Bitmapped Images

Multimedia Techniques & Applications

Yu-Ting Wu

Outline

- Overview
- Image compression
- Image manipulation
- Geometrical transformations

Overview

- Record the value of every pixel in the image
- **Image size** is the main cost for the simplicity

- Images created from external devices are usually in a bitmapped fashion
  - Digital cameras
  - Scanners

digital camera

scanner

Resolution

- A measure of **how finely** a device approximates continuous images using finite pixels
  - Closely related to sampling rates
- Two ways of specifying resolution
  - **Printers and scanners: number of dots per unit**
    - Dots per inch (dpi)
    - Ex: consumer printer (600 dpi), book production (1200 – 2700 dpi), scanners (300 dpi – 3600 dpi)
  - **Video: size of a frame measured in pixels**
    - Ex: 640 x 480, 768 x 576
    - Can translate into the form of dpi if you know the physical dimension of the display device

Image Compression

5

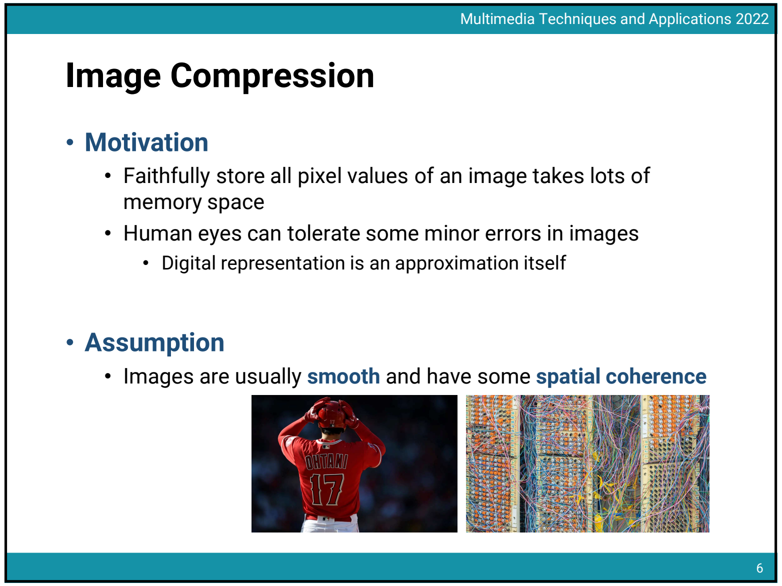
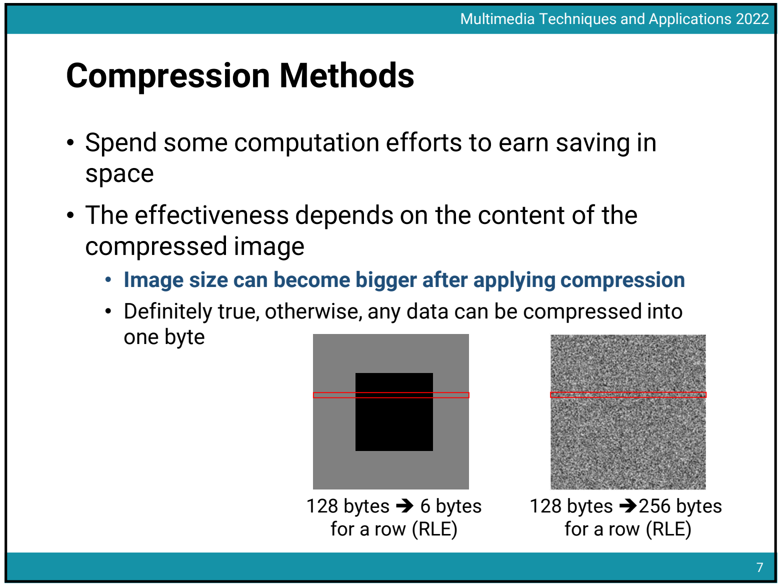


Image Compression

- **Motivation**
  - Faithfully store all pixel values of an image takes lots of memory space
  - Human eyes can tolerate some minor errors in images
    - Digital representation is an approximation itself
- **Assumption**
  - Images are usually **smooth** and have some **spatial coherence**

6



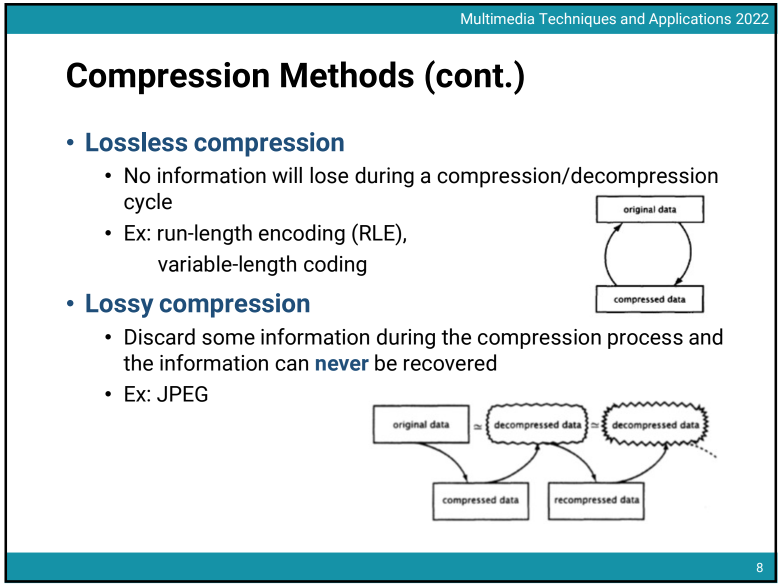
Compression Methods

- Spend some computation efforts to earn saving in space
- The effectiveness depends on the content of the compressed image
  - **Image size can become bigger after applying compression**
  - Definitely true, otherwise, any data can be compressed into one byte

128 bytes ➔ 6 bytes for a row (RLE)

128 bytes ➔256 bytes for a row (RLE)

7

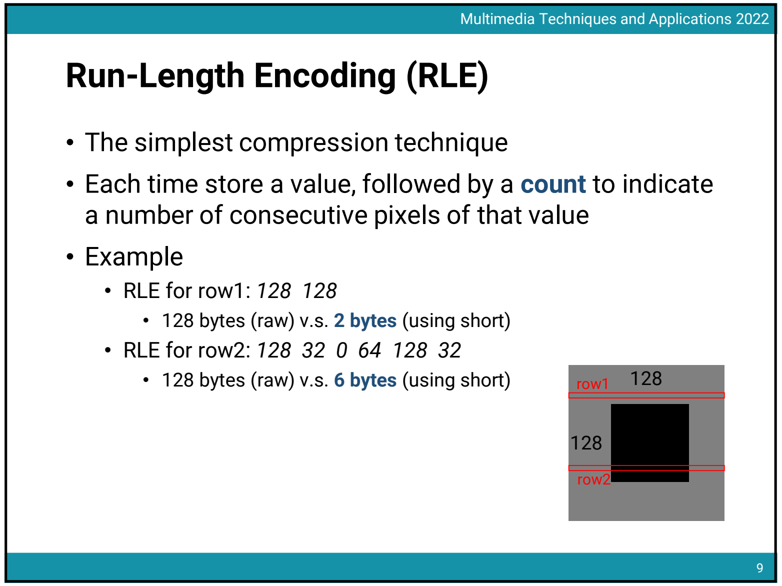


Compression Methods (cont.)

- **Lossless compression**
  - No information will lose during a compression/decompression cycle
  - Ex: run-length encoding (RLE), variable-length coding
- **Lossy compression**
  - Discard some information during the compression process and the information can **never** be recovered
  - Ex: JPEG

original data
compressed data
original data
≃
decompressed data
≃
decompressed data
compressed data
recompressed data

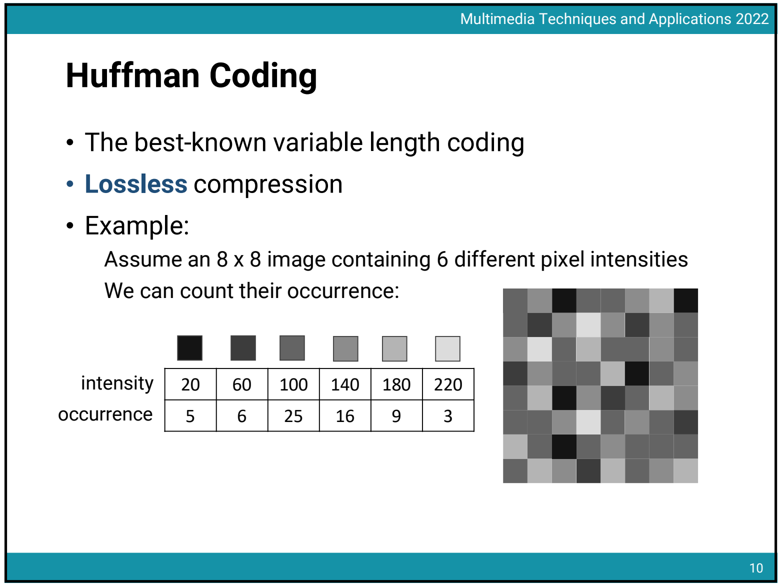
8



Run-Length Encoding (RLE)

- The simplest compression technique
- Each time store a value, followed by a **count** to indicate a number of consecutive pixels of that value
- Example
  - RLE for row1: *128 128*
    - 128 bytes (raw) v.s. **2 bytes** (using short)
  - RLE for row2: *128 32 0 64 128 32*
    - 128 bytes (raw) v.s. **6 bytes** (using short)

9



Huffman Coding

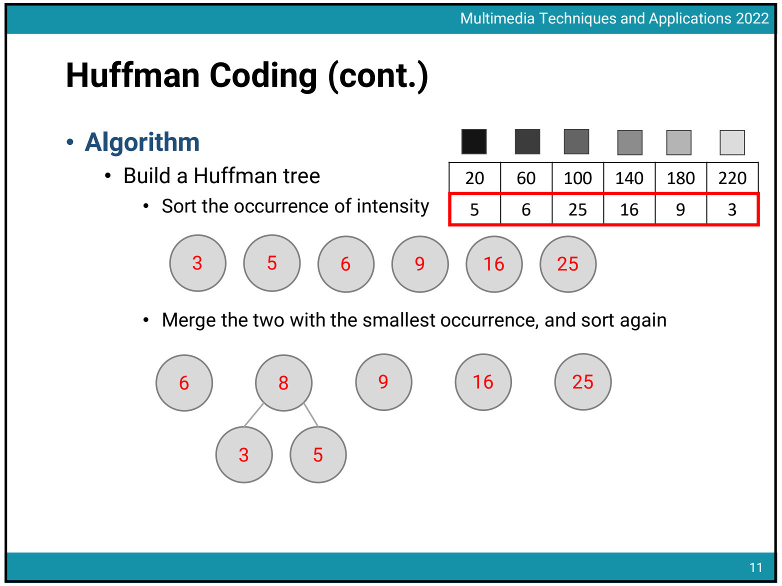
- The best-known variable length coding
- **Lossless** compression
- Example:

  Assume an 8 x 8 image containing 6 different pixel intensities

  We can count their occurrence:

	■	■	■	■	■	■
intensity	20	60	100	140	180	220
occurrence	5	6	25	16	9	3

10



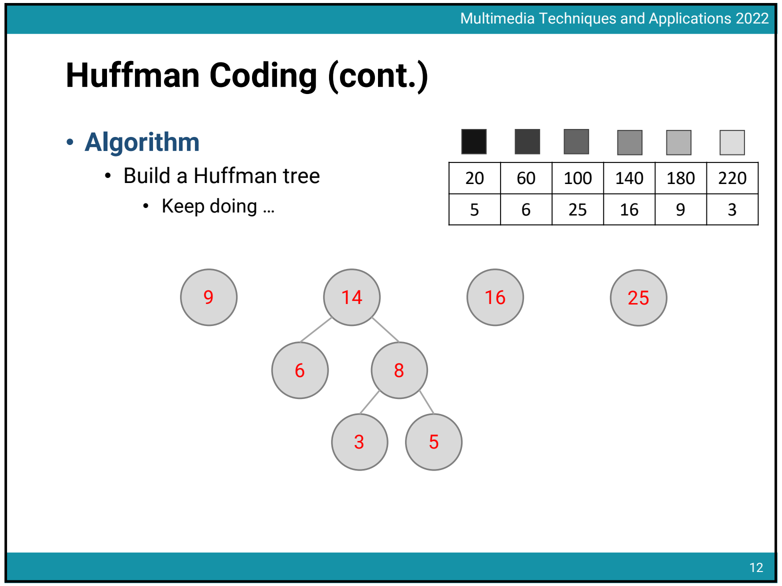
Huffman Coding (cont.)

- **Algorithm**
  - Build a Huffman tree
    - Sort the occurrence of intensity

■	■	■	■	■	■
20	60	100	140	180	220
5	6	25	16	9	3

- Merge the two with the smallest occurrence, and sort again

11

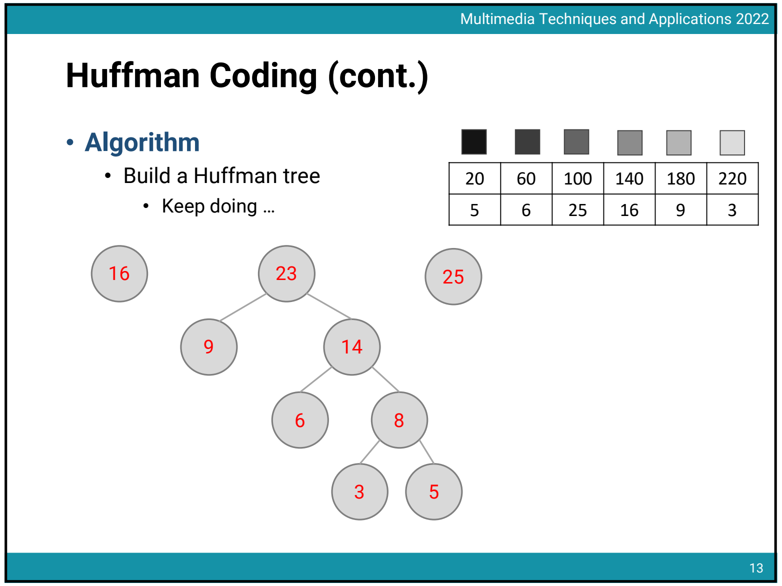


Huffman Coding (cont.)

- **Algorithm**
  - Build a Huffman tree
    - Keep doing ...

■	■	■	■	■	■
20	60	100	140	180	220
5	6	25	16	9	3

12

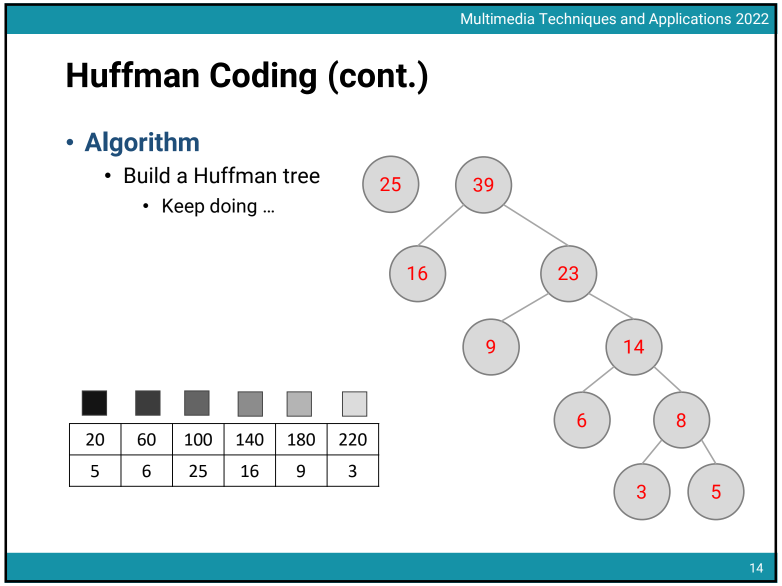


Huffman Coding (cont.)

- **Algorithm**
  - Build a Huffman tree
    - Keep doing …

■	■	■	■	■	□
20	60	100	140	180	220
5	6	25	16	9	3

13

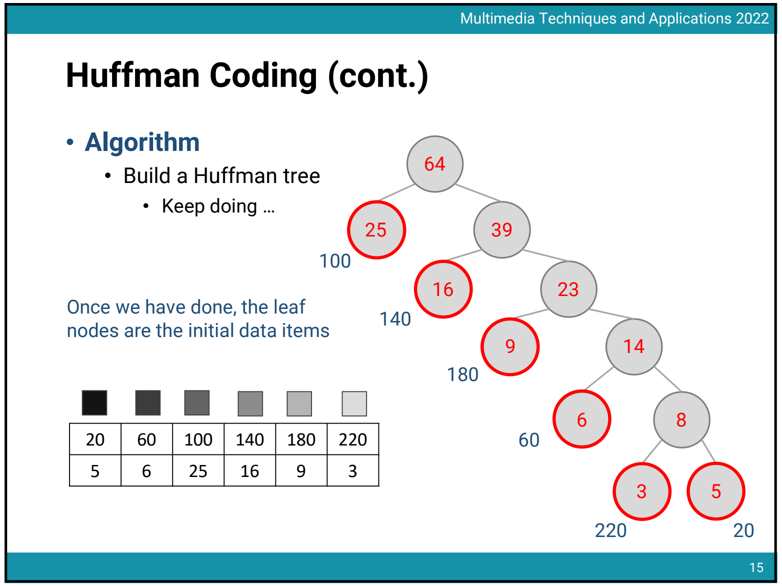


Huffman Coding (cont.)

- **Algorithm**
  - Build a Huffman tree
    - Keep doing …

■	■	■	■	■	□
20	60	100	140	180	220
5	6	25	16	9	3

14



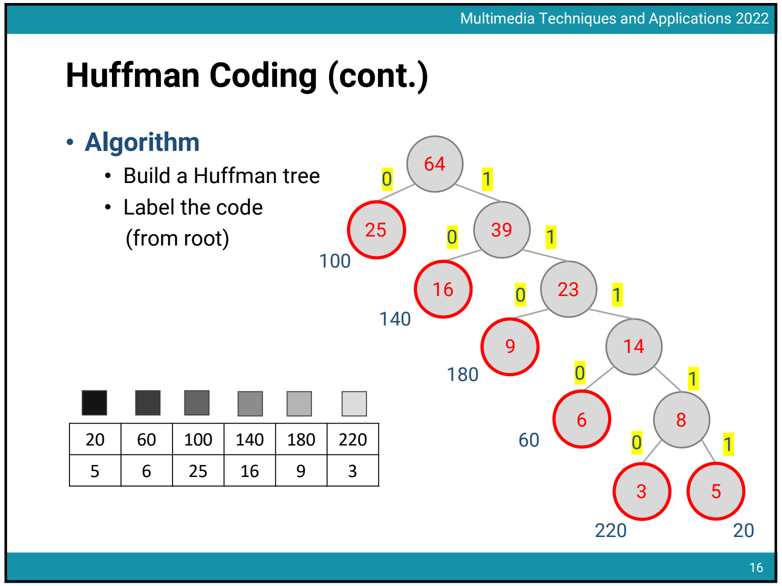
Huffman Coding (cont.)

- **Algorithm**
  - Build a Huffman tree
    - Keep doing …

Once we have done, the leaf nodes are the initial data items

■	■	■	■	■	□
20	60	100	140	180	220
5	6	25	16	9	3

15

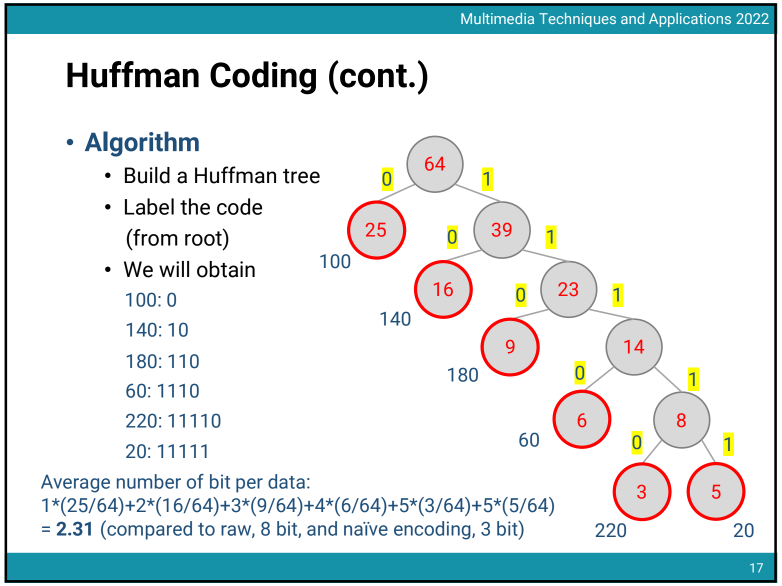


Huffman Coding (cont.)

- **Algorithm**
  - Build a Huffman tree
  - Label the code (from root)

■	■	■	■	■	□
20	60	100	140	180	220
5	6	25	16	9	3

16



Multimedia Techniques and Applications 2022

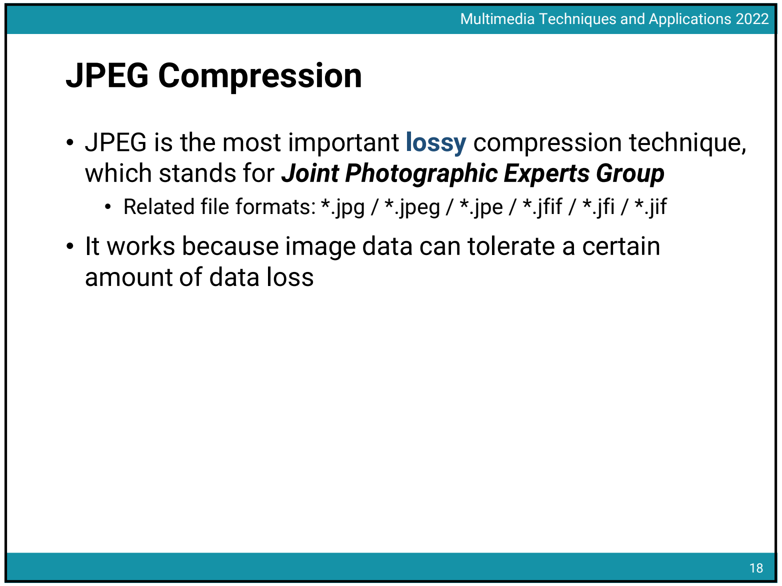
Huffman Coding (cont.)

- **Algorithm**
  - Build a Huffman tree
  - Label the code (from root)
  - We will obtain
    - 100: 0
    - 140: 10
    - 180: 110
    - 60: 1110
    - 220: 11110
    - 20: 11111

Average number of bit per data:
1*(25/64)+2*(16/64)+3*(9/64)+4*(6/64)+5*(3/64)+5*(5/64)
= **2.31** (compared to raw, 8 bit, and naïve encoding, 3 bit)

17

17



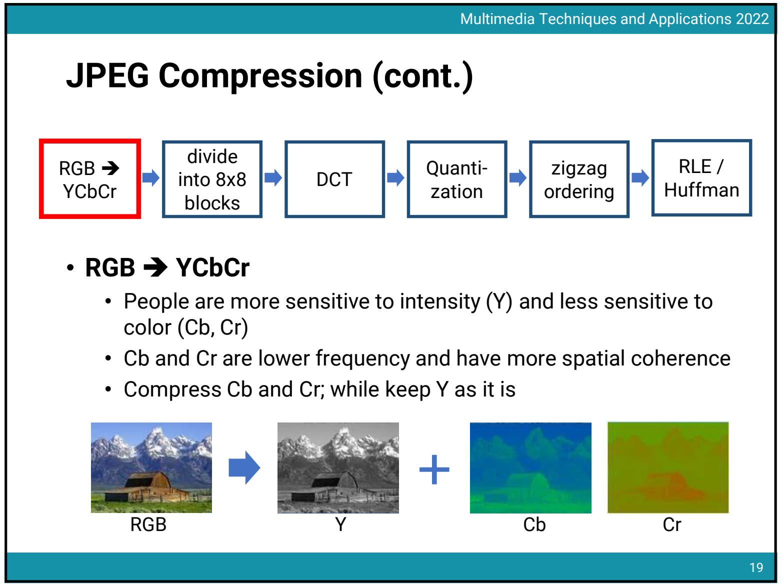
Multimedia Techniques and Applications 2022

JPEG Compression

- JPEG is the most important **lossy** compression technique, which stands for ***Joint Photographic Experts Group***
  - Related file formats: *.jpg / *.jpeg / *.jpe / *.jfif / *.jfi / *.jif
- It works because image data can tolerate a certain amount of data loss

18

18



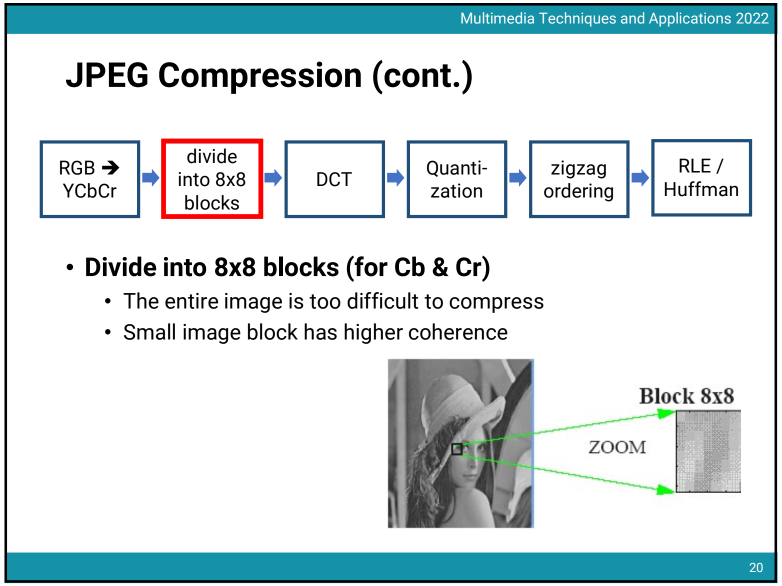
Multimedia Techniques and Applications 2022

JPEG Compression (cont.)

- **RGB ➔ YCbCr**
  - People are more sensitive to intensity (Y) and less sensitive to color (Cb, Cr)
  - Cb and Cr are lower frequency and have more spatial coherence
  - Compress Cb and Cr; while keep Y as it is

19

19



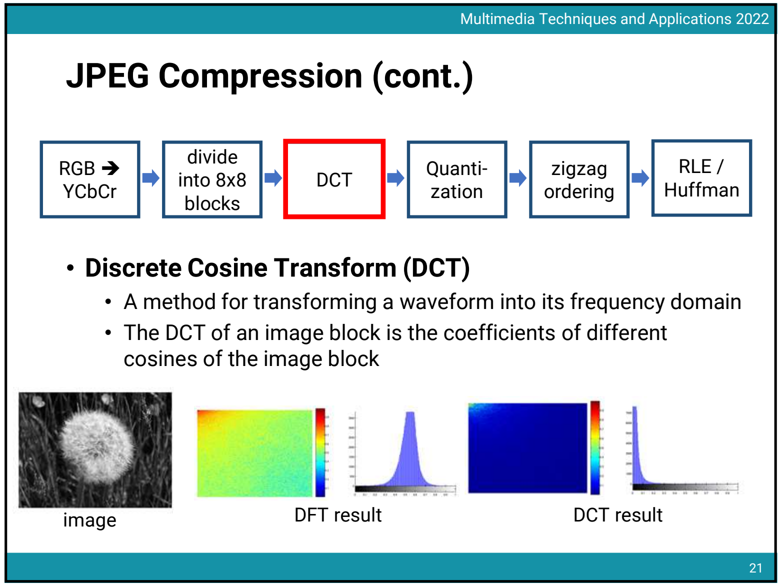
Multimedia Techniques and Applications 2022

JPEG Compression (cont.)

- **Divide into 8x8 blocks (for Cb & Cr)**
  - The entire image is too difficult to compress
  - Small image block has higher coherence

20

20

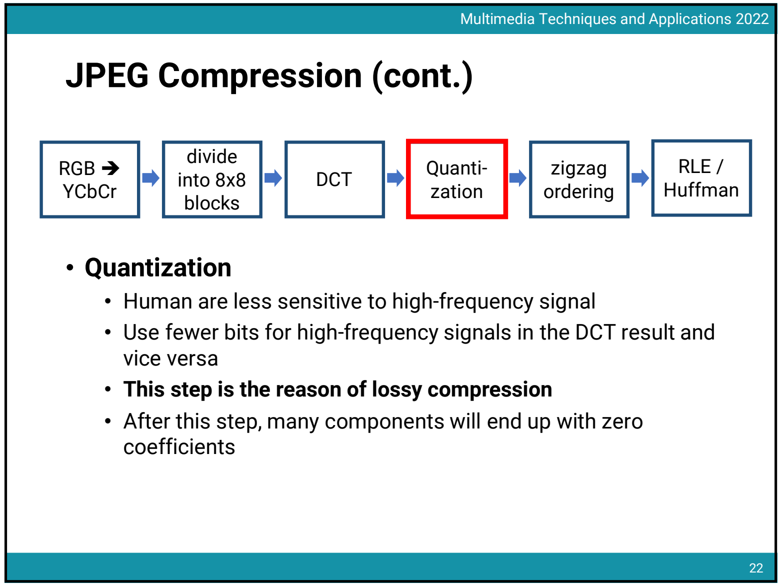


JPEG Compression (cont.)

RGB ➔ YCbCr → divide into 8x8 blocks → DCT → Quanti-zation → zigzag ordering → RLE / Huffman

- **Discrete Cosine Transform (DCT)**
  - A method for transforming a waveform into its frequency domain
  - The DCT of an image block is the coefficients of different cosines of the image block

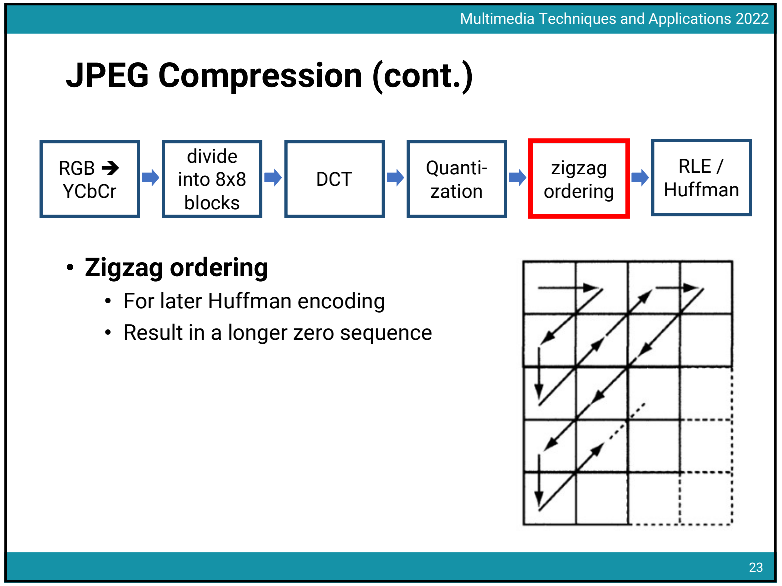
image
DFT result
DCT result



JPEG Compression (cont.)

RGB ➔ YCbCr → divide into 8x8 blocks → DCT → Quanti-zation → zigzag ordering → RLE / Huffman

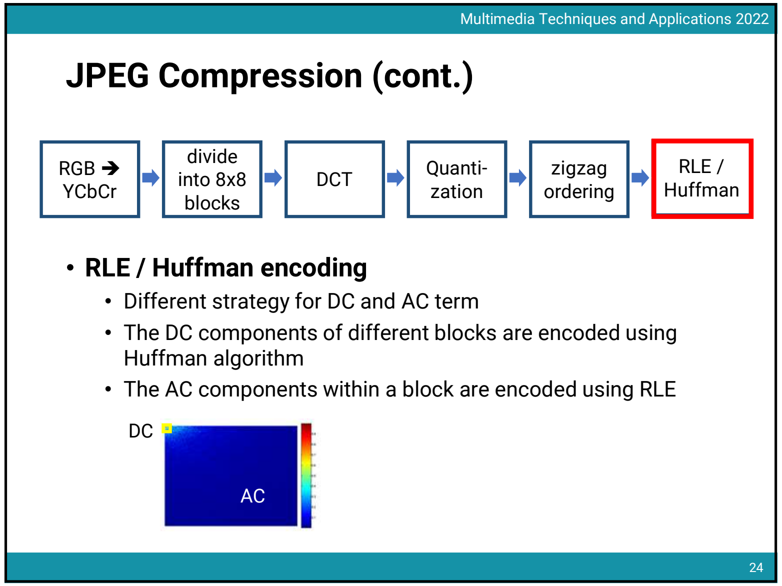
- **Quantization**
  - Human are less sensitive to high-frequency signal
  - Use fewer bits for high-frequency signals in the DCT result and vice versa
  - **This step is the reason of lossy compression**
  - After this step, many components will end up with zero coefficients



JPEG Compression (cont.)

RGB ➔ YCbCr → divide into 8x8 blocks → DCT → Quanti-zation → zigzag ordering → RLE / Huffman

- **Zigzag ordering**
  - For later Huffman encoding
  - Result in a longer zero sequence



JPEG Compression (cont.)

RGB ➔ YCbCr → divide into 8x8 blocks → DCT → Quanti-zation → zigzag ordering → RLE / Huffman

- **RLE / Huffman encoding**
  - Different strategy for DC and AC term
  - The DC components of different blocks are encoded using Huffman algorithm
  - The AC components within a block are encoded using RLE

JPEG Compression (cont.)

- The decompression of JPEG data is done by reversing the compression process
- We can control the degree of compression by altering the amount of quantization
- JPEG compression usually achieves very high compression rate for natural images (5% of the original size)

Image Manipulation

Image Manipulation

- **Motivations**
  - Correct deficiencies in an image (e.g., noise, red-eye)
  - Create images that are difficult or impossible to make naturally (e.g., glow)

- Type of image manipulations
  - Pixel **point** processing
  - Pixel **group** processing

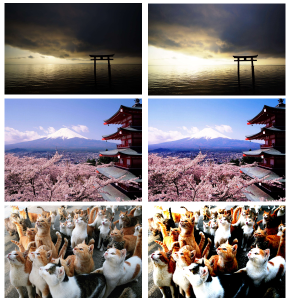
Pixel Point Processing

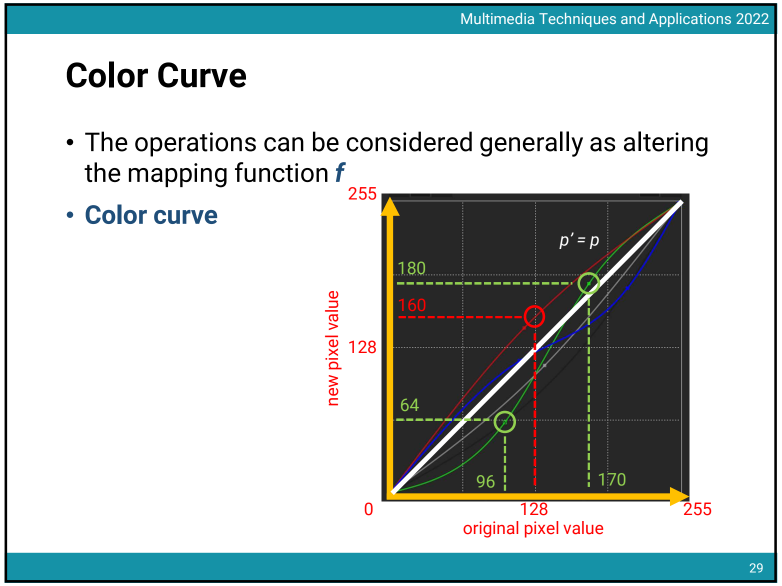
- Compute a pixel's new value solely on the basis of its old value

mapping function

$$p' = \boxed{f}(p)$$

- Some examples
  - Adjustment of brightness
  - Adjustment of contrast
  - Change the black and white levels

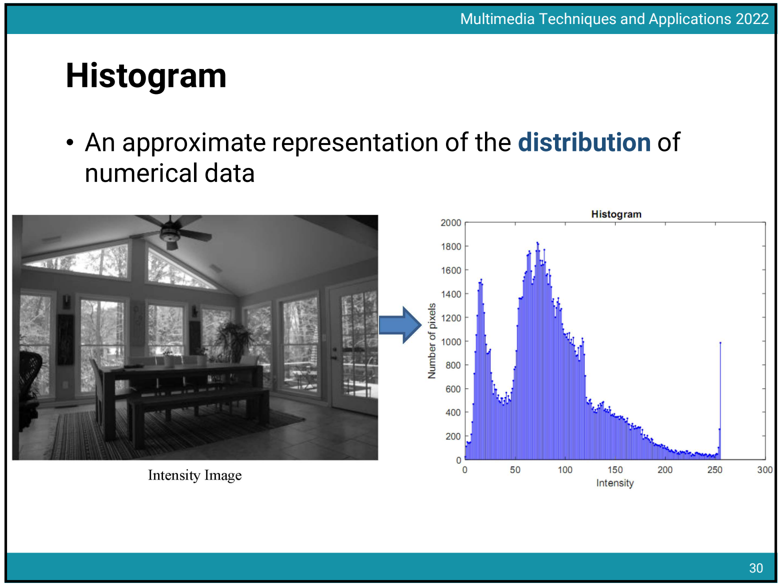




Color Curve

- The operations can be considered generally as altering the mapping function ***f***
- **Color curve**

29

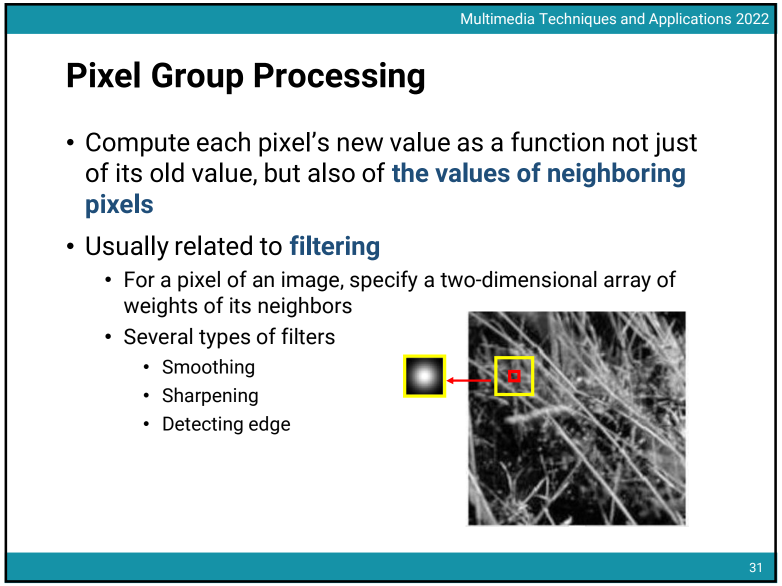


Histogram

- An approximate representation of the **distribution** of numerical data

Intensity Image

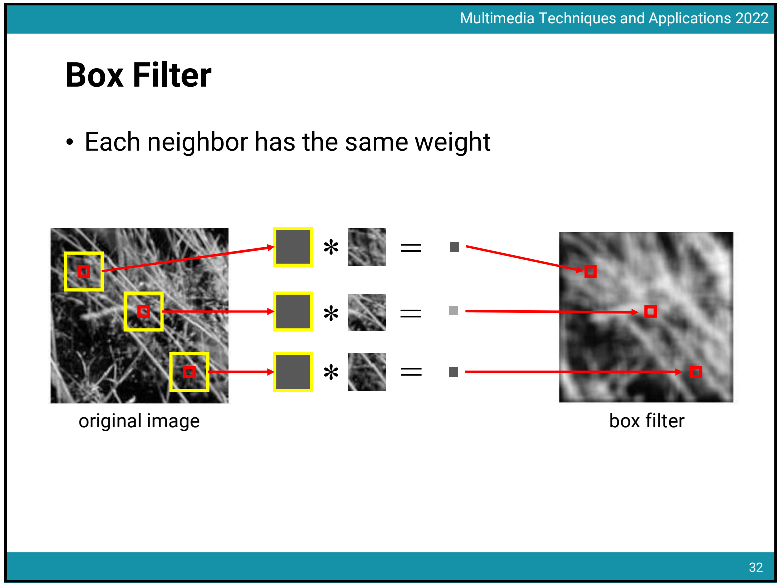
30



Pixel Group Processing

- Compute each pixel’s new value as a function not just of its old value, but also of **the values of neighboring pixels**
- Usually related to **filtering**
  - For a pixel of an image, specify a two-dimensional array of weights of its neighbors
  - Several types of filters
    - Smoothing
    - Sharpening
    - Detecting edge

31



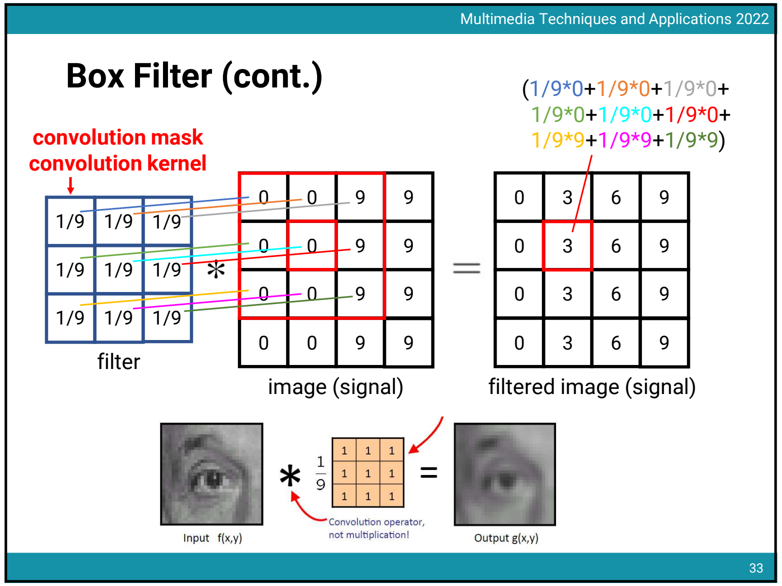
Box Filter

- Each neighbor has the same weight

original image

box filter

32



Multimedia Techniques and Applications 2022

Box Filter (cont.)

(1/9*0+1/9*0+1/9*0+
1/9*0+1/9*0+1/9*0+
1/9*9+1/9*9+1/9*9)

convolution mask
convolution kernel

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

filter

*

0	0	9	9
0	0	9	9
0	0	9	9
0	0	9	9

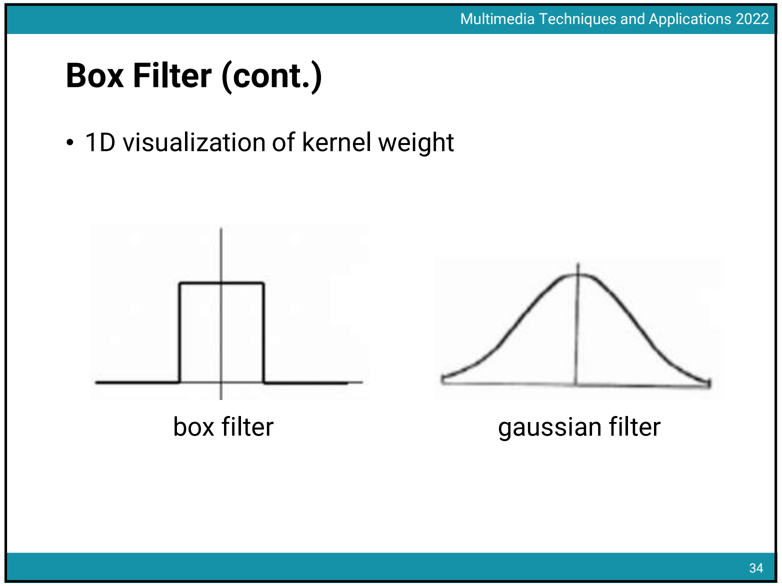
image (signal)

=

0	3	6	9
0	3	6	9
0	3	6	9
0	3	6	9

filtered image (signal)

33

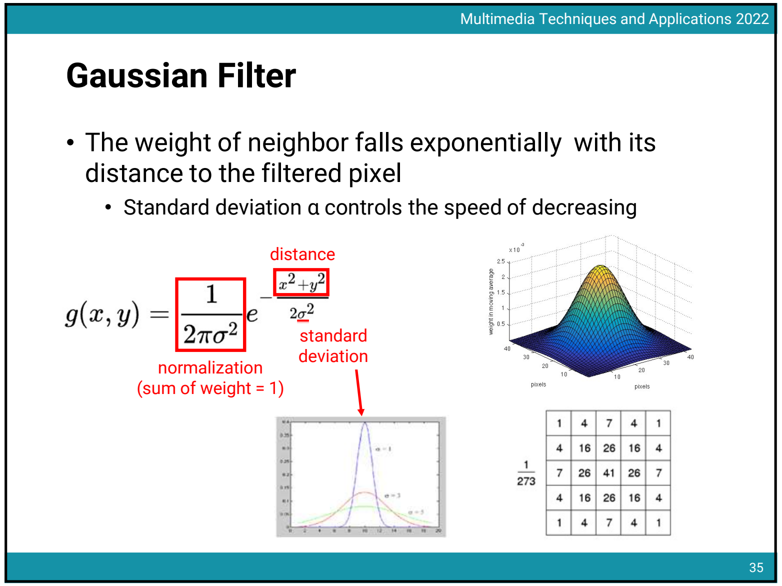


Multimedia Techniques and Applications 2022

Box Filter (cont.)

- 1D visualization of kernel weight

34



Multimedia Techniques and Applications 2022

Gaussian Filter

- The weight of neighbor falls exponentially with its distance to the filtered pixel
  - Standard deviation α controls the speed of decreasing

$$g(x,y)=\boxed{\frac{1}{2\pi\sigma^2}}e^{-\frac{\boxed{x^2+y^2}}{2\sigma^2}}$$

distance

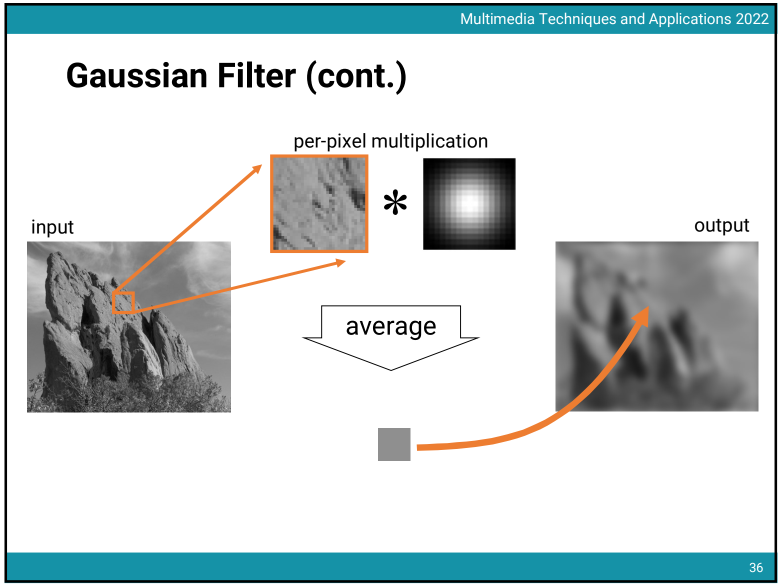
standard deviation

normalization (sum of weight = 1)

$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

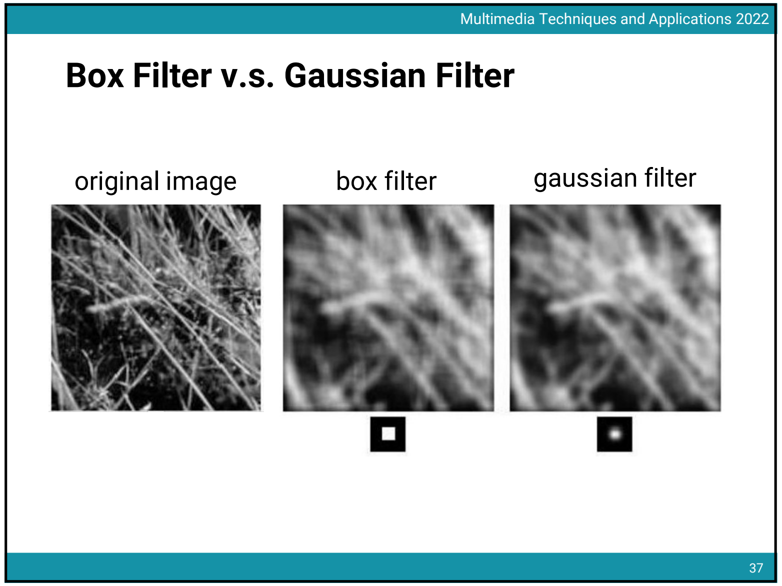
35



Multimedia Techniques and Applications 2022

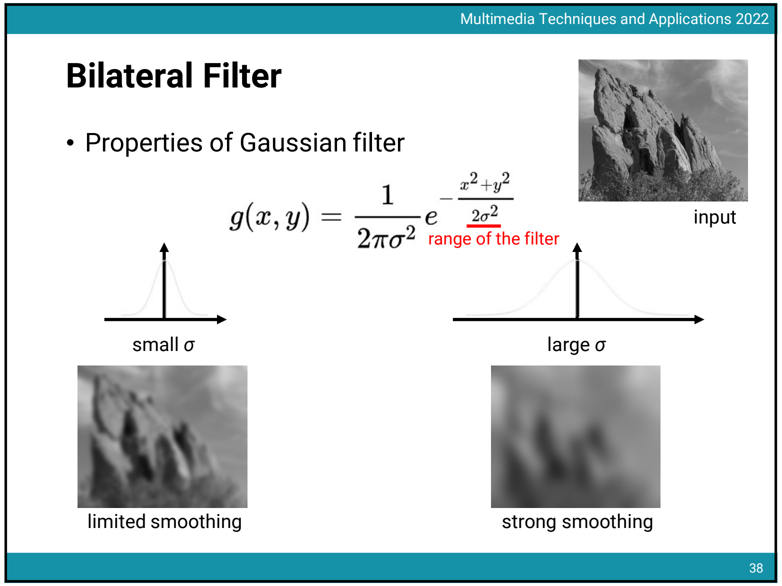
Gaussian Filter (cont.)

36



Box Filter v.s. Gaussian Filter

37



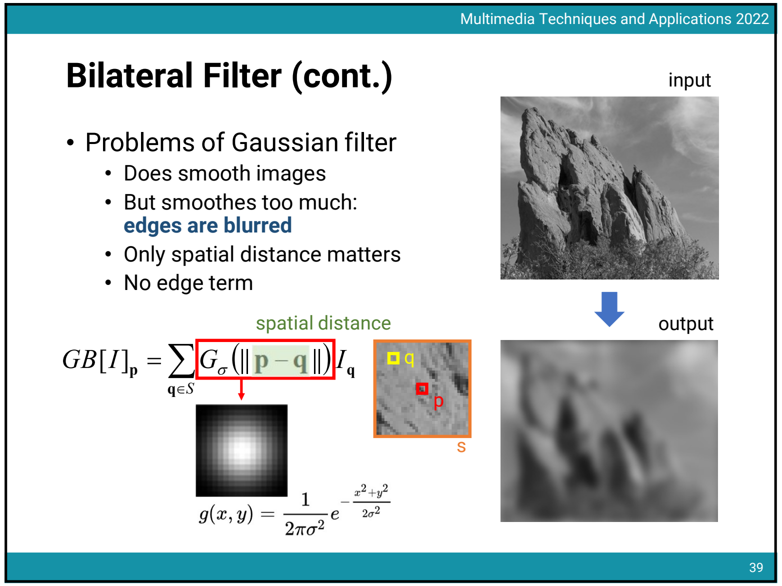
Bilateral Filter

- Properties of Gaussian filter

$$g(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{\underline{2\sigma^2}}}$$

range of the filter

38



Bilateral Filter (cont.)

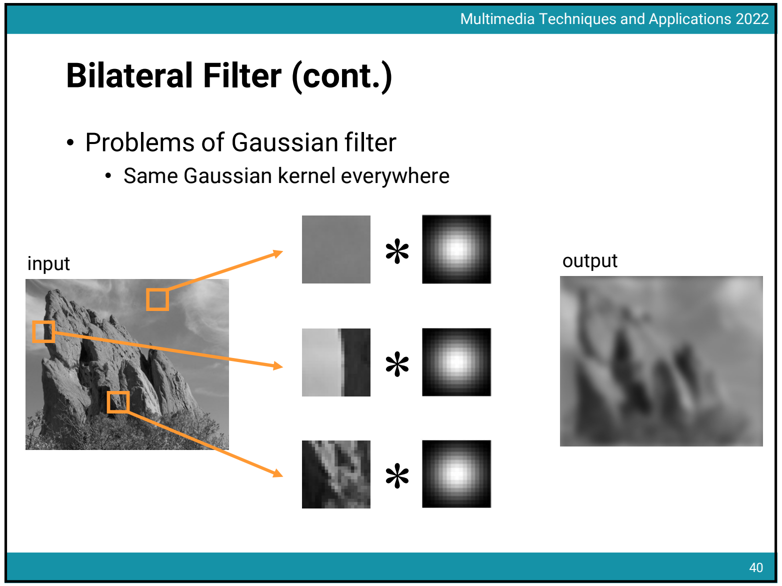
- Problems of Gaussian filter
  - Does smooth images
  - But smoothes too much: **edges are blurred**
  - Only spatial distance matters
  - No edge term

spatial distance

$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q}\in S} \boxed{G_\sigma(\|\mathbf{p}-\mathbf{q}\|)} I_{\mathbf{q}}$$

$$g(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

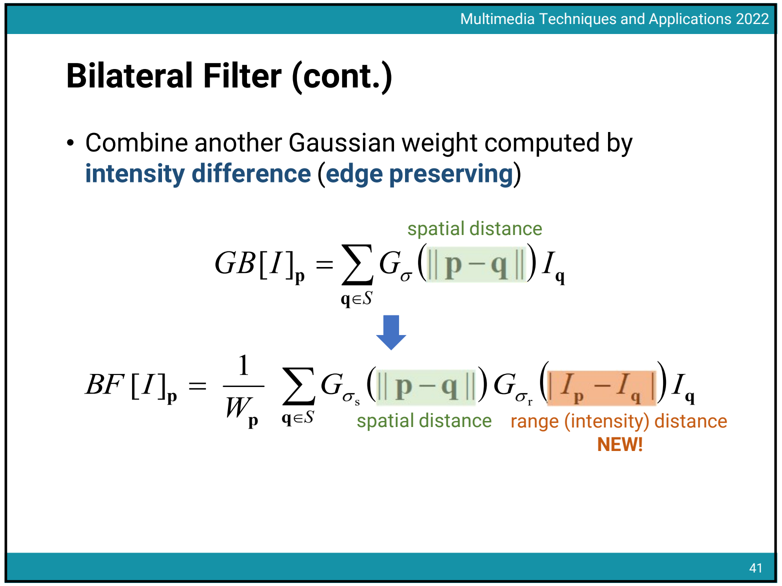
39



Bilateral Filter (cont.)

- Problems of Gaussian filter
  - Same Gaussian kernel everywhere

40



Multimedia Techniques and Applications 2022

Bilateral Filter (cont.)

- Combine another Gaussian weight computed by **intensity difference** (**edge preserving**)

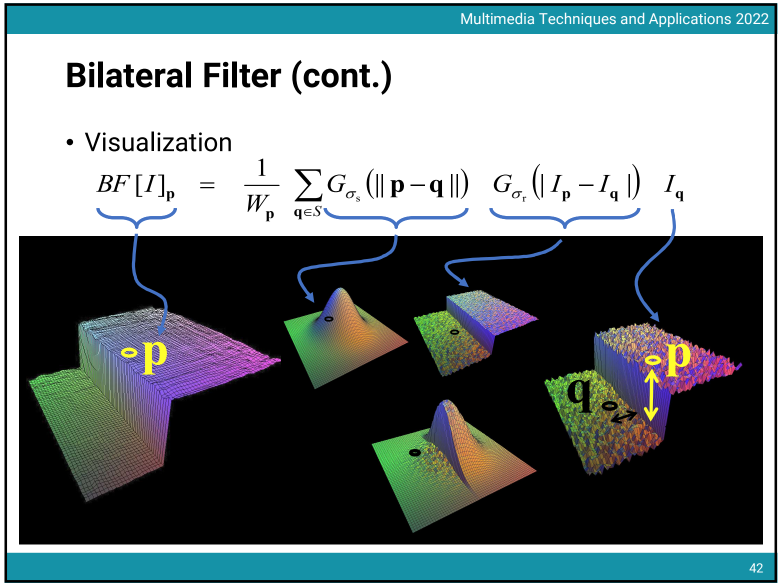
$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q}\in S} G_{\sigma}\left(\underbrace{\|\mathbf{p}-\mathbf{q}\|}_{\text{spatial distance}}\right) I_{\mathbf{q}}$$

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}\in S} G_{\sigma_s}\left(\underbrace{\|\mathbf{p}-\mathbf{q}\|}_{\text{spatial distance}}\right) G_{\sigma_r}\left(\underbrace{|I_{\mathbf{p}}-I_{\mathbf{q}}|}_{\text{range (intensity) distance}}\right) I_{\mathbf{q}}$$

NEW!

41

41



Multimedia Techniques and Applications 2022

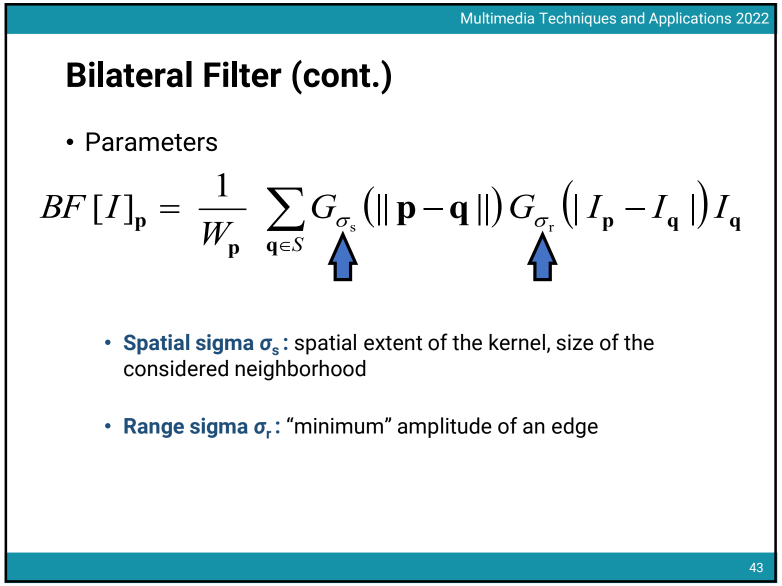
Bilateral Filter (cont.)

- Visualization

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}\in S} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)\; G_{\sigma_r}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)\; I_{\mathbf{q}}$$

42

42



Multimedia Techniques and Applications 2022

Bilateral Filter (cont.)

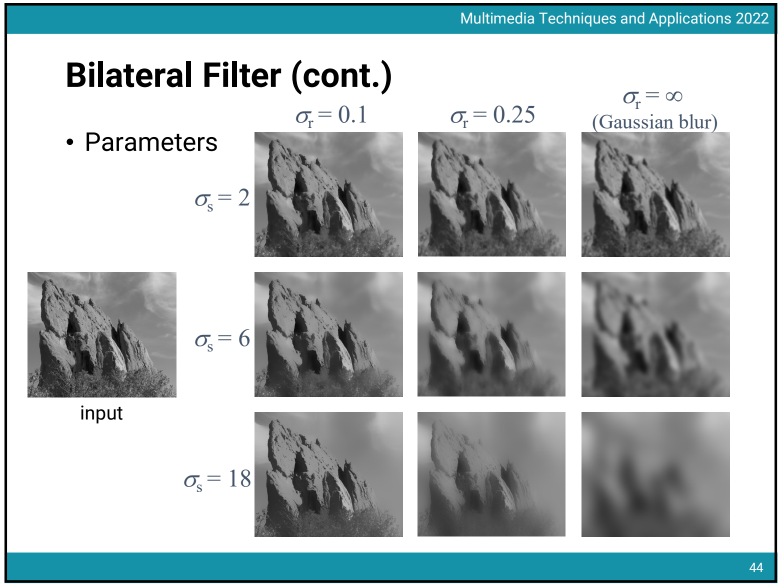
- Parameters

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}\in S} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_r}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)\, I_{\mathbf{q}}$$

  - **Spatial sigma σ_s :** spatial extent of the kernel, size of the considered neighborhood
  - **Range sigma σ_r :** “minimum” amplitude of an edge

43

43



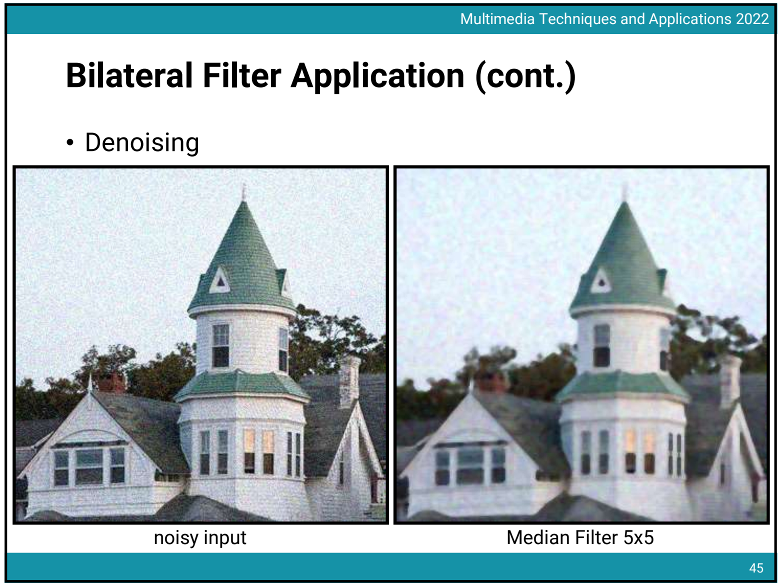
Multimedia Techniques and Applications 2022

Bilateral Filter (cont.)

- Parameters

44

44



Multimedia Techniques and Applications 2022

Bilateral Filter Application (cont.)

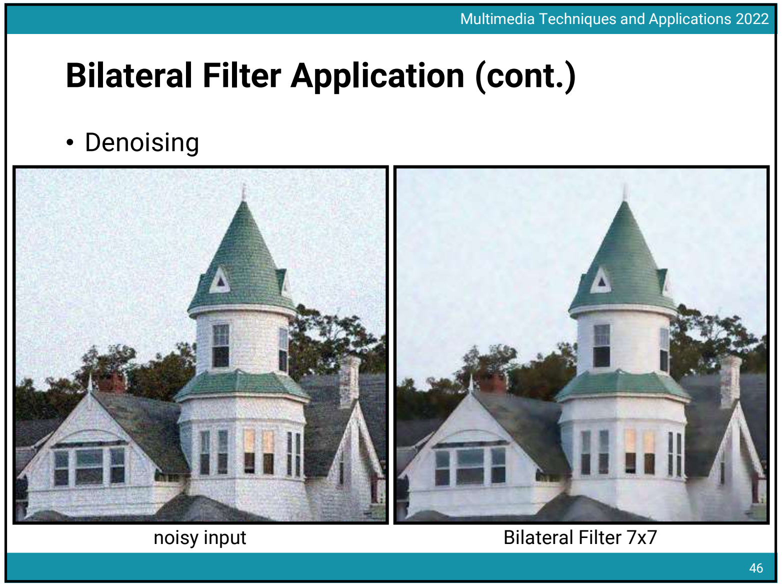
- Denoising

noisy input

Median Filter 5x5

45

45



Multimedia Techniques and Applications 2022

Bilateral Filter Application (cont.)

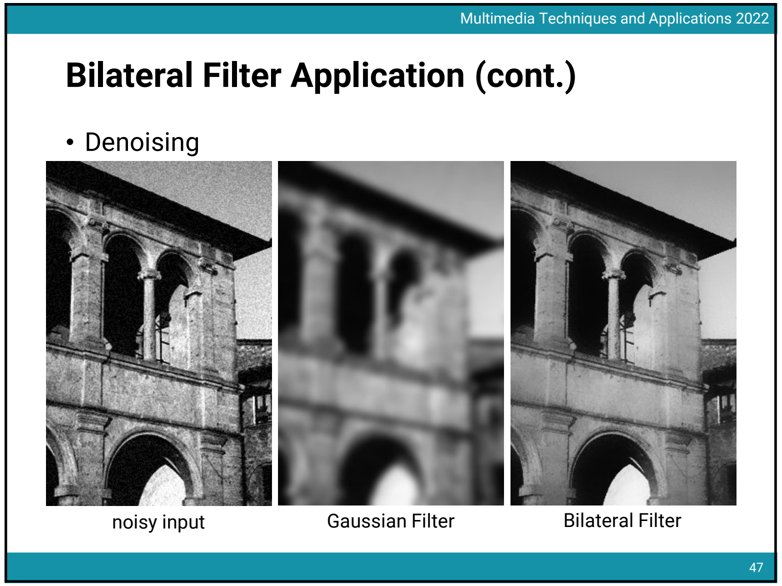
- Denoising

noisy input

Bilateral Filter 7x7

46

46



Multimedia Techniques and Applications 2022

Bilateral Filter Application (cont.)

- Denoising

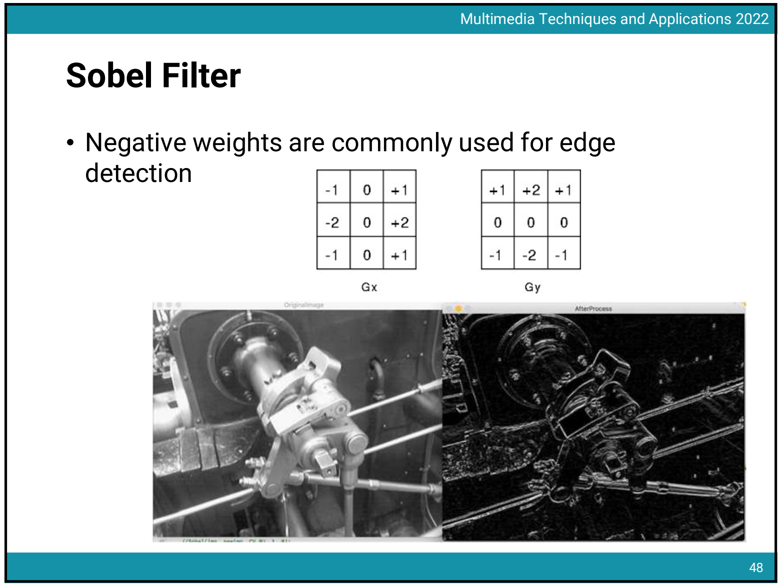
noisy input

Gaussian Filter

Bilateral Filter

47

47



Multimedia Techniques and Applications 2022

Sobel Filter

- Negative weights are commonly used for edge detection

-1	0	+1
-2	0	+2
-1	0	+1

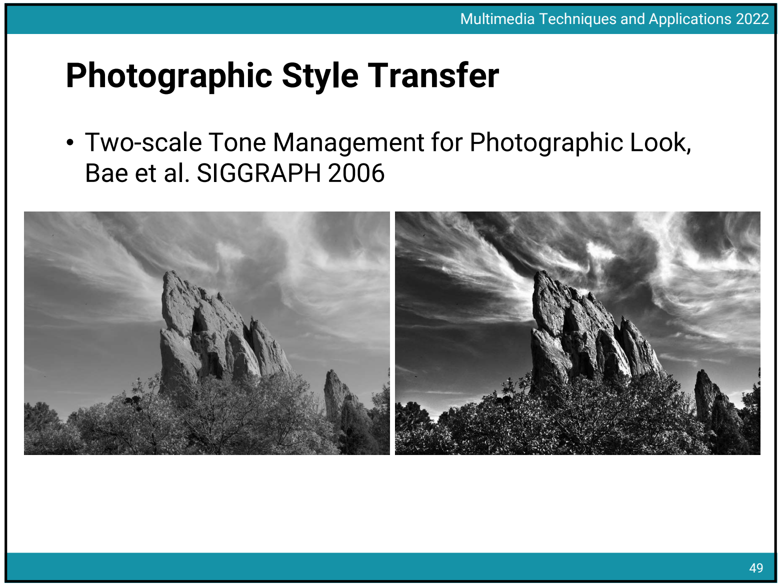
Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

48

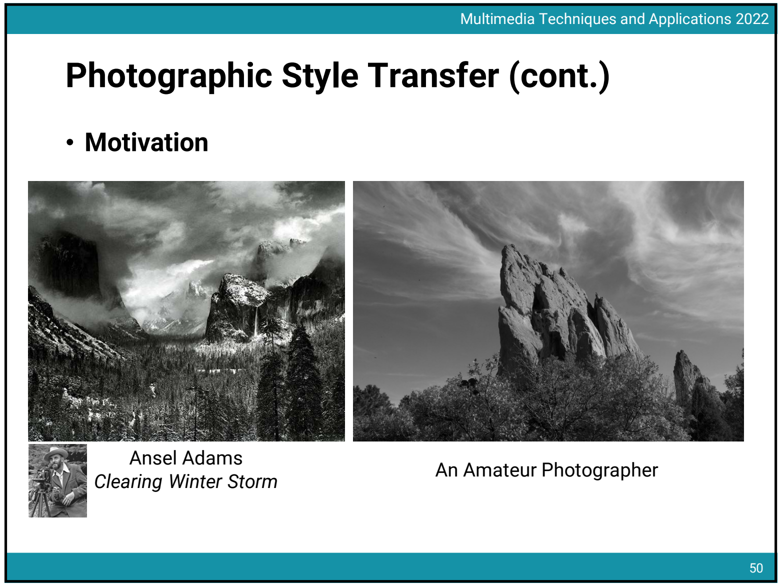
48



Photographic Style Transfer

- Two-scale Tone Management for Photographic Look, Bae et al. SIGGRAPH 2006

49



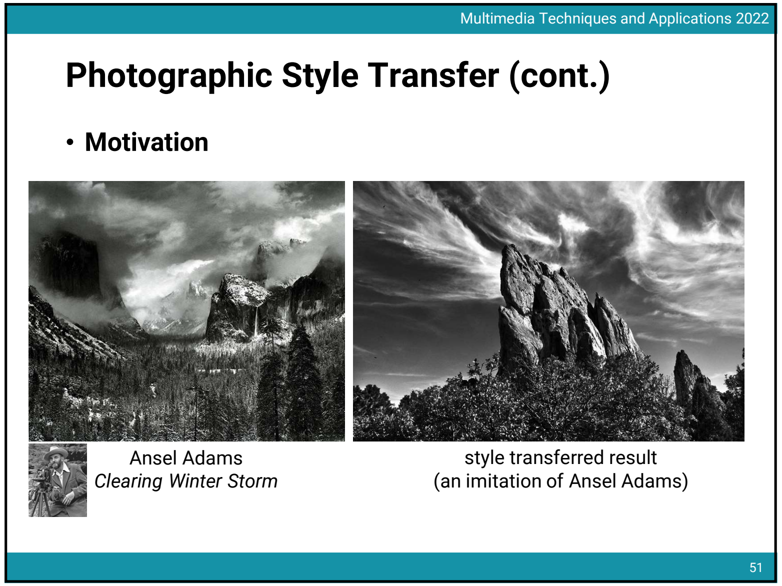
Photographic Style Transfer (cont.)

- **Motivation**

Ansel Adams
Clearing Winter Storm

An Amateur Photographer

50



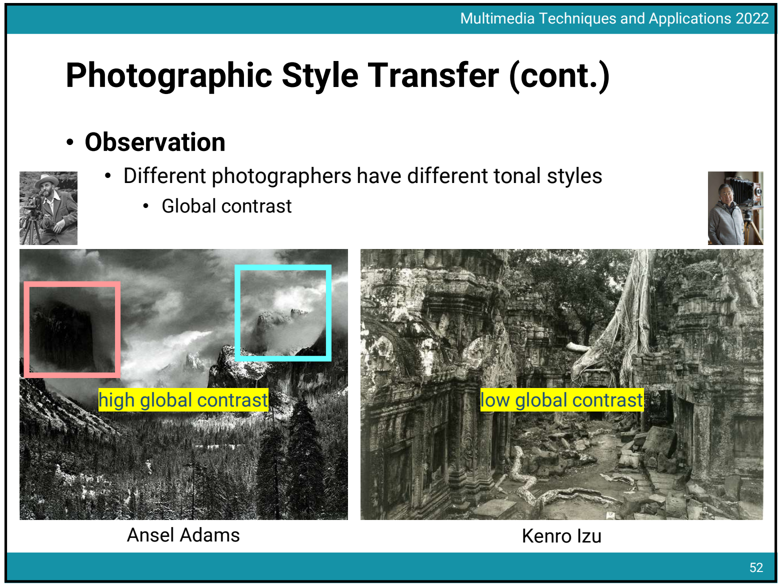
Photographic Style Transfer (cont.)

- **Motivation**

Ansel Adams
Clearing Winter Storm

style transferred result
(an imitation of Ansel Adams)

51



Photographic Style Transfer (cont.)

- **Observation**
  - Different photographers have different tonal styles
    - Global contrast

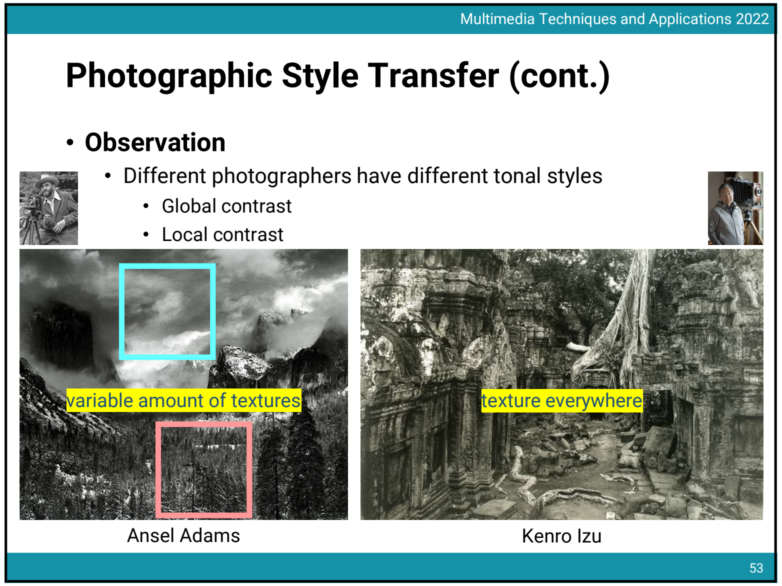
high global contrast

low global contrast

Ansel Adams

Kenro Izu

52



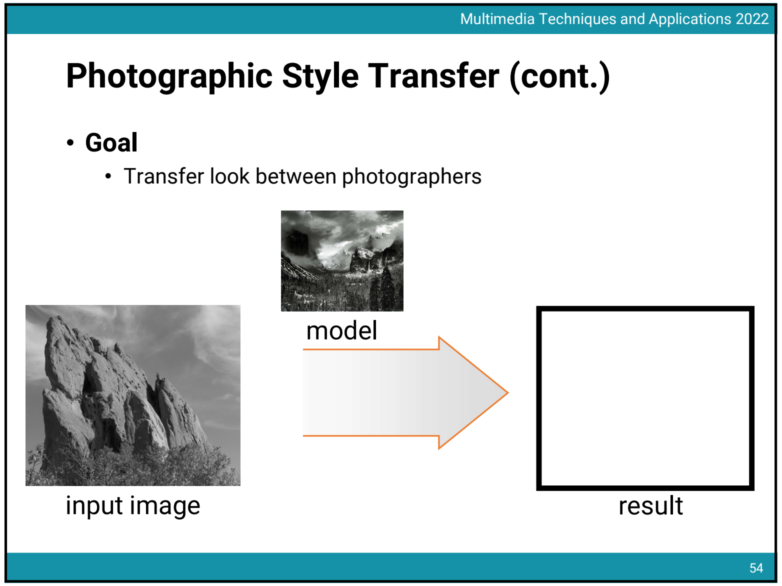
Photographic Style Transfer (cont.)

- **Observation**
  - Different photographers have different tonal styles
    - Global contrast
    - Local contrast

Ansel Adams

Kenro Izu

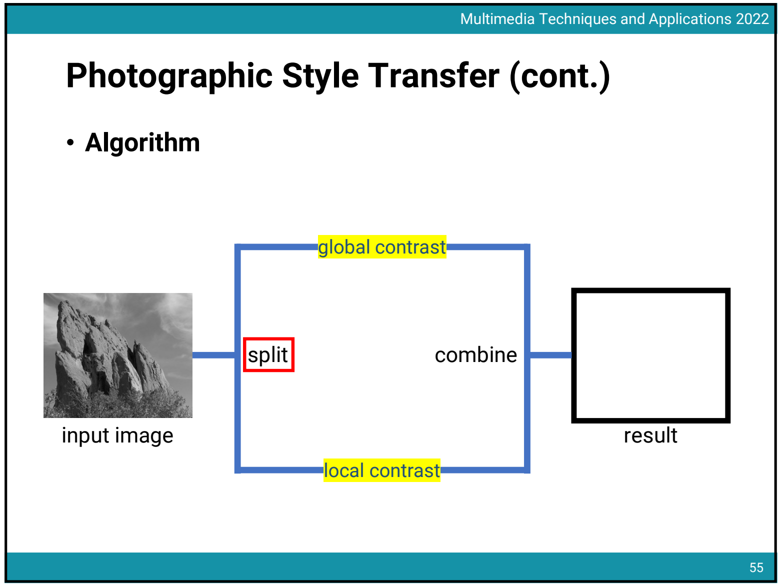
53



Photographic Style Transfer (cont.)

- **Goal**
  - Transfer look between photographers

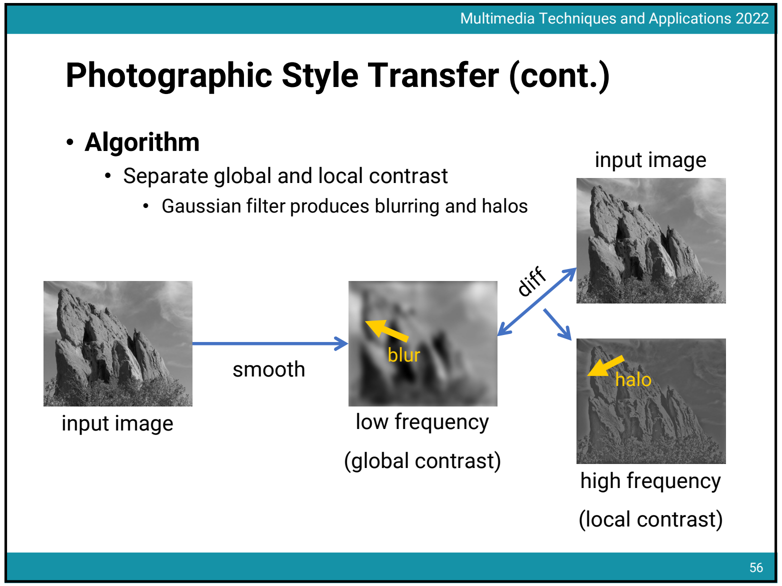
54



Photographic Style Transfer (cont.)

- **Algorithm**

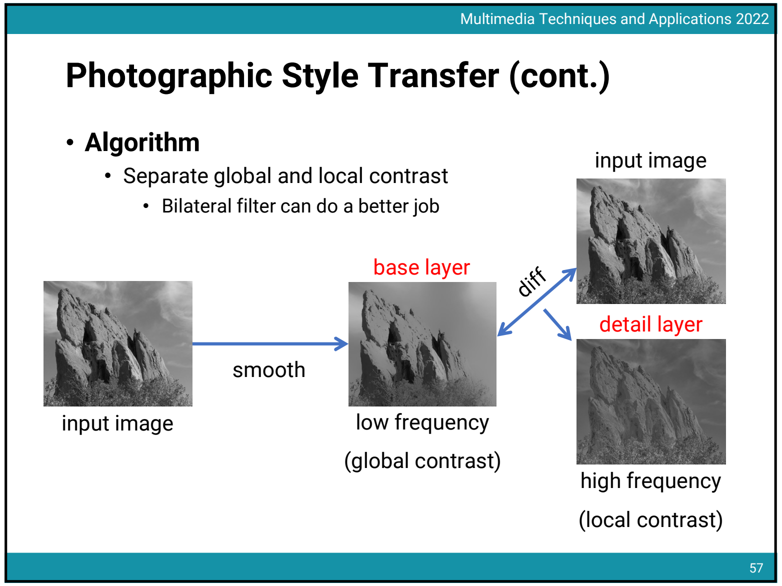
55



Photographic Style Transfer (cont.)

- **Algorithm**
  - Separate global and local contrast
    - Gaussian filter produces blurring and halos

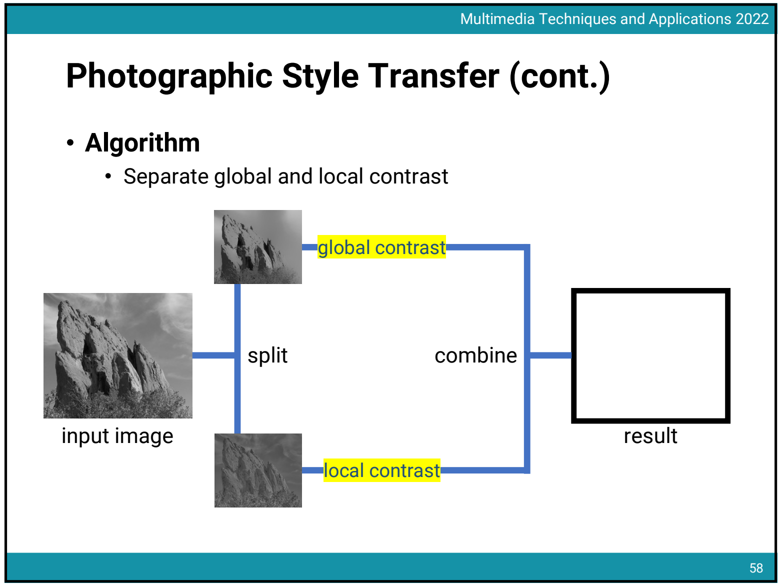
56



Photographic Style Transfer (cont.)

- **Algorithm**
  - Separate global and local contrast
    - Bilateral filter can do a better job

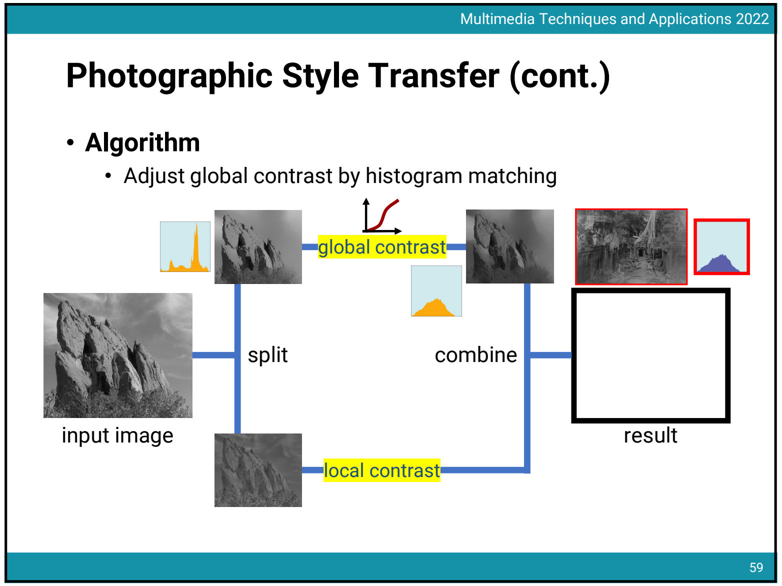
57



Photographic Style Transfer (cont.)

- **Algorithm**
  - Separate global and local contrast

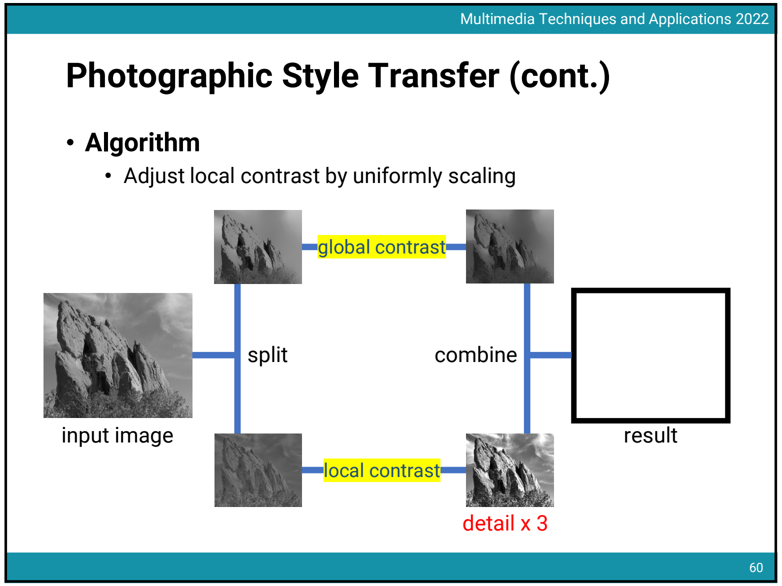
58



Photographic Style Transfer (cont.)

- **Algorithm**
  - Adjust global contrast by histogram matching

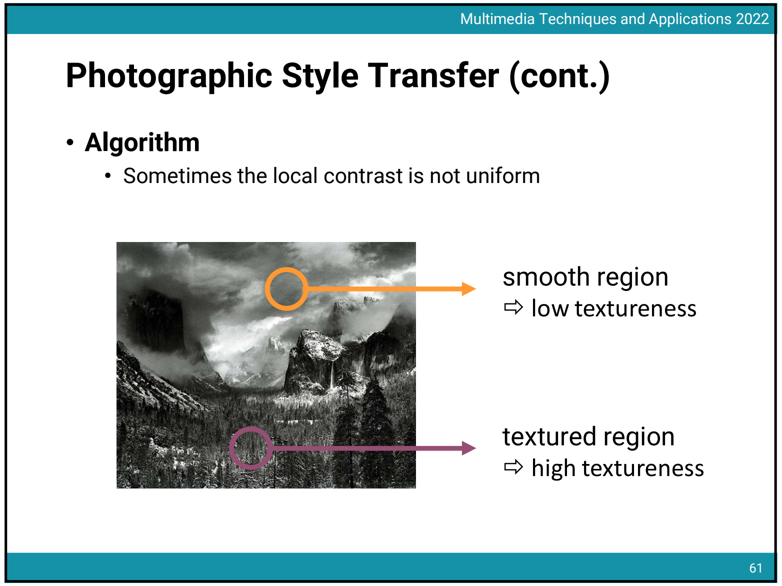
59



Photographic Style Transfer (cont.)

- **Algorithm**
  - Adjust local contrast by uniformly scaling

60



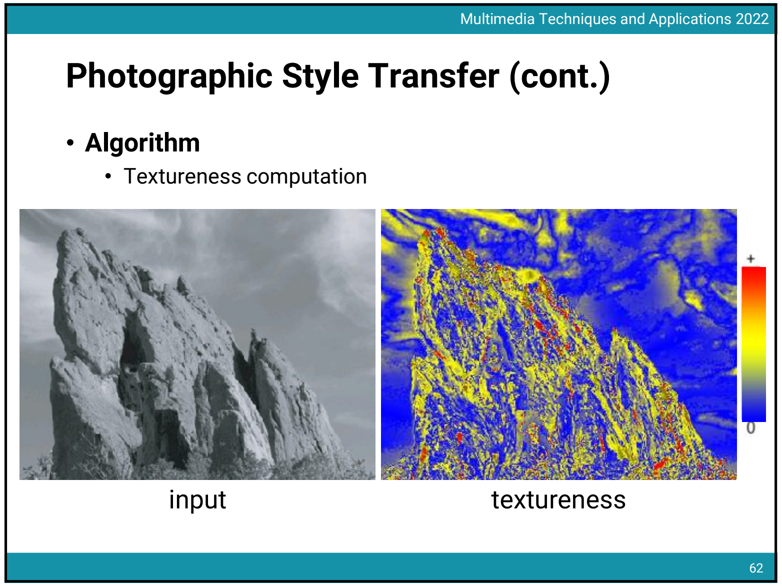
Multimedia Techniques and Applications 2022

Photographic Style Transfer (cont.)

- **Algorithm**
  - Sometimes the local contrast is not uniform

61

61



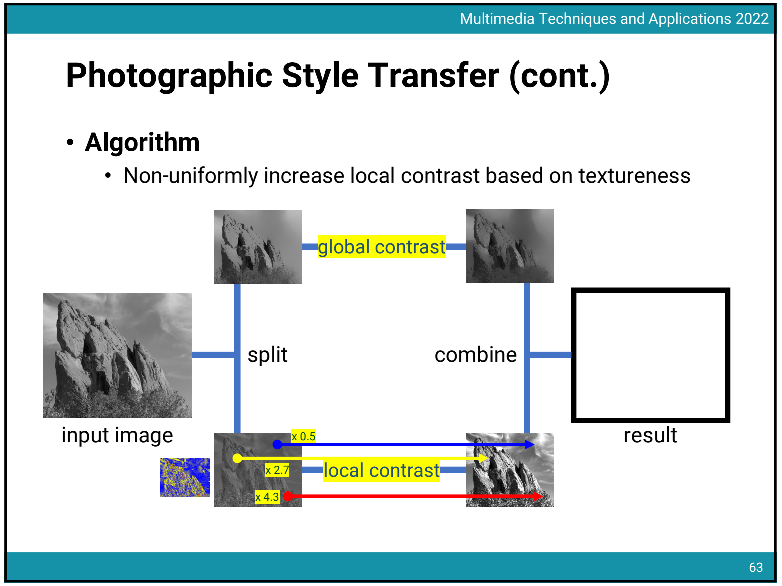
Multimedia Techniques and Applications 2022

Photographic Style Transfer (cont.)

- **Algorithm**
  - Textureness computation

62

62



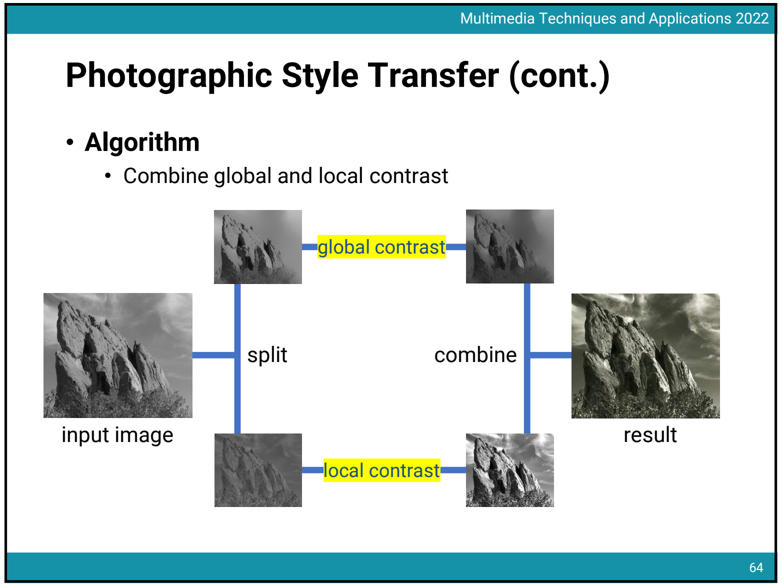
Multimedia Techniques and Applications 2022

Photographic Style Transfer (cont.)

- **Algorithm**
  - Non-uniformly increase local contrast based on textureness

63

63



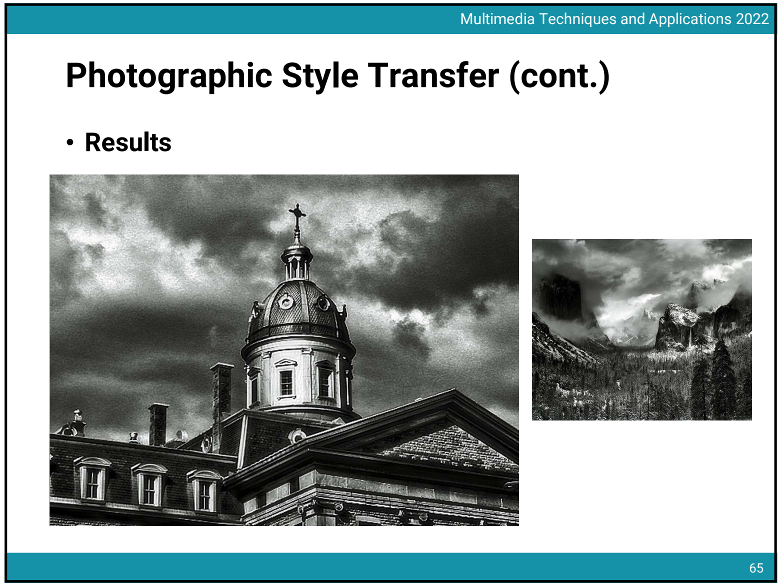
Multimedia Techniques and Applications 2022

Photographic Style Transfer (cont.)

- **Algorithm**
  - Combine global and local contrast

64

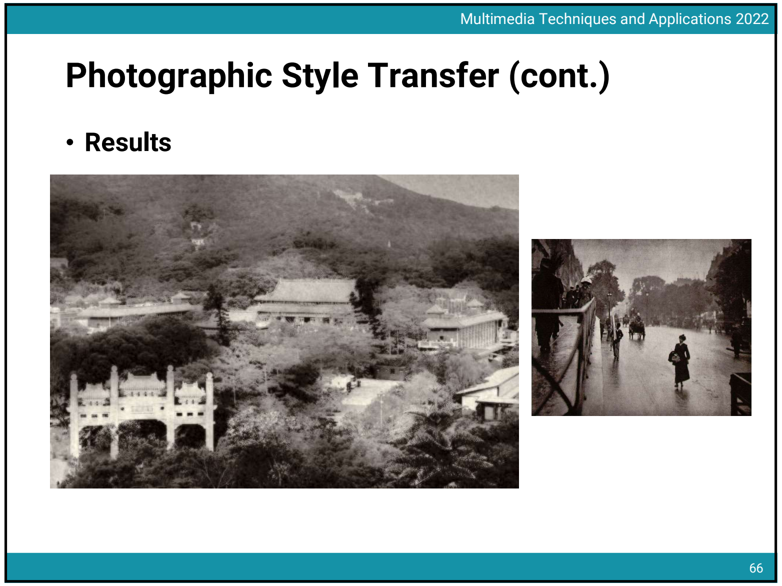
64



Photographic Style Transfer (cont.)

- **Results**

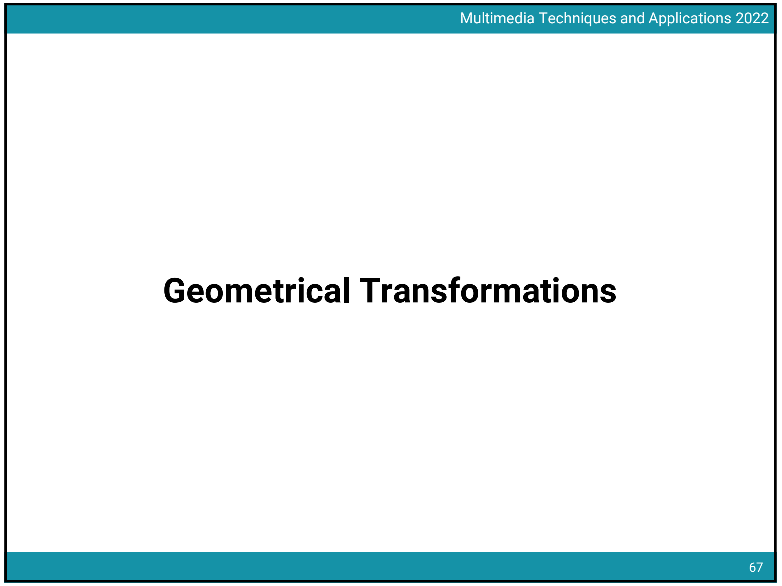
65



Photographic Style Transfer (cont.)

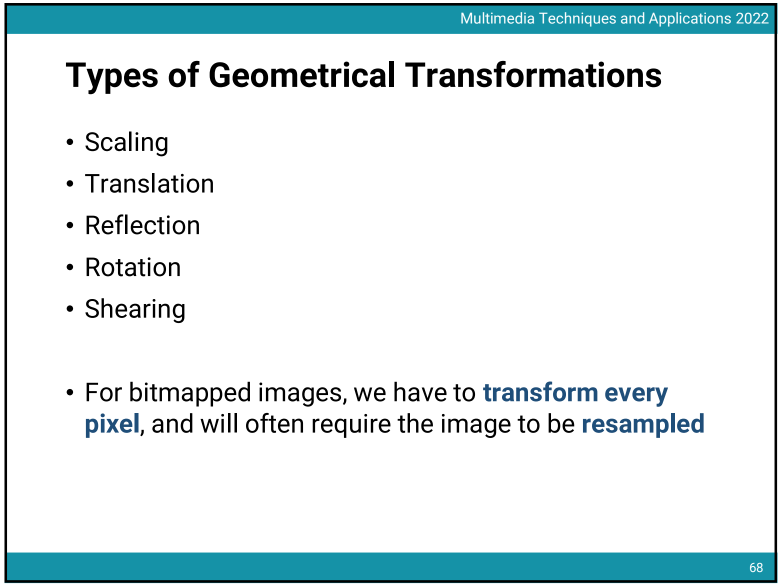
- **Results**

66



Geometrical Transformations

67

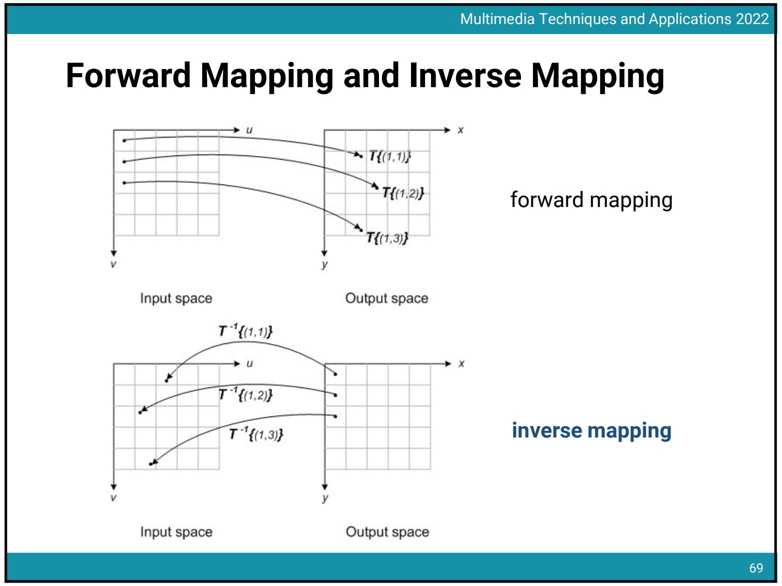


Types of Geometrical Transformations

- Scaling
- Translation
- Reflection
- Rotation
- Shearing

- For bitmapped images, we have to **transform every pixel**, and will often require the image to be **resampled**

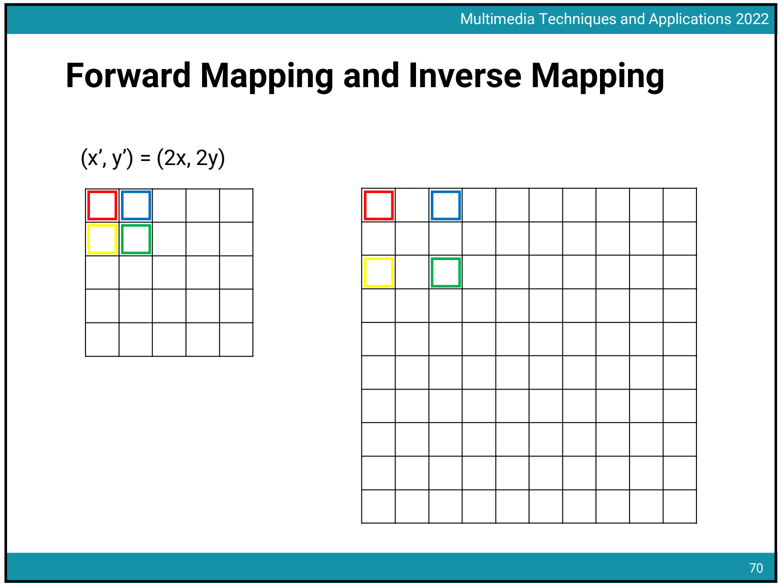
68



Forward Mapping and Inverse Mapping

forward mapping

inverse mapping



Forward Mapping and Inverse Mapping

(x', y') = (2x, 2y)

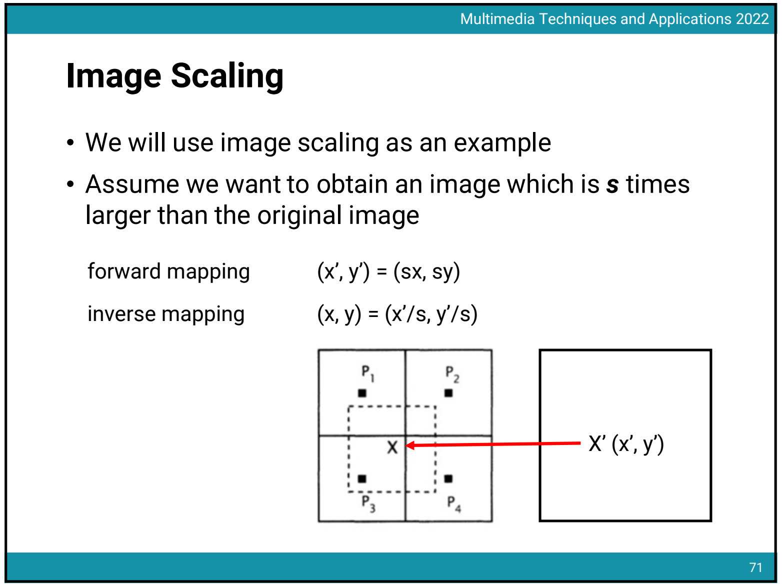


Image Scaling

- We will use image scaling as an example
- Assume we want to obtain an image which is ***s*** times larger than the original image

forward mapping	(x', y') = (sx, sy)
inverse mapping	(x, y) = (x'/s, y'/s)

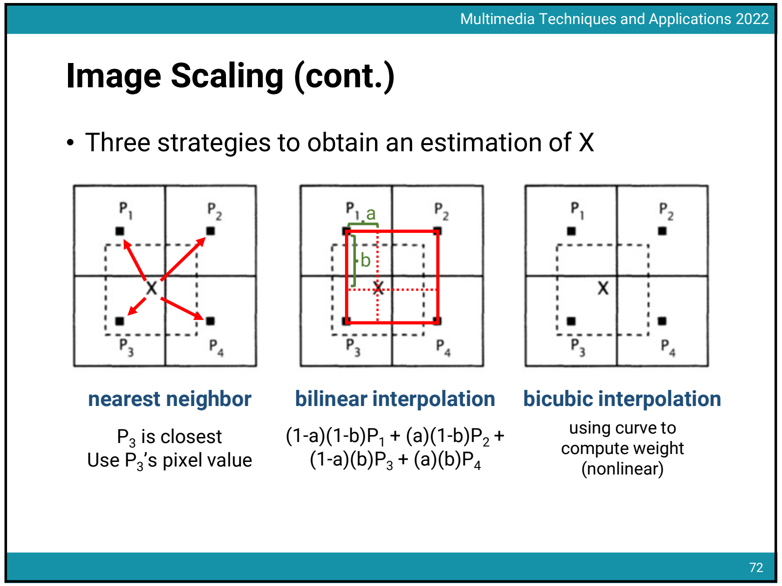


Image Scaling (cont.)

- Three strategies to obtain an estimation of X

nearest neighbor

P_3 is closest
Use P_3's pixel value

bilinear interpolation

$(1\text{-}a)(1\text{-}b)P_1 + (a)(1\text{-}b)P_2 + (1\text{-}a)(b)P_3 + (a)(b)P_4$

bicubic interpolation

using curve to
compute weight
(nonlinear)

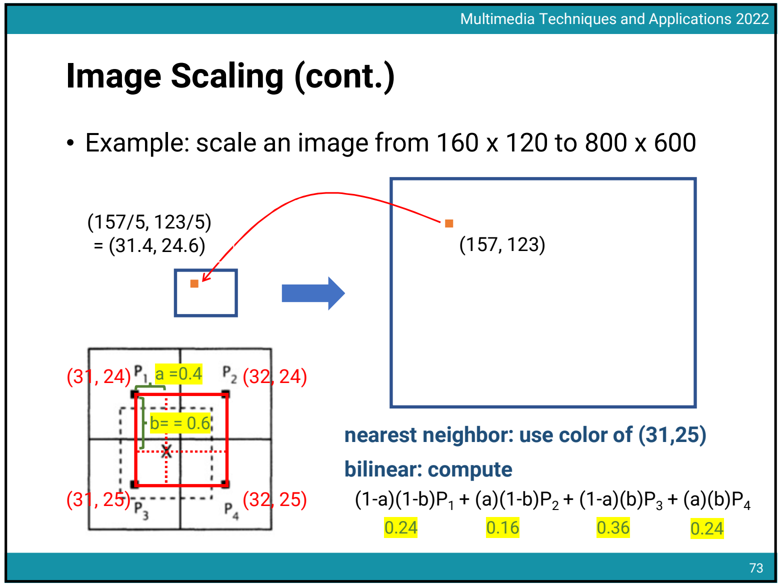


Image Scaling (cont.)

- Example: scale an image from 160 x 120 to 800 x 600

nearest neighbor: use color of (31,25)

bilinear: compute

$(1-a)(1-b)P_1 + (a)(1-b)P_2 + (1-a)(b)P_3 + (a)(b)P_4$

0.24 0.16 0.36 0.24

73

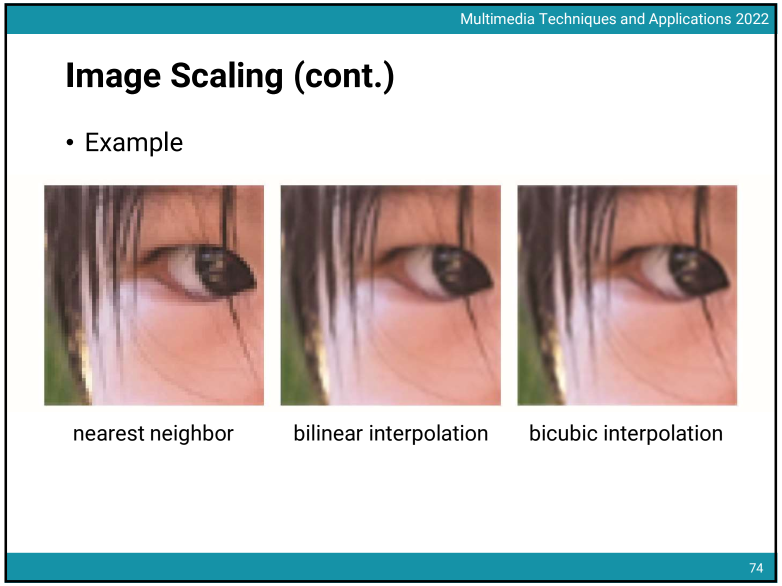


Image Scaling (cont.)

- Example

nearest neighbor bilinear interpolation bicubic interpolation

74

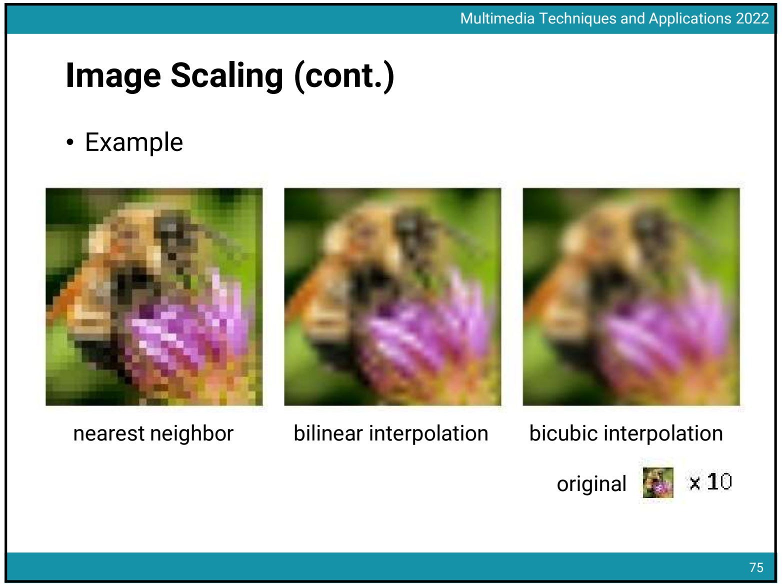


Image Scaling (cont.)

- Example

nearest neighbor bilinear interpolation bicubic interpolation

original x 10

75

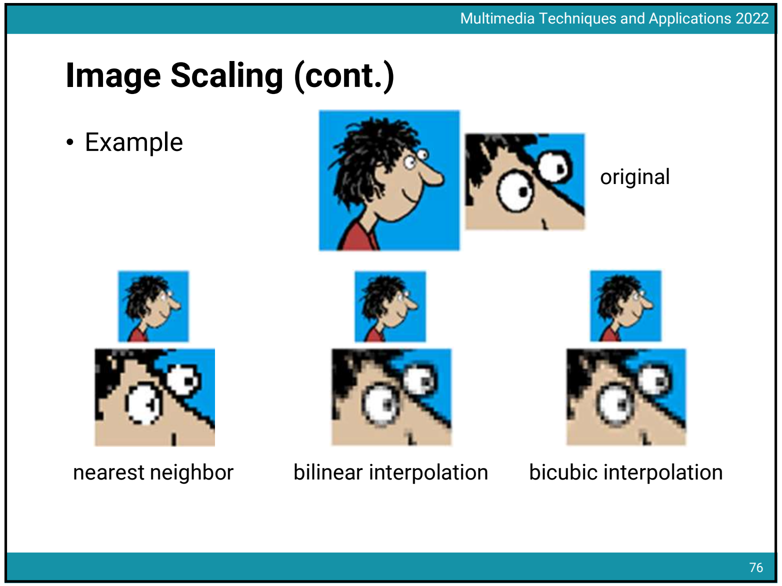


Image Scaling (cont.)

- Example

original

nearest neighbor bilinear interpolation bicubic interpolation

76

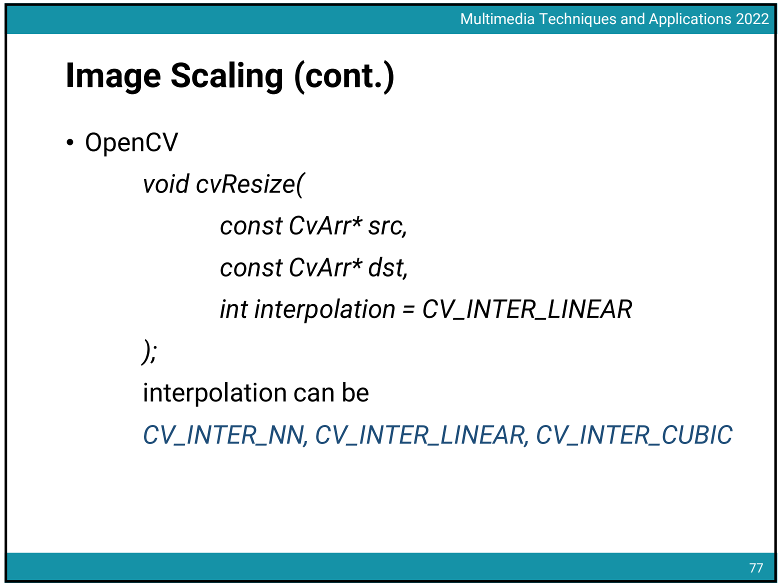


Image Scaling (cont.)

- OpenCV

```
void cvResize(
        const CvArr* src,
        const CvArr* dst,
        int interpolation = CV_INTER_LINEAR
);
```

interpolation can be

CV_INTER_NN, CV_INTER_LINEAR, CV_INTER_CUBIC