MACHINE LEARNING LAB

Introduction to OpenCV, PIL

MUNADI SIAL

SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

Overview

This lab will be centered on the following:

- Understand the image data as an array of pixels
- Load and Save images
- Display images in different windows
- Access and modify pixels in images
- Access and modify regions in images
- Place lines, rectangles, circles and text in images
- Resize images at various scales
- Rotate images at various angles

OpenCV

- OpenCV is an imaging library used widely for computer vision tasks such as feature extraction, stereovision, image stitching, pose estimation and many more

- Computer vision is the field that involves getting information from image data

- OpenCV is available in Python, C++ and MatLab

- To use OpenCV in Python, we import the library:

```
import cv2
```

Images

An image is an array of unit squares called “pixels”

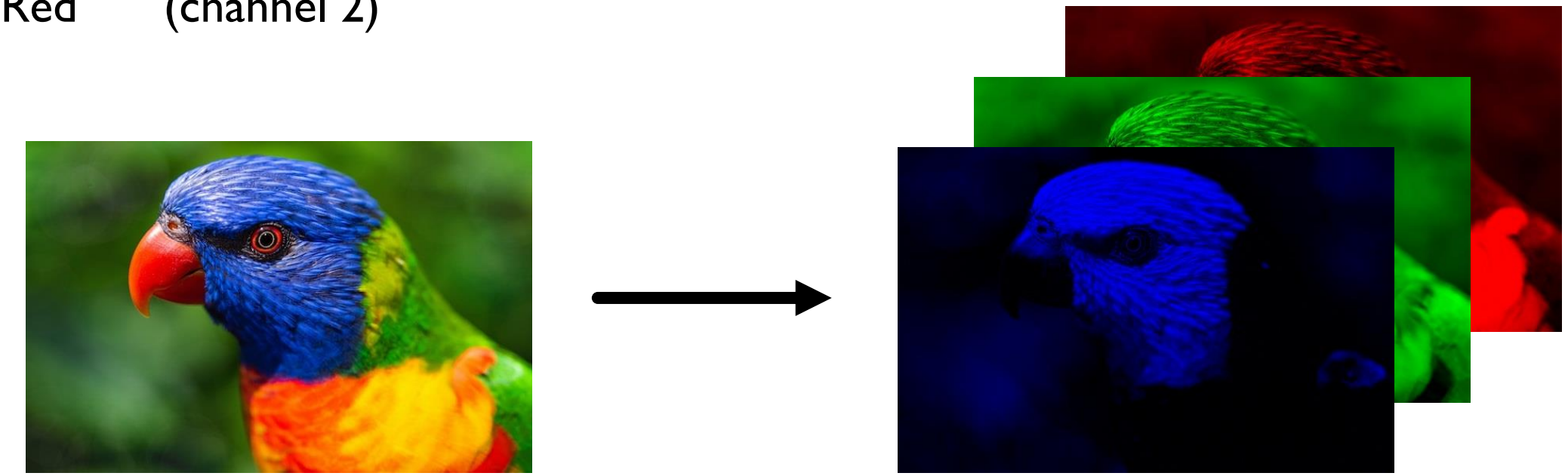
Each pixel has a location (x,y) in the image array

Each pixel has some color values (intensity level)

Images

A colored image has 3 channels, i.e. there are 3 color values in every pixel:

- Blue (channel 0)
- Green (channel 1)
- Red (channel 2)

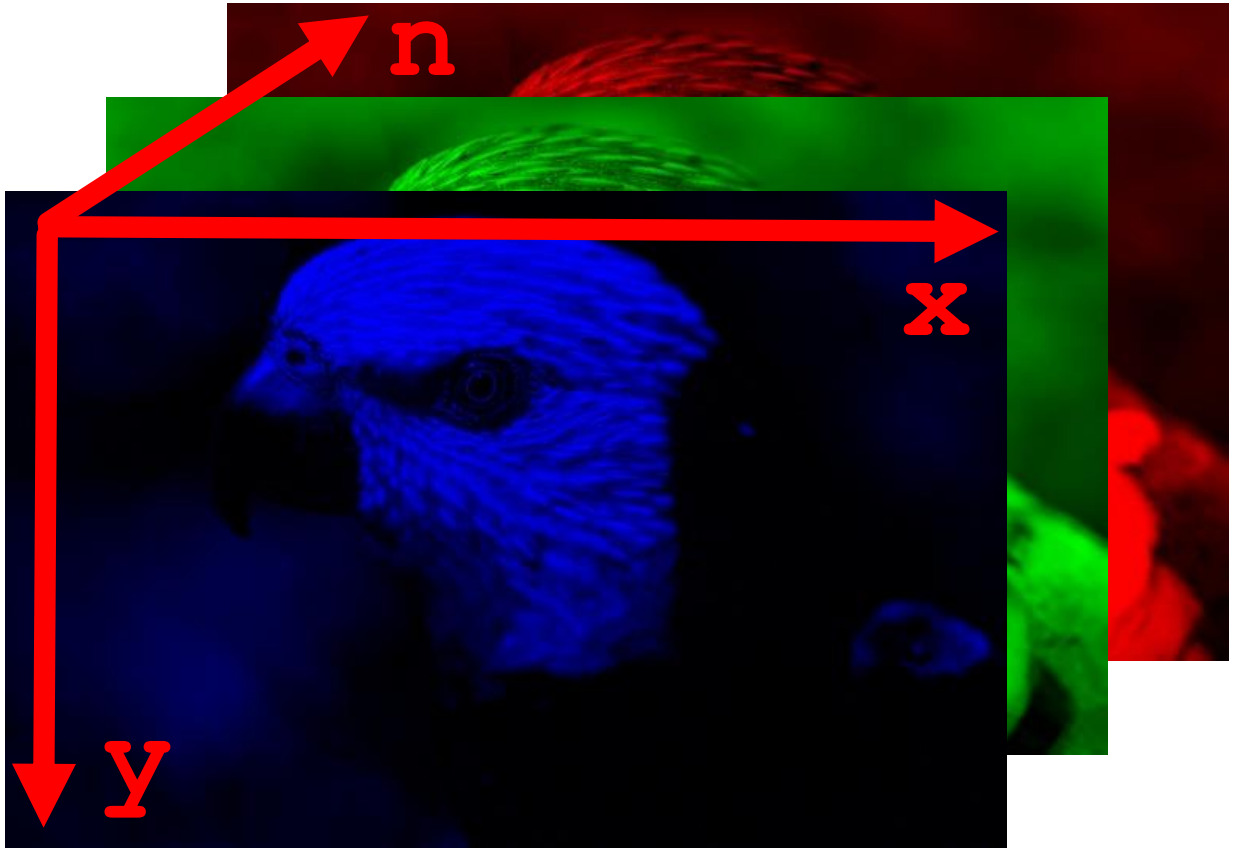


By default, OpenCV stores channels as BGR (not RGB)

Images

An image is essentially a 3-D array defined by its height (y), width (x) and number of channels (n)

Notice that the y-axis points downwards in the pixel frame



Load Images

To load an image, we use the **imread** function:

```
imgA = cv2.imread("my_image.jpg", 1)
```

The above function will load the image file and place it in *imgA*.

The *imgA* is like a variable that stores an image and we can use it in our program to reference the image

(The second argument in the imread function chooses the color mode;
0 is for grayscale, 1 is for colored and -1 is for alpha-channel inclusive)

Load Images

If the image is in the same directory as the script file, the image filename is directly used:

```
imgA = cv2.imread(“p3at.jpg”, 1)
```

If the image is in a different directory then the entire path to the image must be provided:

```
imgB = cv2.imread('D://docs/Robot Pics/turtlebot.jpg', 1)
```

Display Images (IDE)

Once the image is loaded, it can be displayed using the **imshow** function:

```
cv2.imshow('image1',imgA)
```

The first argument is the *window* name. This name appears at the top of the window which shows the image.

The second argument is the image object that is to be displayed

Display Images (IDE)

Once the image is loaded, it can be displayed using the **imshow** function:

```
cv2.imshow('image1',imgA)
cv2.waitKey(0)
```

When the imshow function is used, the image appears only for a very small time before it closes. To avoid this, the **waitKey** function is used to hold the image.

The argument for the waitkey function specifies the time (in milliseconds) to hold the image. If the argument is 0, then the image is held for infinite time until the user presses a key.

Display Images (IDE)

The following code will display two images

```
cv2.imshow('image1',imgA)
cv2.imshow('image1',imgB)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

To ensure all windows are closed, the **destroyAllWIndows** function is used.

(Another function called *destroyWindow('window_name')* can be used to close a specific window)

Display Images (CoLab)

When using Google CoLab, the **cv2.imshow** function may not work.

To display images in the CoLab browser window, use the **cv2_imshow** function (notice the underscore)

```
from google.colab.patches import cv2_imshow

imgC = cv2.imread("my_image.jpg", 1)

cv2_imshow(imgC)
```

Save Images

To save an image on disk, the **imwrite** function is used:

```
cv2.imwrite('myImageRotated.jpg',imgA)
```

The first argument is the name of the image file to be saved

The second argument is the image object that we are saving

Basic Operations

Consider the image of size 620 x 420

The x-axis goes from 0 to 619
The y-axis goes from 0 to 419

(The channels go from 0 to 2)

Basic Operations

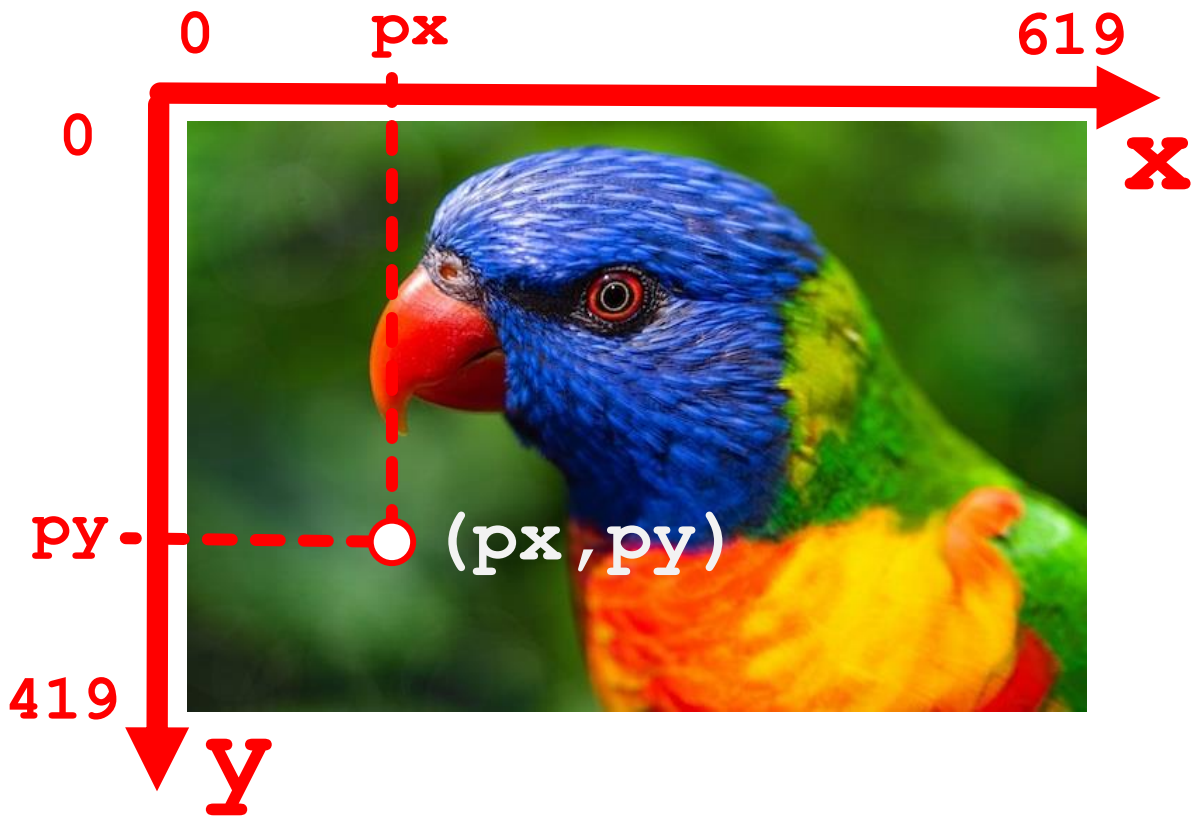
To get the BGR values of a specific pixel at location (px, py), we use:

```
img = cv2.imread('bird.jpg', 1)

val = img[py, px, :]
```

The first index (py) is the row of the array. It corresponds to the y-axis

The second index (px) is the column of the array. It corresponds to the x-axis

The third index (:) is the channel



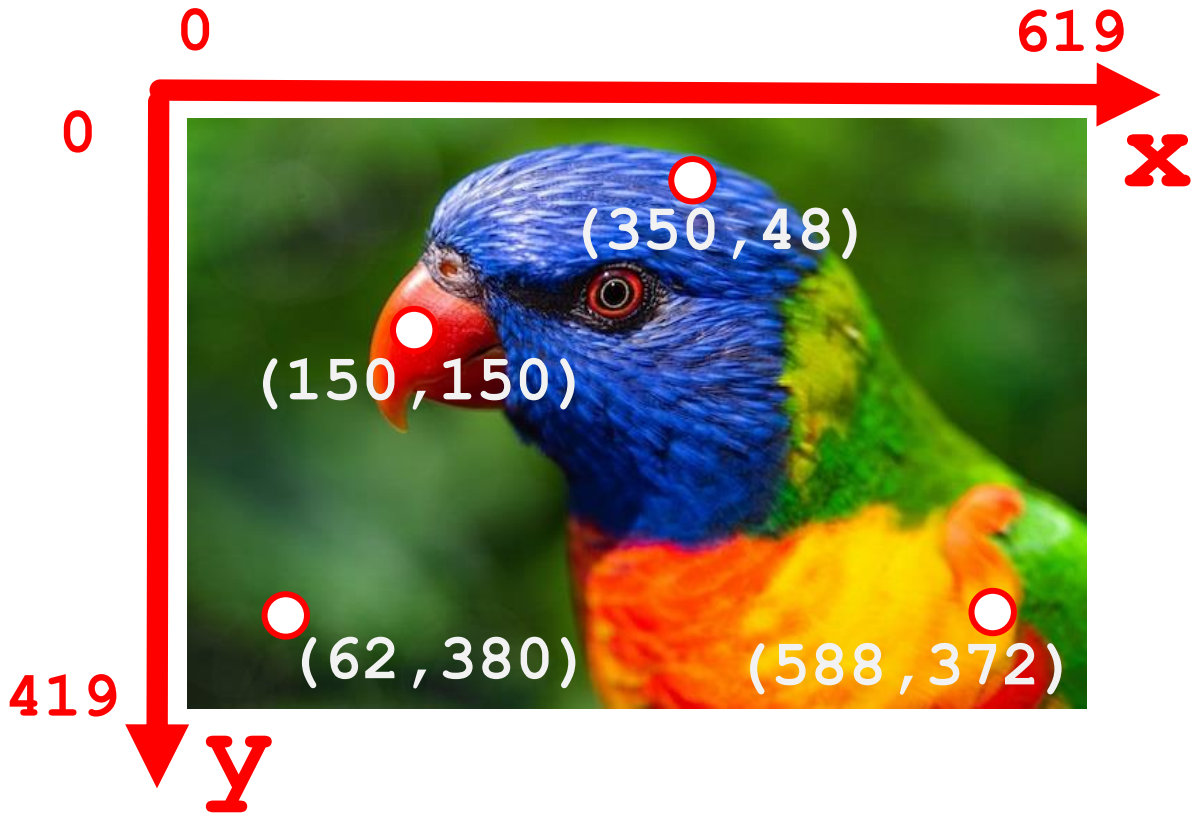
Basic Operations

To get the BGR values at a number of pixels:

```
img = cv2.imread('bird.jpg', 1)

print(img[380,62,:])
print(img[150,150,:])
print(img[48,350,:])
print(img[372,588,:])
```

Output:

```
[27 59 42]
[  2  54 251]
[251 151 127]
[  0 102 184]
```



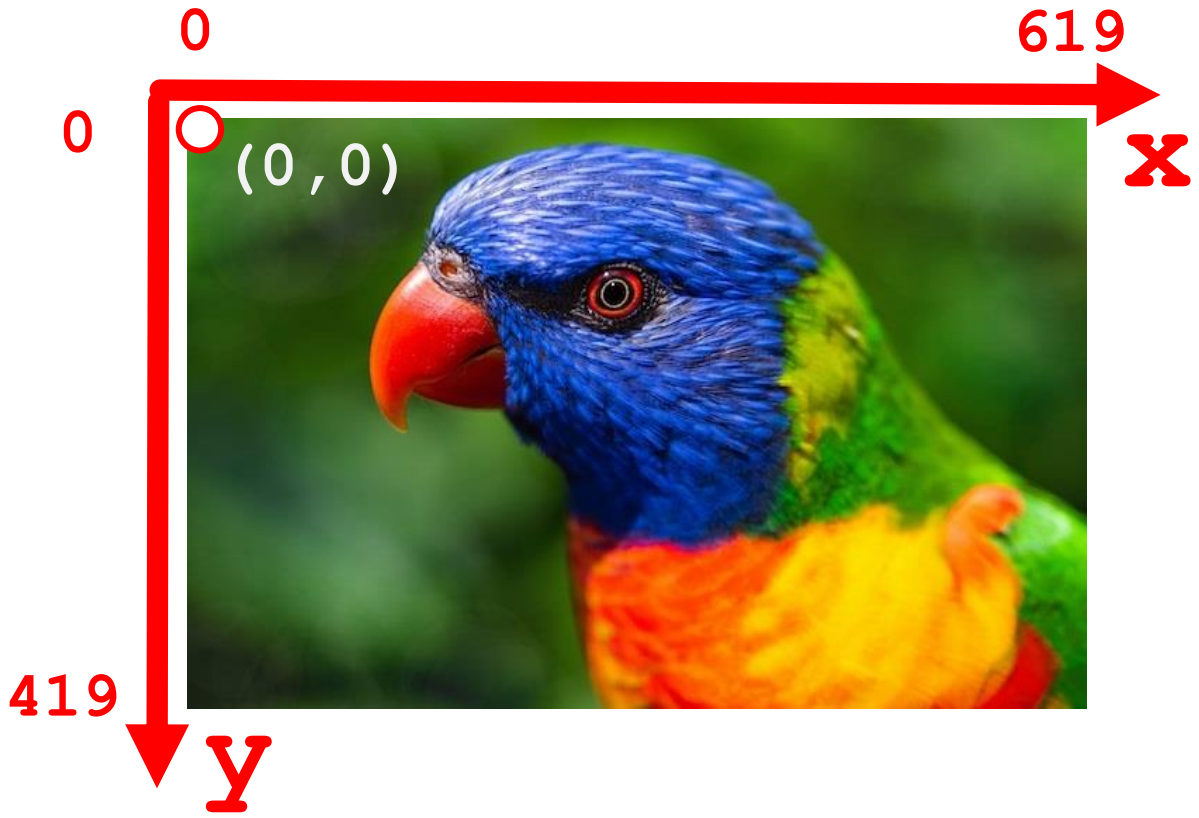
Basic Operations

To get the BGR values at location (0,0) pixel:

```
img = cv2.imread('bird.jpg', 1)

print(img[0,0,:]) # BGR
print(img[0,0,0]) # B
print(img[0,0,1]) # G
print(img[0,0,2]) # R
```

Output:

```
[22 143  63]
22
143
63
```



Basic Operations

The **shape** attribute is used to get the dimensions of the array:

```
img = cv2.imread('bird.jpg', 1)
rows = img.shape[0]
cols = img.shape[1]

print(img.shape)
print(rows)
print(cols)
```

Output:

```
(420, 620, 3)
420
620
```

Basic Operations

The following code will load an image and display its color channels

```
img = cv2.imread('bird.jpg', 1)

imgB = img[:,:,0]
imgG = img[:,:,1]
imgR = img[:,:,2]

cv2.imshow('Original',img)
cv2.imshow('Blue',imgB)
cv2.imshow('Green',imgG)
cv2.imshow('Red',imgR)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

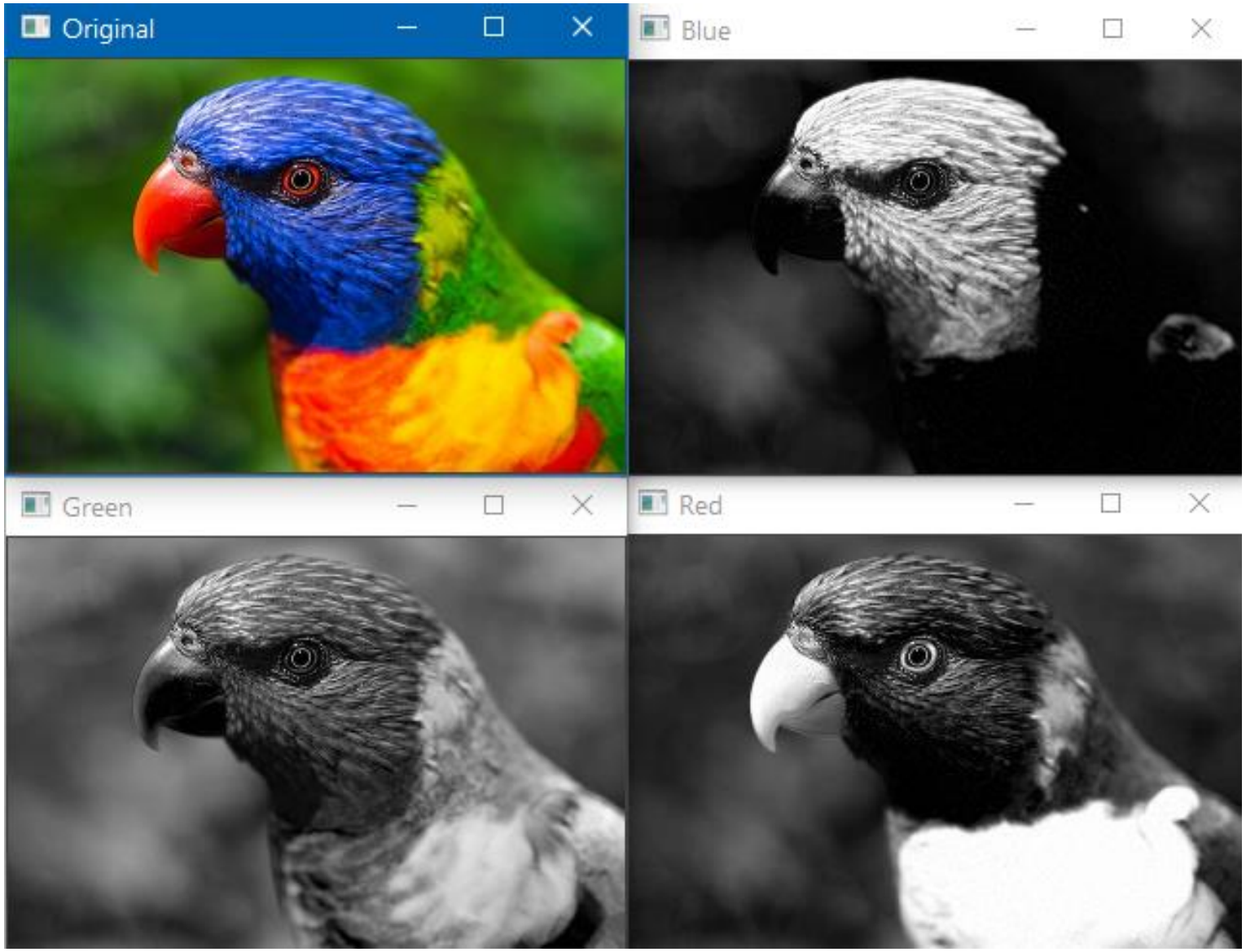
Basic Operations

The results are shown.

The more white color indicates large values of the channel color

Note that the color channels are ordered as BGR (instead of RGB) in OpenCV:

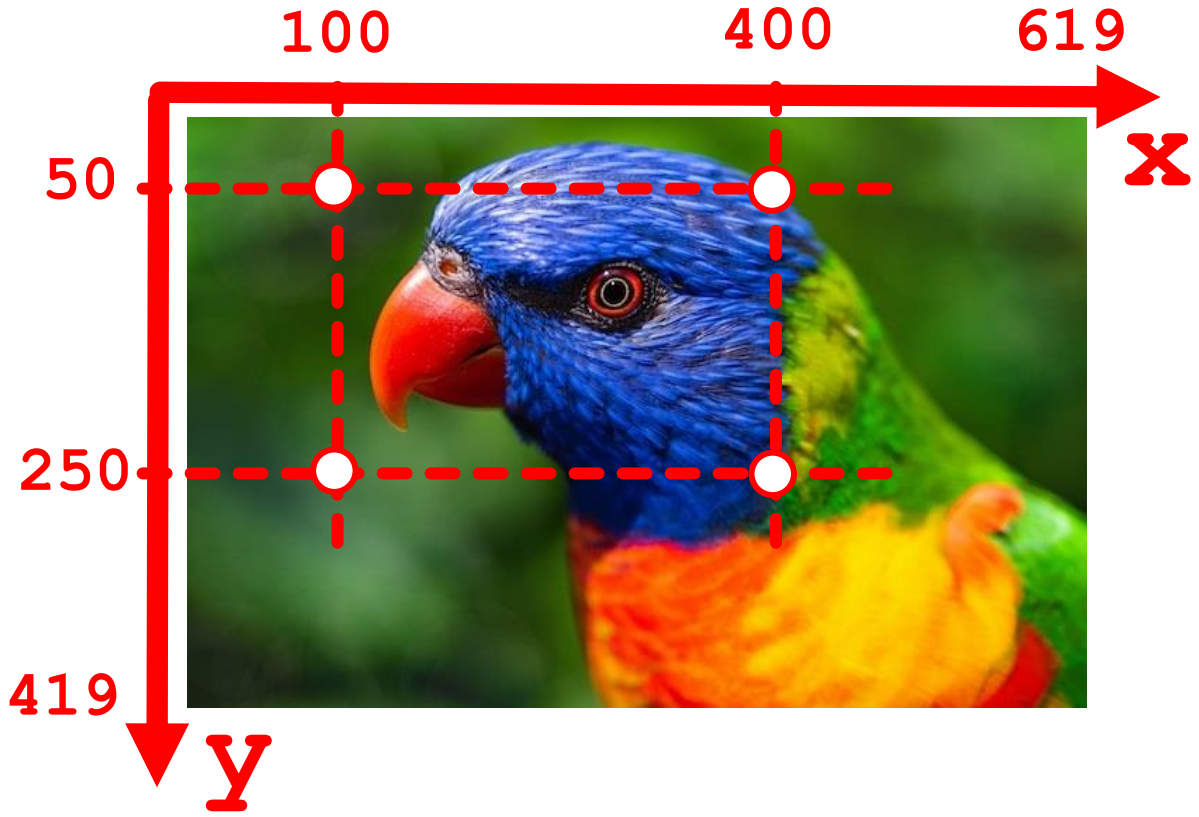
```
imgB = img[:,:,0]
imgG = img[:,:,1]
imgR = img[:,:,2]
```



Cropping Images

Cropping a region of image (ROI) is similar to accessing pixels
A range of pixel values can be taken using a slice operation (:)

```
img = cv2.imread('bird.jpg', 1)
img_crop = img[50:250,100:400]
cv2.imshow('ROI', img_crop)
cv2.waitKey(0)
```



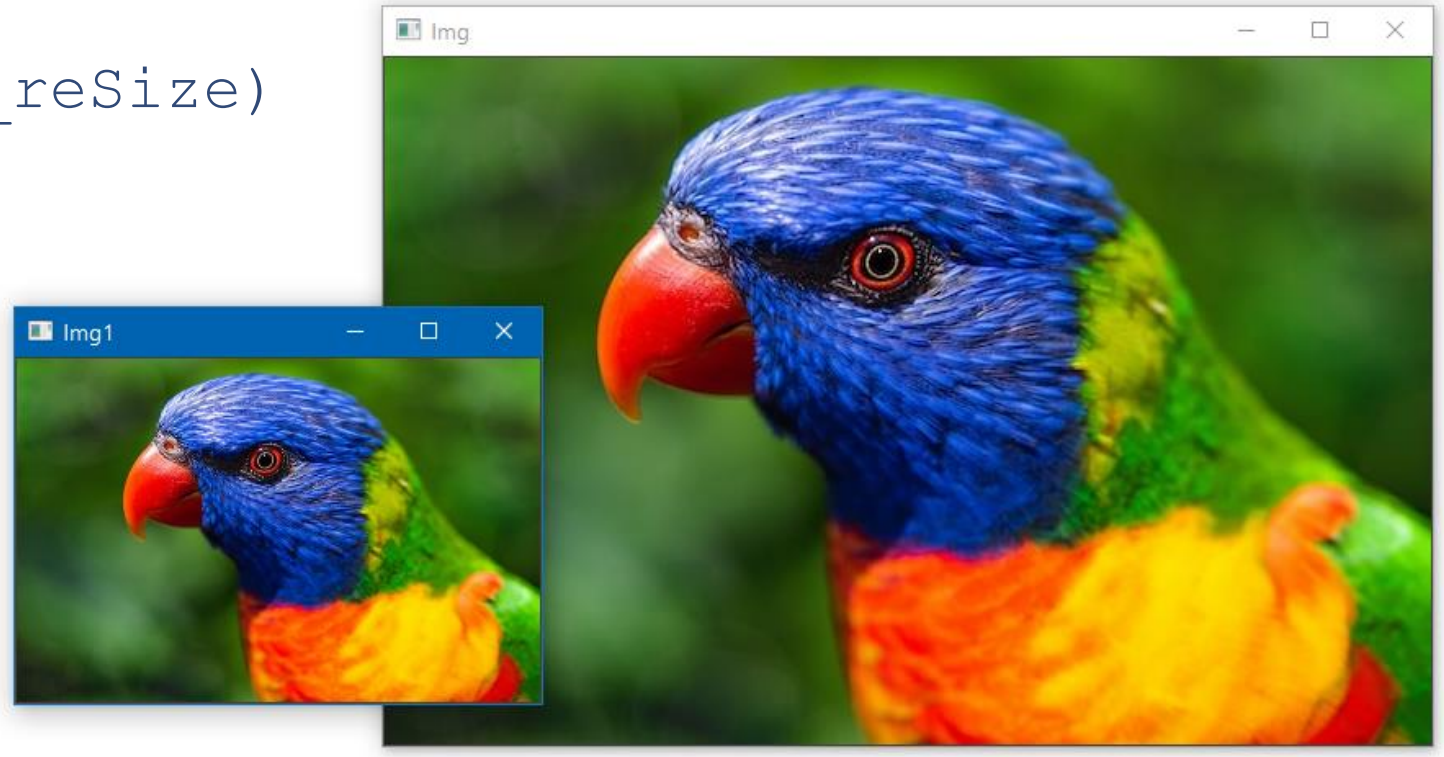
Scaling Images

The **resize** function is used to change the image size

```
imgResize = cv2.resize(img,None,fx=0.5, fy=0.5,
interpolation = cv2.INTER_CUBIC)

cv2.imshow('Img', img)
cv2.imshow('Img1', img_reSize)
```

fx and fy are the scaling factors

Scaling factor of 0.5 means half of original length



Scaling Images

If the scaling factors fx and fy are not equal, the image gets stretched

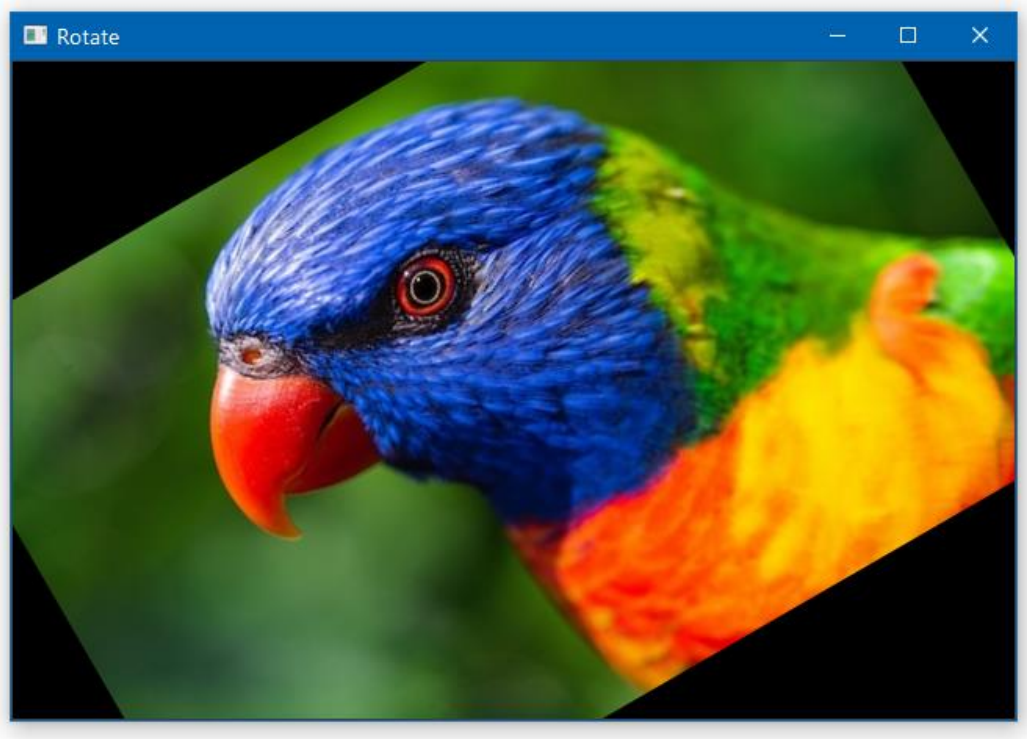
```
img_resized1 = cv2.resize(img,None,fx=1,
fy=0.4, interpolation = cv2.INTER_CUBIC)

img_resized2 = cv2.resize(img,None,fx=0.3,
fy=1, interpolation = cv2.INTER_CUBIC)
```

Rotating Images

To rotate the image, 2 functions are used:

- **getRotationMatrix2D** to compute the transformation matrix
- **warpAffine** to apply the rotation transformation to the image matrix

Rotated at 30 degrees



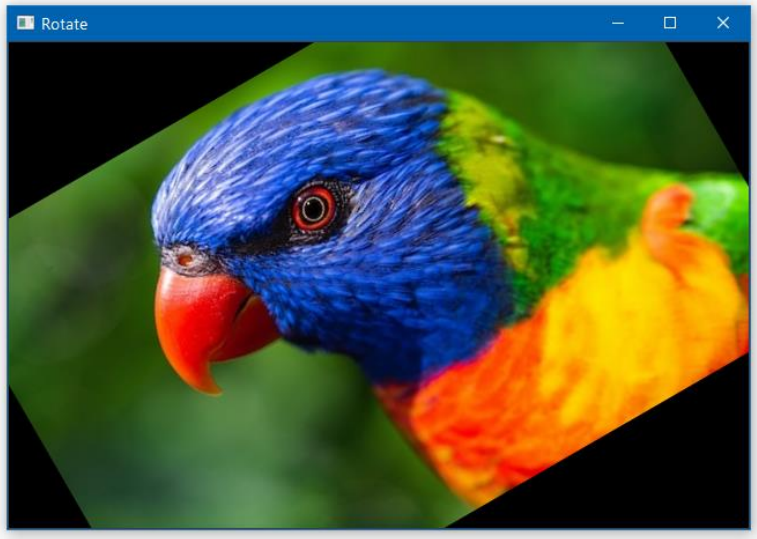

```
img = cv2.imread('bird.jpg', 1)
rows = img.shape[0]
cols = img.shape[1]

M = cv2.getRotationMatrix2D((cols/2,rows/2),30, 1)
rot = cv2.warpAffine(img,M,(cols,rows))

cv2.imshow('Rotate', rot)
```

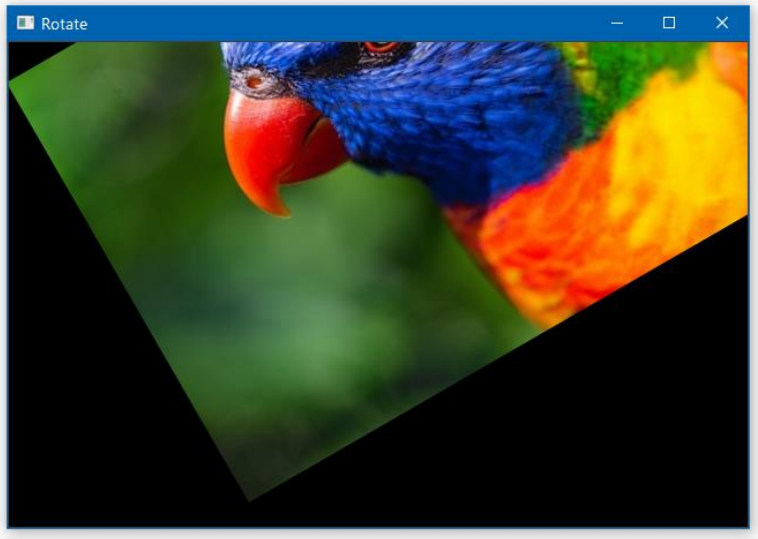
Rotating Images

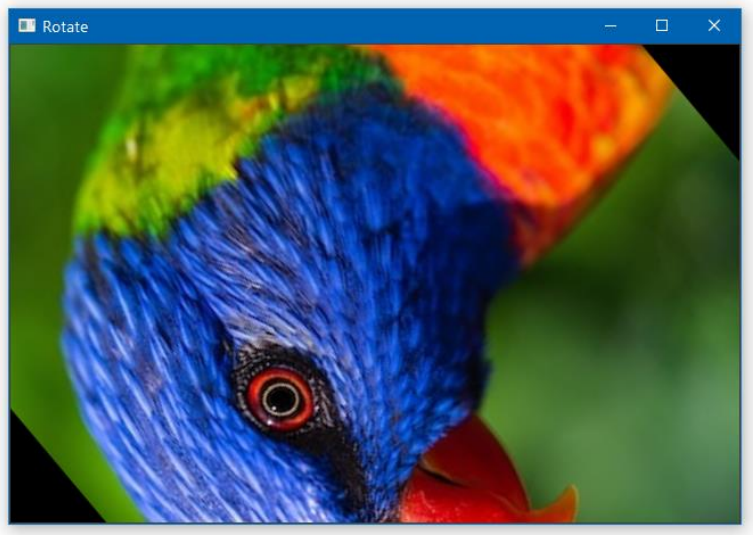
The **getRotationMatrix2D** takes the following inputs:

- Center of rotation (cx, cy)
- Angle of rotation A
- Scale factor k

```
M = cv2.getRotationMatrix2D((cx, cy), A, k)
```





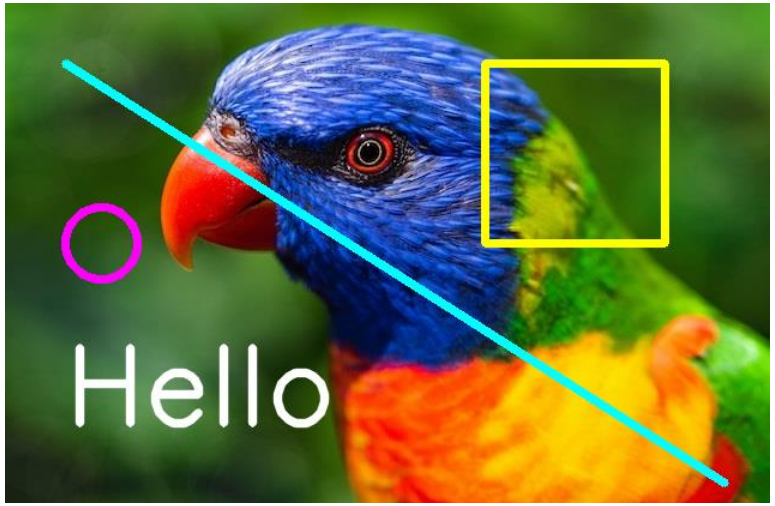


Placing Shapes

Various shapes can be placed in the image:

```
img1 = cv2.line(img,(x1,y1),(x2,y2),(B,G,R),t)
img2 = cv2.rectangle(img,(x1,y1),(x2,y2),(B,G,R),t)
img3 = cv2.circle(img,(xc,yc), radius, (B,G,R), t)
```

(x1,y1)	start point
(x2,y2)	end point
(xc,yc)	center point
(B,G,R)	color values from 0 to 255
thickness	line thickness (-1 for solid color)



Text can also be placed in the image:

```
cv2.putText(img,'Hello', (x,y), cv2.FONT_HERSHEY_SIMPLEX,
size, (B,G,R), thickness, cv2.LINE_AA)
```

Placing Shapes

The following example places some shapes and text in the image

```
img = cv2.line(img,(50,50),(600,400),(255,255,0),5)
img = cv2.rectangle(img,(400,50),(550,200),(0,255,255),5)
img = cv2.circle(img,(80,200), 30, (255,0,255), 5)

font = cv2.FONT_HERSHEY_SIMPLEX

cv2.putText(img,'Hello',
(50,350), font, 3,
(255,255,255), 5, cv2.LINE_AA)

cv2.imshow('Shapes', img)
```

Lab Tasks

- Download the manual from LMS
- Perform the Lab Tasks as given in the manual and submit it on LMS
- Remember to execute scripts with the terminal