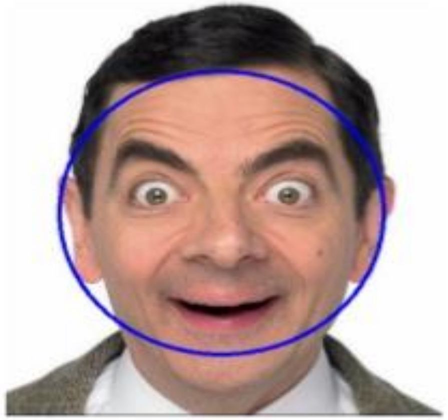
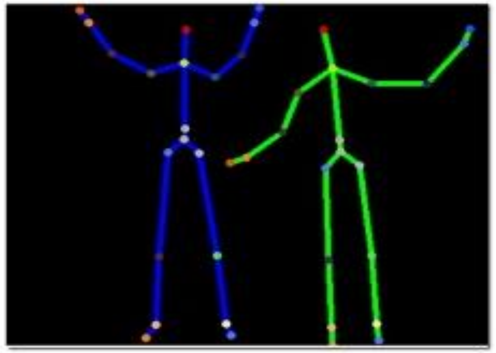
OpenCV Workshop





Session 2: playing with images

Elhan Shabaninia

Reading images from files

- OpenCV supports reading images in different formats, such as PNG, JPEG, and TIFF.

imread(path, flag)

- The image should be in the working directory or a full path of image should be given.
- Second argument is a flag which specifies the way image should be read (color image, grayscale, …).

```
img = cv2.imread(‘path', cv2.IMREAD_COLOR)
```

python

```
Mat image;
image = imread(“path”, CV_LOAD_IMAGE_COLOR);
```

C++

- Instead of these three flags, you can simply pass integers 1, 0 or -1.
- Even if the image path is wrong, it won’t throw any error.

Display an image

imshow(name, image)

- First argument is a window name which is a string.
- second argument is our image. You can create as many windows as you wish, but with different window names.

waitKey(time)

- Its argument is the time in milliseconds.
- The function waits for specified milliseconds for any keyboard event. If you press any key in that time, the program continues. If 0 is passed, it waits indefinitely for a key stroke.

Display an image

destroyAllWindows()

- destroyAllWindows() simply destroys all the windows we created. If you want to destroy any specific window, use the function destroyWindow() where you pass the exact window name as the argument.

namedWindow(name , flag)

- There is a special case where you can already create a window and load image to it later. In that case, It is done with the function namedWindow().

```
cv2.namedWindow('image', cv2.WINDOW_NORMAL)
 cv2.imshow('image',img)
 cv2.waitKey(0)
 cv2.destroyAllWindows()
```

python

```
namedWindow( "Display window", WINDOW_AUTOSIZE );
 imshow( "Display window", image );
  waitKey(0);
```

C++

Write an image

imwrite()

- First argument is the file name,
- second argument is the image you want to save.

```
cv2.imwrite('messigray.png',img)
```

python

```
imwrite("alpha.png", mat )
```

C++

Let's write a program

loads an image in grayscale, displays it, save the image if you press ‘s’ and exit, or simply exit without saving if you press *ESC* key.

Capturing and reading frames from a camera

VideoCapture

- Class for video capturing from video files or cameras.

cv2.VideoCapture(filename) → <VideoCapture object>
cv2.VideoCapture(device) → <VideoCapture object>

python

VideoCapture::VideoCapture(const string& **filename**)
VideoCapture::VideoCapture(int **device**)

C++

Capturing and reading frames from a camera

read()

- This is the most convenient method for reading video files.

```
cv2.VideoCapture.read([image]) → retval, image
```

python

```
bool VideoCapture::read(Mat& image)
```

C++

Capturing and reading frames from a camera

Example:

```
cap = cv2.VideoCapture(0)

while(True): # Capture frame-by-frame
        ret, frame = cap.read()
        …
        cv2.imshow('frame',gray)
        if cv2.waitKey(30) == ord('q'):
                break

capture.release()
cv2.destroyAllWindows()
```

python

```
VideoCapture cap(0); // open the default camera
if(!cap.isOpened()) // check if we succeeded
      return -1;
Mat frame;
namedWindow(“frame",1);
for(;;)
{
        Mat frame;
        cap >> frame; // get a new frame from camera ….
        …
        imshow(“frame", frame);
        if(waitKey(30) >= 0) break;
}
```

C++

writing frames into a video file

VideoWriter

Video writer class

```
cv2.VideoWriter([filename, fourcc, fps, frameSize[,
isColor]]) → <VideoWriter object>
```

python

```
VideoWriter::VideoWriter(const string& filename, int fourcc,
double fps, Size frameSize, bool isColor=true)
```

C++

writing frames into a video file

VideoWriter::write

Writes the next video frame

```
cv2.VideoWriter.write(image) → None
```

python

```
void VideoWriter::write(const Mat& image)
```

C++

Simple image transformations— resizing

resize()

Python:
cv2.resize(src, dsize, dst, fx, fy, interpolation) → dst

C++:
 void resize(InputArray src, OutputArray dst, Size dsize, double fx=0, double fy=0, int interpolation=INTER_LINEAR)

- **src** – input image.
- **dst** – output image;
- **dsize** – output image size;
- **fx** – scale factor along the horizontal axis.
- **fy** – scale factor along the vertical axis.
- **interpolation**

Simple image transformations— resizing

Python:

OpenCV offers several ways of using the cv2.resize function.

- We can set the target size (width, height) in pixels as the second parameter:

```
width, height = 128, 256
resized_img = cv2.resize(img, (width, height))
```

- Resize by setting multipliers of the image's original width and height:

```
w_mult, h_mult = 0.25, 0.5
resized_img = cv2.resize(img, (0, 0), resized_img, w_mult, h_mult)
```

Simple image transformations— flipping

Flips a 2D array around vertical, horizontal, or both axes.

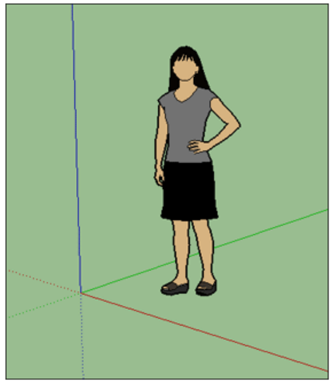
Python: cv2.flip(src, flipCode, dst) → dst

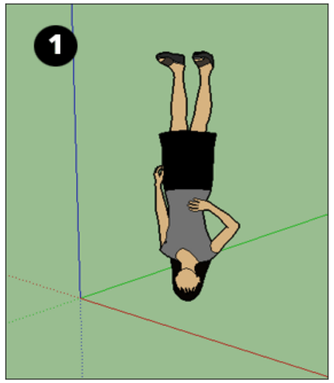
C++: void flip(InputArray src, OutputArray dst, int flipCode)

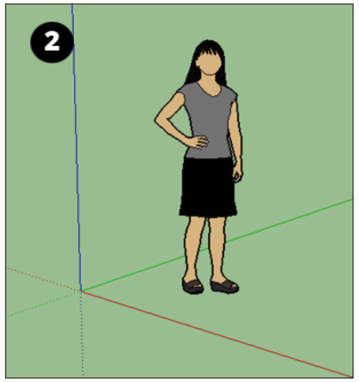
src – input array.

- **dst** – output array of the same size and type as src.
- **flipCode** – a flag to specify how to flip the array;

0: around the x-axis
positive value (for example, 1): flipping around y-axis.
Negative value (for example, -1) means flipping around both axes







Arithmetic Operations on Images

•arithmetic operations on images like addition, subtraction, bitwise operations etc.

add() :

You can add two images by OpenCV function.

Python:

cv2.add(src1, src2, dst, mask, dtype) → dst

There is a difference between OpenCV addition and Numpy addition. OpenCV addition is a saturated operation while Numpy addition is a modulo operation.

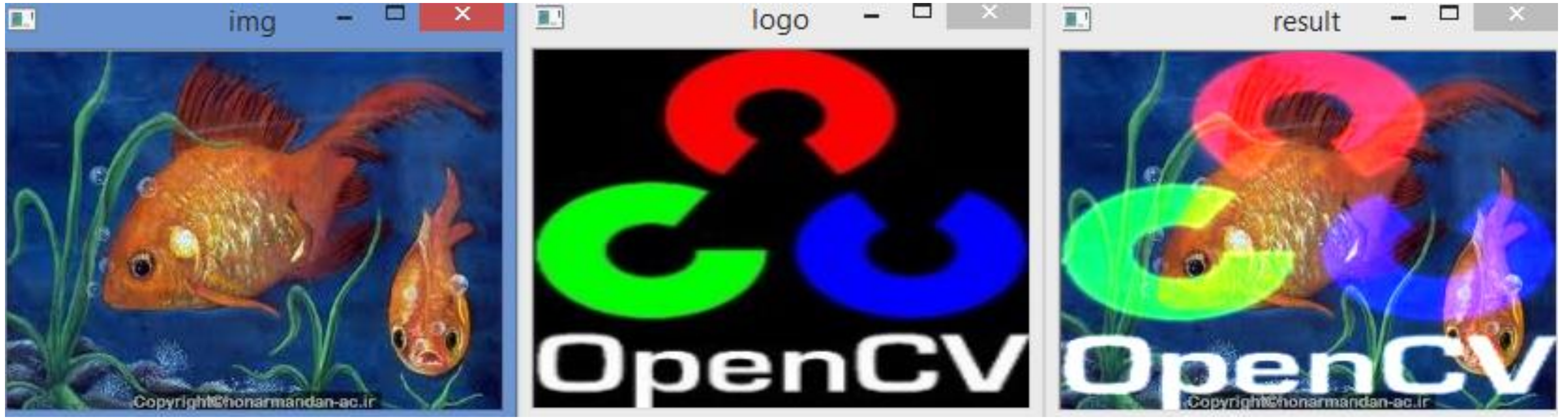
C++: void add(InputArray src1, InputArray src2, OutputArray dst, InputArray mask=noArray(), int dtype=-1)

Both images should be of same depth and type, or second image can just be a scalar value.

mask – optional: specifies elements of the output array to be changed

Let's write a program

I want to put OpenCV logo above an image.



Arithmetic Operations on Images

- arithmetic operations on images like addition, subtraction, bitwise operations etc.

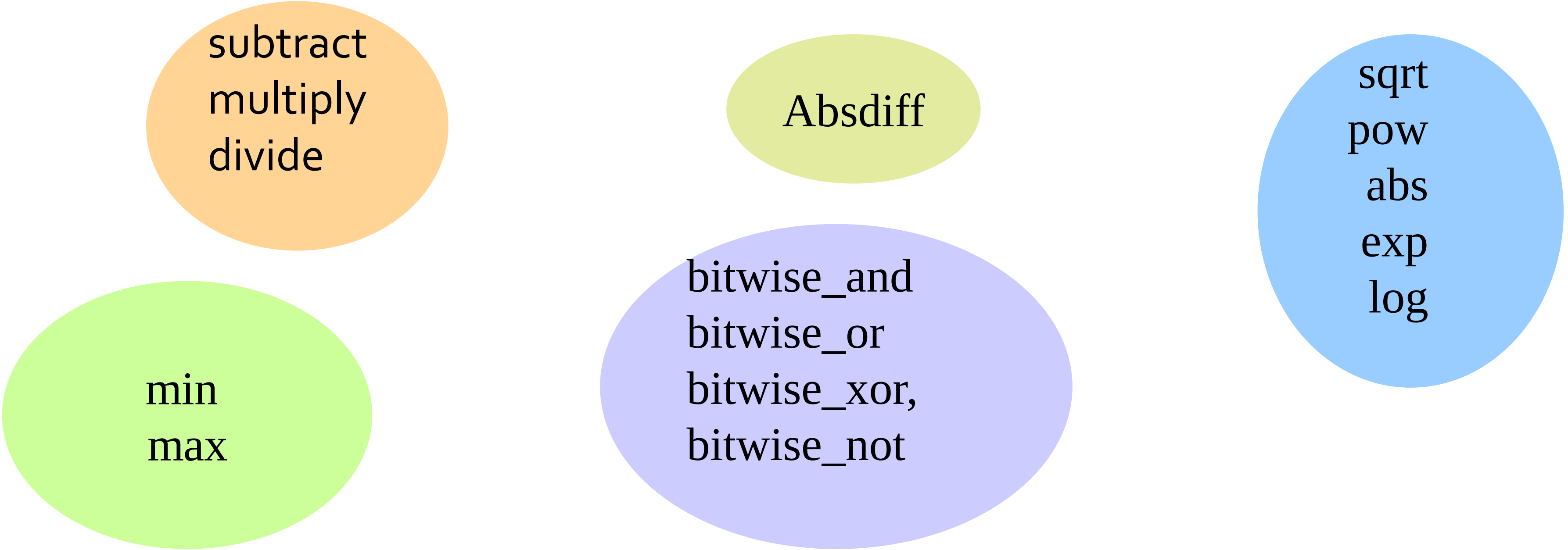
addWeighted:

Python: cv2.addWeighted(src1, alpha, src2, beta, gamma, dst, dtype]) → dst

C++: void addWeighted(InputArray src1, double alpha, InputArray src2, double beta, double gamma, OutputArray dst, int dtype=-1)

$$\texttt{dst}(I) = \texttt{saturate}(\texttt{src1}(I) * \texttt{alpha} + \texttt{src2}(I) * \texttt{beta} + \texttt{gamma})$$

Arithmetic Operations on Images



subtract
multiply
divide

Absdiff

sqrt
pow
abs
exp
log

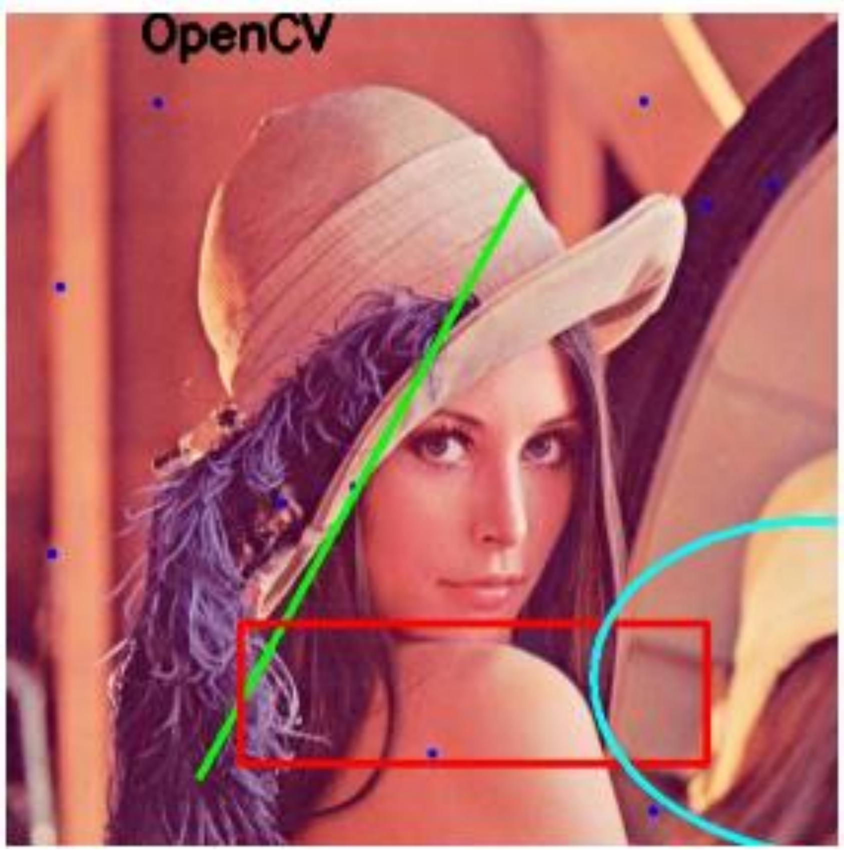
min
max

bitwise_and
bitwise_or
bitwise_xor,
bitwise_not

Most C++ operators have been overloaded. Among them are the arithmetic operators +, -, *, / and bitwise operators &, |, ^, ~ and .....

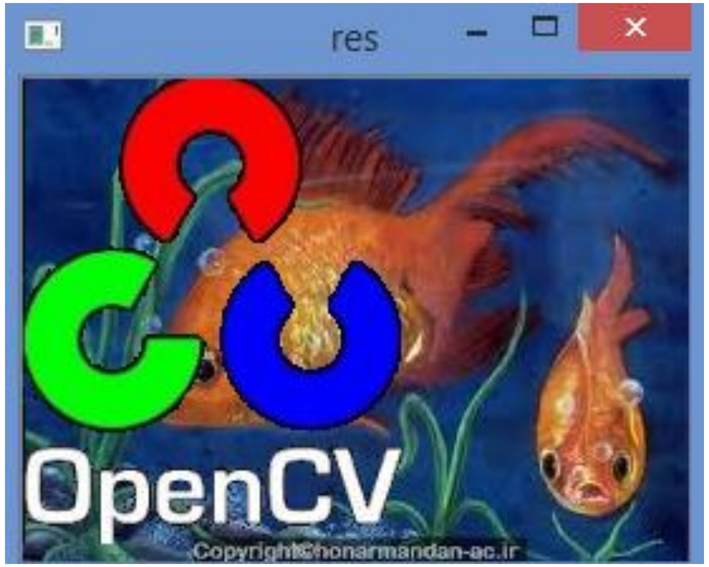
Drawing functions—markers, lines, ellipses, rectangles, text and …

circle
ellipse
line
arrowedLine
rectangle
putText



Let's write a program

I want to put OpenCV logo at the top-left corner of image



Let's write a program

I want to add rain to my video

Manipulating matrices-creating, filling, accessing elements,

- **Matrices in OpenCV's Python interface are presented with NumPy arrays.**
- **so you can use numpy functions for images.**

np.full ( ): create full matrice
np.zeros(): create zero matrice

- **Another feature of a matrix is its element type. The element type defines which data type is used to represent element values. For example, each pixel can store values in the [0-255] range—in this case, it is np.uint8. Or, it can store float (np.float32) or double (np.float64) values.**

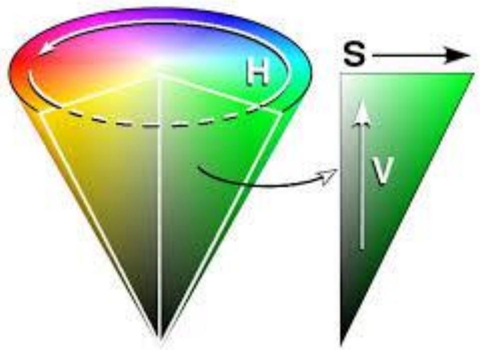
- **for historical reasons, OpenCV stores color values for RGB representation in BGR format—so be careful.**

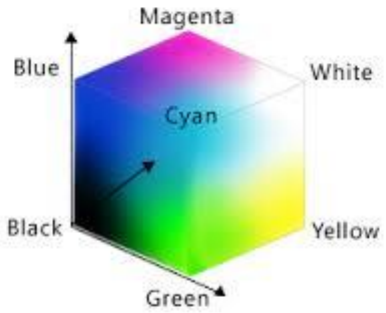
Let's write a program

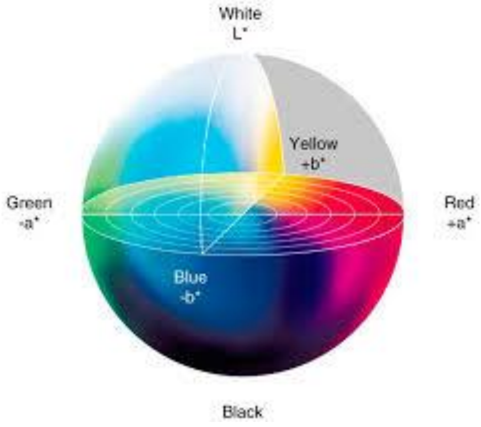
I want to create an image, fill it with desired value and access its elements….

Converting images from one color space to another

- By default, full color images in OpenCV are presented in RGB color space. But for some cases it's necessary to move to other color representations; for example, to have a separate channel for intensity.







Converting images from one color space to another

cvtColor:

Python: cv2.cvtColor(src, code, dst, dstCn]) → dst

Convert the image to grayscale:
gray =
cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)

Convert the image to HSV color space:
hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

C++: void cvtColor(InputArray src, OutputArray dst, int code, int dstCn=0)