Image Processing

OpenCV

Prepared by: NOU Sotheany

Using OpenCV library

- Everything in OpenCV library is defined in a namespace called *cv*.
- To access functions or classes you may use the *cv::* specifier
- Or *using namespace cv;* directive
- Include statements *“opencv2/module_name/module_name.hpp”*
  - *#include “opencv2/core/core.hpp”*
  - *#include “opencv2/highgui/highgui.hpp”*

Class Mat

- The data structure used to store images in OpenCV is called *Mat*.
  - *Mat* can be used to store several types of images:
    - Grayscale
    - True-color (BGR)
    - Binary image
  - Defined in the core module of OpenCV
  - *#include "opencv2/core/core.hpp"*
  - *cv::Mat img;*

imread()

- *imread()* is used to load image from a file to *Mat*.
- *Mat imread(const string& filename, int flags=1)*
- Parameters:
  - *filename* – Name of an image file to be loaded
  - *flags* – Flags specifying the color type of a loaded image
  - *CV_LOAD_IMAGE_COLOR* - If set, always convert image to the color one
  - *CV_LOAD_IMAGE_GRAYSCALE* - If set, always convert image to the
- Defined in the *highgui* module

imwrite()

- *imwrite()* is used to save image stored in *Mat* to a file.
  - *bool imwrite(const string& filename, Mat& img)*
- Parameters:
  - *filename* – Name of the image file
  - *img* – Image to be saved
- Defined in the *highgui* module

imshow()

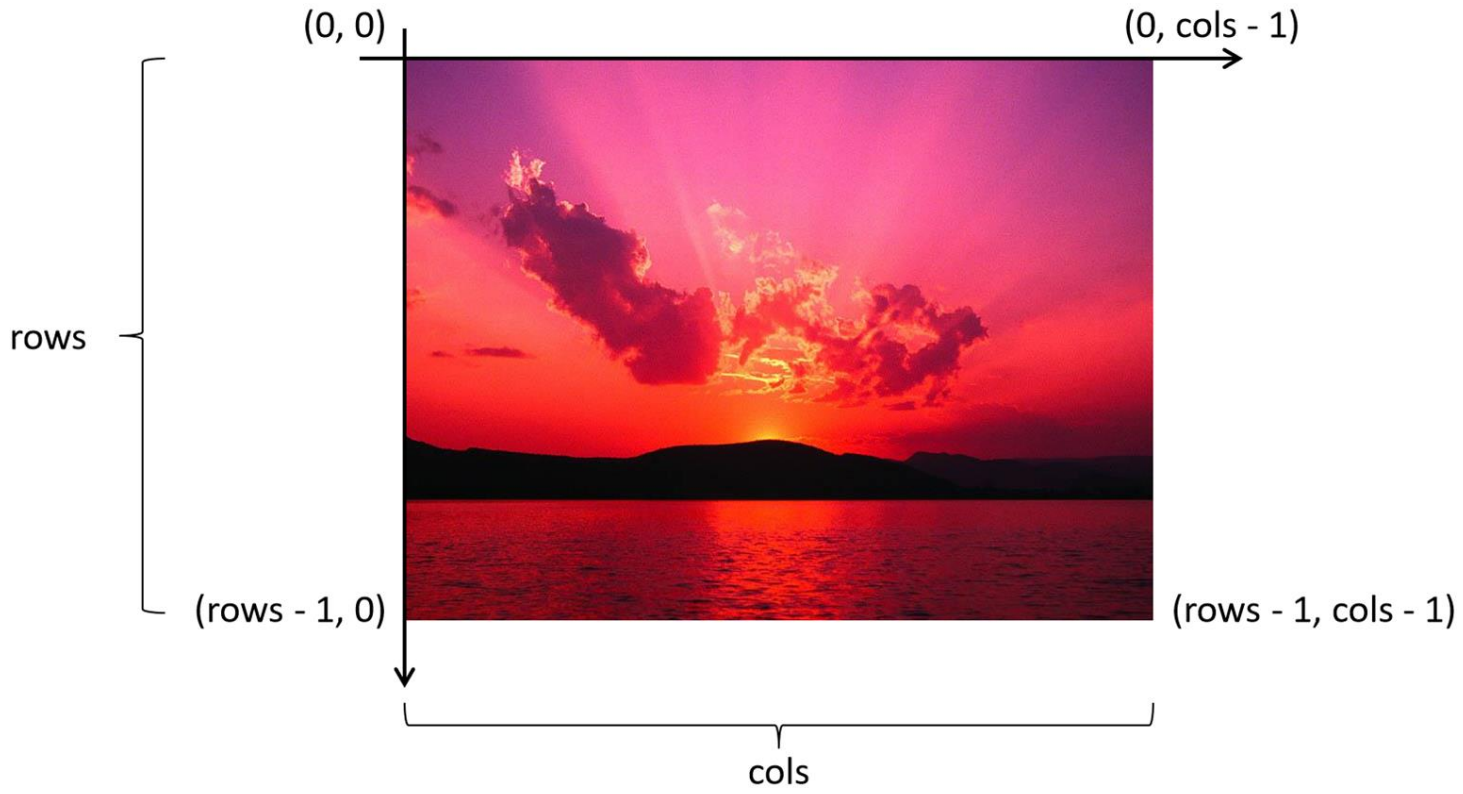
- *imshow()* is used to display an image in a specific window.
  - *void imshow(const string& winname, const Mat& img)*
- Parameters:
  - *winname* – Name of the window used to display the image
  - *img* – Image to be displayed
- Defined in the *highgui* module

waitKey()

- *waitKey()* makes the program wait for a pressed key.
  - *int waitKey(int delay=0)*
- Parameters:
- *delay* – Delay time in milliseconds. The values of 0 or negative mean “forever” (terminate when a key is pressed).
- Defined in the *highgui* module

Class Mat: rows and cols

- *Mat* class has two attributes *rows* and *cols* that define the size (spatial resolution) of image



Accessing image's pixels

- Grayscale image

  cv::Mat img;

  img.at<uchar>(i,j);

- Color image
  - OpenCV uses BGR format for color images
    - Channel 0 – blue
    - Channel 1 – green
    - Channel 2 – red

Accessing image's pixels

- The type of color pixel is *cv::Vec3b*
  - A vector containing three elements

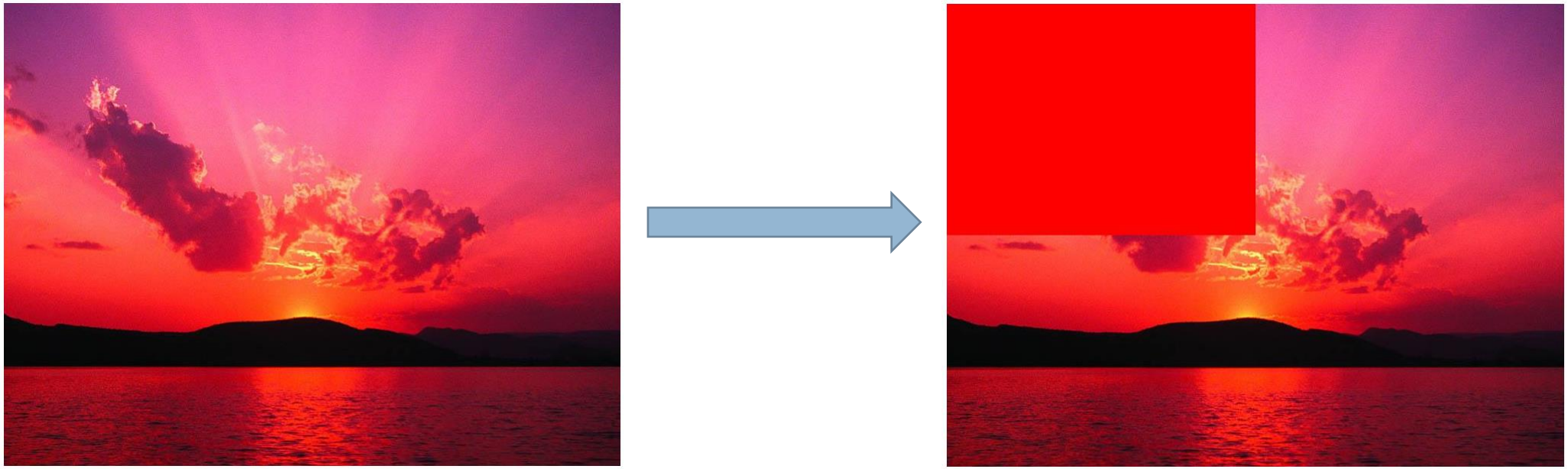
    cv::Mat img;

    img.at<cv::Vec3b>(i,j)[0]

    img.at<cv::Vec3b>(i,j)[1]

    img.at<cv::Vec3b>(i,j)[2]

TP

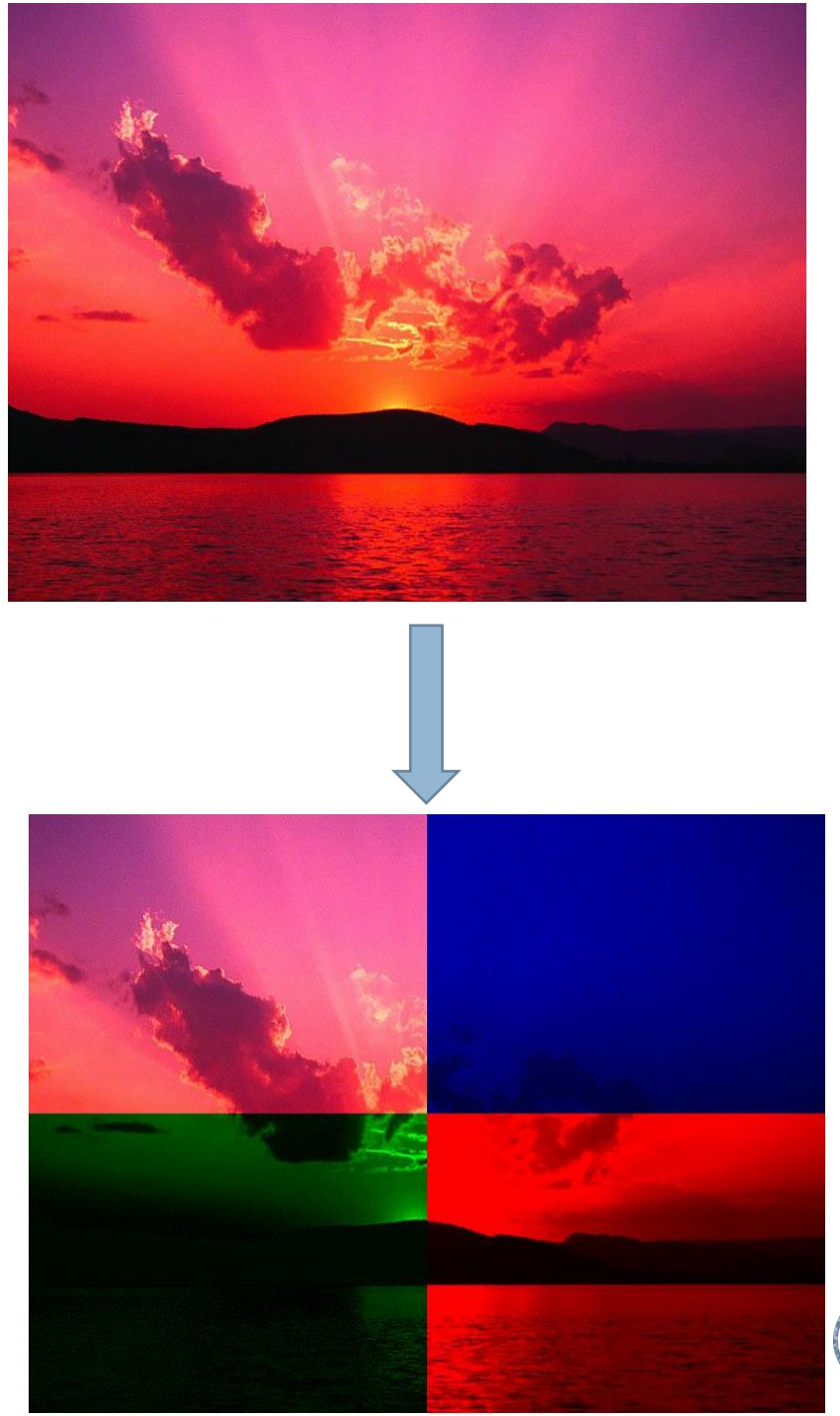


TP

- Write a program to convert the upper image

to the lower image

- The image is divided into four parts:
  - Top-left: the original color
  - Top-right: no red and green components
  - Bottom-left: no red and blue components
  - Bottom-right: no green and blue components



TP

- Write your own function that converts an BGR image into a grayscale image
- Using the following formula:

$$I = 0.299 \times R + 0.587 \times G + 0.114 \times B$$