PROGRAMMING

Lecture 14

Hanbat National University
Dept. of Computer Engineering
Changbeom Choi

OUTLINE

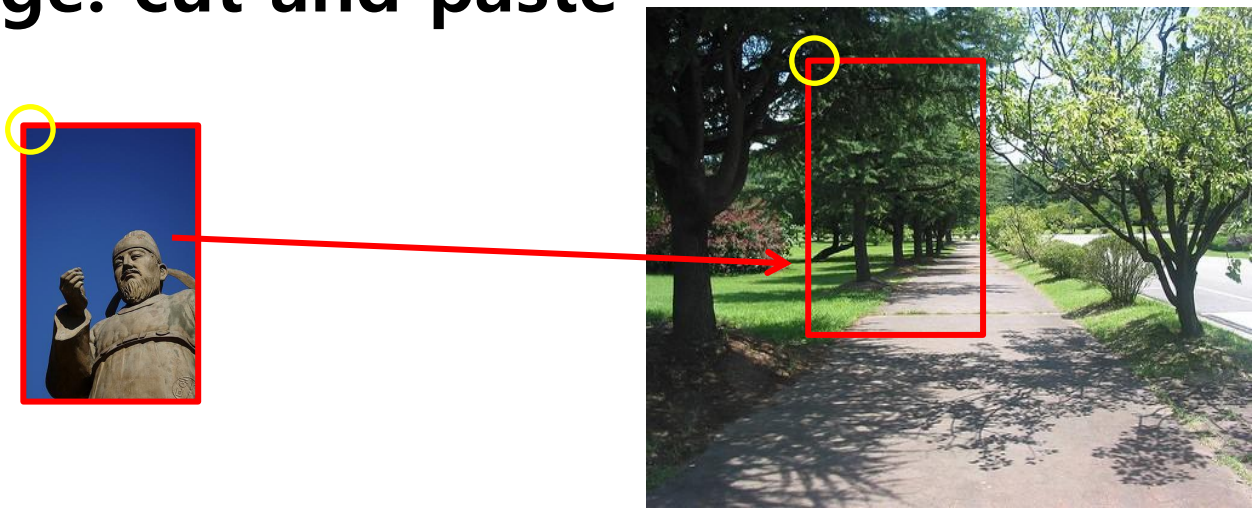
More on photo processing

Word play

DSU patterns

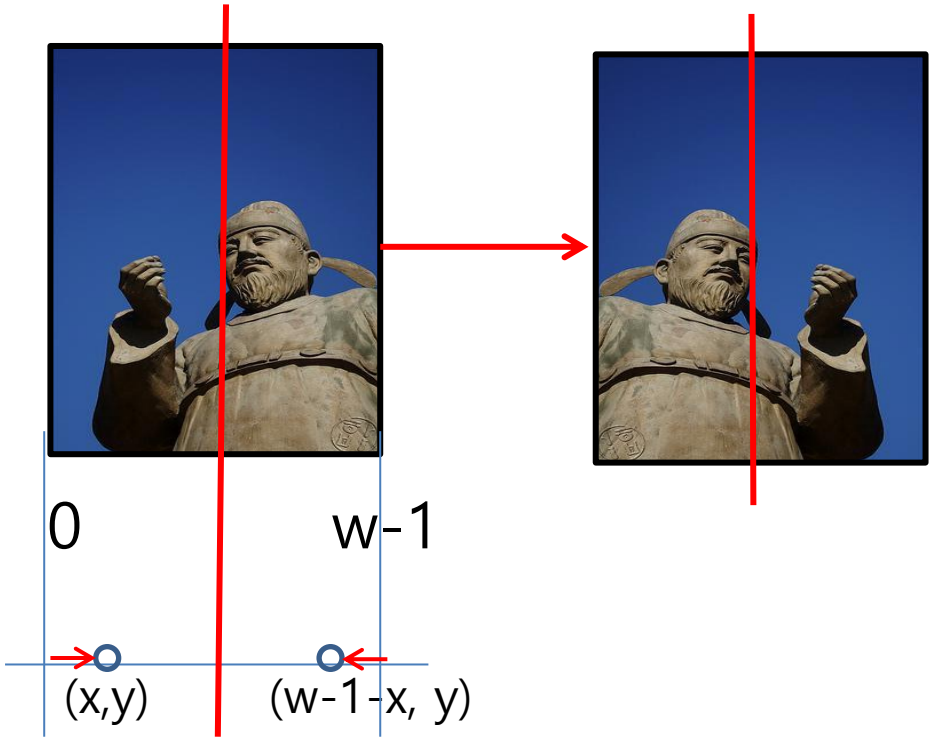
MORE ON PHOTO PROCESSING

Collage: cut and paste



```
def paste(canvas, img, x1, y1):
    w, h = img.size()
    for y in range(h):
        for x in range(w):
            p_color = img.get(x, y)
            canvas.set(x1 + x, y1 + y, p_color)
```

Reflection

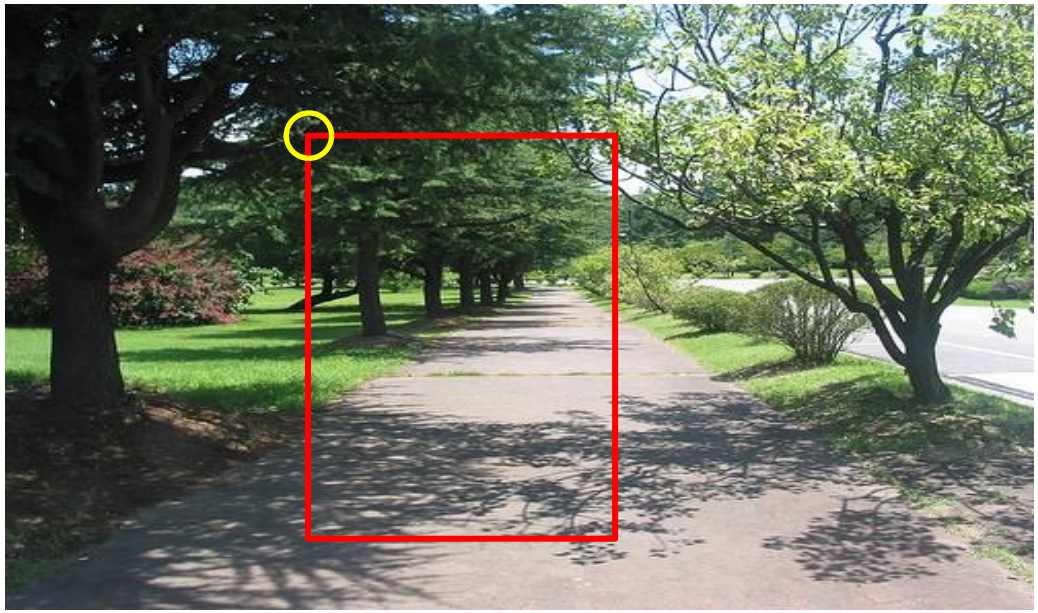



```
def reflection(img):
  w, h = img.size()
  for y in range(0,h)
     for x in range(0, w//2)
        pl = img.get(x, y)
        pr = img.get(w-1–x, y)
        img.set(x, y, pr)
        img.set(w-1-x, y, pl)
```

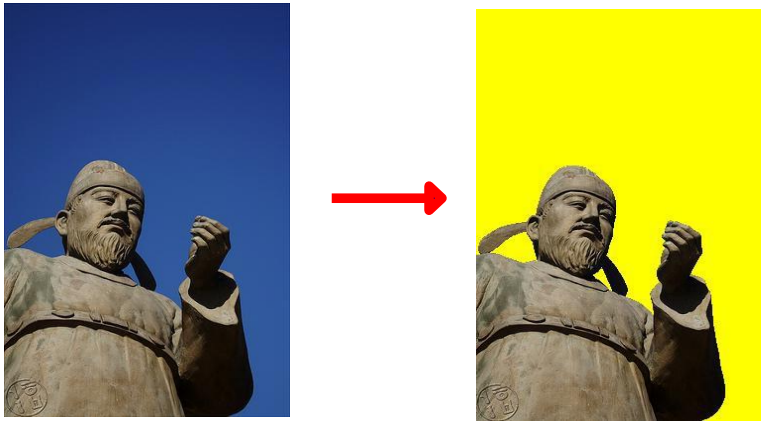
Any better way ?

Chromakey

- A technique to overlay one scene on top of another one.
- Commonly used for weather maps.



Blue screen image for identifying a foreground object



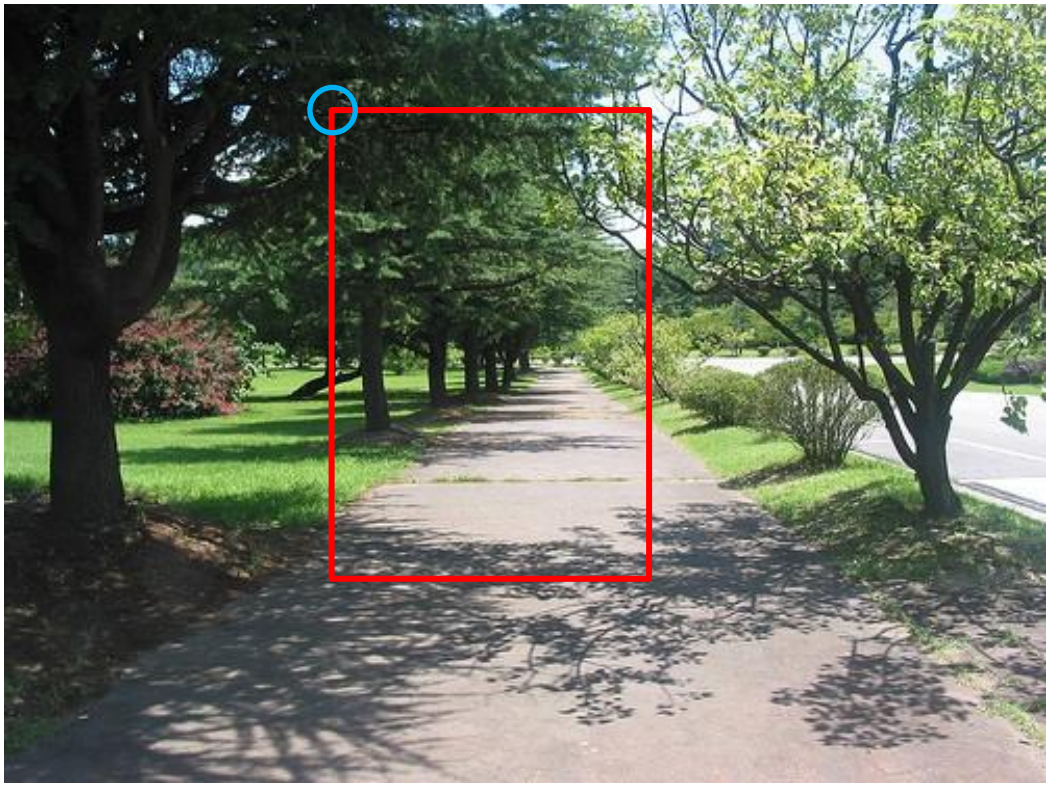
```
def chroma_yellow(img,key,threshold):
  w, h = img.size()
  for y in range(h):
    for x in range(w):
          p = img.get(x, y)
          if dist(p, key) < threhold:
               img.set(x, y, Color.yellow)
```

background color

How to measure color difference

```
def dist(color, back):
    r1, g1, b1 = color
    r2, g2, b2 = back
    return( math.sqrt((r1 – r2) ** 2)
                + (g1 – g2)**2
                + (b1 – b2)**2 ))
```

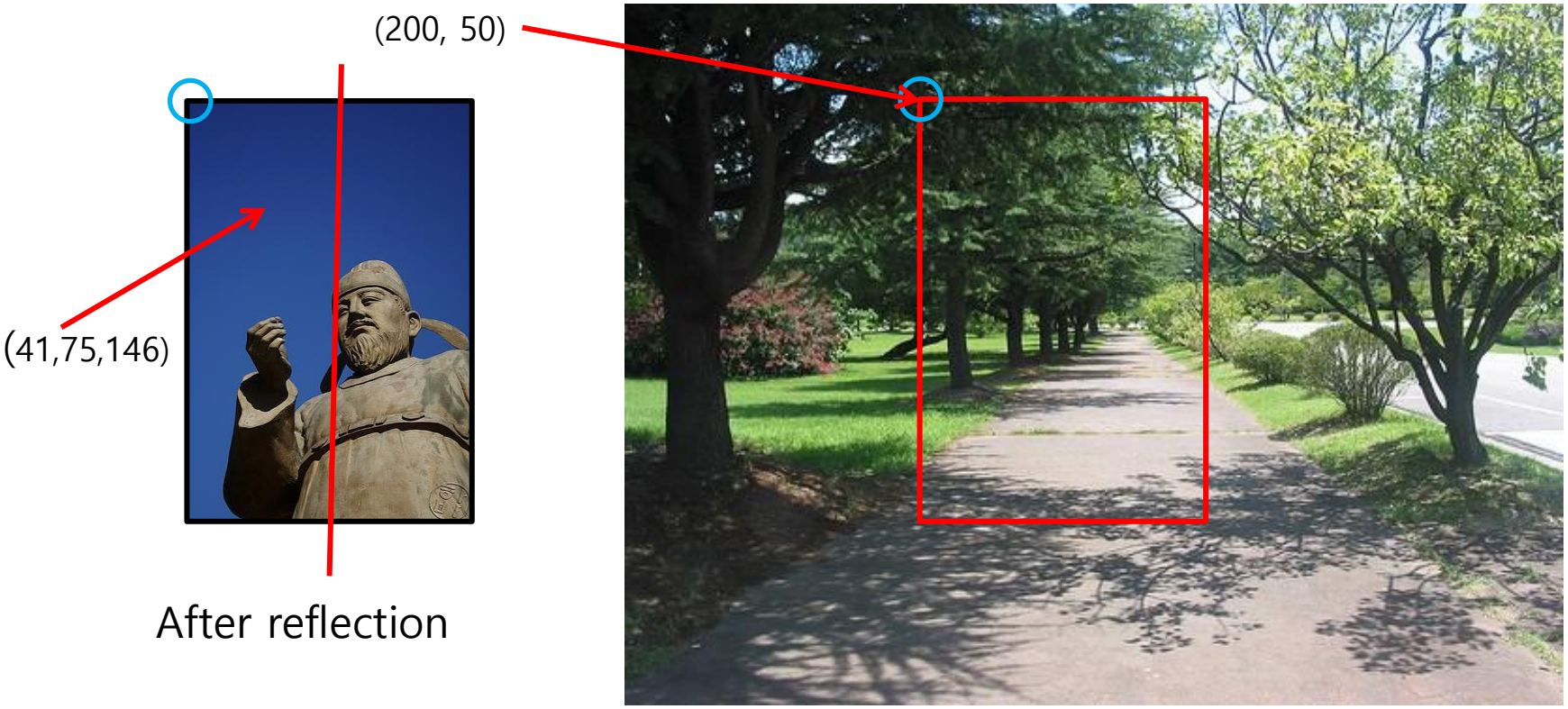
Euclidian distance in the (r, g, b) color space!



PROBLEM 1: CHROMAKEY

Given a **campus image** and another image containing the **foreground object**(**statue**), place the foreground object(statue) of the latter in the former starting from the pixel at (200,50), that is, **(x1, y1) = (200, 50),** using the chromakey technique. You may assume that the latter is taken with a blue screen. Use (41, 75, 146) as the key, that is, **key = (41, 75, 146),** to identify the background (a blue screen) of the latter. You may set threshold = 70.

(Continued)


(200, 50)
(41,75,146)
After reflection

```
def chroma(x1,y1,key,threshold):
    w, h = statue.size()
    for y in range(h):
        for x in range(w):
            p = statue.get(x, y)
            if dist(p, key) > threhold: foreground object
                campus.set(x1+x, y1+y,p)


    campus.show()
```

```
from cs1media import *
import math
campus = load_picture("photos/trees1.jpg")
statue = load_picture("photos/statue1.jpg")

Fill in this box.


def main():
    reflection()
    chroma(200, 50, (41, 75, 146), 70)

main()
```

WORD PLAY

PROBLEM 2: PLAYING WITH WORDS

For this problem we need a **list** of **English words**. We are going to use the word list collected and contributed to the public domain by **Grady Ward** (http://wikipedia.org/wiki/Moby_Project). It is a list of 113,809 words. You can download it from our homepage (words.txt). Please name your file also as **words.txt**. We will play with words in this file.

Selecting long words

2-1. Count all words in word.txt longer than 18 letters

Print these words and the number of such words.

anticonservationist
comprehensivenesses
counterdemonstration
counterdemonstrations
counterdemonstrator
.....
representativenesses
no. of words = 24

```
file = open("words.txt", "r")
count = 0
for word in file:
    word = word.strip()
    if len(word) > 18:
        count += 1
        print word
print "no. of words = ", count
```

Finding palindromes

2-2. A palindrome is a word that reads the same way forwards and backwards. Print out all palindromes in a file, “words.txt”. You may use the function shown on the left.

```
def is_Palindrome(s):
    start = 0
    end = len(s) - 1
    for i in range(len(s)//2):
        if s[start] != s[end]:
            return False
        else:
            start += 1
            end -= 1
    return True
```

2-3. Count all words without the **letter "e"** and print the number of such words.

2-4. Is there a word with **triple letters**(three of the same letters in a row)?

2-5. An **abecedarian** is a word in which letters are sorted. Count the number of abecedarians and print these words and the number of such words.

Read Chapter 9 of your textbook to find some solutions.

DSU PATTERNS

DSU is abbreviation of "**Decorate-Sort-Undecorate**" , which is a pattern that involves **building a list of tuples, sorting the list,** and **extracting the result from the list.**

Example: Sort the words in "words.txt" in the decreasing order of their lengths(from longest to shortest):

The length of word is not available in "words.txt" !

```
list1 = ["four", "by", "pot", "a"]
list2 = []
for word in list1:
    list2.append((len(word), word))

list2.sort(reverse = True)

res = []
for length, word in list2:
    res.append(word)
print res
```

["four", "by", "pot", "a"]

[(4, "four"), (2, "by"),(3, "pot"), (1, "a")]

Decorate

Sort

[(4, "four"), (3, "pot"), (2, "by"),((1, "a")]

Undecorate

["four", "pot", "by", "a"]

DSU PATTERN

PROBLEM 3: 20 MOST FREQUENTLY USED WORDS

The file **“emma.txt”** contains the text of a novel Emma by Jane Austen. Suppose that you are going to find the **20 most frequently-used words** in this book, of which their **lengths between 5 and 10** inclusively. You are to use the word list in the file **“words.txt”** to count the frequency of each of such words while **discarding all words which is not in the file**. The book contains **header information** at the beginning and **whitespace** and **punctuation** in each line.

(Continued)

Before analyzing the book, your program should skip the **header information.** For each line**,** your program should strip all **whitespace and punctuation** and convert **uppercase letters** to lowercase letters for correct analysis. You may employ the **DSU pattern** to develop your program.

Pseudo code

1. Skip the header information of the book.
2. Create a dictionary from the file words.text
3. For each line in the book, do the followings;
   3.1 Split the line into words — Decorate
   3.2 Strip whitespace and punctuation
   3.3 Process each word in the line
   4 **Build a list of tuples with frequency updated**
5. **Sort the list** — Sort
6. **Extract 20 most frequently-used words** — Undecorate

Step 1: Skip the header information

```
fp = open("emma.txt", "r")
for line in fp:
    if line.startswith('*END*THE SMALL PRINT!'):
        break
```

Step 2: Create a dictionary from the file "words.txt"

Select **all words** such that **5** ≤ **len(word)** ≤ **10**.

Create a list of the selected words.

Initialize the **dictionary** as follows:

{word 1: 0, word 2: 0, ... , word n: 0 }

Step 3: For each line in the book, do the followings:

```
import string
for word in line.split():
    word = word.strip(string.punctuation +
                      string.whitespace)
    word = word.lower()
```

step 3.1

Step 3.2

Step 3.3: Process each word in the line:

If the word is in the dictionary,

frequency[word] += 1

Step 4: Build a list of tuples with frequency updated

After scanning the entire book, read the dictionary to create a **list** of **tuples for sorting.**

Steps 5 and 6: Easy