# Introduction To OpenCV

“Sooner or later all things are numbers, yes?”

— Terry Pratchett,

# OpenCV : (Open Source Computer Vision)

- Originally developed by Intel
- Currently maintained by Itseez
- supports the Deep Learning frameworks TensorFlow, Torch/PyTorch and Caffe.
- OpenCV is used as the primary vision package in ROS.
- Written in C/C++
- Wiki
- https://opencv.org/

# Getting Started with Images:

- **cv2.imread(<image path> , FLAG) :** read an image
  - cv2.IMREAD_COLOR
  - cv2.IMREAD_GRAYSCALE (or simply use : 0)
  - cv2.IMREAD_UNCHANGED
- **cv2.imshow(<frame name> , imageMat) :** display an image
  - cv2.waitKey( ) & 0xFF
  - cv2.destroyAllWindows( )
  - cv2.namedWindow('image', cv2.WINDOW_NORMAL)
- **cv2.imwrite( 'image.png' , imageMat ) :** save an image
- **RGB vs. BGR**
- Write a program to convert a color image to grayscale image and save it.
  - [ Additional Info: **cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)** ]

# Getting Started with Videos:

- **cap = cv2.VideoCapture( Arg ) :**
  - **Arg : can be the device index (eg., Webcam : 0) or it can be a video path.**
  - **Creates a VideoCapture object to capture the video frames from the interface. Interface can either be a Video or a Webcam.**
- Additional Info **: cv2.VideoWriter( ) :** used to save a video.
- Write a program which captures video from the WebCam and save it to the local drive.

# Drawing Functions:

- **cv2.line( )**
- **cv2.circle( )**
- **cv2.rectangle( )**
- **cv2.ellipse( )**
- **cv2.putText( )**

- Additional Info: **cv2.setMouseCallback() : Use this function to make a paintbrush application. Refer to the OpenCV documentation for details.**

# Operations on Image:

- **Accessing & Modifying a Pixel:**
  - `Px = imgMat [ row , col ]  # accessing`
  - `imgMat [ row , col ] = 100  # modifying`

- **Warning: Numpy is a optimized library for fast array calculations. So simply accessing each and every pixel values and modifying it will be very slow and it is discouraged.**

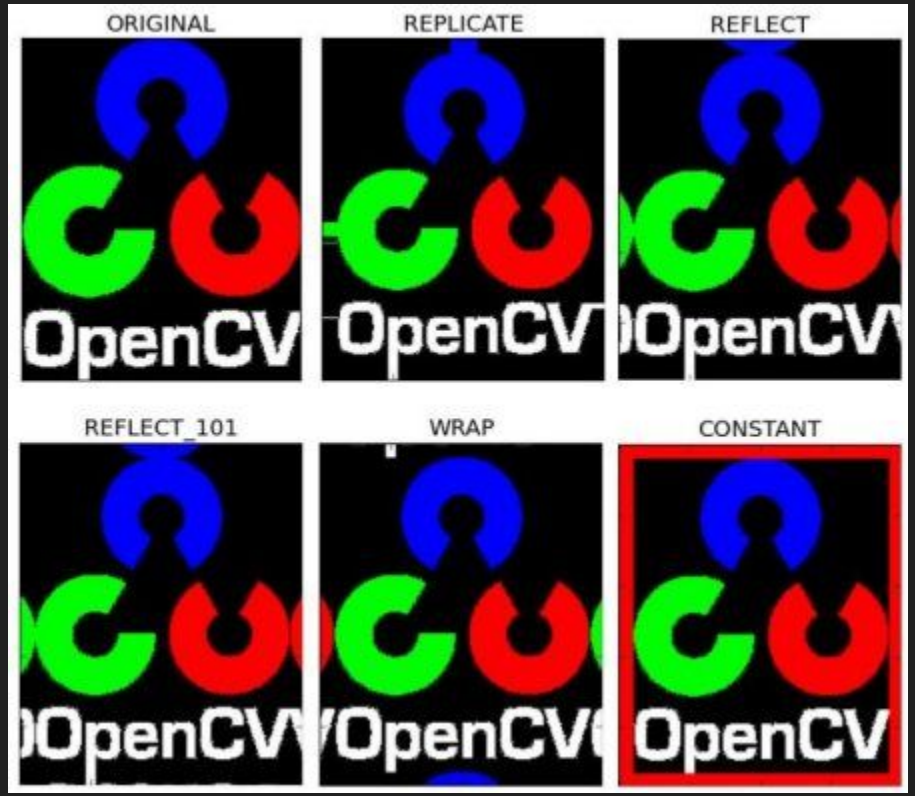
# Image Props:

- **Image Shape :**
  - **Row , Col , channels  = imgMat.shape  # for a color image**
  - **Row , Col  = imgMat.shape  # for a grayscale image**
- **Image Size :** total number of pixels
  - **Total_pixels = imgMat.size**
- **Image Datatype :**
  - **Img_data_type = imgMat.dtype**
- **ROI (Region of Images):**
  - **Roi = imgMat [ startRow : endRow  , startCol : endCol ]  # numpy array indexing**

# Image Padding: (Used in Convolution)

- **cv2.copyMakeBorder( ) :** make image border
- **Arguments:**
  - **Src : input image**
  - **top, bottom, left, right - border width in number of pixels in corresponding directions**
  - **borderType - Flag defining what kind of border to be added.**
    - **cv2.BORDER_CONSTANT**
    - **cv2.BORDER_REFLECT**
    - **cv2.BORDER_REFLECT_101 or cv2.BORDER_DEFAULT**
    - **cv2.BORDER_REPLICATE**
    - **cv2.BORDER_WRAP**

# Image Padding: (Used in Convolution)



# Image Addition:

- **cv2.add() :** simple addition of two images
  - Note: There is a difference between OpenCV addition and Numpy addition. OpenCV addition is a saturated operation while Numpy addition is a modulo operation.
- **cv2.addWeighted() :** blend two images
  - $dst = \alpha \cdot img1 + \beta \cdot img2 + \gamma$

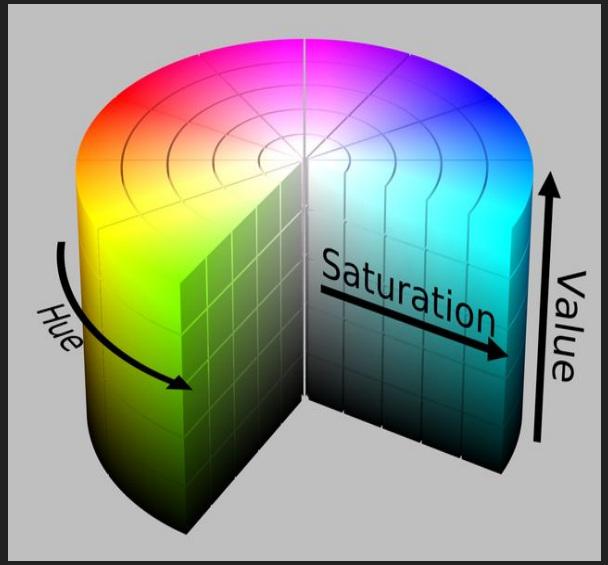
- **Create a slideshow of images in a folder with smooth transition between images using cv2.addWeighted function.**

# Bitwise operations:

# Image Processing In OpenCV:

- **CHANGING COLORSPACES :**
  - **BGR ↔ HSV , BGR ↔ GRAY , etc.,**
  - **cv2.cvtColor(input_image, cv2.COLOR_BGR2GRAY)**
  - **cv2.cvtColor(input_image, cv2.COLOR_BGR2HSV)**

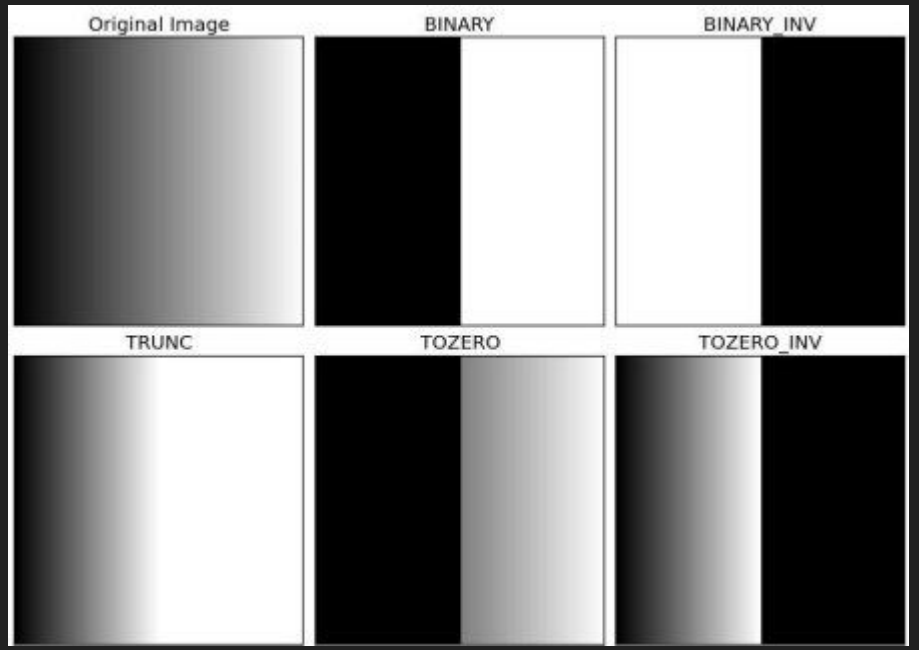
- **Note:** For HSV, Hue range is [0,179], Saturation range is [0,255] and Value range is [0,255].



# Object Tracking:

# Thresholding:

- **cv2.threshold()**
  - Arguments:
  - Src : input image
  - Thresh : threshold value
  - Maxval : maximum value to use with the THRESH_BINARY and THRESH_BINARY_INV thresholding types.
  - Type :
    - THRESH_BINARY
    - THRESH_BINARY_INV
    - THRESH_TRUNC
    - THRESH_TOZERO
    - THRESH_TOZERO_INV



# Adaptive Thresholding:

- **cv2.adaptiveThreshold()**
  - Arguments:
  - Src : input image
  - Maxval : maximum value
  - adaptiveMethod:
    - ADAPTIVE_THRESH_MEAN_C
    - ADAPTIVE_THRESH_GAUSSIAN_C
  - thresholdType :
    - THRESH_BINARY
    - THRESH_BINARY_INV
  - blockSize: Size of a pixel neighborhood
  - C : Constant subtracted from the mean or weighted mean