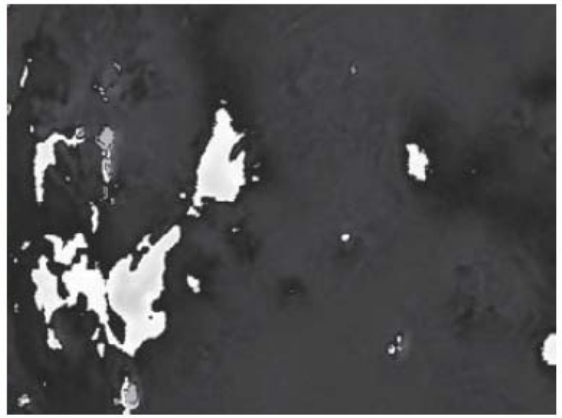
- An image can be segmented in any color space (RGB, HSI, CMY, etc.).
- Image segmentation based on colors in HSI space:
  - Color is conveniently represented in the hue image.
  - Saturation is used as a mask image in order to isolate further regions of interest in the hue image.
  - Intensity is infrequently used for segmentation, since it carries no color information.

Segmentation in HSI Space

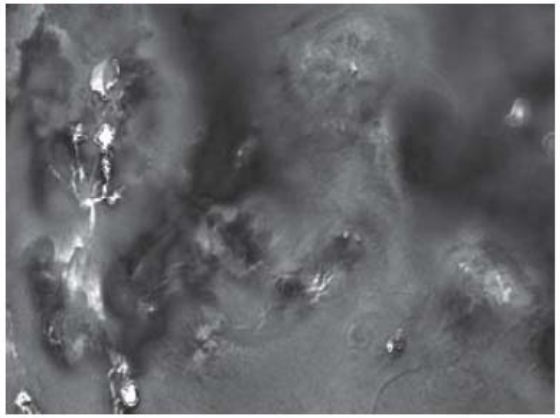
(a) Original



(b) Hue

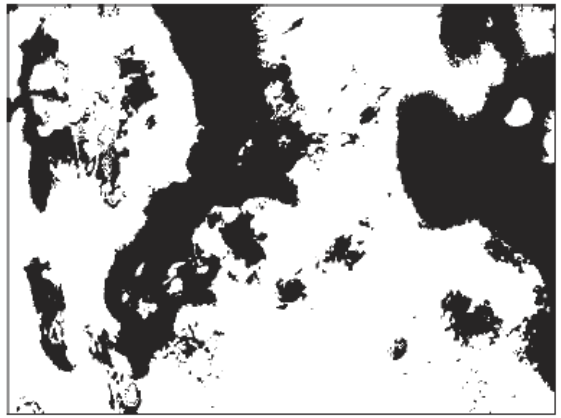


(c) Saturation

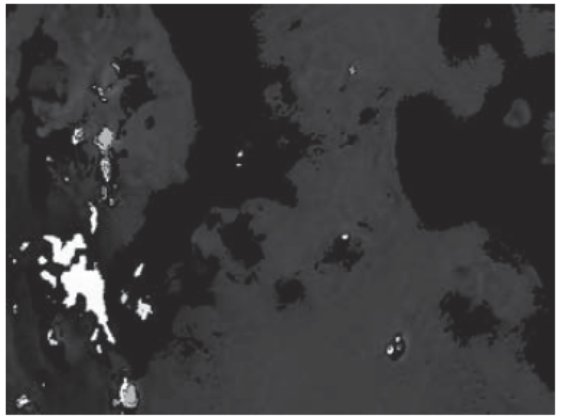


(d) Intensity

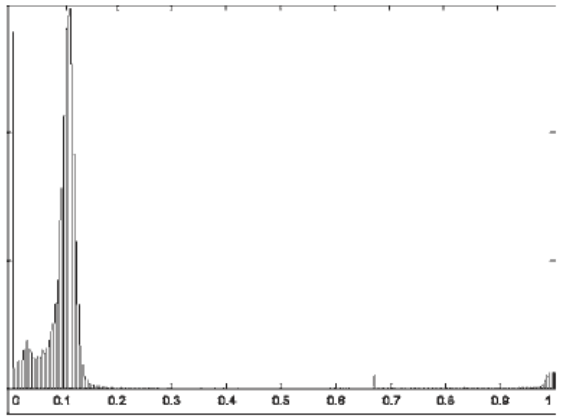
(e) Binary saturation mask (black = 0)



(f) Product of (b) and (e)



(g) Histogram of (f)



(h) Segmentation by thresholding (f)

Segmentation in RGB Space

- Segmentation in RGB Space
  - HSI space: more intuitive.
  - RGB space: better results.

- Objective: Segment objects of a specified color range in an RGB image.
- Given a set of sample color points representative of the colors of interest, estimate the “average” color that we wish to segment.

Segmentation in RGB Space

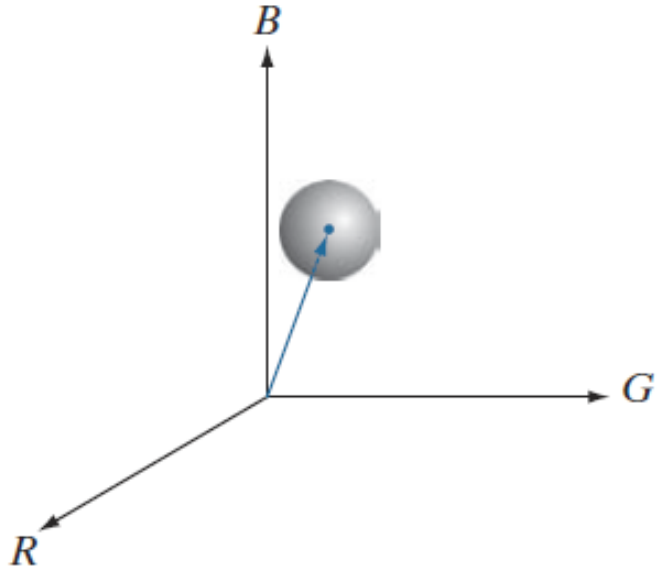
- Let the average color be denoted by the RGB vector **a**.
- Using a similarity-measure to identify whether a color in the specified range.
- One of the simplest measures is the Euclidean distance.

$$D(\mathbf{z},\mathbf{a}) = \|\mathbf{z}-\mathbf{a}\|$$

$$= \left[(\mathbf{z}-\mathbf{a})^T(\mathbf{z}-\mathbf{a})\right]^{\frac{1}{2}}$$

$$= \left[(\mathbf{z}_R-\mathbf{a}_R)^2+(\mathbf{z}_G-\mathbf{a}_G)^2+(\mathbf{z}_B-\mathbf{a}_B)^2\right]$$

- Let **z** denote an arbitrary point in RGB space. **z** is similar to **a** if the distance between them is less than a specified threshold, D_0.

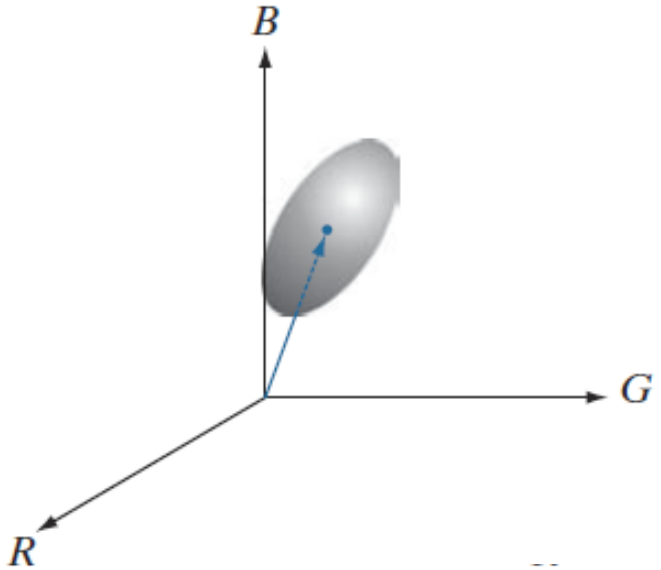


Segmentation in RGB Space

- A generalization of Euclidean distance is Mahalanobis distance:

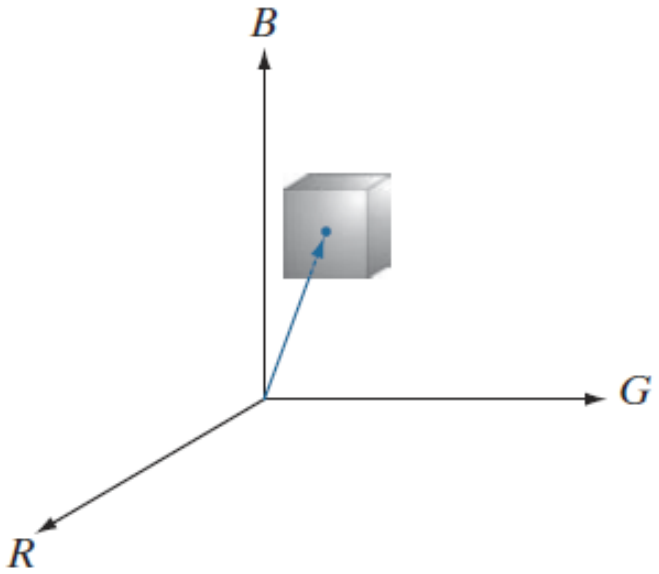
$$D(\mathbf{z},\mathbf{a}) = \left[(\mathbf{z}-\mathbf{a})^T \boldsymbol{C}^{-1}(\mathbf{z}-\mathbf{a})\right]^{\frac{1}{2}}$$

where **C** is the covariance matrix of the samples representative of the color we wish to segment. When **C** = **I**, Mahalanobis distance is just the Euclidean distance.



Segmentation in RGB Space

- However, implementing of last two methods is computationally expensive for images of practical size, even if the square roots are not computed.
- A compromise is to use a bounding box.
- Given an arbitrary color point, we segment it by determining whether or not it is on the surface or inside the box.



Segmentation in RGB Space

- Select samples by a rectangular region, compute its mean vector and standard deviation vector in the RGB space as $[a_R, a_G, a_B]^T$ and $[\sigma_R, \sigma_G, \sigma_B]^T$, respectively.
- The bounding box could be obtained as:

$$(a_R - 1.25\sigma_R) \sim (a_R + 1.25\sigma_R)$$

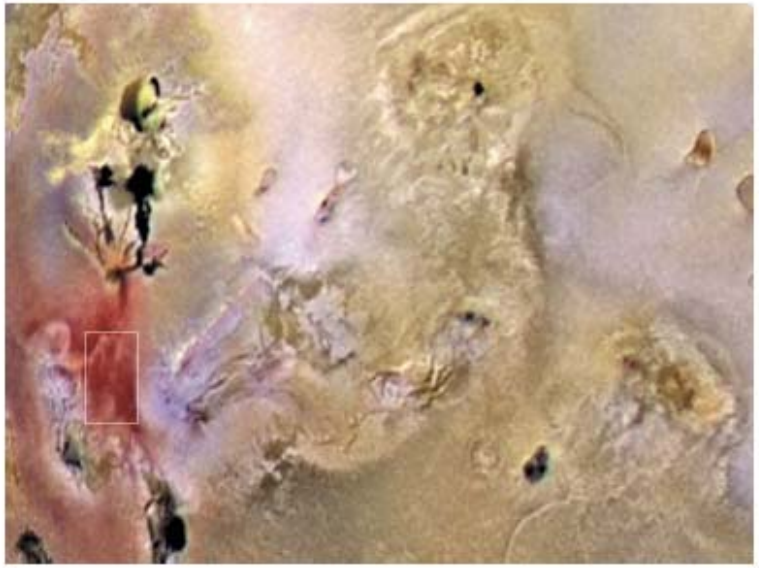
$$(a_G - 1.25\sigma_G) \sim (a_G + 1.25\sigma_G)$$

$$(a_B - 1.25\sigma_B) \sim (a_B + 1.25\sigma_B)$$

- Coding： 1—in the box; 0 — else.

Segmentation in RGB Space

Original image with colors of interest shown enclosed by a rectangle



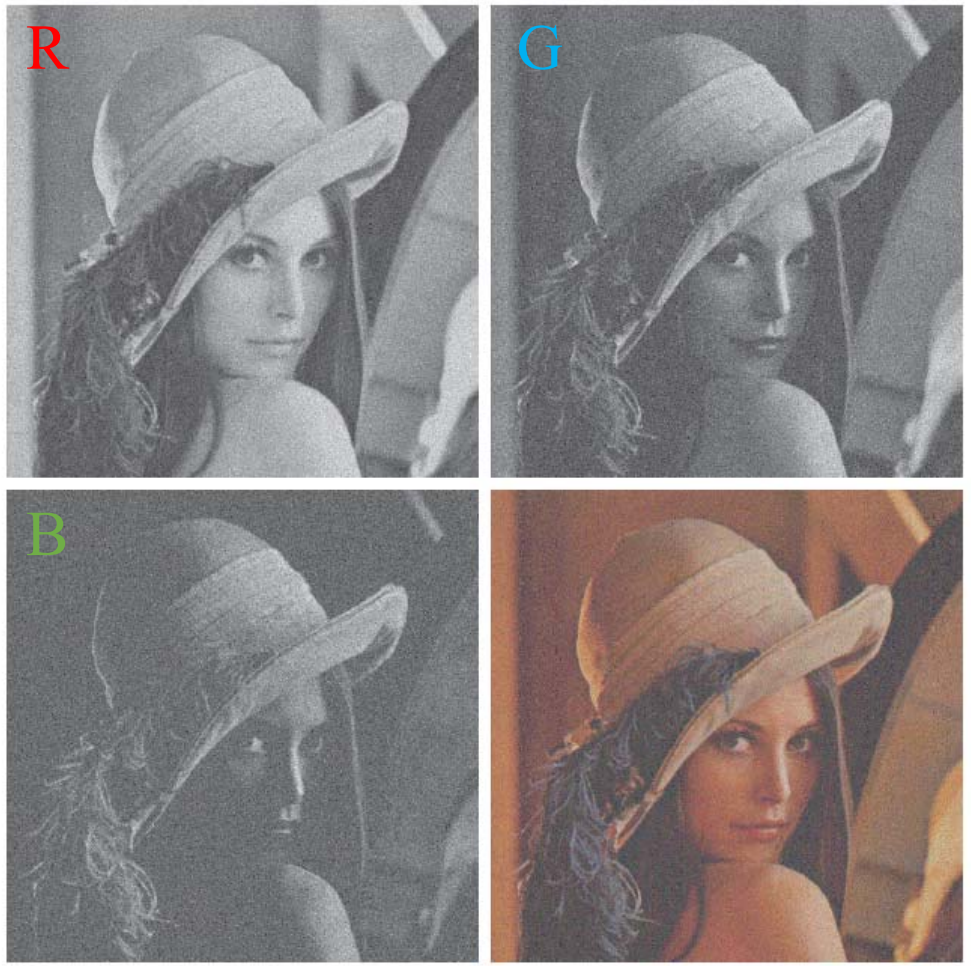
Result of segmentation in RGB vector space

Result of segmentation in HSI space

Noise in Color Images

- Take a brief look at noise in color images and how noise carries over when converting from one color model to another.

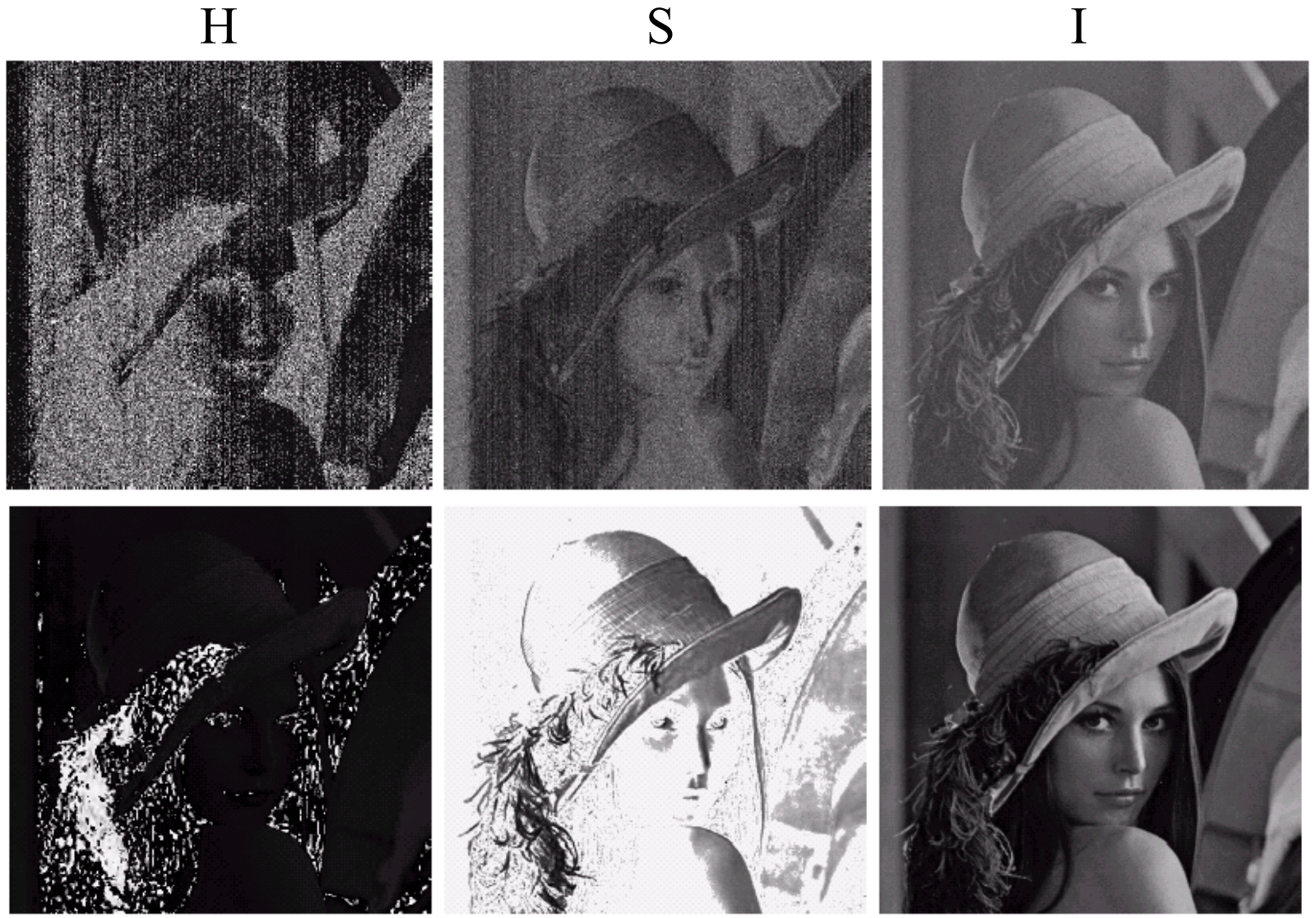
Red, green, and blue 8-bit component images corrupted by additive Gaussian noise of mean 0 and standard deviation of 28 intensity levels.



Fine-grain noise tends to be less visually noticeable in a color image than it is in a monochrome image.

Resulting
RGB image

Noise in Color Images



Noise in Color Images

Note:

- Hue and saturation components of noisy image are significantly degraded.
  - This is due to the nonlinearity of the *cos* and *min* operations in the RGB to HSI transformation.
- On the other hand, the intensity component is slightly smoother than three noisy RGB components.
  - This is because the intensity is the average of RGB, and averaging reduces random noise.

$$H = \begin{cases} \theta & \text{if } B \le G \\ 360 - \theta & \text{if } B > G \end{cases}$$

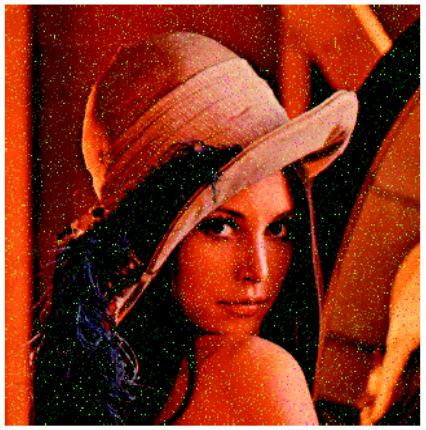
$$\text{with} \quad \theta = \cos^{-1}\left\{ \frac{\frac{1}{2}\left[(R-G)+(R-B)\right]}{\left[(R-G)^2+(R-B)(G-B)\right]^{1/2}} \right\}$$

$$S = 1 - \frac{3}{(R+G+B)}\left[\min(R,G,B)\right]$$

$$I = \frac{1}{3}(R+G+B)$$

Noise in Color Images

- In cases when only one RGB channel is affected by noise, conversion to HSI spreads the noise to all HSI component images.



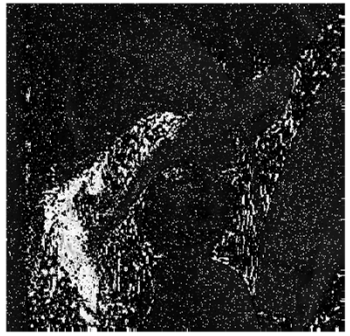
R

G

B

Only green component image is corrupted by salt-and-pepper noise

H



S

I

Summary

- In this lecture we have learnt:
  - Full-color processing
    - ➢Color transformations
    - ➢Smoothing and sharpening
    - ➢Color segmentation
    - ➢Noise in color images