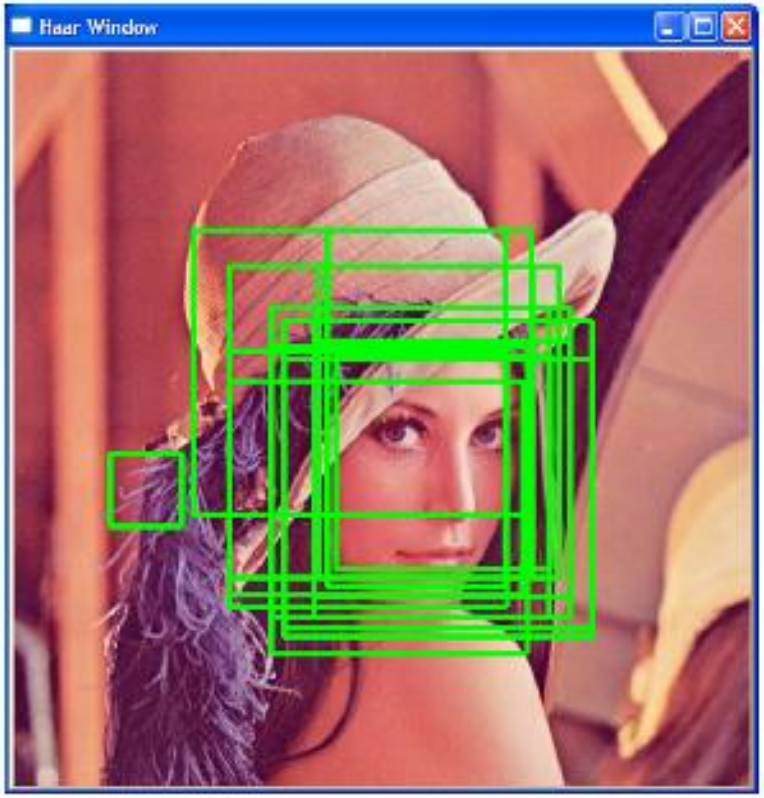
IMAGE PROCESSING

ROBOTICS CLUB SUMMER CAMP'12

What is Image Processing?



- IMAGE PROCESSING = IMAGE + PROCESSING

WHAT IS IMAGE?

- IMAGE = Made up of PIXELS
- Each Pixels is like an array of Numbers.
- Numbers determine colour of Pixel.

- TYPES OF IMAGES

1. BINARY IMAGE
2. GREYSCALE IMAGE
3. COLOURED IMAGE

Binary Image

Each Pixel has either 1 (White) or 0 (Black)

Depth =1 (bit)

Number of Channels = 1

0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0

Grayscale

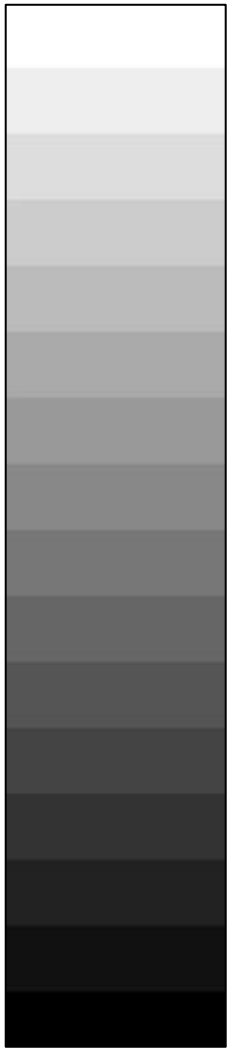
Each Pixel has a value from 0 to 255.

0 : black and 1 : White

Between 0 and 255 are shades of b&w.

Depth=8 (bits)

Number of Channels =1



Grayscale Image

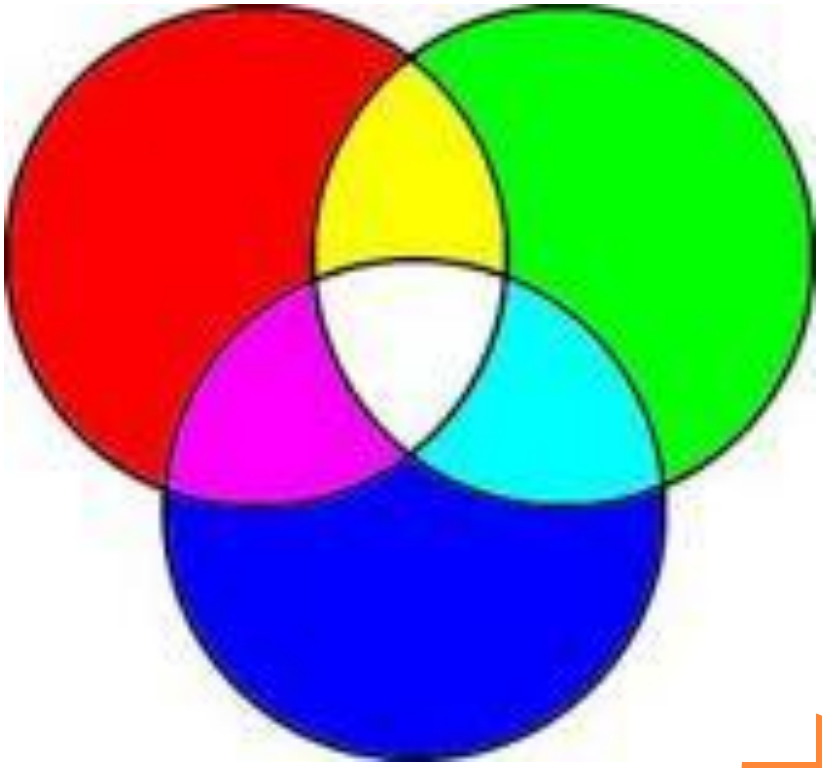
RGB Image

Each Pixel stores 3 values :-

R : 0- 255

G: 0 -255

B : 0-255



Depth=8 (bits)

Number of Channels = 3

RGB IMAGE

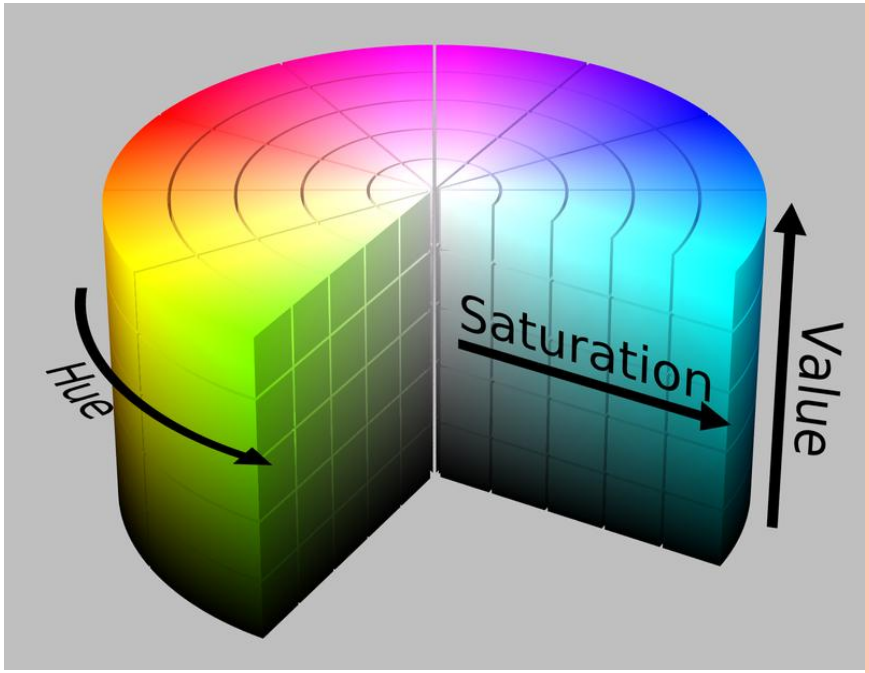
HSV IMAGE

Each pixel stores 3 values :-

H (hue) : 0 -180

S (saturation) : 0-255

V (value) : 0-255



Depth = 8 (bits)

Number of Channels = 3

Note : Hue in general is from 0-360 , but as hue is 8 bits in OpenCV , it is shrinked to 180

Starting with OpenCV

OpenCV is a library for C language developed for Image Processing

To embed opencv library in Dev C complier , follow instructions in :-

http://opencv.willowgarage.com/wiki/DevCpp

Header Files in C

After embedding openCV library in Dev C include following header files:-

#include "cv.h"

#include "highgui.h"

IMAGE POINTER

An image is stored as a structure ***IplImage*** with following elements :-

int height

int width

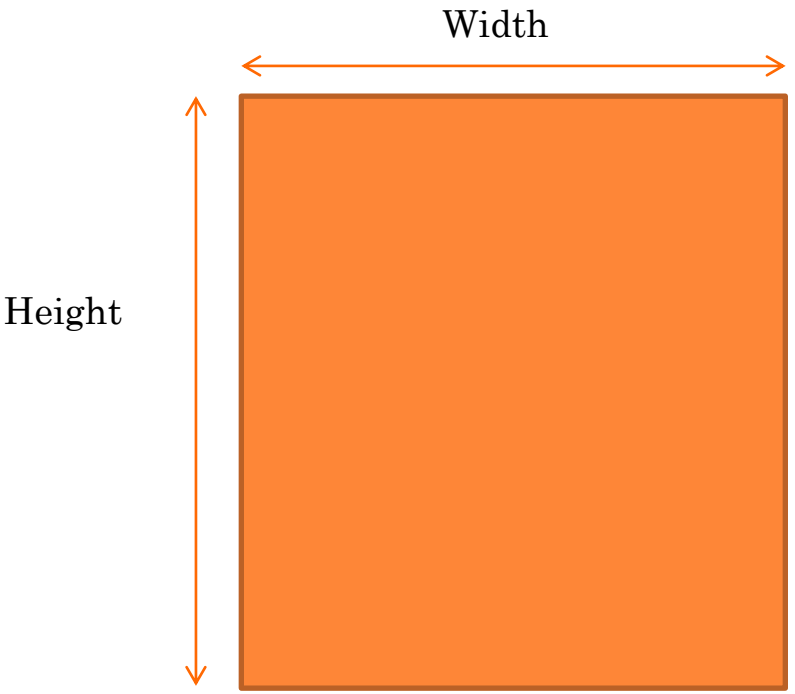
int nChannels

int depth

char *imageData

int widthStep

….. So on



- Initialising pointer to a image (structure) :-

  IplImage input*

- Load image to the pointer [0=gray;1=colored]

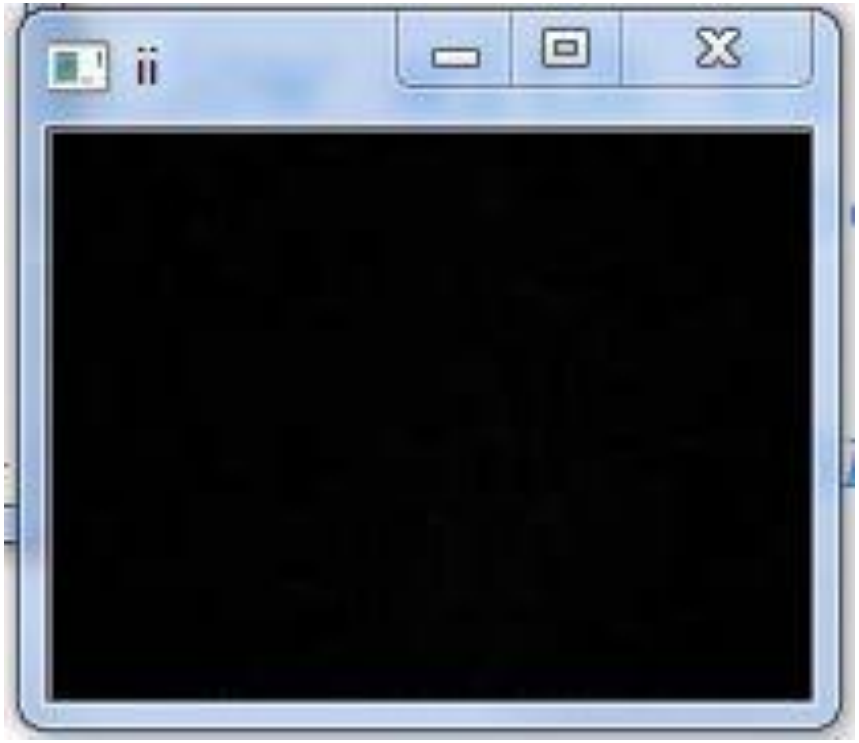
  input = cvLoadImage("apple.jpg",1)

Note :The image apple.jpg must be in same folder where you save your C program

cvNamedWindow("ii",1)

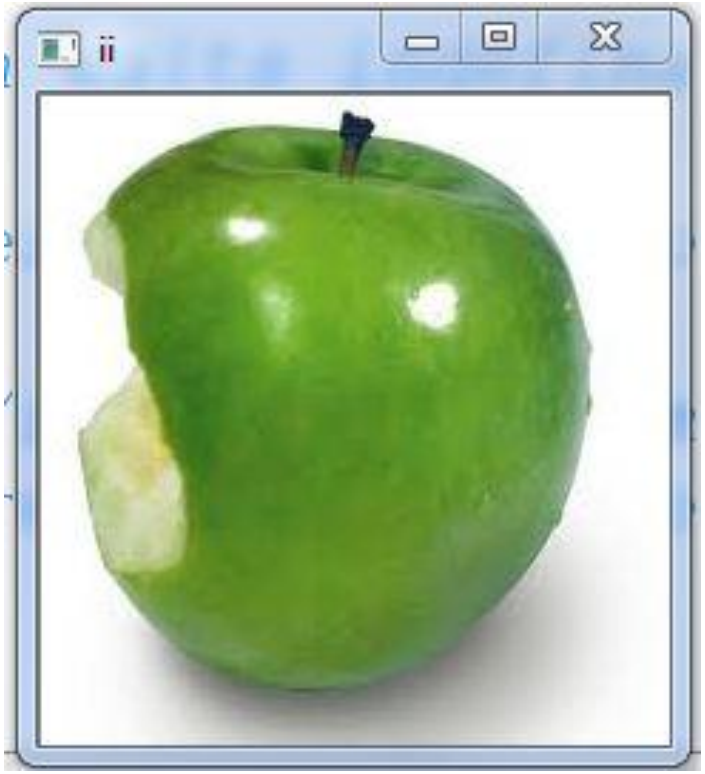
Creates a window named ii

1 = Coloured 0 = Grayscale



cvShowImage("ii",input)

Shows image pointed by input , in the window named ii



Create an Image

To create an image you need to specify its :-

- Size (height and width)
- Depth
- Number of Channels

output=cvCreateImage(cvGetSize(input),IPL_DEPTH_8U,3)

cvWaitKey(*a number*)

- If 0 or negative number is given as input:-

  Waits indefinitely till key press and returns the ASCII value of the key pressed

- If positive number is given as input :-

  Waits for corresponding milliseconds.

Command	Function
cvDestroyWindow("ii")	Destroys window named *ii*
cvReleaseImage(&input)	Releases image pointer *input* from memory
output=cvCloneImage(input)	Copies image from input to output
cvCvtColor(input, output, conversion type) Conv. type : CV_BGR2GRAY ,CV_BGR2HSV	Saves input image in output pointer in other color space
cvSaveImage("output.jpg",output)	Saves image pointed by output naming it output
cvDilate(input , output, NULL, iterations)	Dilates an image for given number of iterations and saves it in output
cvErode(input,erode,NULL,iterations); Note : here NULL is a structural element	Erodes an image for given number of iterations and saves it in output

cvThreshold(input, output, threshold, maxValue, thresholdType)

Threshhold types:-

- CV_THRESH_BINARY

  max value if more than threshold, else 0

- CV_THRESH_BINARY_INV

  0 if more than threshold , else max value

- CV_THRESH_TRUNC

  threshhold if more than threshold , else no change

- CV_THRESH_TOZERO

  no change if more than threshold else 0

- CV_THRESH_TOZERO_INV

  0 if morethan threshold , else no change

imageData

An image's data is stored as a character array whose first element is pointed by :-

Input->imageData (char pointer)

widthStep

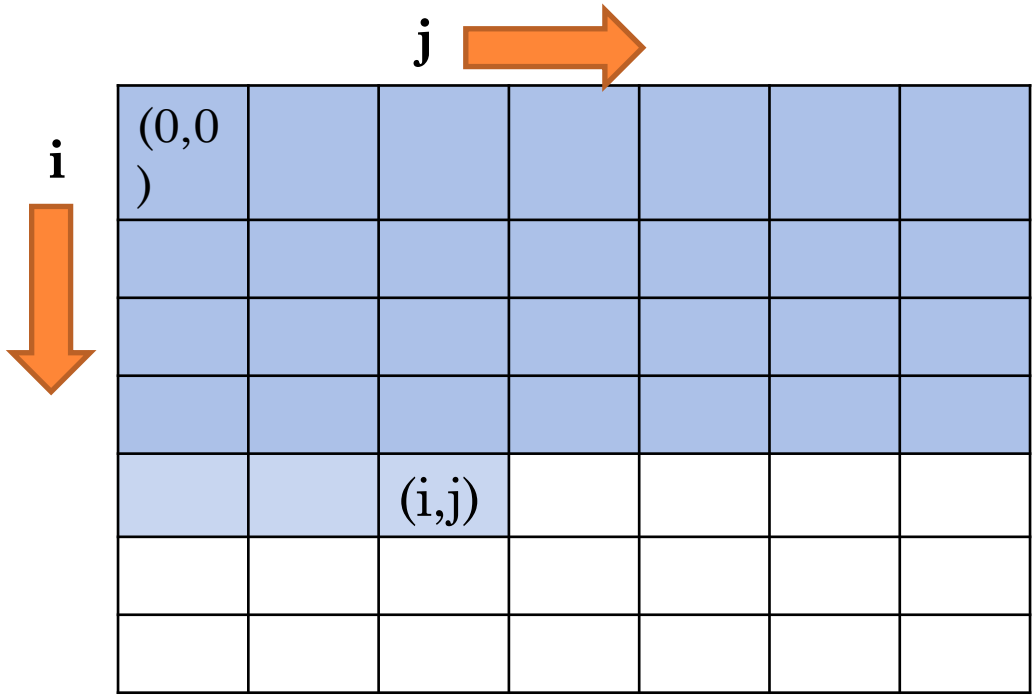
Number of array elements in 1 row is stored in :-

input->widthStep

Accessing (i,j) pixel of an image

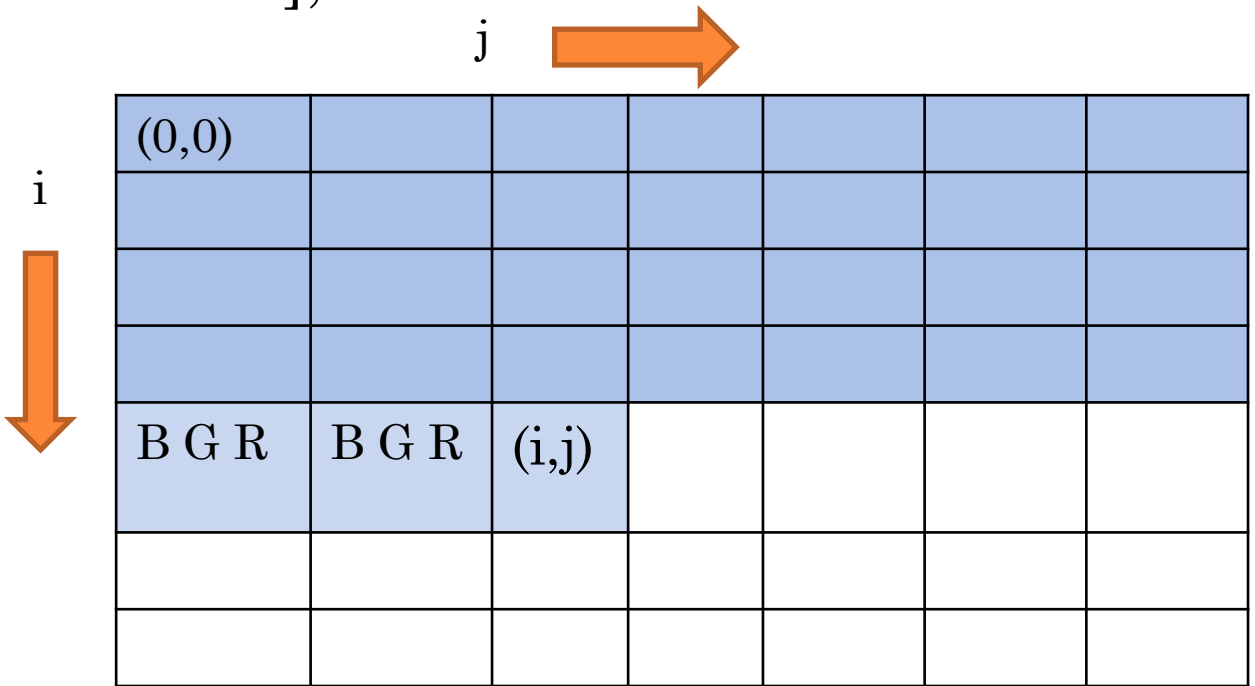
- Grayscale

```
uchar *pinput  =  (uchar*)input->imageData;
int c = pinput[i*input->widthStep + j];
```



- 3 channel image (BGR):-

```
uchar *pinput   =  (uchar*)input->imageData;
int b= pinput[i*input->widthStep + j*input->nChannels+0];
int  g= pinput[i*input->widthStep + j*input->nChannels+1];
int  r= pinput[i*input->widthStep + j*input->nChannels+2];
```



VIDEO POINTER

CvCapture capture* - is a video pointer.

- To take video from camera :-

```
CvCapture
*capture=cvCreateCameraCapture(0);
```

Note : Here 0 - Default & 1 - External

- To take video from a saved video file :-

```
CvCapture*
capture=cvCreateFileCapture("trial.avi" );
```

Taking image from Camera

```
 CvCapture
*capture=cvCreateCameraCapture(0);
 for(int i=0;i<100000000;i++);
 if(capture!=NULL)
                  IplImage
*frame=cvQueryFrame(capture);
```

Note : Here for loop is used to compensate time of initialization of camera in Windows

PLAYING VIDEO

```
CvCapture *capture=cvCreateCameraCapture(0);
 IplImage *frame;
 if(capture!=NULL){
                        frame=cvQueryFrame(capture );
                            while(1){
                        cvShowImage("Video",frame);
                        frame=cvQueryFrame(capture);
                        c=cvWaitKey(1);// frame rate
                        if(c>0&&c<255)
                                break;
                            }
                         }
```

Mouse Pointer Information

```
void my_mouse_callback( int event, int x, int y, int flags, void*
param ){
        uchar *pimage   =  (uchar*)image->imageData;
        int r=pimage[y*image->widthStep + x*image-
>nChannels+2];
        int g=pimage[y*image->widthStep + x*image-
>nChannels+1];
        int b=pimage[y*image->widthStep + x*image-
>nChannels+0];
         printf( " x=%d y=%d r=%d  g=%d  b=%d\n",x,,y,,r,g,b);
}
main(){ ……..
        cvNamedWindow("image",1);
        cvSetMouseCallback("image", my_mouse_callback, NULL);
        cvShowImage("image",image);             }
```

Note : cvSetMouseCallback is set for a NamedWindow and not for an image

IP Problem Statements

In general , all IP problem Statements have to discard one color and accept another in output image .

Input Image → Output Binary Image

```
If( color pixel value > threshhold)
        output pixel=255;
else
        output pixel =0;
```

Note : In general , HSV format is highly useful to distinguish RGB colors (Why ?)

Questions