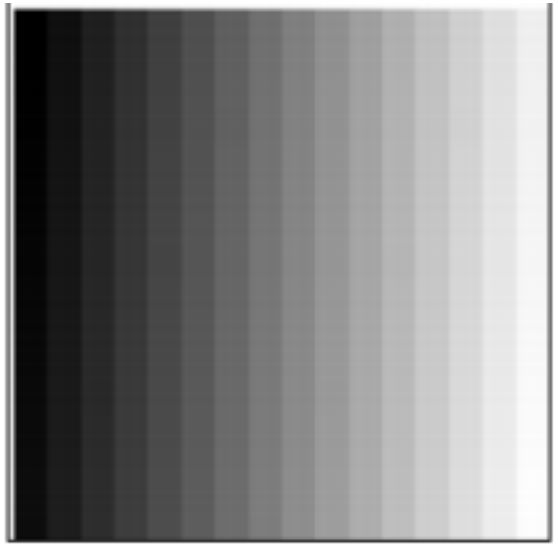
Lecturer: Bui Ha Duc, PhD

Email:

ducbh@hcmute.edu.vn

- Image Representation
  - Types of image
  - Image Features
- Introduction to OpenCV
  - Load Image
  - Access pixel value
  - Set pixel value
- Digital image is presented by **pixel matrix**
- Image processing operation in a computer may be observed as a **matrix operation**

x



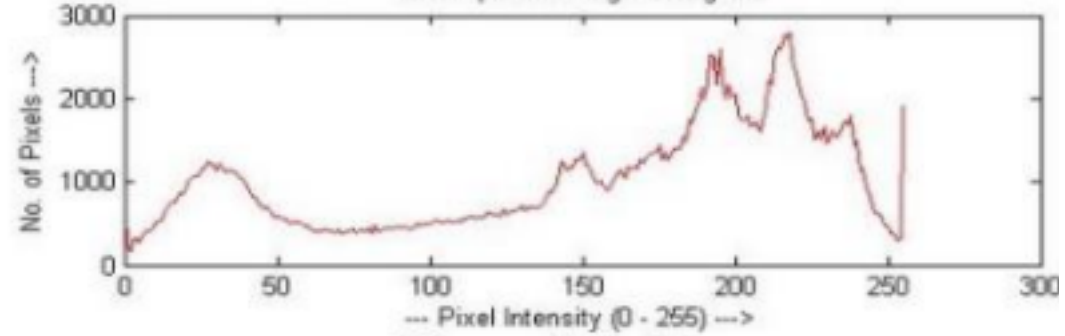
0	16	32	48	64
1	17	33	49	65
2	18	34	50	66
3	19	35	51	67
4	20	36	52	68
5	21	37	53	69
6	22	38	54	70
7	23	39	55	71
8	24	40	56	72
9	25	41	57	73
10	26	42	58	74
11	27	43	59	75
12	28	44	60	76
13	29	45	61	77
14	30	46	62	78
15	31	47	63	79

(R,G,B)=(146,185,216)

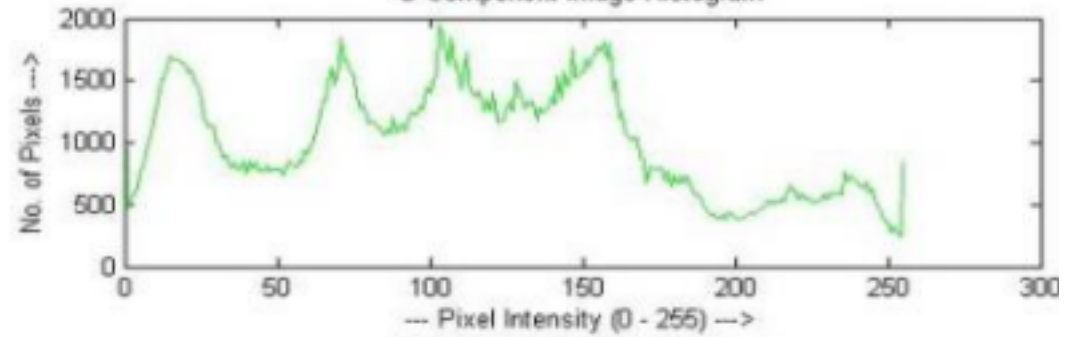
Image(3,4) = ?

- Represented by **3 matrices**
- Colors are seen as variable combination of primary colors Red (R), Green (G), and blue (B)
- Each element are integer number range from **0 to 255**

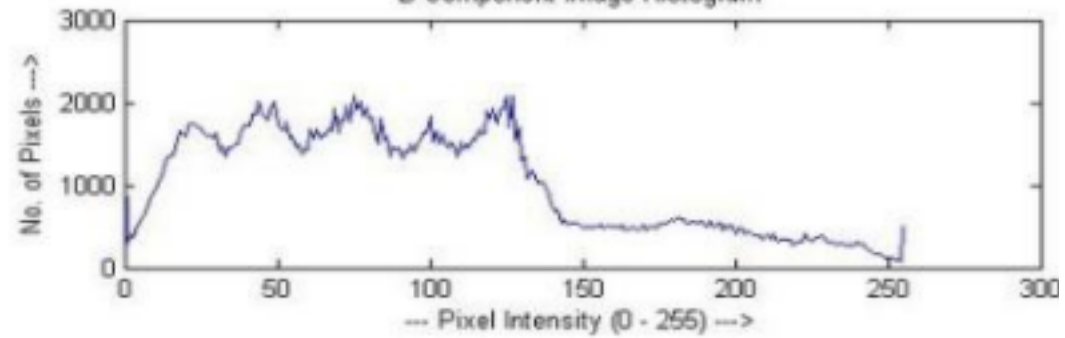
- Intensity of the pixel with respect to the color
- In RGB system, it’s possible to create 256^3=16777216 different colors



G-Component Image



B-Component Image



Color Image

RGB components

R/G/B histogram

Color image processing is generally challenging

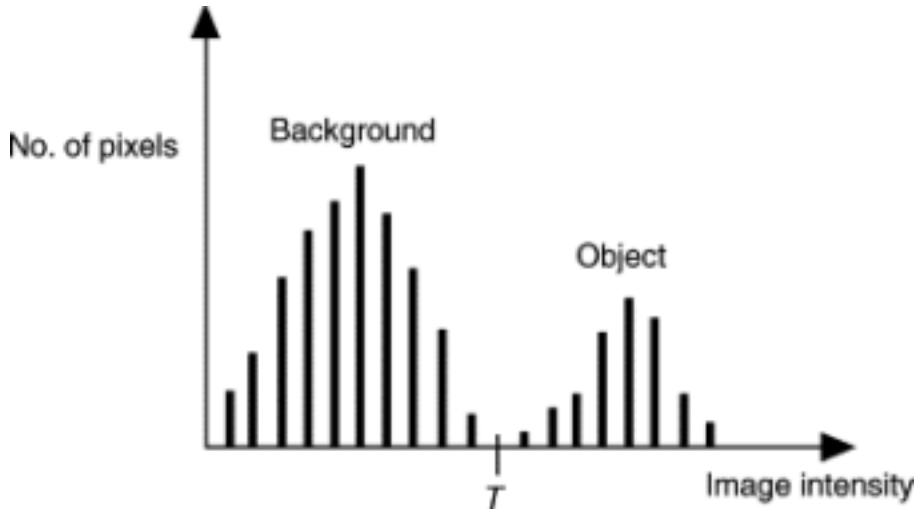
- Three separate, "independent" channels
- Time consuming
- Require nonlinear approaches

- Represented by **1 matrix**
  - Each element of the matrix is the intensity of the corresponding pixel
  - Range from **0 (black) to 255 (white)**
- Covert from RGB image:

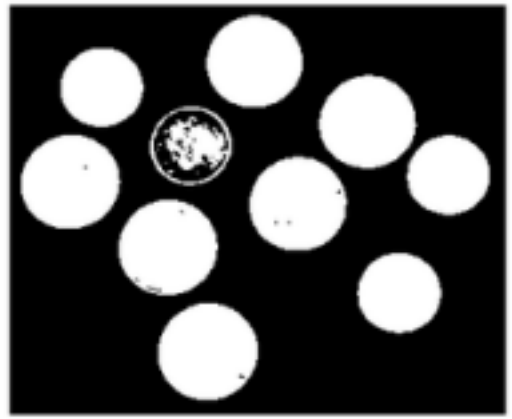
$$\text{??} = \text{??}.\text{????????} + \text{??}.\text{????????} + \text{??}.\text{????????}$$

intensity=177

- Also called Boolean images
- Represented by a matrix
- All elements are **0** and **1**
- 0 is black
  - 1 is white
- Result of thresholding operations
- Important in segmentation



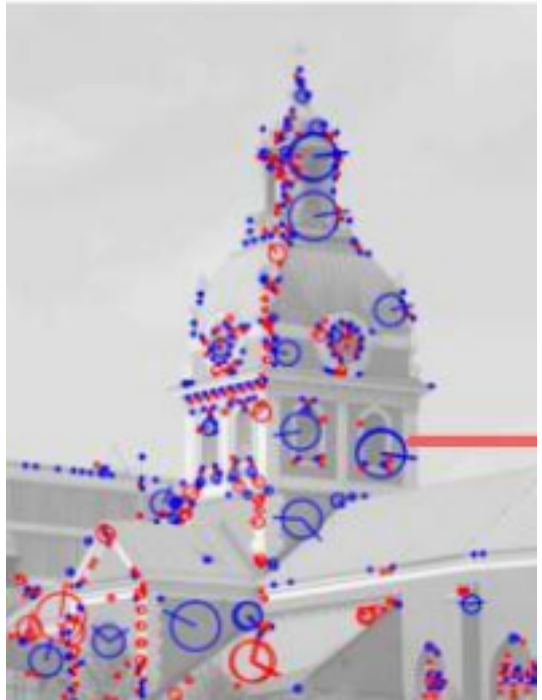
Input Image



threshold output

Basic features

- Histogram
- Color
- Edge
- Corner



Edge

- Centroid, size, shape
- Lines
- Equations
- Start, end points,
- orientation
- Key points
- Location
- Direction
- scale

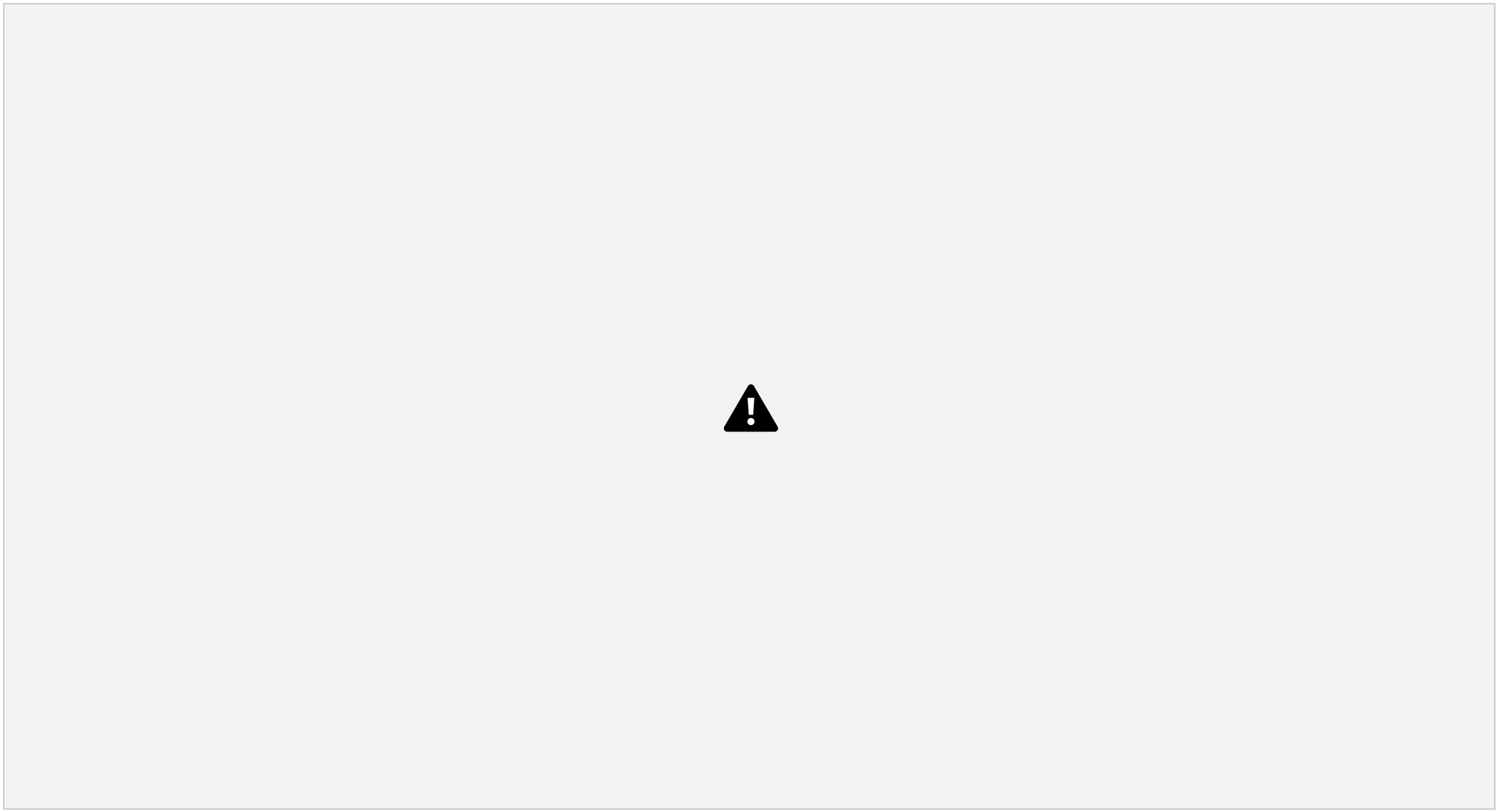
Key points

Advanced features - Regions

- **OpenCV: Open Source Computer Vision & Machine Learning software library**
  - Created in 1999 by Intel
  - Supported from 2008 by **Willow Garage**
  - Willow Garage also supported the Robotic Operating system (ROS) and Point Cloud Library (PCL)
- OpenCV is a cross-plaform
  - Available in Windows, Linux, Android, MacOS …
- OpenCV support a wide range of programming languages:

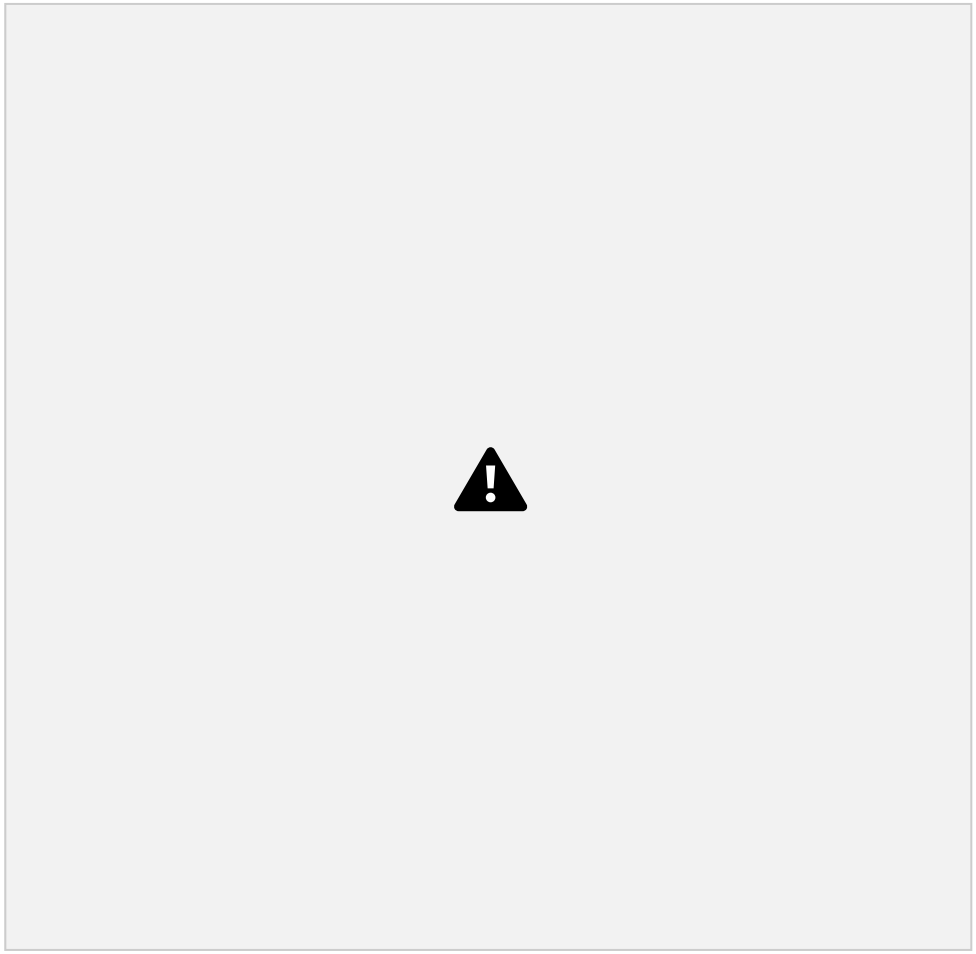
- OpenCV is written in C/C++
  - Stable source code: opencv.org/release
  - Developments
    - Source code github.com/opencv
    - Online documentation docs.opencv.org
- Current stable version is 4.4.0
  - Written for C++ 11
  - Better video object detection and tracking
  - More support for Deep learning

- **core** ▶ Base data structures & core routines
- **imgproc** ▶ Image processing routines
  - Linear / nonlinear image filtering
  - Geometric image transforms
  - Shape descriptors
  - Basic image operators
  - Histograms
  - Basic feature detection
- **video** ▶ Video analysis routines
  - Motion estimation
  - Motion segmentation
  - Background subtraction
  - Object tracking

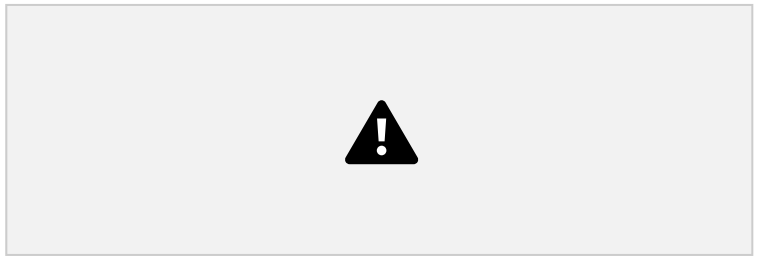


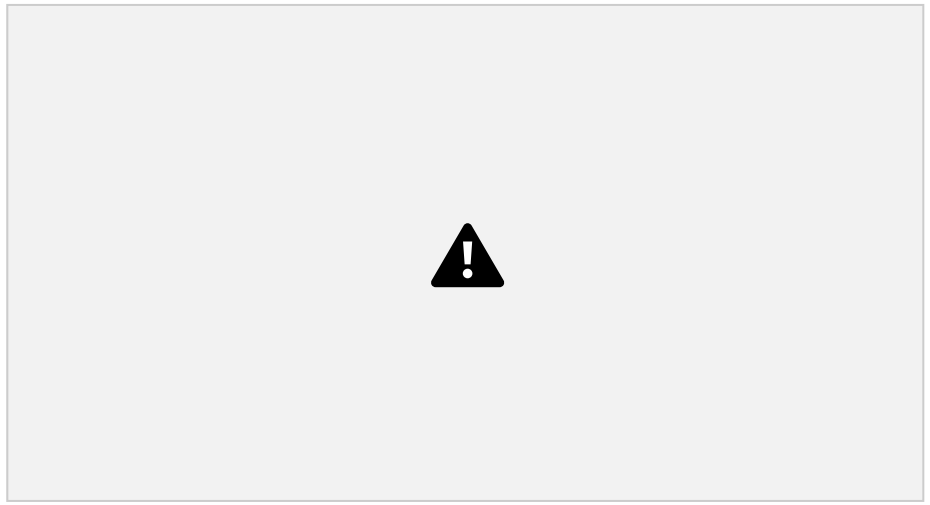
Include headers of openCV modules

All OpenCV classes and functions are in the cv namespace



Without using name space **cv**





Multi-channel array class for image data

BGR instead of RGB

- Mat is a dense multi-channel array class used to store image data
- Mat class is the epic center of the OpenCV library
- Majority of functions in OpenCV are
  - member of Mat class,
  - take Mat as an argument
  - or return Mat as a return value
- Mat contains 2 data part:
  - Header: containing information such as the size of the matrix, the method used for storing, at which address is the matrix stored

- Pointer: containing the pixel values

- Create a Mat constructor

```
Mat img(int rows, int cols, int type)
```

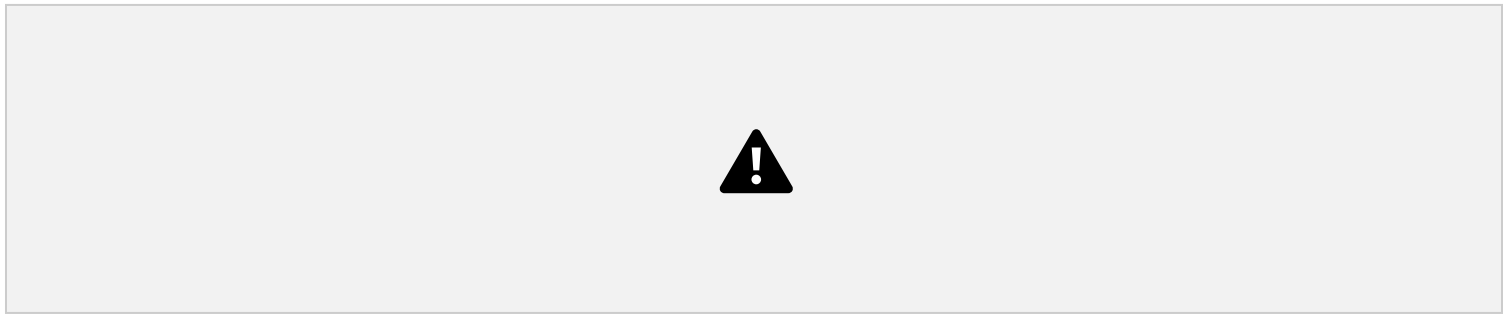
Example

```
Mat img(3, 10,
CV_32FC3)
```

- Mat data type:

```
CV_<bit_depth><data_type>C<nb_channels>
```

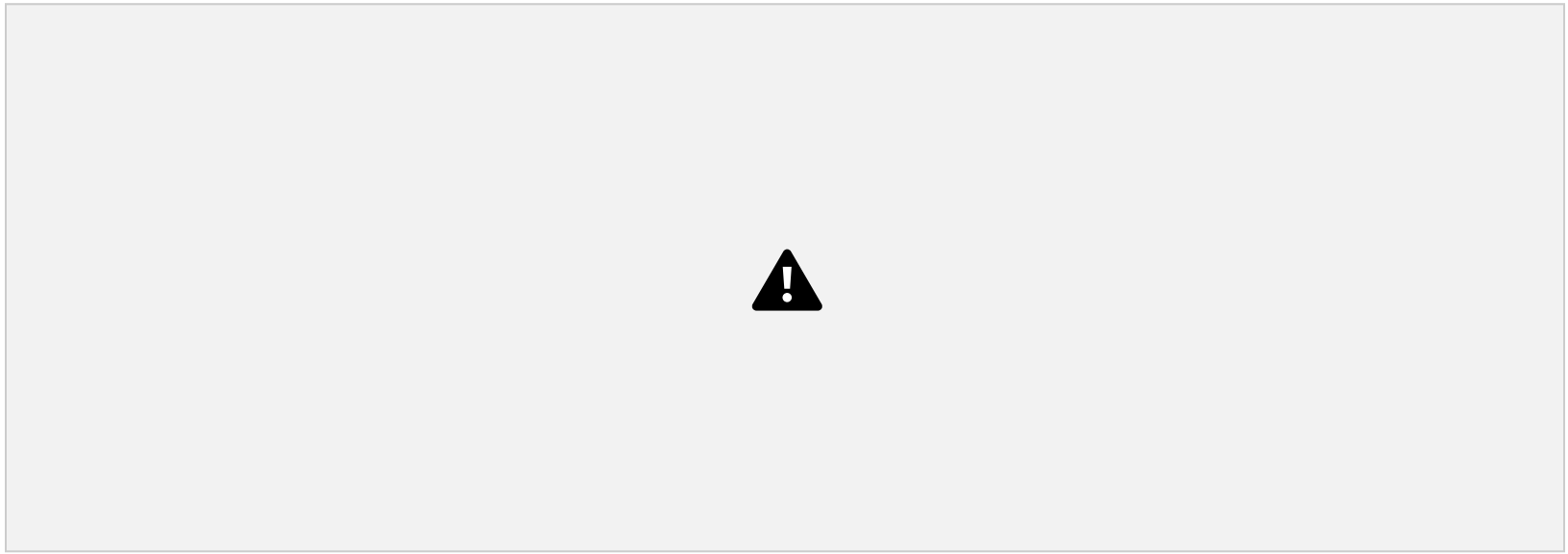
img is a 2-D array with 3 rows, 10 columns, 3 channels, data type is Float 32 bit



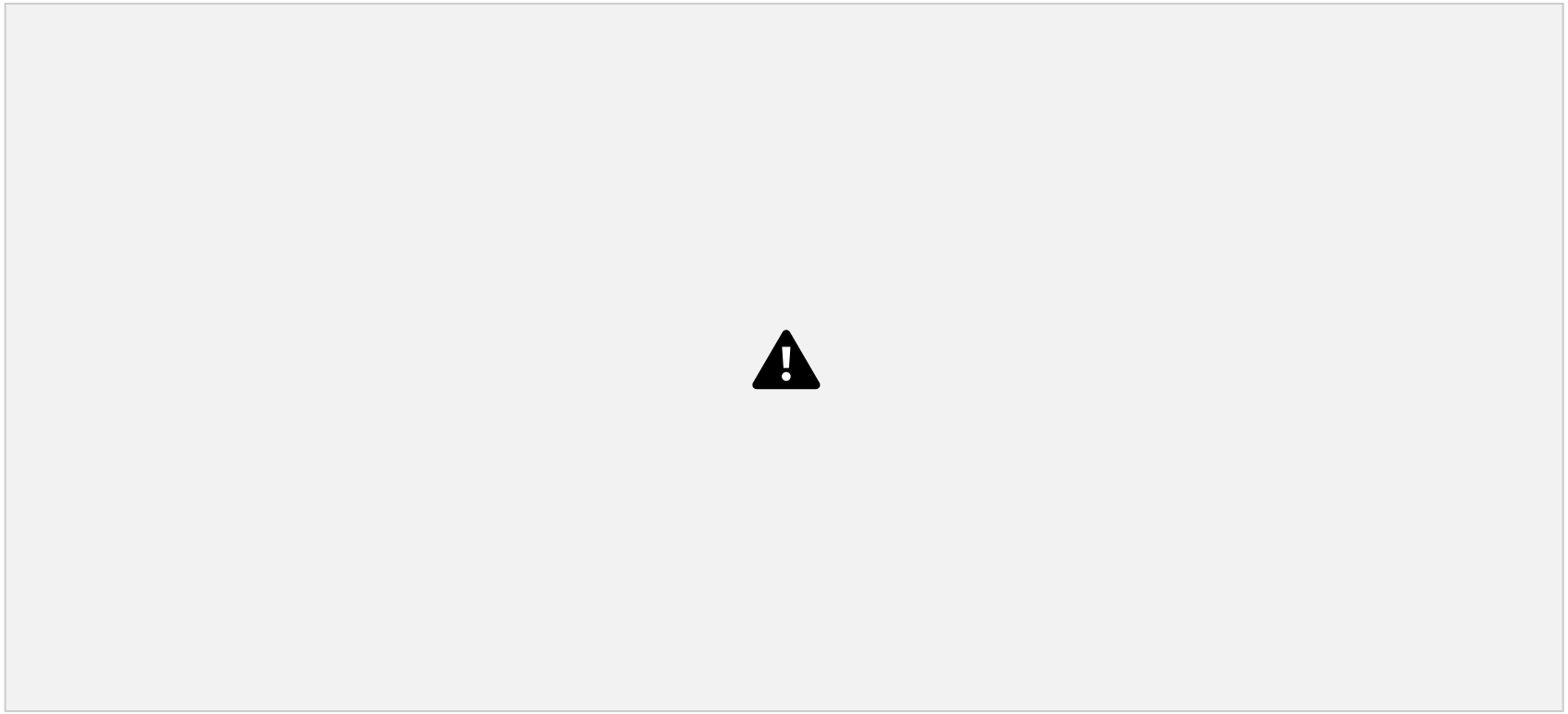
- Examples
  - `Mat M1 (2, 3, CV_8UC1)`

    (2x3) single channel array with 8-bit unsigned char data
  - **Mat A (10, 10, CV_16SC3)**

    (10x10) 3-channel array with 16-bit signed short integer data
- Point and Point3 are classes used to store Cartesian coordinates of 2D/3D points



- Class for short numerical vector specified by its Cartesian coordinates



Most common Mat Types:

- **CV_8UC1** : 8-bit, 1-channel for gray scale images
- **CV_32FC1** : for 32-bit 1 channel gray images
- **CV_8UC3** : 8-bit 3 channel for color images

- **CV_32FC3 :** for 32-bit 1 channel gray images

The default setting with imread function is CV_8UC3

Access individual pixels value Mat::at<type>(Point2i(x,y))

- Use **Mat::at<type>(r,c)** method
- E.g. For CV_8UC1 Mat: uchar a = img.at<uchar>(r,c) For CV_8UC3 Mat: Vec3b a = img.at<Vec3b>(r,c)

method

Notes:
r: row → y C: column → x

- Use

- Use **Mat::ptr<type>(int r)** to get pointer of the beginning of a row → **ptr[c]** to get pixel value at row r, column c

Access individual pixels value

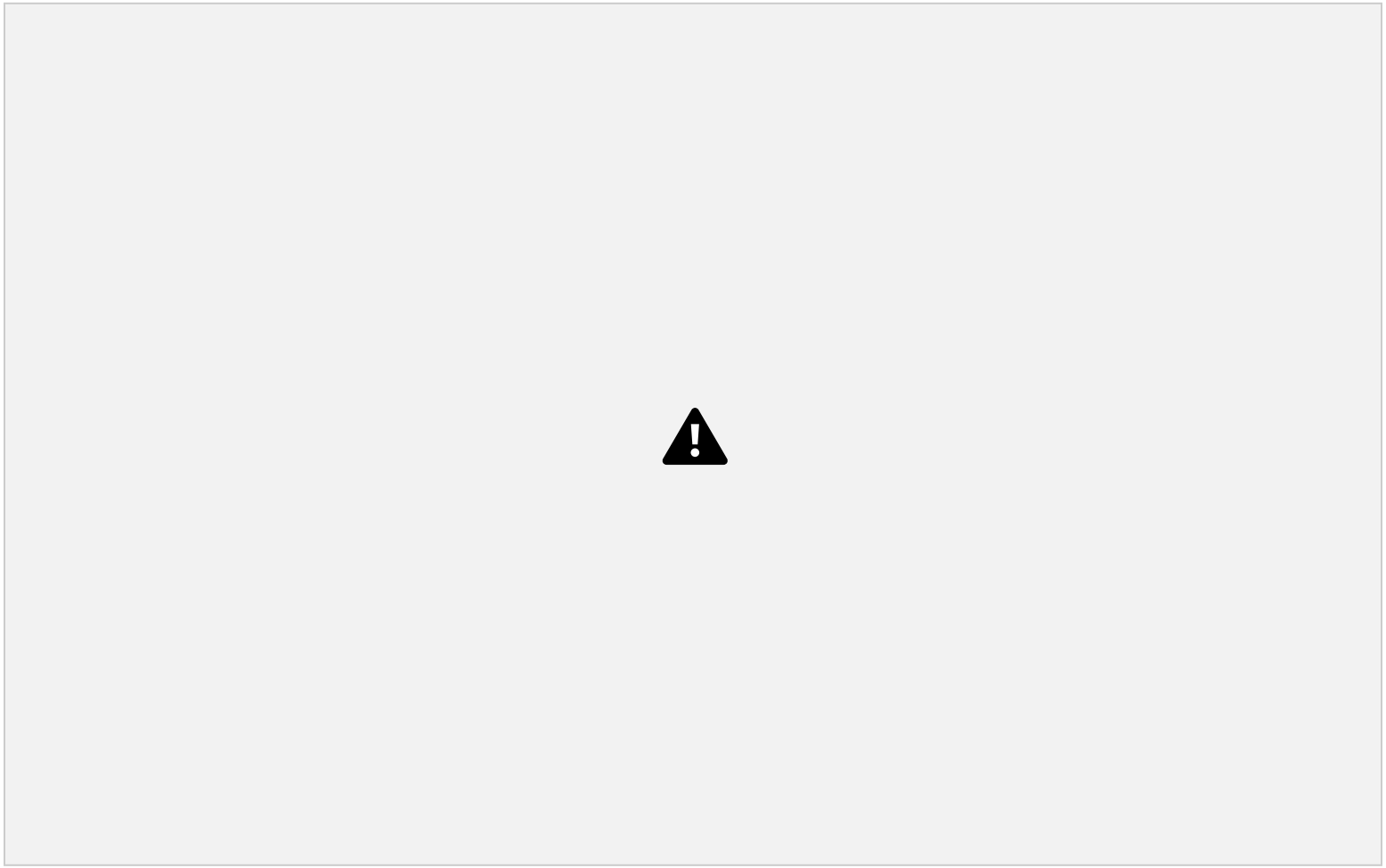
- Use **Mat::ptr<type>(int r)** to get pointer of the beginning of a row → **ptr[c]** to get pixel value at row r, column c

e.g.

```
// Iterate over all pixels of the image
for(int r = 0; r < img.rows; r++) {
   // Obtain a pointer to the beginning of row r
   Vec3b* ptr = img.ptr<Vec3b>(r);
```

```
            for(int c = 0; c < img.cols; c++) {
            // Invert the blue and red values of the pixel
             ptr[c] = Vec3b(ptr[c][2], ptr[c][1], ptr[c][0]);
            }
        }
```

Access block pixels value



Set value for a pixel

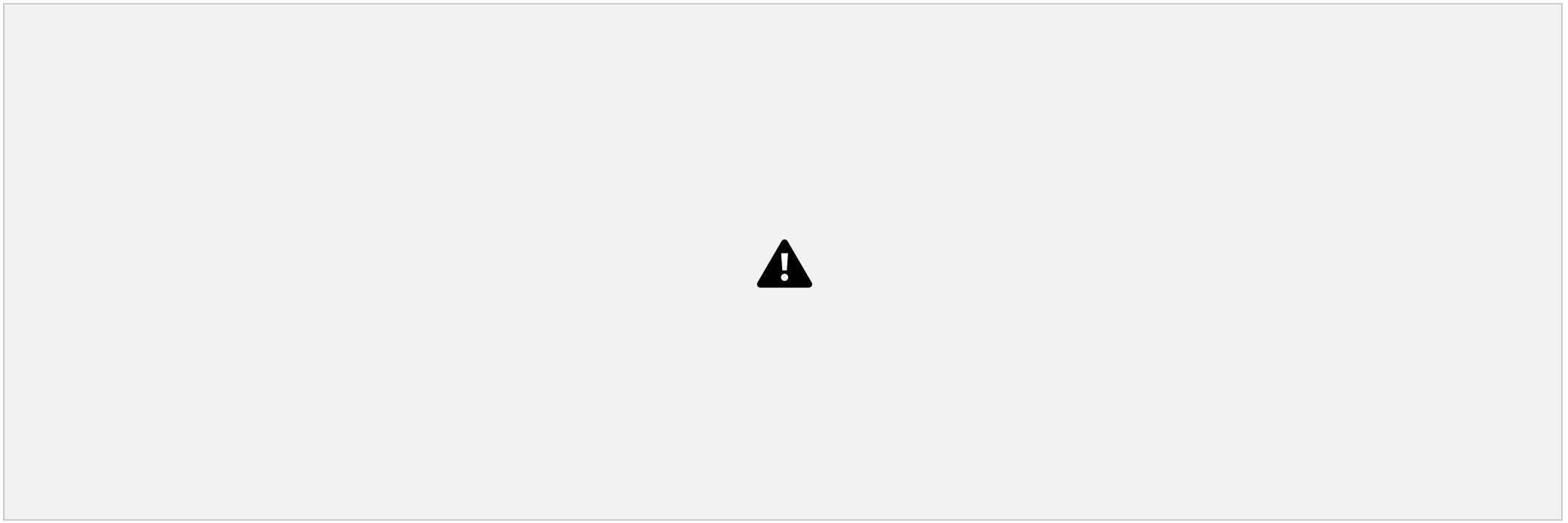
- Use **Mat::at<type>(r,c) = value**
  - E.g. For CV_8UC1 Mat: img.at<uchar>(r,c) = uchar a

For CV_8UC3 Mat: img.at<Vec3b>(r,c) = Vec3b b

```
for (int x = 0; x<img.rows; x++) {
      for (int y = 0; y<img.cols; y++) {
          // Accesssing values of each pixel
          if (img.at<uchar>(x, y) >= threshhold) {
              output.at<uchar>(x, y) = 254; // Setting the pixel values to 255 if it's above the
          threshold }
          else if (img.at<uchar>(x, y) < threshhold) {
              output.at<uchar>(x, y) = 0; // Setting the pixel values to 255 if it's below the
          threshold }
      }
}
```

Set value for a block of pixels

- Use **cv::addWeighted()** → page 104 “Learning OpenCV 3”



Set value for a block of pixels

```
Mat img1 = imread(argv[1],1);
Mat img2 = imread(argv[2],1);

Mat roi1(img1,Rect(50,50,100,100));
Mat roi2(img2,Rect(0,0,100,100));
```

```
addWeighted(roi1, 0.0, roi2, 1.0, 0.0, roi1);
```