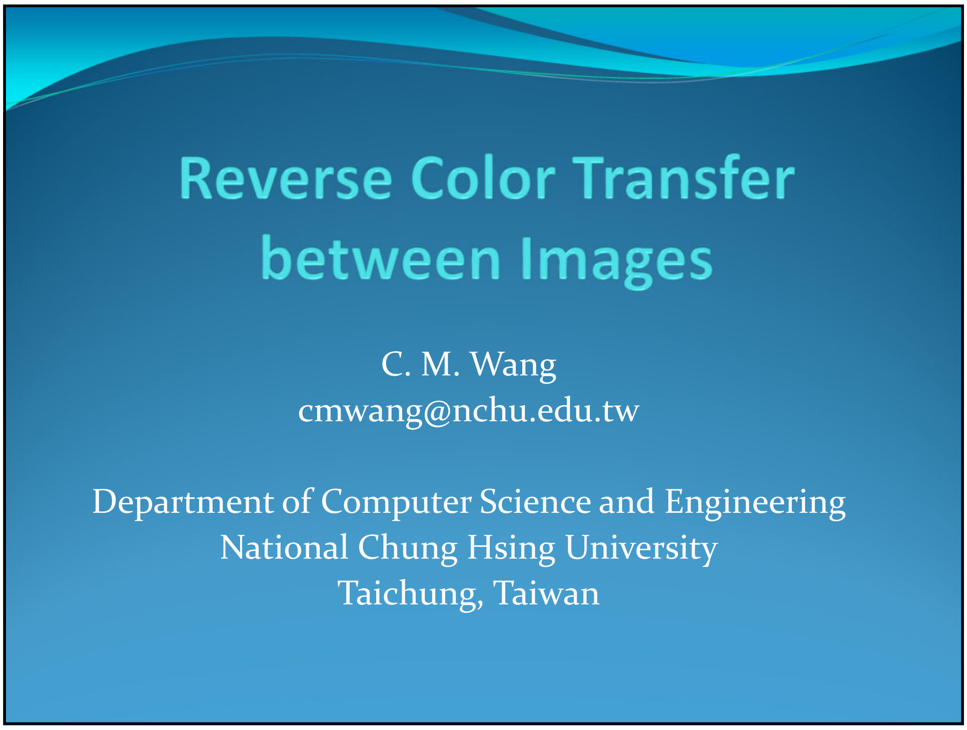
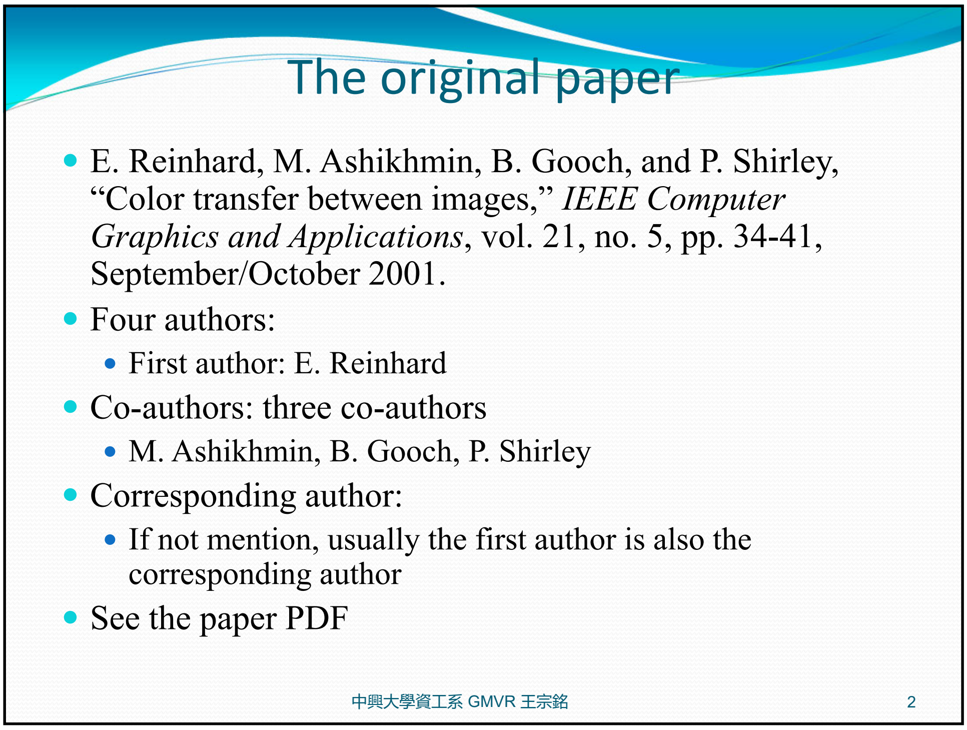


Reverse Color Transfer between Images

C. M. Wang
cmwang@nchu.edu.tw

Department of Computer Science and Engineering
National Chung Hsing University
Taichung, Taiwan

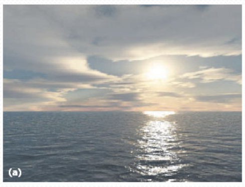


The original paper

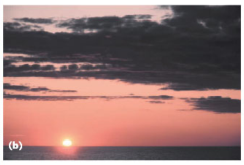
- E. Reinhard, M. Ashikhmin, B. Gooch, and P. Shirley, "Color transfer between images," *IEEE Computer Graphics and Applications*, vol. 21, no. 5, pp. 34-41, September/October 2001.
- Four authors:
  - First author: E. Reinhard
- Co-authors: three co-authors
  - M. Ashikhmin, B. Gooch, P. Shirley
- Corresponding author:
  - If not mention, usually the first author is also the corresponding author
- See the paper PDF

Color Transfer

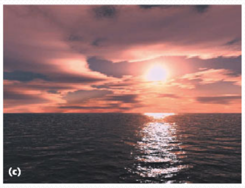
- One of tasks for image processing: altering an image's color
- Goal of this paper
  - Describing a method that borrows one image's color characteristics from another
  - Source image: the original (standard image)
  - Target image: the colors of an image intends to be altered



source

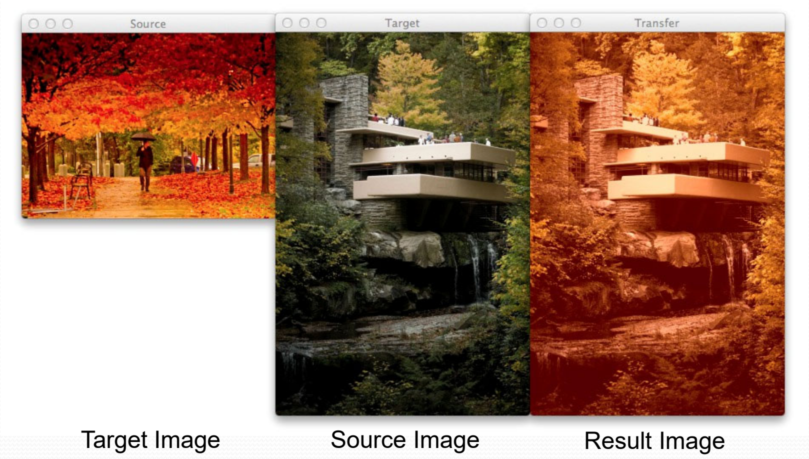


target



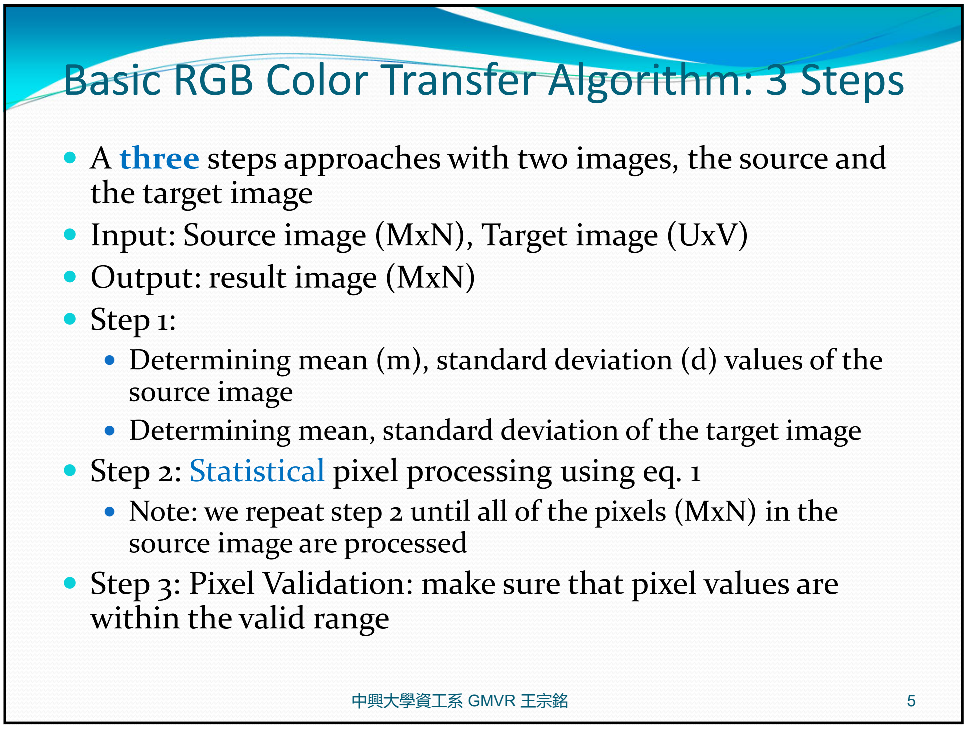
Result

Example



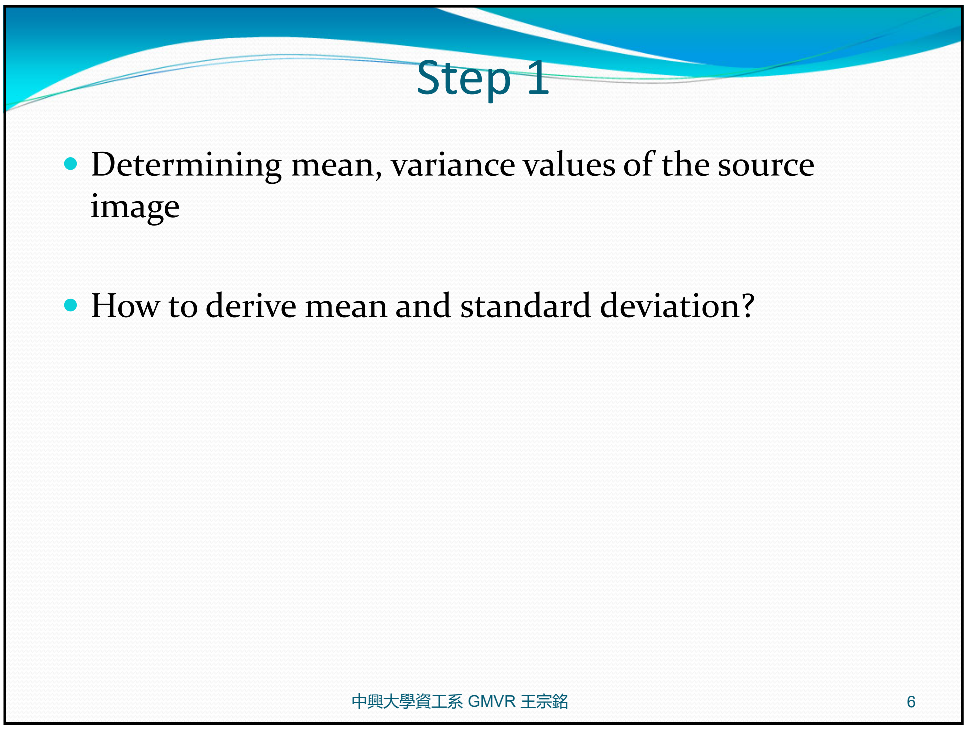
Target Image Source Image Result Image

http://www.pyimagesearch.com/2014/06/30/super-fast-color-transfer-images/



Basic RGB Color Transfer Algorithm: 3 Steps

- A **three** steps approaches with two images, the source and the target image
- Input: Source image (MxN), Target image (UxV)
- Output: result image (MxN)
- Step 1:
  - Determining mean (m), standard deviation (d) values of the source image
  - Determining mean, standard deviation of the target image
- Step 2: Statistical pixel processing using eq. 1
  - Note: we repeat step 2 until all of the pixels (MxN) in the source image are processed
- Step 3: Pixel Validation: make sure that pixel values are within the valid range



Step 1

- Determining mean, variance values of the source image
- How to derive mean and standard deviation?

Mean and Standard Deviation

- Mean and Standard Deviation

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}, \quad \text{where} \quad \mu = \frac{1}{N}\sum_{i=1}^{N} x_i.$$

- Example: considering the following 8 values

$$2,\ 4,\ 4,\ 4,\ 5,\ 5,\ 7,\ 9.$$

- Mean is 5 since

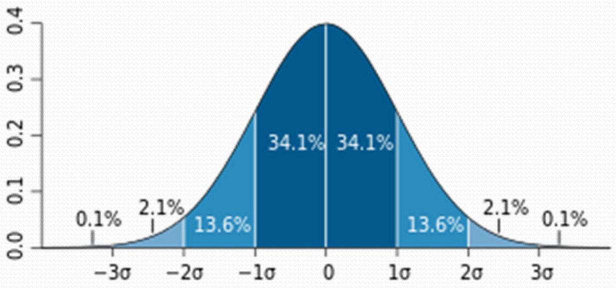
$$\frac{2+4+4+4+5+5+7+9}{8} = 5.$$

$$\begin{array}{ll} (2-5)^2 = (-3)^2 = 9 & (5-5)^2 = 0^2 = 0 \\ (4-5)^2 = (-1)^2 = 1 & (5-5)^2 = 0^2 = 0 \\ (4-5)^2 = (-1)^2 = 1 & (7-5)^2 = 2^2 = 4 \\ (4-5)^2 = (-1)^2 = 1 & (9-5)^2 = 4^2 = 16. \end{array}$$

- Standard deviation is 2 since $\sqrt{\frac{9+1+1+1+0+0+4+16}{8}} = 2.$

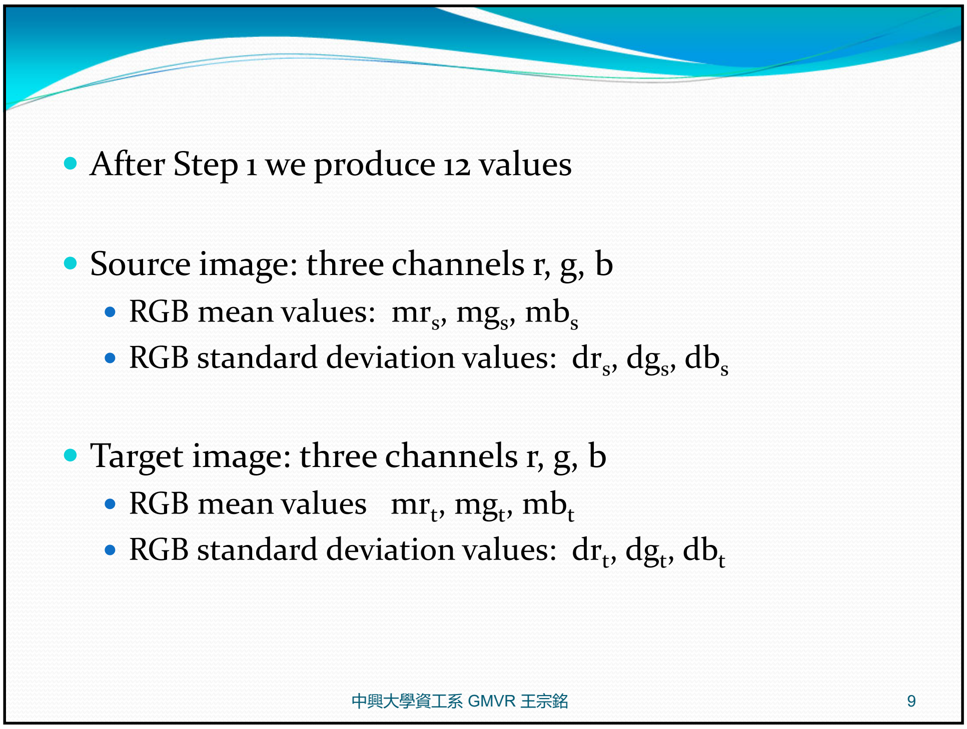
Standard Deviation (SD) (σ)

- a measure used to **quantify** the amount of variation (變化程度)or dispersion (偏差) of a set of data values
- σ is near 0 indicating that the data points tend to be very close to the mean

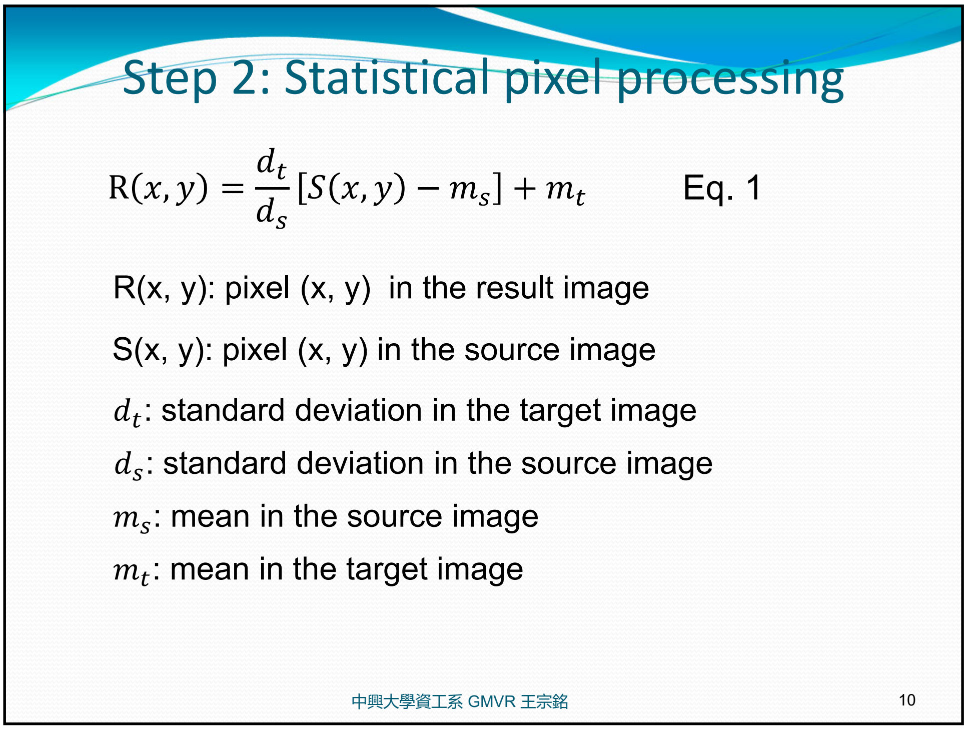


A plot of a normal distribution (or bell-shaped curve) where each band has a width of 1 standard deviation
Rule: 68.2-95.4-99.7

http://en.wikipedia.org/wiki/Standard_deviation



- After Step 1 we produce 12 values
- Source image: three channels r, g, b
  - RGB mean values: mr_s, mg_s, mb_s
  - RGB standard deviation values: dr_s, dg_s, db_s
- Target image: three channels r, g, b
  - RGB mean values mr_t, mg_t, mb_t
  - RGB standard deviation values: dr_t, dg_t, db_t



Step 2: Statistical pixel processing

$$\mathrm{R}(x,y) = \frac{d_t}{d_s}[S(x,y) - m_s] + m_t \qquad \text{Eq. 1}$$

R(x, y): pixel (x, y) in the result image

S(x, y): pixel (x, y) in the source image

d_t: standard deviation in the target image

d_s: standard deviation in the source image

m_s: mean in the source image

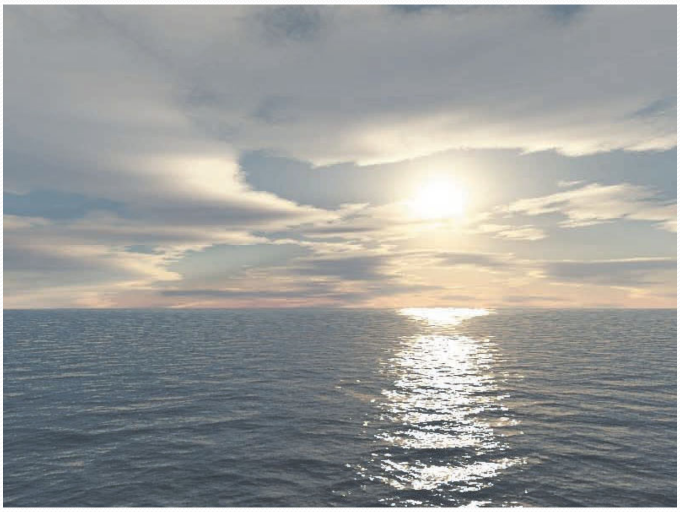
m_t: mean in the target image

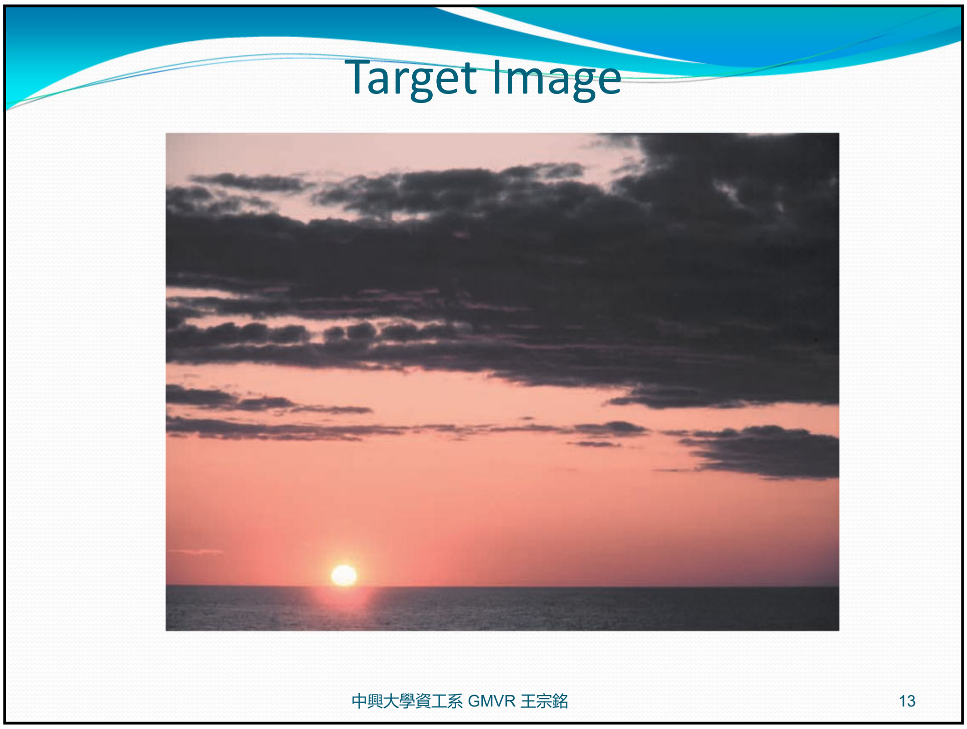
Step 3: Pixel Validation

- We need to derive an integer representation after pixel processing.
- Use Floor function to derive an integer that is close to the floating value
- Make sure the pixel value is within [0, 255]

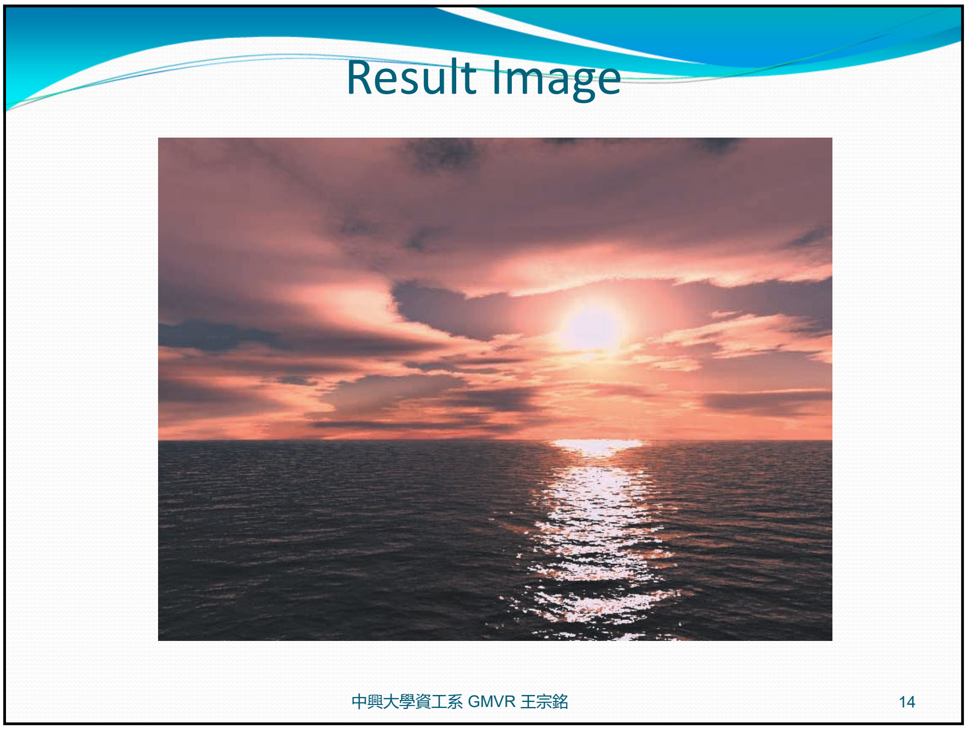
$$R(x,y) = \left\lfloor \frac{d_t}{d_s}[S(x,y) - m_s] + m_t + 0.5 \right\rfloor$$

Source Image

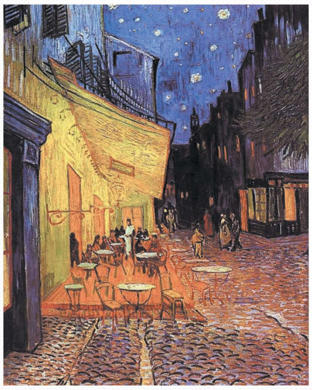




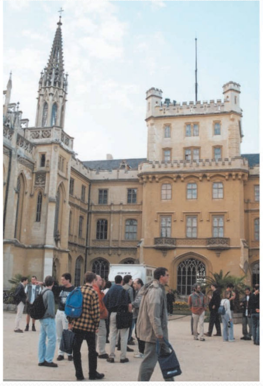
Target Image



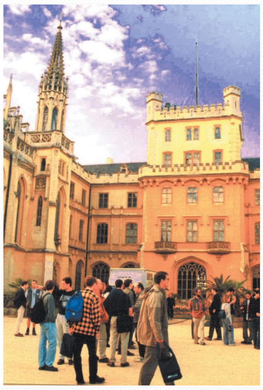
Result Image



Target Image



Source Image

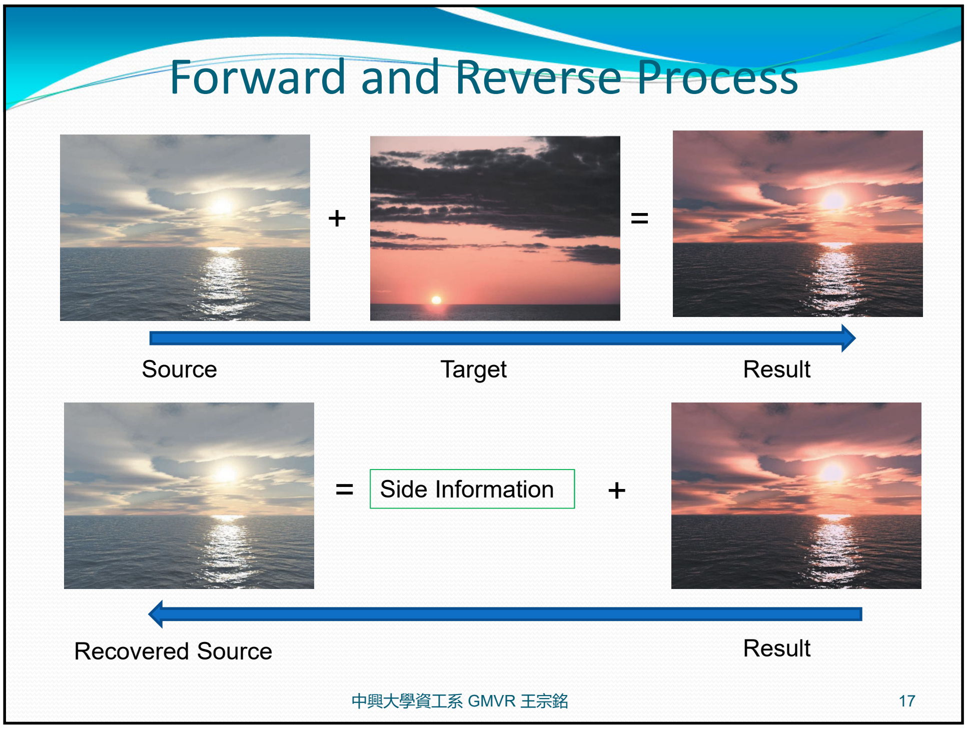


Result Image

Reversible Color Transfer

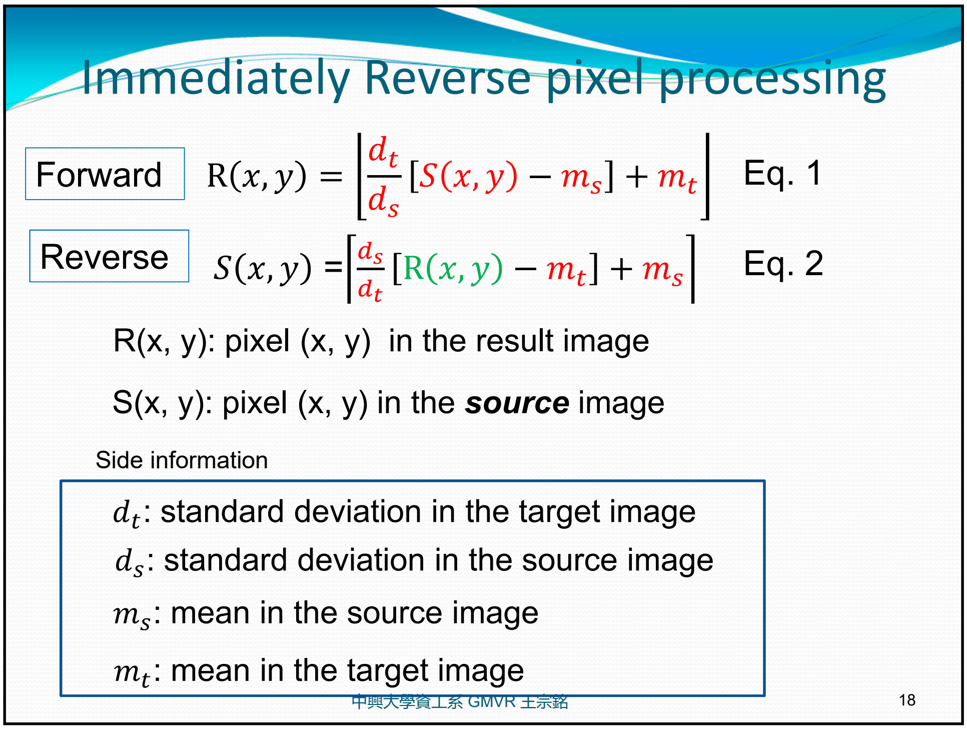
- Color transfer is a forward process:
- source + target → result image

- Can we reverse color transfer process?
- Reverse process:
- result + side information → recovered source
- Source and Recovered Source may have a subtle difference
- How to measure the difference between two images?



Forward and Reverse Process

+
=
Source
Target
Result
=
Side Information
+
Recovered Source
Result



Immediately Reverse pixel processing

Forward $$\mathrm{R}(x,y) = \left|\frac{d_t}{d_s}[S(x,y) - m_s] + m_t\right|$$ Eq. 1

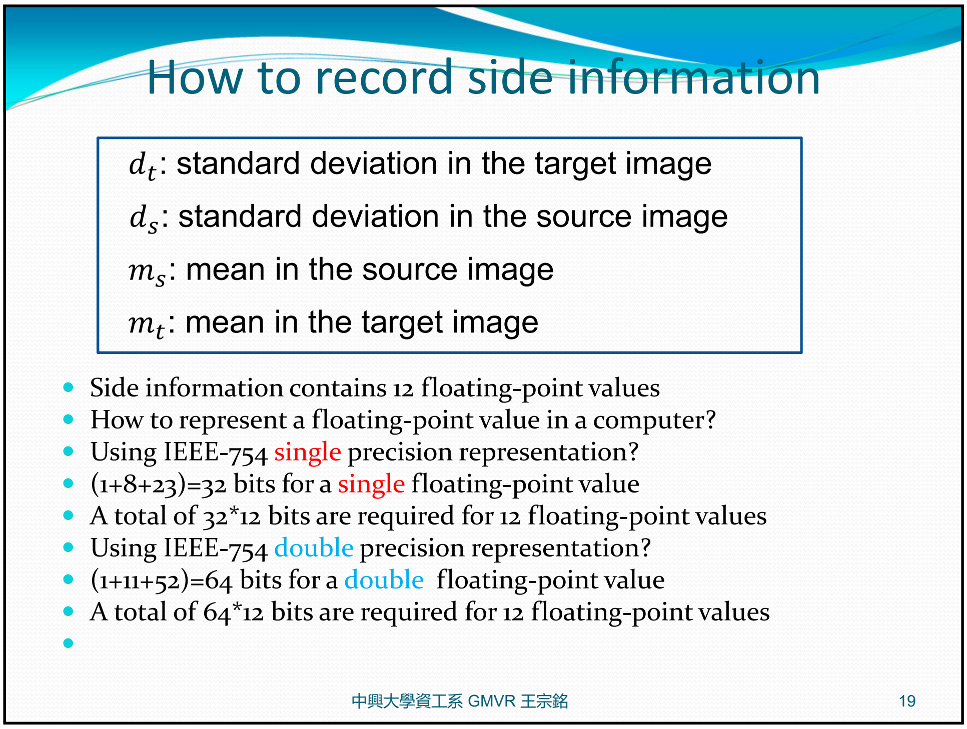
Reverse $$S(x,y) = \left|\frac{d_s}{d_t}[\mathrm{R}(x,y) - m_t] + m_s\right|$$ Eq. 2

R(x, y): pixel (x, y) in the result image

S(x, y): pixel (x, y) in the ***source*** image

Side information

d_t: standard deviation in the target image
d_s: standard deviation in the source image
m_s: mean in the source image
m_t: mean in the target image



How to record side information

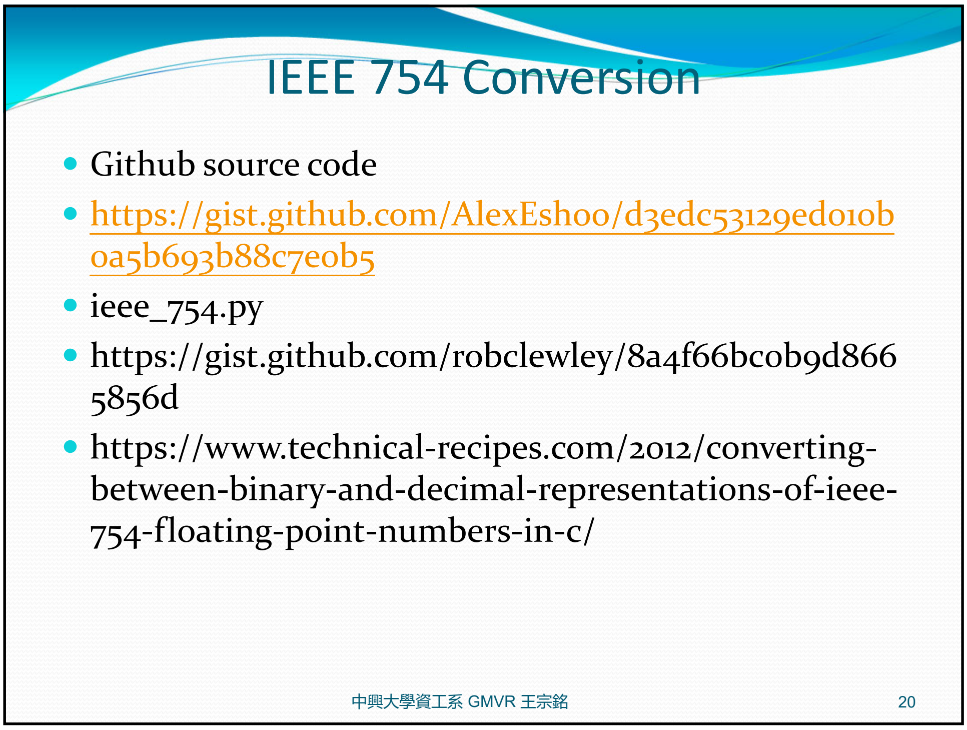
d_t: standard deviation in the target image

d_s: standard deviation in the source image

m_s: mean in the source image

m_t: mean in the target image

- Side information contains 12 floating-point values
- How to represent a floating-point value in a computer?
- Using IEEE-754 single precision representation?
- (1+8+23)=32 bits for a single floating-point value
- A total of 32*12 bits are required for 12 floating-point values
- Using IEEE-754 double precision representation?
- (1+11+52)=64 bits for a double floating-point value
- A total of 64*12 bits are required for 12 floating-point values

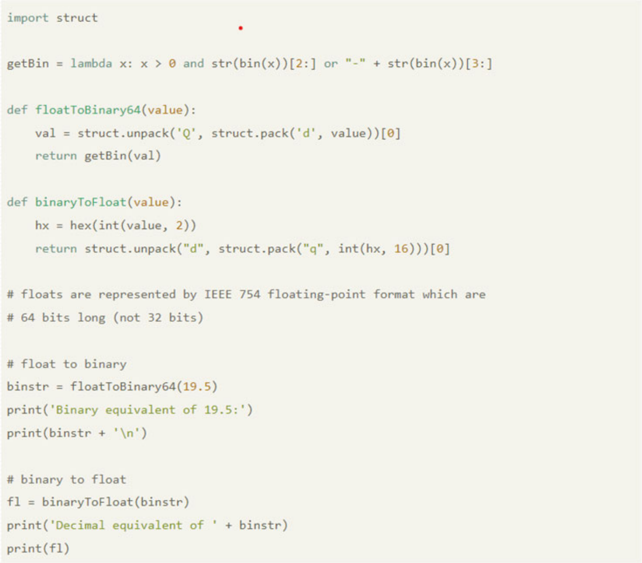


IEEE 754 Conversion

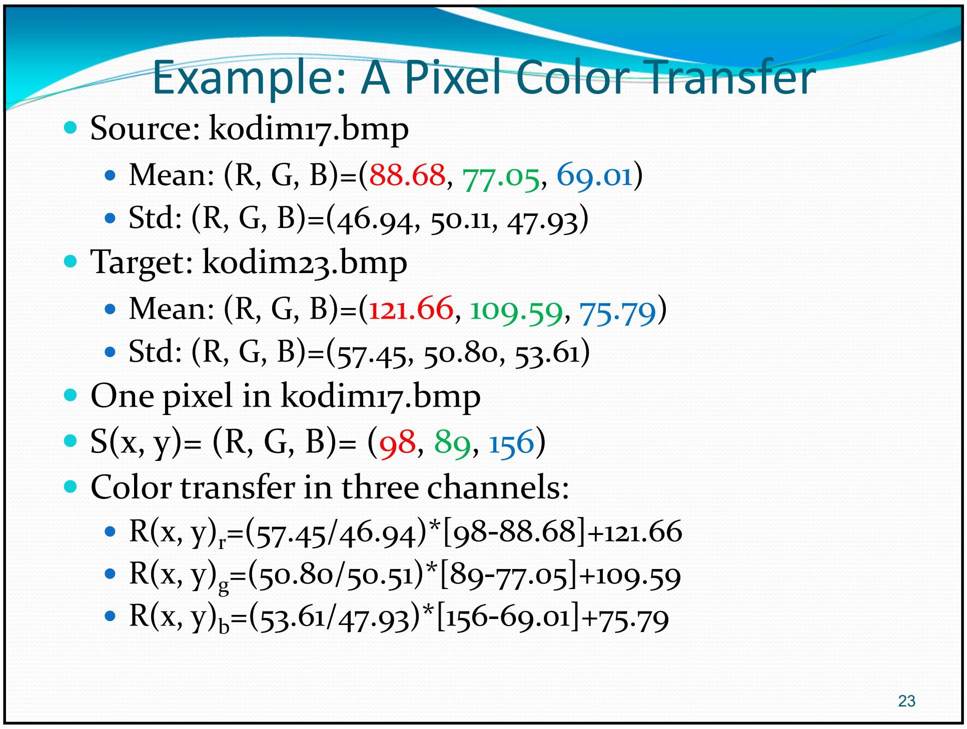
- Github source code
- https://gist.github.com/AlexEshoo/d3edc53129ed010b0a5b693b88c7e0b5
- ieee_754.py
- https://gist.github.com/robclewley/8a4f66bc0b9d8665856d
- https://www.technical-recipes.com/2012/converting-between-binary-and-decimal-representations-of-ieee-754-floating-point-numbers-in-c/

- Python source code
- Remember that in Python floats are represented by IEEE 754 floating-point format which are 64 bits long – not 32 bit

https://www.796t.com/content/1549004412.html



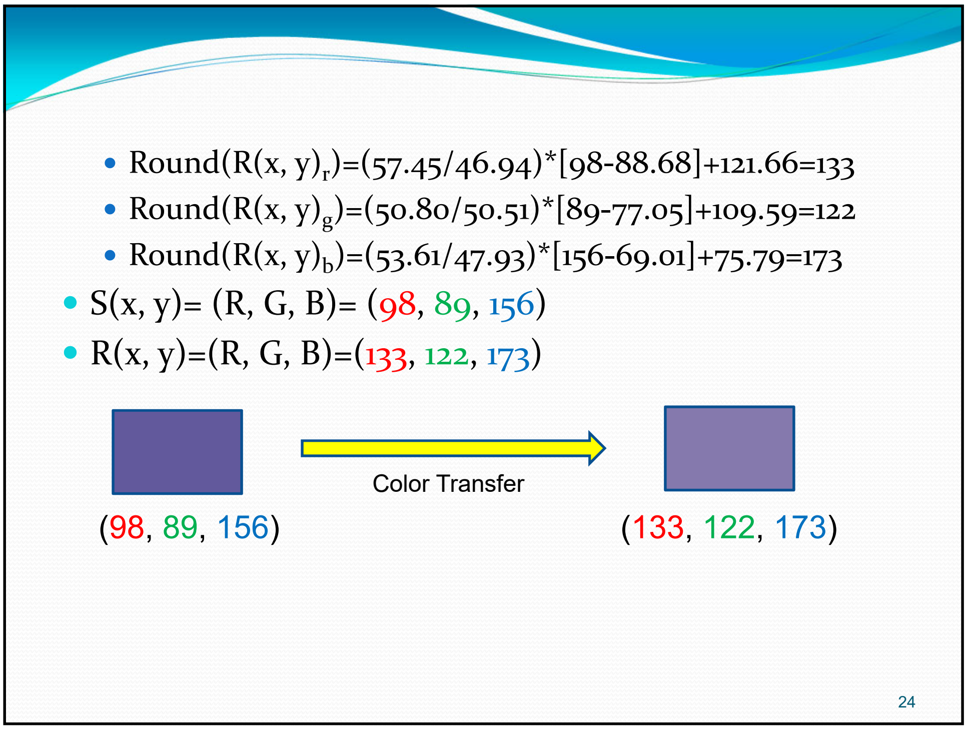
```python
import struct

getBin = lambda x: x > 0 and str(bin(x))[2:] or "-" + str(bin(x))[3:]

def floatToBinary64(value):
    val = struct.unpack('Q', struct.pack('d', value))[0]
    return getBin(val)

def binaryToFloat(value):
    hx = hex(int(value, 2))
    return struct.unpack("d", struct.pack("q", int(hx, 16)))[0]

# floats are represented by IEEE 754 floating-point format which are
# 64 bits long (not 32 bits)

# float to binary
binstr = floatToBinary64(19.5)
print('Binary equivalent of 19.5:')
print(binstr + '\n')

# binary to float
fl = binaryToFloat(binstr)
print('Decimal equivalent of ' + binstr)
print(fl)
```



Example: A Pixel Color Transfer

- Source: kodim17.bmp
  - Mean: (R, G, B)=(88.68, 77.05, 69.01)
  - Std: (R, G, B)=(46.94, 50.11, 47.93)
- Target: kodim23.bmp
  - Mean: (R, G, B)=(121.66, 109.59, 75.79)
  - Std: (R, G, B)=(57.45, 50.80, 53.61)
- One pixel in kodim17.bmp
- S(x, y)= (R, G, B)= (98, 89, 156)
- Color transfer in three channels:
  - $R(x, y)_r=(57.45/46.94)*[98-88.68]+121.66$
  - $R(x, y)_g=(50.80/50.51)*[89-77.05]+109.59$
  - $R(x, y)_b=(53.61/47.93)*[156-69.01]+75.79$

23



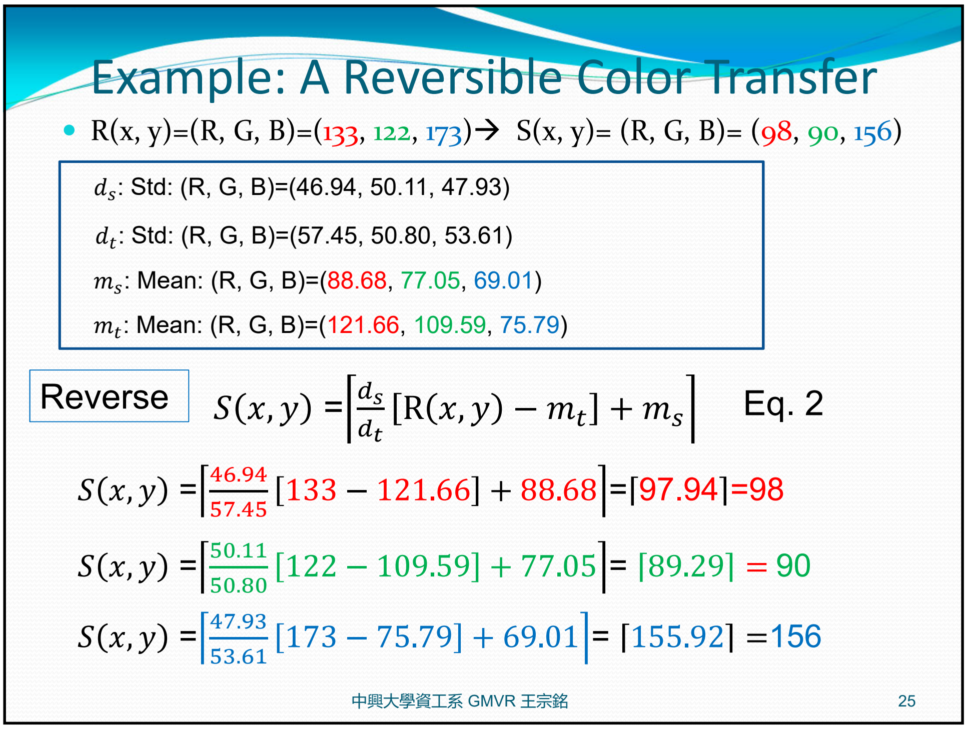
- $\text{Round}(R(x, y)_r)=(57.45/46.94)*[98-88.68]+121.66=133$
- $\text{Round}(R(x, y)_g)=(50.80/50.51)*[89-77.05]+109.59=122$
- $\text{Round}(R(x, y)_b)=(53.61/47.93)*[156-69.01]+75.79=173$
- S(x, y)= (R, G, B)= (98, 89, 156)
- R(x, y)=(R, G, B)=(133, 122, 173)

Color Transfer

(98, 89, 156)

(133, 122, 173)

24



Example: A Reversible Color Transfer

- R(x, y)=(R, G, B)=(133, 122, 173)→ S(x, y)= (R, G, B)= (98, 90, 156)

d_s: Std: (R, G, B)=(46.94, 50.11, 47.93)

d_t: Std: (R, G, B)=(57.45, 50.80, 53.61)

m_s: Mean: (R, G, B)=(88.68, 77.05, 69.01)

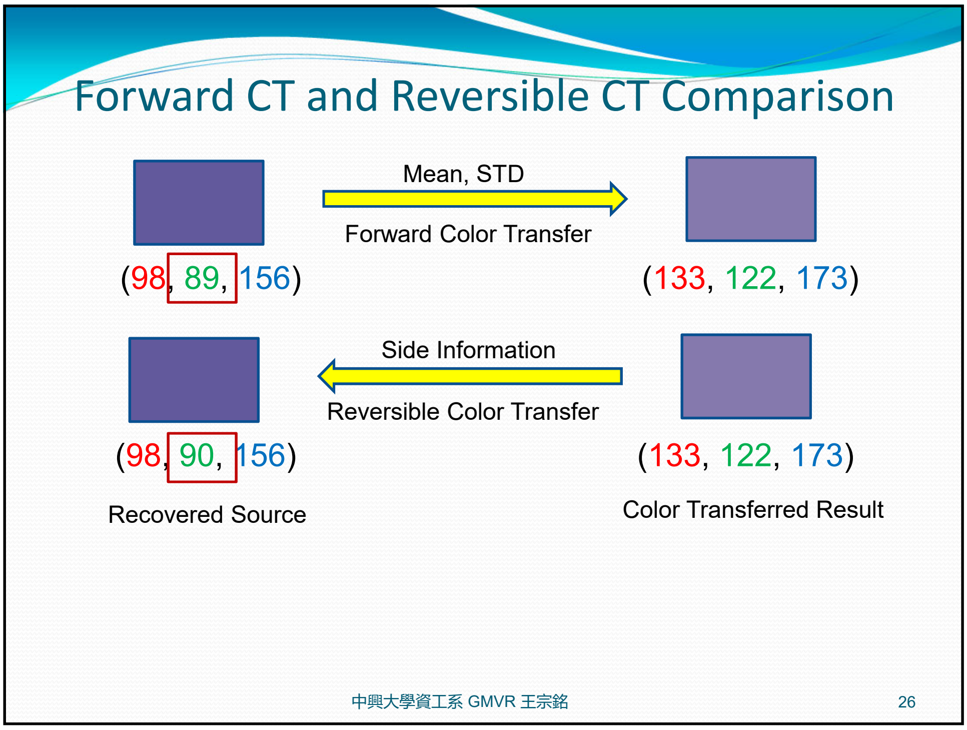
m_t: Mean: (R, G, B)=(121.66, 109.59, 75.79)

Reverse $$S(x,y) = \left\lceil \frac{d_s}{d_t}[R(x,y) - m_t] + m_s \right\rceil$$ Eq. 2

$$S(x,y) = \left\lceil \frac{46.94}{57.45}[133 - 121.66] + 88.68 \right\rceil = [97.94] = 98$$

$$S(x,y) = \left\lceil \frac{50.11}{50.80}[122 - 109.59] + 77.05 \right\rceil = [89.29] = 90$$

$$S(x,y) = \left\lceil \frac{47.93}{53.61}[173 - 75.79] + 69.01 \right\rceil = [155.92] = 156$$



Forward CT and Reversible CT Comparison

Mean, STD

Forward Color Transfer

(98, 89, 156)

(133, 122, 173)

Side Information

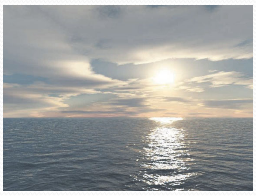
Reversible Color Transfer

(98, 90, 156)

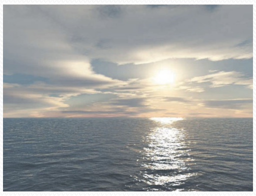
(133, 122, 173)

Recovered Source

Color Transferred Result



Source



Recovered Source

Source and Recovered Source may not be exactly identical

How similarities are they?

Quantify image quality using a metric

Peak Signal Noise Ratio: PSNR

https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio

- **Structure Similarity Index (SSIM)**
- **https://zh.wikipedia.org/wiki/%E7%B5%90%E6%A7%8B%E7%9B%B8%E4%BC%BC%E6%80%A7**

- 計算兩影像的**PSNR**、**SSIM**和**Python**程式碼