**Mini-project-HPC**

**Object recognition using open cv in real time**

import cv2

import numpy as np

MIN\_MATCH\_COUNT=30

detector=cv2.SIFT()

FLANN\_INDEX\_KDITREE=0

flannParam=dict(algorithm=FLANN\_INDEX\_KDITREE,tree=5)

flann=cv2.FlannBasedMatcher(flannParam,{})

trainImg=cv2.imread("TrainingData/TrainImg.jpg",0)

trainKP,trainDesc=detector.detectAndCompute(trainImg,None)

cam=cv2.VideoCapture(0)

while True:

ret, QueryImgBGR=cam.read()

QueryImg=cv2.cvtColor(QueryImgBGR,cv2.COLOR\_BGR2GRAY)

queryKP,queryDesc=detector.detectAndCompute(QueryImg,None)

matches=flann.knnMatch(queryDesc,trainDesc,k=2)

goodMatch=[]

for m,n in matches:

if(m.distance<0.75\*n.distance):

goodMatch.append(m)

if(len(goodMatch)>MIN\_MATCH\_COUNT):

tp=[]

qp=[]

for m in goodMatch:

tp.append(trainKP[m.trainIdx].pt)

qp.append(queryKP[m.queryIdx].pt)

tp,qp=np.float32((tp,qp))

H,status=cv2.findHomography(tp,qp,cv2.RANSAC,3.0)

h,w=trainImg.shape

trainBorder=np.float32([[[0,0],[0,h-1],[w-1,h-1],[w-1,0]]])

queryBorder=cv2.perspectiveTransform(trainBorder,H)

cv2.polylines(QueryImgBGR,[np.int32(queryBorder)],True,(0,255,0),5)

else:

print "Not Enough match found- %d/%d"%(len(goodMatch),MIN\_MATCH\_COUNT)

cv2.imshow('result',QueryImgBGR)

if cv2.waitKey(10)==ord('q'):

break

cam.release()

cv2.destroyAllWindows()

**OUTPUT🡪**