Why Python ?

Basic Python

- Spaces prettier than curly Braces. Also no semicolons !
- Variables can be declared without mention type
- Arrays*, Matrices and Vectors are easy to manipulate
- Intuitive syntax
- Community Support and Libraries

Work Flow

Basic Python

- Write Script
- Run Script through Command Prompt / Terminal
- Fix Space Indentation (most typical error)
- Re Run Script
- Debug and Optimize
- Repeat

Yo World

Basic Python

```
# Single Line Comments begin with Hashtags


'''
Multi Line Comments are between Triple Quotes
like this. These type of comments are included
at the beginning of each function to serve as
a description of the function
'''

# No semi colons needed after lines

print("Yo World")

print('Yo World')

print("'Yo World'")
```

Variables

Basic Python

```
'''
cant start with numbers
cant use operators (+, -, !, <, >,= etc.)
please use sensible names, ok ?
'''

# int and floats -> try +  -  / * % **
mama = 30
dada = -10.0

# strings - both single/double quotes
Lalpori_CardBERRY_123 = 'Bhiaya add meh'

msg = "Bhara hoilo %d" % mama
```

```
# lists -> like arrays but different -> more later !
cg = [4.0, 3.9, 3, -3, -3.6]

# list of lists
cglist = [ [4, 4, 4], [2, 2, 2] ]

# tuples - for all purposes, similar to lists
coord = (2,-3,1)

# bool
isHigh = False
isCool = True


# manual input (as string)
val = input("What is the value?")
```

Converting Variables Type

Basic Python

```
# all inputs are strings
# but can be converted to any type
age = input("Enter age : ")
age = int(age)

yob = 2018 - age

num = input("Enter number : ")
num = float(num)

sq = num * num
sq = num**2       # ** = to the power
```

print (almost anything)

Basic Python

```
print(Lalpori_CardBerry_123)

print("My Cg")
print(cg)

# normally all prints end with newline ('\n')
# but you can replace that with ANYTHING
print("My Cg : " , end= ' ')
print(cg)

balance = -125.25

print("Balance at 30 Apr 2018 : ", end = '\t')
print(balance)

# alternatively
print("Balance at 30 Apr 2018 : \t %d" % balance)
print("Balance at 30 Apr 2018 : \t %3.1f" % balance)
```

```
# print more than one variable
pa, dha, ni, sa = 1,2,3,4
msg = "yoyo"

print("%s %d %d %d %d" % (msg, ni, dha, pa+sa, sa))
```

Lists

Basic Python

```python
# declaring numeric lists
A = [1,2,3,4,5]

# using range function to generate list
B = range(7,8)

# Counting backwards, inc/dec can be any *integer*
C = range(5,0,-1)

# list of 0,1,2 repeated 10 times
D = [0,1,2]*10

# functions associated with lists
aLength = len(A)

aMax = max(A)

aMin = min(A)

aSum = sum(A)
```

```python
# returning sorted array
aSorted = sorted(A)

# reversing order of list
A.reverse()

# adding entries to end of list
A.append(6)

# adding entries from another list
A.extend(B)

# getting index of particular entry in list
idx = A.index(10)

# removing entry from list
a = A.pop(idx)

# inserting entry into list
A.insert(20,idx)
```

Lists Can Be Mixed Data

Basic Python

```
F = ['BUET', 1206, '+029862344', (23.4,95.3), True]
G = ['DU',   1000, '+028831461', (27.2,92.3), True]

# list of two lists
unis = [F,G]

# appending to empty list
pubUni = []
pubUni.append(F)
pubUni.append(G)

# extending lists
pubUni = []
pubUni.extend(F)
pubUni.extend(G)
```

Indexing Lists and Tuples

Basic Python

```
# index from start
F[0]
F[1]

# index from end
F[-1]
F[-2]

# slicing
s1 = F[0:3] # entries 0,1 and 2

s2 = F[1:4] # entries 1,2 and 3

s3 = F[1:]  # entries from 1 to end

s4 = F[:4]  # entries from start to 3
```

Conditionals

Basic Python

```
# can use any type of logical expression as condn
if 2+2 == 4:
  print("Reality")
else:
  print("Dream")

# nested ifs
a, b, c = 10, 0, 2

if a and b :
  print("And = True")

elif a or b:
  print("Or = True")

  if a >= b:
    print ("Greater or Equal")

else:
  pass
```

```
# one liners
dream = True if 2+2 != 4 else False

div = a/b if b else "infinity“


# is operator
if b is 0:
  print ("Divide by Zero")

# in operator
if a in [1,5,10,15]:
  print ("a in list")

# not operator (invert)
if not False:
  print ("It is True")
```

while Loops

Basic Python

```
i = 100

while i>2 :
  print(i)
  i /= 2 # divide by and assign


# determining prime
i, num = 3, 59

while i <= num/2 and num % i:
  i+=2

if num % i:
  print("%d is prime" % num)
else:
  print("%d is not prime" % num)
```

for Loops

Basic Python

```python
# iterate over a list
# can be list of anything - number, image, audio
aList = ['1', 2, '3', 'Mama', 3.0]

for i in aList :
  print (i , end = ' ')

# iterating over a list of lists
anotherList = [ [1,2,3,4], [-1,-2,-3,-4] ]

for i in anotherList:
  print(i)
```

```python
# nested for
for i in anotherList:

  for j in i:
    print(j*j , end = ' ')

  print("\n")
```

More for Loops

Basic Python

```
# getting both item and index - enumerate

azim = ['L4T2', 'EEE 13', '1306150']
afia = ['L4T2', 'EEE 13', '1306128']
abir = ['L4T1', 'EEE 14', '1406005']

aTeam = [azim, afia, abir]

for idx, value in enumerate(aTeam):
  print (" %d \t %s " % (idx+1, value) )
```

```
# iterating over multiple lists at the same time - zip

names = ['raied',   'rabby',   'prithul', 'messal', 'billah']
ID =    ['1306114', '1306112', '1406111', '1406222','1406333']

batch13 = []
batch14 = []

for i,n in zip(ID, names):

  if i[:2] == '13':
    batch13.append(n)
  elif i[:2] == '14':
    batch14.append(n)

print(batch13)
print(batch14)
```

BUET ROBOTICS SOCIETY
discovering new degrees of freedom

Dictionaries

Basic Python

```
# dictionaries are like lists
# but instead of index, they have keys
# keys can be any type - string, int, float
IDs = { 'shahruk' : 1306119, 'raied': 1306114 }

# entries for each key can be anything too
# can be a int, float, a list, another dict
phoneBook = { 'ananta' : [01720202020, 894232],
              'barsha':  ['basrsha@gmail.com']  }


# To get an entry, index like list but with key
instead of number
print(phoneBook['ananta'])
```

```
# can be used in for loops inplace of list
for entry in phoneBook:
  print(entry)

for entry in phoneBook:
  print( phoneBook[entry] )
```

Importing Libraries

Basic Python

```python
# general way
import nameOfLibrary

# accessing functions / submodules of library
nameOfLibrary.functionName()

# give nickname
import largeOutstandingLibrary as lol

lol.functionName(in1)

# import only specific functions from libraay
from thisLibrary import function1, function2

function1(in1, in2)

# import all functions from library (not recommended)
from thisLibrary import *

function999(in2)
```

```python
# Useful Libraries

import glob
import os
import pickle
import re
import cv2
import PIL
import pytesseract as ocr
import numpy as np
import scipy as sp
import skimage as si

from optparse import OptionParser
from matplotlib import pyplot as plt
```

Functions

Basic Python

```python
# function skeleton
def functionName(in1, in2, in4=True, in5=3.1412)

  '''
  do stuff here with variables and other functions
  and return anything you like : list, multiple
  variables, string etc.
  '''

  return out1, out2, out3
```

```python
import numpy as np

# roots of qudradtic
def root(a,b,c):
  det = b**2 - 4 * a * c

  if det < 0:
    Im = np.sqrt( abs(det) ) / 2
    Re = -b / 2

    x1 = (Re, Im)
    x2 = (Re, -Im)

  else:
    x1 = (-b + np.sqrt(det))/2
    x2 = (-b - np.sqrt(det))/2

  return x1, x2
```

numpy

Basic Python

```python
import numpy as np

aList = [ [1,2,3,4], [5,6,7,8], [9,10,11,12] ]

# numpy arrays from lists
mat = np.array(aList, dtype = np.int8)
print(mat, end = '\n\n')

# .shape returns (nRows, nCols)
r, c = mat.shape
print(r,c, end = '\n\n')

# indexing is similar to list, axis separated by ,
firstRow = mat[0, :]
thirdCol = mat[:, 2]
secondElement = mat[0,2]
coMatrix1 = mat[1:,1:]

print(coMatrix1, end = '\n\n')
```

```python
# you can reshape arrays into different dims
# total num of elements must be equal 4x3 = 12 = 6x2
mat1 = mat.reshape(6,2)
mat2 = mat.reshape(2,6)

# matrix multiplication
mat3 = np.matmul(mat1,mat2,)
print ( mat3 , end = '\n\n')

# transpose
mat4 = np.transpose(mat)
```

numpy

Basic Python

```python
# elementwise multiplication -> use *
mask = np.zeros( mat.shape )
mask[:, 2:] = 1
mat = mat * mask

print (mat, end = '\n\n')
```

```python
# getting 10 random values from
# uniform distribution between 1 and 100
samples = np.random.uniform(1,100,10)
samples = np.random.randint(1,100,10)

print(samples, end = '\n\n')
```

Input

Basic OpenCV

```python
import cv2

# ./ represents current directory
imgPath = './chobi.jpg'

# read image as numpy array
img = cv2.imread(imgPath)

# show image in window named 'Chobi'
cv2.imshow('Chobi', img)

# display image for 5 seconds
cv2.waitKey(5000)

# close all windows
cv2.destroyAllWindows()
```

```python
# vid is now like a film reel of the video
vid = cv2.VideoCapture('./lfr.mp4')

# vid.read() tries to read a frame from the reel
success, frame = vid.read()

while(success):
cv2.imshow('Video', frame)
cv2.waitKey(100)

success,frame = vid.read()

cv2.destroyAllWindows()
```

Output

Basic OpenCV

```python
import cv2

# write image to disk
cv2.imwrite('./chobi2.jpg',img)

# compress
encodeParam = [cv2.IMWRITE_JPEG_QUALITY, 20]
cv2.imwrite('./chobi2.jpg',img,encodeParam)

# ------------------------------------------------

x, y, fps = 640, 480, 30

encoder = cv2.VideoWriter('./output.avi', -1, fps, (x,y))

vid = cv2.VideoCapture(0)

success, frame = vid.read()
```

```python
while success:
  # write frame to disk
  encoder.write(frame)
  success, frame = vid.read()

  cv2.imshow('Capture', frame)
  key = cv2.waitKey(30)

  # if pressed key is q, break
  if key == ord('q'):
    cv2.destroyAllWindows()
    vid.release()
    encoder.release()
    break
```

Drawing & Writing

Basic OpenCV

```python
import cv2

# drawing circles
cv2.circle(img, (xC,yC), radius, (b,g,r), thickness )

# drawing rectangles
cv2.rectange(img, (TLx, TLy), (BRx, BRy), (b,g,r), thickness)

# writing text
font = cv2.FONT_HERSHEY_SIMPLEX
fontScale = 2
cv2.putText(img, text, (x,y), font, fontScale, (255,255,255), 2)
```

Image Array

Basic OpenCV

- Unsigned 8-bit
- Numpy Array
- 0 – 255 values
- 3rd Dim is Channel (Red, Blue, Green)

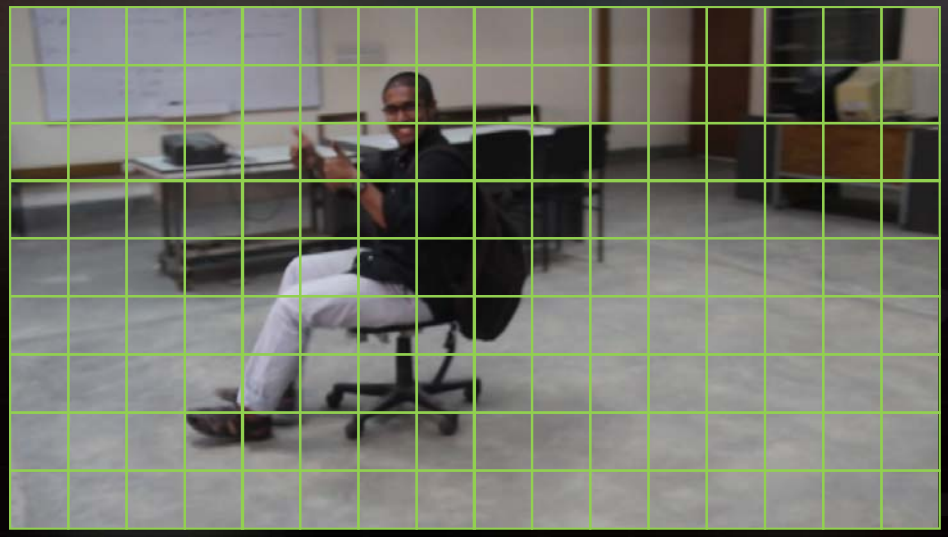


Image Array

Basic OpenCV

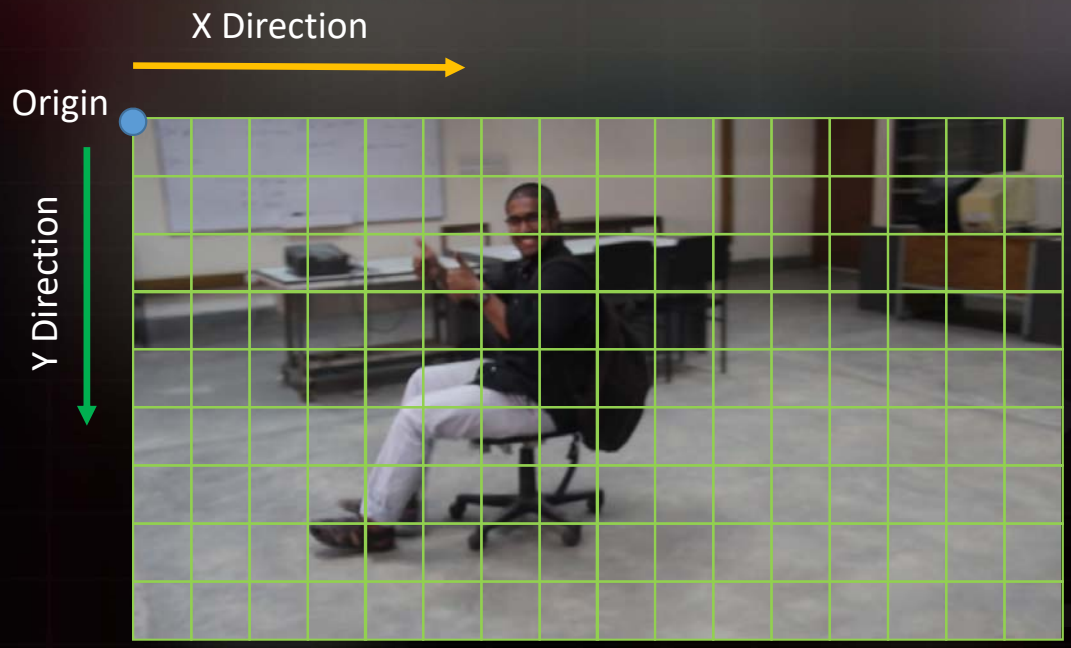


Image Array

Basic OpenCV

2 : Red

1 : Green

0 : Blue

Image Stitching

Basic OpenCV

```
import cv2

# horizontal stack
imgStack = np.hstack( (img1,img2,img3))

# vertical stack
imgStack = np.vstack( (img1,img2,img3))

# depth stack
imgStack = np.dstack( (img1,img2,img3))
```

Image Transformations – Crop, Scale, Flip

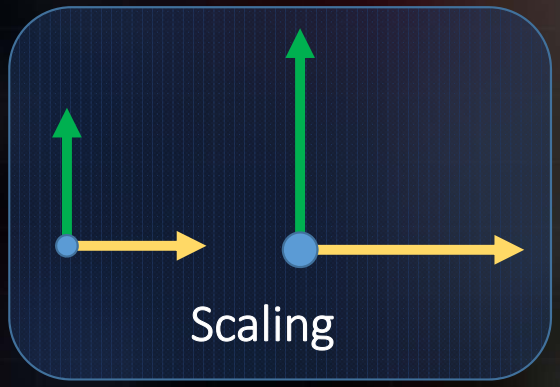
Basic OpenCV

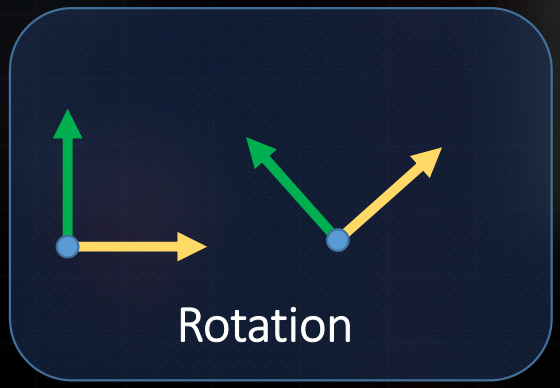
```python
import cv2

img = cv2.imread('./chobi.jpg')

# y = height, x = width, ch = no. of channels
y, x, ch = img.shape
print( x, y, ch)

# cropping across all 3 channels
cropped = img[100:300, 725:875, : ]
cv2.imshow('Cropped',cropped)
cv2.waitKey(0)

# resizing by dimension
xNew, yNew = 400, 400
resized = cv2.resize(cropped, (xNew,yNew) )
cv2.imshow('Resized',resized)
cv2.waitKey(0)
```

```python
# resizing by scale
xRatio, yRatio = 3, 0.5
resized = cv2.resize(cropped, None, fx=xRatio, fy=yRatio)
cv2.imshow('Scaled',resized)
cv2.waitKey(0)

# flip vertically
flipped = cv2.flip(cropped, 0)
cv2.imshow('V-Flip', flipped)
cv2.waitKey(0)

# flip horizontally
flipped = cv2.flip(cropped, 1)
cv2.imshow('H-Flip',flipped)
cv2.waitKey(0)
```

Image Transformations - Matrix

Basic OpenCV





$$M = \begin{pmatrix} \underbrace{1 & 0}_{\text{Rotation \& Scale}} & \underbrace{T_x}_{\text{Translation}} \\ 0 & 1 & T_y \end{pmatrix}$$

Affine Transformation

Image Transformations - Rotation

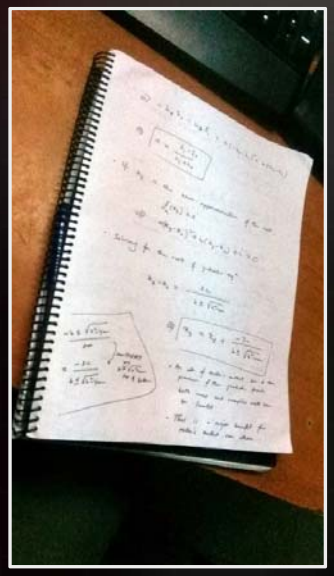
Basic OpenCV

```python
import cv2

# a simple function to display image
def show(img, delay = 0, name = 'Image'):

  cv2.imshow(name, img)
  key = cv2.waitKey(delay)

  return key
```

```python
# the 0 tells cv2 load image as grayscale
img = cv2.imread('./chobi.jpg', 0)
y,x = img.shape

# getting center of image (floor division)
xC, yC = x//2, y//2
scale = 1
degrees = 45

# rotation
M = cv2.getRotationMatrix2D( (xC, yC), degrees, scale)
rotated = cv2.warpAffine( img, M, (img.shape[:2]) )
show(rotated)
```

Image Transformations - Perspective

Basic OpenCV

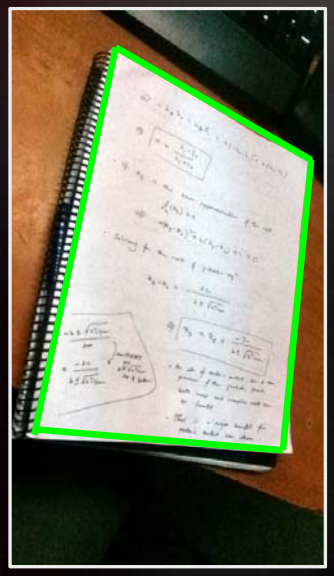


Chotha Scanner

Free

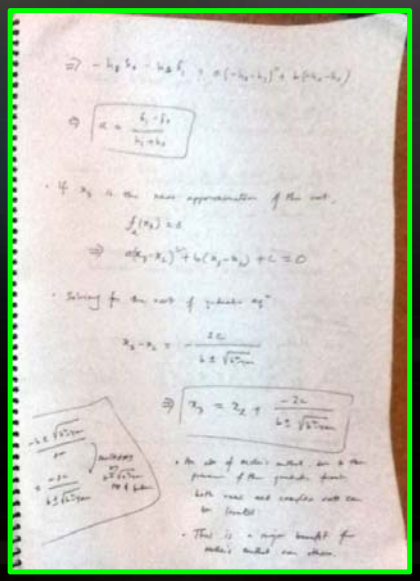
Image Transformations - Perspective

Basic OpenCV



Chotha Scanner
Free

cv2.warpPerspective

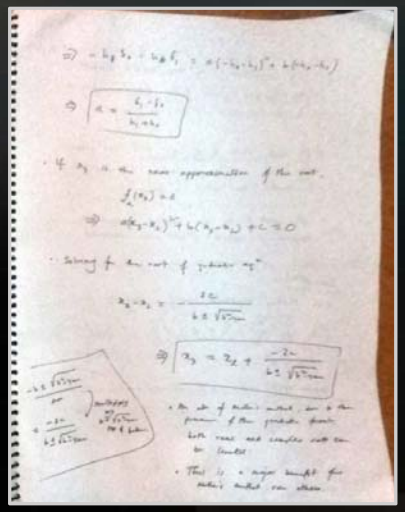


Chotha Scanner
Pro

Image Manipulations - Thresholding

Basic OpenCV

- Set Threshold Pixel Intensity
- Values **above** Threshold boosted to 255
- Values **below** Threshold set to 0
- Done on Grayscale Images (single channel)



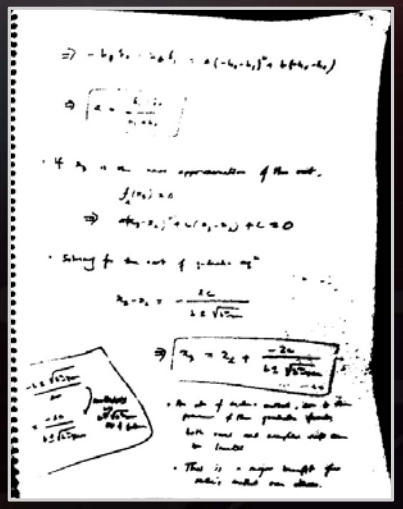


Image Manipulations - Thresholding

Basic OpenCV

```python
# simple global thresholding

img = cv2.imread('./chotha2.jpg',0)

thLevel = 200
maxVal = 255

ret, th = cv2.threshold(img, thLevel, maxVal,
                        cv2.THRESH_BINARY)

print(ret)
show(th)
```

```python
# using Otsu's Alg to decide threshold (bimodal)
ret, th = cv2.threshold(img, thLevel, maxVal,
                        cv2.THRESH_OTSU)
print(ret)
show(th)


# using Adaptive Thresholding (Local Thresholding)

# window size must be odd
ws = 17

th = cv2.adaptiveThreshold(img, maxVal,
                           cv2.ADAPTIVE_THRESH_MEAN_C,
                           cv2.THRESH_BINARY, ws, 2)

show(th)
```

Image Manipulations - Denoising

Basic OpenCV

```python
img = cv2.imread('./noisy.jpg',0)
y, x = img.shape
y, x = y//2, x//2

# making size manageable
img = cv2.resize(img, ( x, y ) )

# Fast Non Local Means Denoising Alg.
h, ws, ts = 8, 21, 7
img2 = cv2.fastNlMeansDenoising(img, None, h, ws, ts)
show(img2, name = 'Denoised')

# Median Blur ; other blurs also available
img3 = cv2.medianBlur(img2,3)
show(img3, name = 'Blurred')

# Thresholding
_, img4 = cv2.threshold(img3,137,255,cv2.THRESH_BINARY)
show(img4, name = 'Thresholded')
```

```python
cv2.destroyAllWindows()

# Need to translate left
M = np.float32( [ [1, 0,-20],
                  [0, 1, 0] ] )

img5 = cv2.warpAffine(img4,M,(x,y))
show(img5, name = "Translated")

# Need to Rotate by about 40 degrees CCW
M = cv2.getRotationMatrix2D( (x//2,y//2), -40, 1 )
img5 = cv2.warpAffine(img5,M,(x,y))
show(img5, name = "Rotated")
```

Image Manipulations - Morphing

Basic OpenCV

```
# Getting rid of spots (by eroding) & gaps (by dilating)
img5 = cv2.erode(img5,None,iterations=2)
show(img5, name = "Eroded")

img5 = cv2.dilate(img5,None,iterations=3)
show(img5, name = "Dilated")

img5 = cv2.erode(img5,None,iterations=2)
show(img5, name = "Eroded2")

img5 = cv2.dilate(img5,None,iterations=3)
show(img5, name = "Dilated2")
```

```
# erosion and dilation may use
# specially constructed kernels
# kernels are simply a weighting matrix
k = cv2.MORPH_RECT
k = cv2.MORPH_ELLIPSE
k = cv2.MORPH_CROSS

ws = 5
kernel = cv2.getStructuringElement(k, (ws,ws))

n = 2
cv2.erode( img, kernel, iterations=n)
cv2.dilate( img, kernel, iterations=n)
```

Image Manipulations - Morphing

Basic OpenCV

- cv2.morphologyEX( ) does a combination of erode and dilate
- Different combinations possible
- Try it out with different kernels

```
img = cv2.imread('./ko.png',0)

opt = [cv2.MORPH_CLOSE,
       cv2.MORPH_OPEN,
       cv2.MORPH_GRADIENT]

kern = [ cv2.MORPH_RECT,
         cv2.MORPH_ELLIPSE,
         cv2.MORPH_CROSS ]

kernel = cv2.getStructuringElement(kern[0], (5,5))

img1 = cv2.morphologyEx(img, opt[2], kernel )
show(img1)
```

Optical Character Recognition : OCR

Basic OpenCV

- Open Source Library : Tesseract
- Uses Neural Networks
- Can be trained for different languages
- Downside : Slow for large Images, Sensitive
- Many Optimizations possible, but that's for another time

Optical Character Recognition : OCR

Basic OpenCV

- image_to_string( ) needs PIL Image format
- Language can be changed to Bangla by setting **lang = 'ben'**
- Explore other parameters !
- Preprocess bad images

```
# OCR - that simple !
import pytesseract as ocr
from PIL import Image

img = Image.fromarray(img)
txt = ocr.image_to_string(img, lang='eng')
print(txt)
```

Color Detection

Basic OpenCV

```
img = cv2.imread('./ball.jpg')
show(img)

# Min-Max Color threshold (BGR)
# for object, in this case ball
lower = np.array([17,200,200])
upper = np.array([117,255,255])

# filters pixels according to threshold
mask = cv2.inRange(img,lower,upper)
show(mask, name = 'mask')

# applying mask onto image
out = cv2.bitwise_and(img,img,mask=mask)
show(out, name = 'filtered')
```

```
# doing the same thing to a video
vid = cv2.VideoCapture('./ball2.mp4')

s, f = vid.read()

while s:
  lower = np.array([17,200,200])
  upper = np.array([117,255,255])

  mask = cv2.inRange(f,lower,upper)
  out = cv2.bitwise_and(f, f, mask=mask)
  s,f = vid.read()
  show(out,33)

# try applying erode / dilate to mask
# to make it cleaner
```

BUET ROBOTICS SOCIETY
discovering new degrees of freedom

Finding Contours

Basic OpenCV

```python
# finding contour of mask
# retreive mode = external, returns largest outer contour
# contour aprox mode = CHAIN_APPROX SIMPLE -> removes points in between st lines
cnts = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# see more about retrieval mode at
# https://docs.opencv.org/3.4.0/d9/d8b/tutorial_py_contours_hierarchy.html

# cv2.findContours returns image, contours, and hierarchy
# we only need contours for now
cnts = cnts[1]

# getting largest contours from list of contours
c = max(cnts, key=cv2.contourArea)

# drawing contour
((x, y), r) = cv2.minEnclosingCircle(c)
x,y,r = int(x), int(y), int(r)
cv2.circle(img, (x, y), r, (0, 0, 255), 2)
show(img)
```

Calculating Moments & Centroid

Basic OpenCV

```
# finding momemnts
M = cv2.moments(c)

# centroid of contour (x,y)
x,y = M["m10"] / M["m00"], M["m01"] / M["m00"]
x,y = int(x), int(y)
cv2.circle(img, ( x, y ), 5, (255, 0, 0), -1)

show(img)
```

Tracking Movement

Basic OpenCV

- Tracking an object between frames can be done many ways
- Simplest and most naïve approach : keep track of **Δx** and **Δy** and **average**
- Adding an extra tracking mechanism with detecting overcomes problem of **occlusion** and **failure to detect object**

- Advanced techniques :
  - Kalman Filter
  - Extended Kalman Filter
  - Gauss-Newton Filter
  - Particle Filter
  - Lucas-Kanade Method
  - Markov Models
  - Neural Networks

Following Object

Basic OpenCV

- Need a control loop such as PID
- Target : keep object in the center
- Actuate Motors to achieve target
- Use some tracking method to smooth delta values

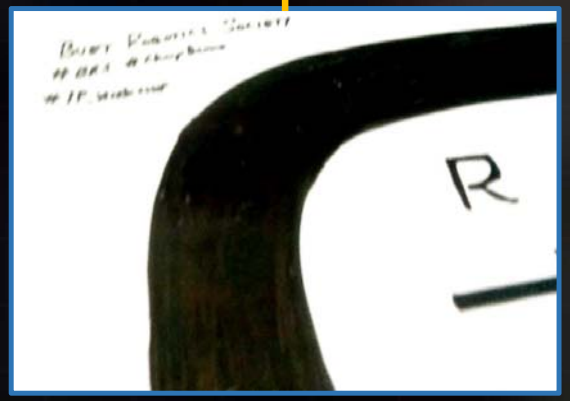
- Example Scenarios:
  - Line Follower
  - Ball Follower
  - Selfie Drones
  - Autonomous Vehicles

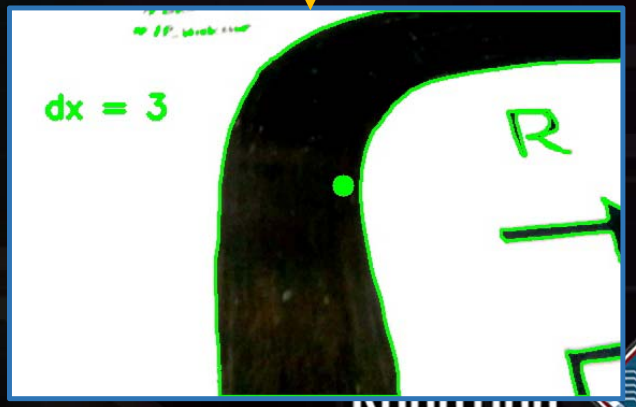
Line Tracking Example

Basic OpenCV

Threshold + Invert

Find Contour and Centroid





Line Tracking Example

Basic OpenCV

```
vid = cv2.VideoCapture('./lfr.mp4')
s,f = vid.read()
h,w, _ = f.shape

# keeping track of centroid cood
x0, y0 = 0, 0

while s:

  f = cv2.resize(f, (w//2,h//2))
  # converting frame to gray scale
  fg = cv2.cvtColor(f,cv2.COLOR_BGR2GRAY)
  ret, th = cv2.threshold(fg, 200, 255, cv2.THRESH_OTSU)

  # inverting to make track white
  th = cv2.bitwise_not(th)

  # finding contours like in color detection
  cnts = cv2.findContours(th, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
  cnts = cnts[1]
```

Line Tracking Example (continued)

Basic OpenCV

```
# this part is inside the while loop
# if contour found
if len(cnts) > 0:
  c = max(cnts, key=cv2.contourArea)

  # finding centroid
  m = cv2.moments(c)
  x,y = m['m10']/m['m00'], m['m01']/m['m00']
  x,y = int(x), int(y)

  cv2.circle(f, (x,y), 10, (0,255,0),-1)
  cv2.drawContours(f,cnts,-1, (0,255,0),2 )
  dx = x-x0
  dy = y-y0
  x0, y0 = x,y

  cv2.putText(f, "dx = %d"%dx, (100,100),
              cv2.FONT_HERSHEY_SIMPLEX, 1, (0,255,0), 4)

  # stitching for pretty preview
  th = np.dstack((th,th,th))
  output = np.hstack((th,f))
  show(output,33)

s,f = vid.read()
```

Shape Detection

Basic OpenCV

- The function approximates the shape of contour
- Uses small line segments to traverse contour
- No. of line segments = No. of Sides => Shape
- Utilizes Douglas-Peucker (DP) algorithm

```
def detectShape (contour):
  shape = 'unidentified'

  # finding length of countour
  perimeter = cv2.arcLength(contour, True)

  # polygonal approximation of contour
  polyApprox = cv2.approxPolyDP ( contour, 0.1*perimeter, True)

  # if 3 lines in polygon, triangle
  if( len(polyApprox) == 3):
    shape = 'triangle'
  elif( len(polyApprox) == 4):
    shape = 'rectangle'
  elif( len(polyApprox)>10):
    shape = 'circle'

  return shape, len(polyApprox)
```

Shape Detection

Basic OpenCV

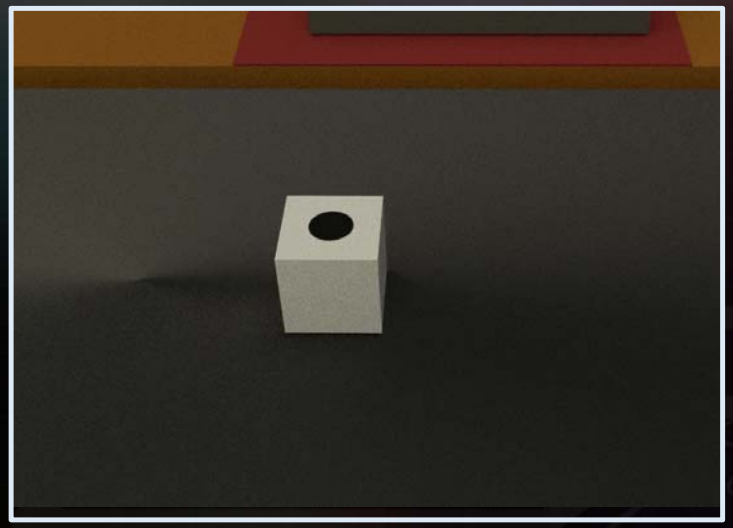
```
img = cv2.imread('./symbols2.jpg')
y,x,ch = img.shape

# resized and converted to grayscale
img2 = cv2.resize(img, (x//2,y//2))
img3 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)

# blur to smooth out
blurred = cv2.GaussianBlur(img3, (5,5), 0)

# threshold
ret, thresh = cv2.threshold(blurred,60,255,cv2.THRESH_OTSU)

# erode
eroded = cv2.erode(thresh, None, iterations=3)

show(eroded, name='eroded')

_ , cnts, hier = cv2.findContours(eroded, cv2.RETR_LIST,
                                  cv2.CHAIN_APPROX_SIMPLE )
```

```
for idx,c in enumerate(cnts):

  shape, verts = detectShape(c)

  if shape =='unidentified':
    continue

  cv2.drawContours(img2, [c], -1, (0,255,0), 2)

  print (shape, '\t(verts=',verts,')\n')

  show(img2)
```

Shape Detection – IAC

Basic OpenCV

- Same Deal as before
- Blur, Threshold to isolate
- Get contours
- Find logic to choose correct Contour
- Approximate Shape of Contour (may need to tune approximation params.)
- Get Shape !



Shape Detection – IAC

Basic OpenCV

```
img = cv2.imread('./symbols2.jpg')
y,x,ch = img.shape

# resized and converted to grayscale
img2 = cv2.resize(img, (x//2,y//2))
img3 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)

# blur to smooth out
blurred = cv2.GaussianBlur(img3, (5,5), 0)

# threshold
ret, thresh = cv2.threshold(blurred,60,255,cv2.THRESH_OTSU)

# erode
eroded = cv2.erode(thresh, None, iterations=3)

show(eroded, name='eroded')

_ , cnts, hier = cv2.findContours(eroded, cv2.RETR_LIST,
                                  cv2.CHAIN_APPROX_SIMPLE )
```

```
for idx,c in enumerate(cnts):

  shape, verts = detectShape(c)

  if shape =='unidentified':
    continue

  cv2.drawContours(img2, [c], -1, (0,255,0), 2)

  print (shape, '\t(verts=',verts,')\n')

  show(img2)
```

Basic OpenCV

Object Recognition using CNN

- CNN = convolutional neural networks
- Pre-trained models available
- These models have been taught by showing them thousands of images
- If you have the resources, may train your own
- You can get models from :
  - Caffe Zoo
  - ImageNet Models
    - VGGnet
    - ResNet
    - Inception

- See cnn.py for an example
- Easy to use models once they are trained !
- Models generalize well