Intelligent Control Systems

Image Processing (3)

—Temporal Operations, Color and Binary Images —

Shingo Kagami

Graduate School of Information Sciences,

Tohoku University

swk(at)ic.is.tohoku.ac.jp

http://www.ic.is.tohoku.ac.jp/ja/swk/

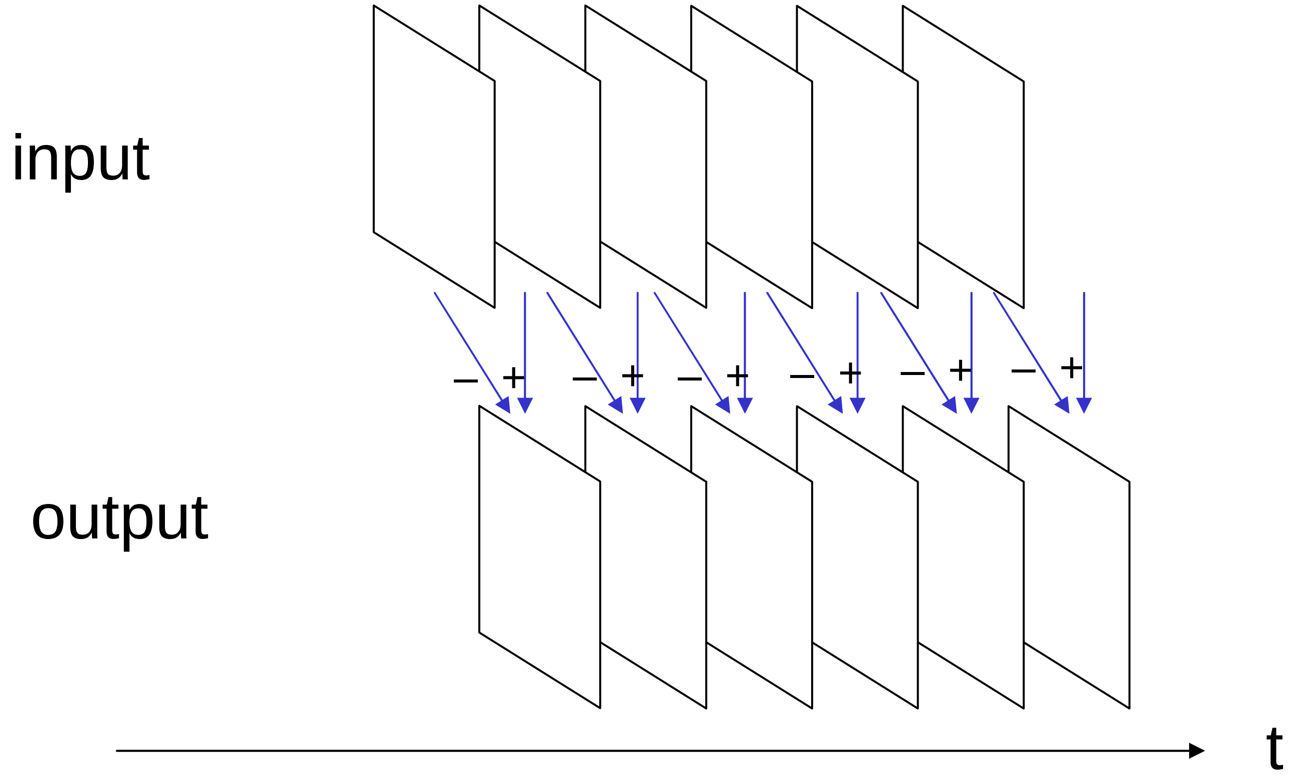
Image Processing Classification

input	output	example
image	image (2-D data)	image to image processing Fourier trans., label image
	1-D data	projection, histogram
	scalar values	position, recognition
image sequence	image sequence	motion image processing
	image	
	1-D data	
	scalar	

Outline

- Temporal Image Processing
- Color Image Processing
- Binary Image Processing

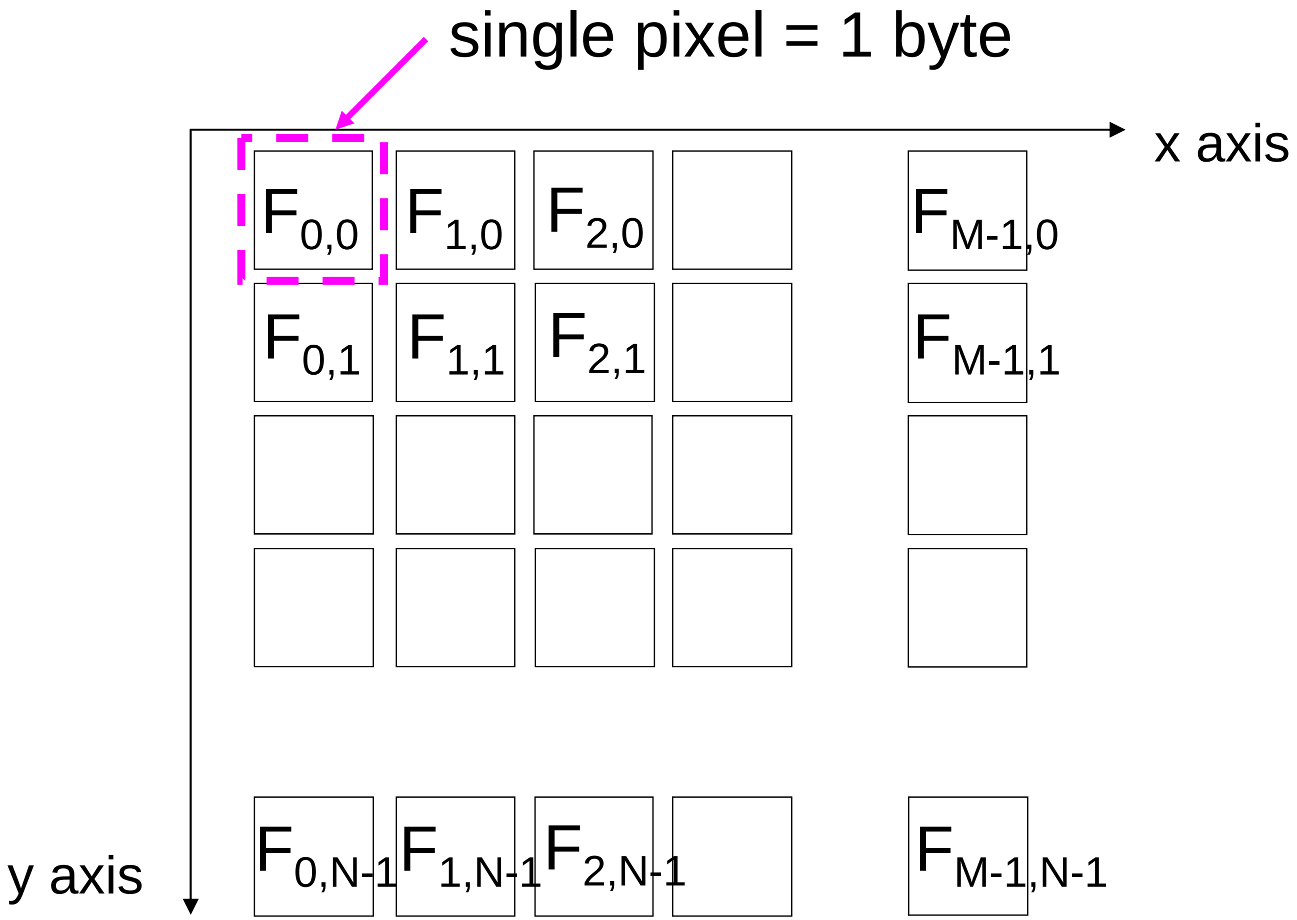
Simple Example: Frame Difference



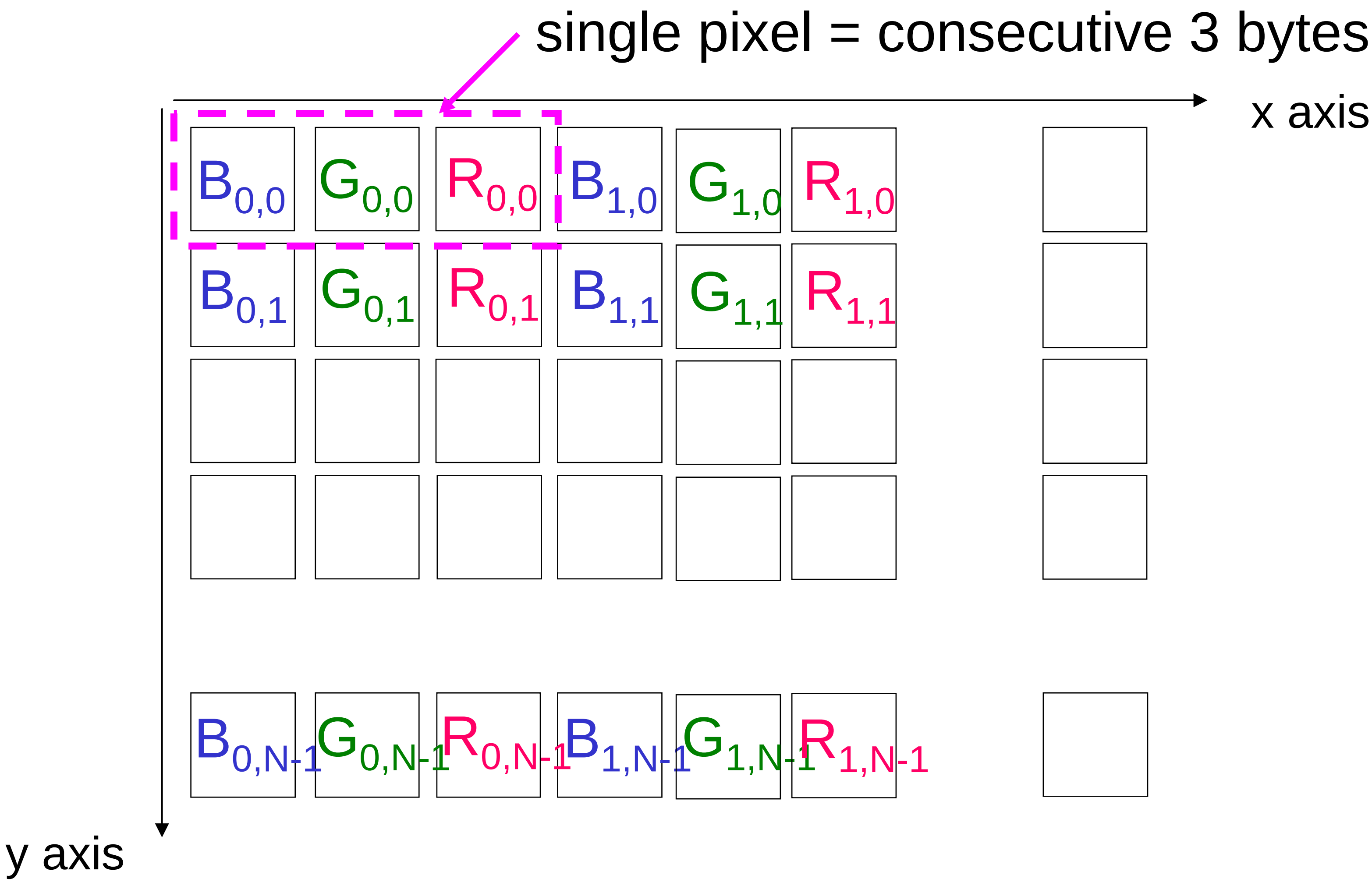
Outline

- Temporal Image Processing
- Color Image Processing
- Binary Image Processing

8-bit Grayscale Image (CV_8U)



24-bit Color Image (CV_8UC3)



Accessing a Color Pixel Value

```
cv::Mat input = cv::imread("lena.jpg", cv::IMREAD_COLOR);
…

for (j = 0; j < height; j++) {
    for (i = 0; i < width; i++) {
        cv::Vec3b pixel = input.at<cv::Vec3b>(j, i);
        uchar blue  = pixel[0];
        uchar green = pixel[1];
        uchar red   = pixel[2];
        …
    }
}
…
```

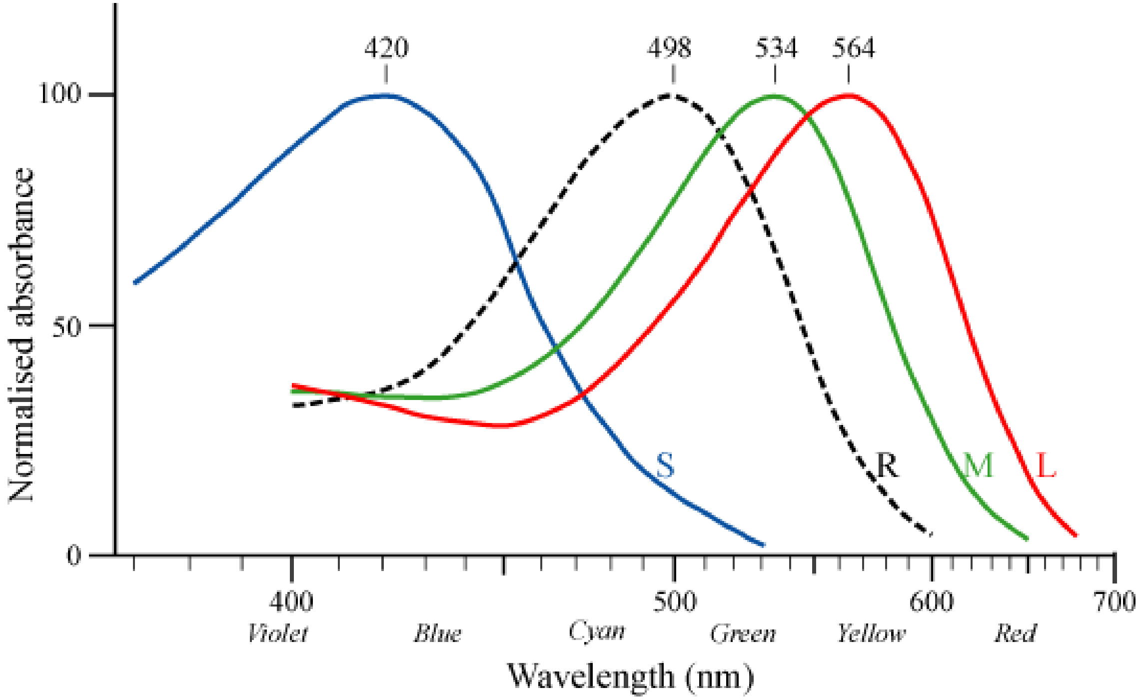
Class of consecutive 3 bytes

operator[] is overloaded so that each component is accessed

RGB Color Space

Why R, G, and B?

- Our eyes have three types of wavelength-sensitive cells (cone cells)
- So, the color space we *perceive* is three-dimensional



http://commons.wikimedia.org/wiki/File:Cone-response.png

Other Color Spaces

XYZ, L*a*b, L*u*v

defined by CIE (Commission Internationale de l'Eclairage)

YIQ, YUV, YCbCr

used in video standards (NTSC, PAL, …)

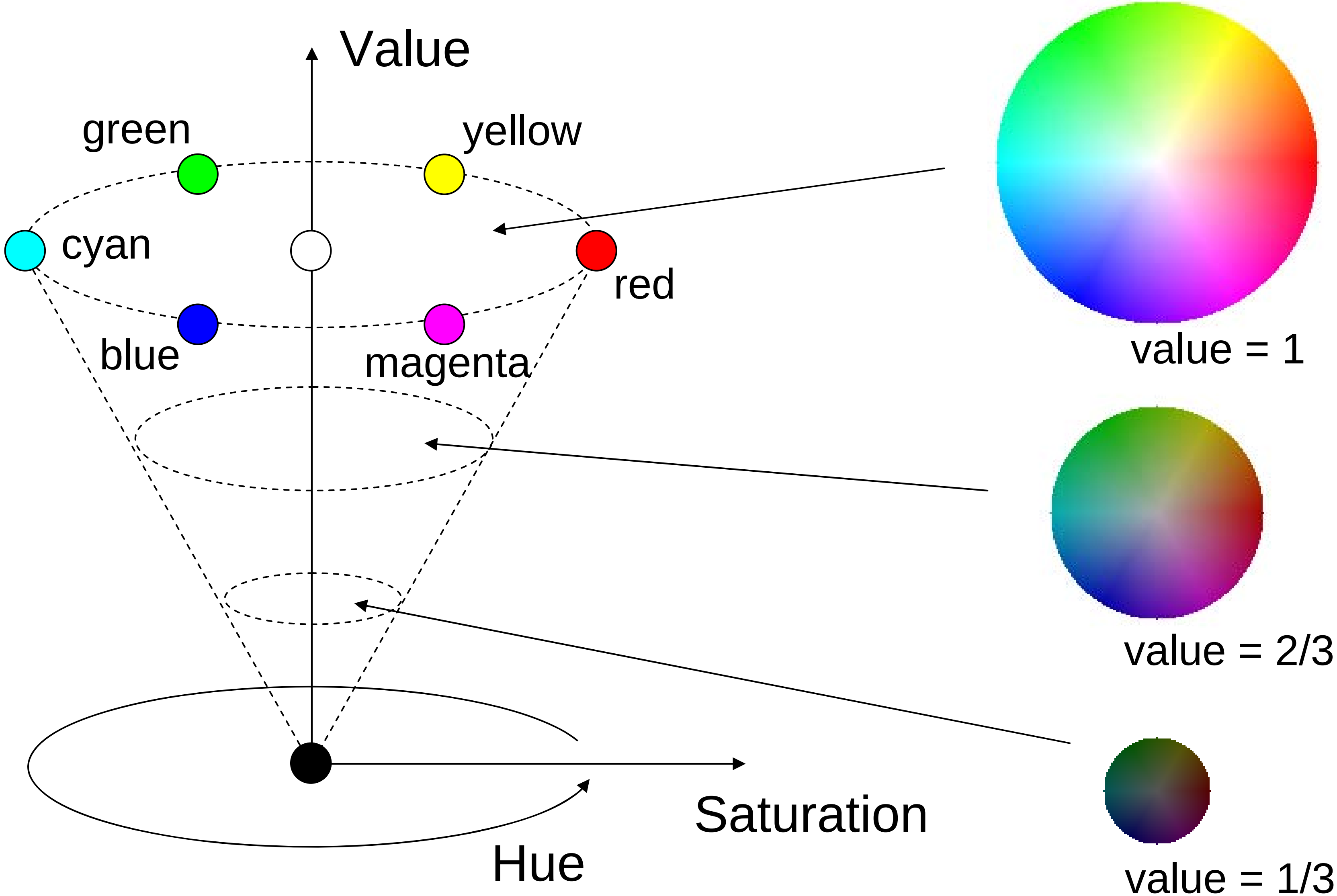
HSV (HSI, HSL)

based on Munsell color system

cf. CMY, CMYK (for printing; subtractive color mixture)

```
cv::cvtColor(input, output, CV_BGR2HSV);
```

HSV Color Space



Notes for cv::Mat data

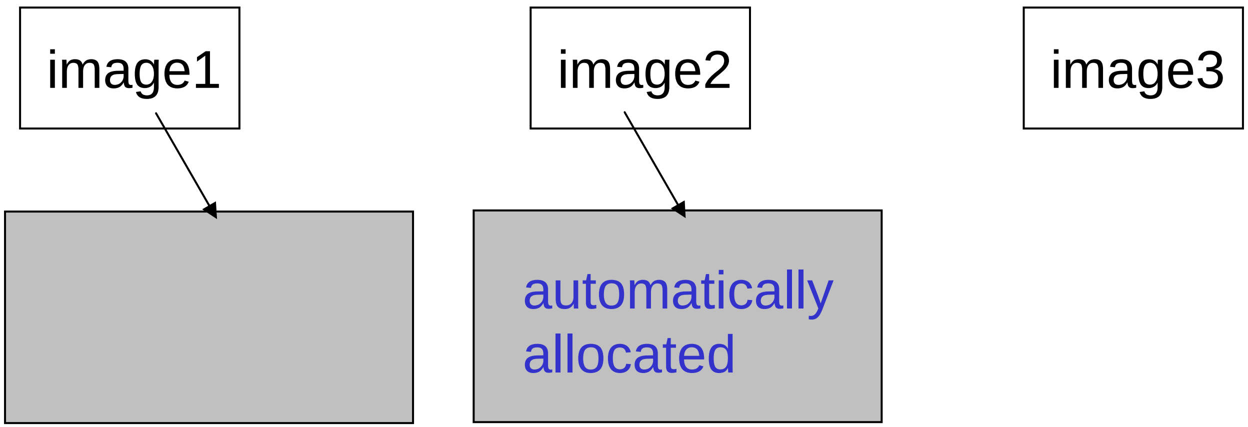
Some parts of the sample codes may seem strange if you are not familiar with OpenCV. It is important to understand:

- In C++, operators can be overloaded and thus even `operator=` can be overloaded
- OpenCV defines `operator=` of `cv::Mat` so that the image data are not copied but shared
- When an image data region becomes shared by no cv::Mat, OpenCV automatically removes the region

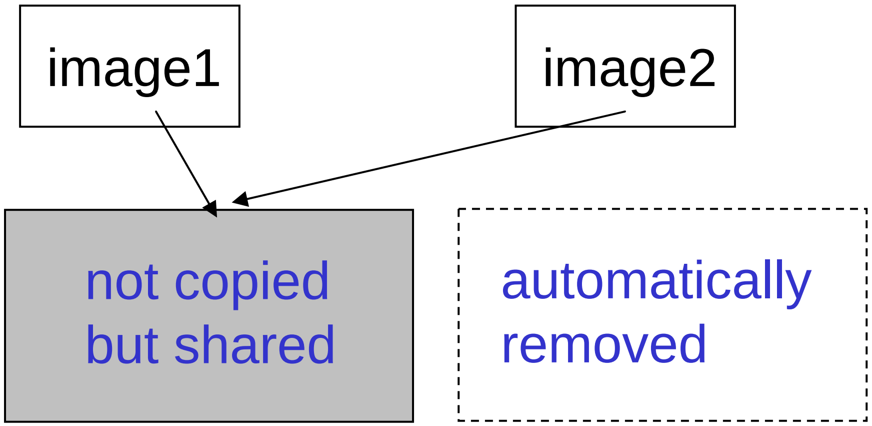
```
cv::Mat image1(480, 640, CV_8U);
cv::Mat image2, image3;

cv::Laplacian(image1, image2, CV_8U);
image2 = image1;
image1.copyTo(image3);
cv::Laplacian(image1, image3, CV_8U);
```

```
cv::Mat image1(480, 640, CV_8U); cv::Mat image2, image3;
```

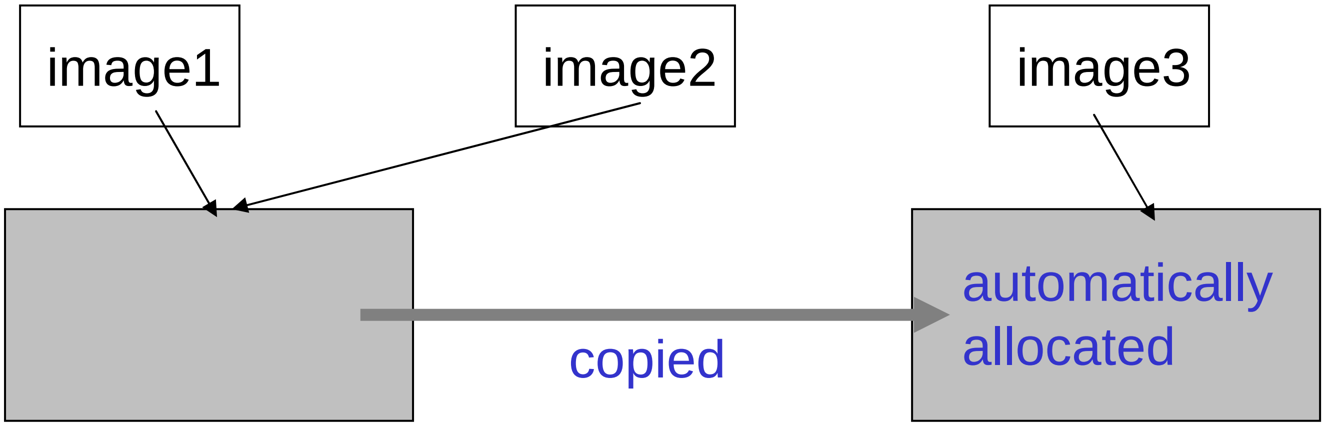

```
cv::Laplacian(image1, image2, CV_8U);
```



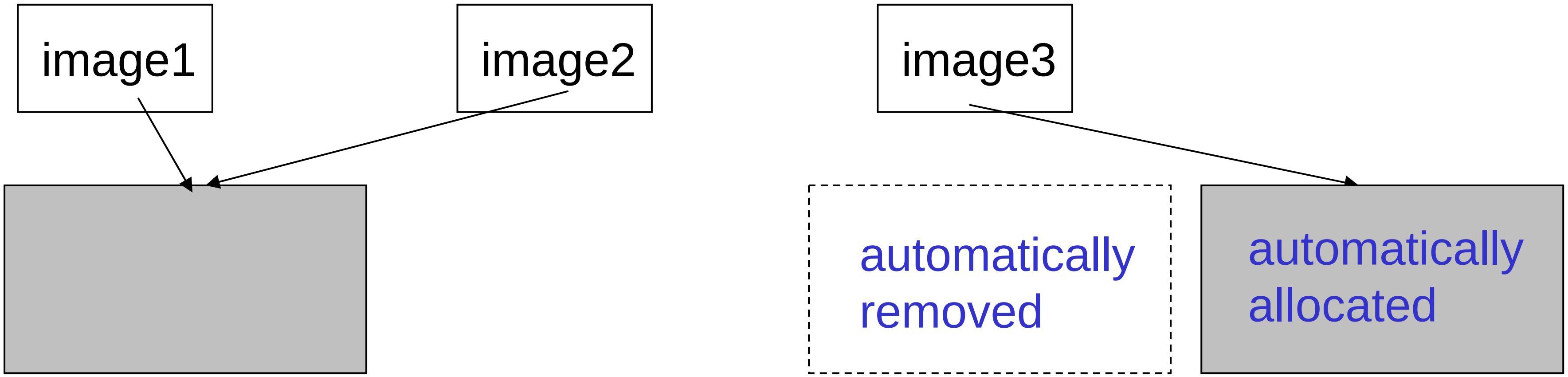
```
image2 = image1;
```




```
image1.copyTo(image3);
```



```
cv::Laplacian(image1, image3, CV_8U);
```



Outline

- Temporal Image Processing
- Color Image Processing
- Binary Image Processing

Binary Image Processing

Processing of binary images is highly developed in a distinctive way other than grayscale/color image processing, because

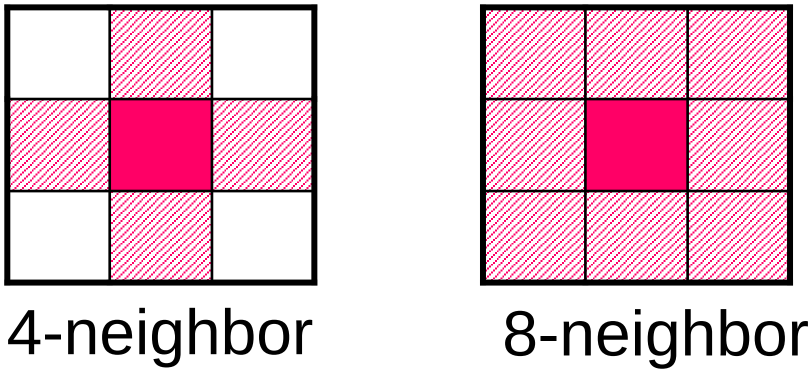
- It found important specific applications (e.g. document processing)
- geometrically rigorous discussion is possible

We have to introduce Digital Geometry:

- Because an image is discretized into pixels, conventional concepts of geometry for continuous shapes (e.g. connectivity, distance) may not be used just as they are.

Connectivity

neighbor: set of pixels that are *near* the pixel of interest.
Many definitions are possible, e.g.



n-adjacent: If two pixels are in n-neighbor of each other, we say they are n-adjacent

n-neighbor connected: If there exists a sequence of pixels p_0, p_1, p_2, $\cdots$, p_{n-1}, p_n where all the p_i have the same pixel value and p_i and p_{i+1} is n-adjacent, we say the pixels in this sequence are n-neighbor connected

Morphological Operations

dilation: “becomes 1 if any of neighbor pixels are 1”

$$G_{i,j} = F_{i,j} \mid F_{i-1,j} \mid F_{i+1,j} \mid F_{i,j-1} \mid F_{i,j+1} \qquad \text{(4-neighbor)}$$

erosion: “becomes 0 if any of neighbor pixels are 0”

$$G_{i,j} = F_{i,j} \,\&\, F_{i-1,j} \,\&\, F_{i+1,j} \,\&\, F_{i,j-1} \,\&\, F_{i,j+1} \qquad \text{(4-neighbor)}$$

opening: erosion, then dilation

closing: dilation, then erosion

Note: These are nonlinear local operatoins

Connected Component Labeling

- or simply “Labeling” for short
- Segments an image into connected components and give each component a unique number (label)
- An image whose pixel values are labels is called a label image

Example Algorithm for 4-neighbor connectivity

- Scan the image from top-left to bottom-right to process “1” pixels
  - If a label has been assigned to either of the upper or left pixels, assign the same label to the current pixel
    - If the upper and the left pixels have different labels, one of them (say, the smaller one) is assigned to the current pixel, and record that both labels refer to the same component to a connectivity table.
  - If none of the upper and the left pixels have labels, a new label is assigned to the current pixel.
- Scan the image again to update the labels referring to the table

								1	1	1				
		2	2											
		2	2	2	2									
		2	2	2	2									
		2	2											
								3	3	3				
									3	3				
						4	4	4	3	3				

3 ⇔ 4

Shape Features

Moment features

$$m_{p,q} = \sum_i \sum_j i^p j^q F_{i,j}$$

$$m_{0,0} = \sum_i \sum_j F_{i,j}$$

0th order moment: i.e. area (for binary images)

$$m_{1,0} = \sum_i \sum_j i F_{i,j}$$

1st order moment in x direction

$$m_{0,1} = \sum_i \sum_j j F_{i,j}$$

1st order moment in y direction

Center of Gravity is computed from 0th and 1st order moments

$(g_x, g_y) = (m_{1,0}/m_{0,0}, m_{0,1}/m_{0,0})$

Higher order moments convey more complicated shape information

Notes for std::vector

- std::vector is part of C++ Standard Template Library (STL)
  - can be used as a better array
- Many other convenient containers are also available:
  - list, map, queue, stack, deque, …

```
std::vector<int> array;
array.push_back(100);
array.push_back(120);                extends automatically
array.push_back(130);

array[2] = array[2] + 50;            can be used almost like an array

printf("array[2] = %d¥n", array[2]);             prints 180
printf("size of array: %d¥n", array.size());     prints 3
```

What’s the common way in OpenCV?

- So far, connected component labeling seems not provided as standard functions of OpenCV

- Contour retrieval using a border following algorithm can be used for similar purpose. See:
  - `cv::findContours()`
  - You can analyze the shape of a retrieved contour by e.g.
    - `cv::arcLength()`
    - `cv::boundingRect()`
    - `cv::contourArea()`
    - …

Summary

- Simple example of video processing
  - frame difference

- Color Image Processing
  - cv::Mat for color images
  - color spaces
    - RGB, HSV, …

- Binary Image Processing
  - connectivity
  - morphological operations
    - dilation, erosion, opening, closing
  - connected component labeling
  - moment features

References

- R. Szeliski: Computer Vision: Algorithms and Applications, Springer, 2010.
- A. Hornberg eds.: Handbook of Machine Vision, Wiley-VCH, 2006.
- G. Bradski and A. Kaebler: Learning OpenCV, O‘Reilly, 2008.
- OpenCV Documentation: http://docs.opencv.org/index.html

(in Japanese)

- ディジタル画像処理編集委員会, ディジタル画像処理, CG-ARTS協会, 2004.
- 田村: コンピュータ画像処理, オーム社, 2002.

Sample codes are in sample20140624.zip available at
http://www.ic.is.tohoku.ac.jp/~swk/lecture/