CIS 666 Artificial Intelligence

Cleveland State University

Department of Electrical Engineering and Computer Science

Project 1

Name: sravani dodla

Csuid: 2725446

Spring 2019

1. Use Harris corner detection technique to find the key points of the enclosed image.

• Display the original image with all detected points on the image.

**Code for Harris corner detection:**

import cv2

import numpy as np

filename = 'im.jpg'

img = cv2.imread(filename)

gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

gray = np.float32(gray)

corners = cv2.goodFeaturesToTrack(gray, 255, 0.04, 12)

corners = np.int0(corners)

for corner in corners:

x, y = corner.ravel()

cv2.circle(img, (x,y), 3 ,255, -1)

cv2.imwrite('result1.jpg',img)

output:



• Display the original image with the strongest 30 points on the image.

import cv2

import numpy as np

filename = 'im.jpg'

img = cv2.imread(filename)

gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

gray = np.float32(gray)

corners = cv2.goodFeaturesToTrack(gray, 30, 0.04, 12)

corners = np.int0(corners)

for corner in corners:

x, y = corner.ravel()

cv2.circle(img, (x,y), 3 ,255, -1)

cv2.imwrite('result1\_1.jpg',img)

Output:



1. Write a program to perform K-means clustering technique using the points of part 1.

• Display the original image with the detected clusters on it (use different colors).

import cv2

import numpy as np

filename = 'im.jpg'

img = cv2.imread(filename)

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

gray = np.float32(gray)

dst = cv2.cornerHarris(gray, 2, 3, 0.06)

# Threshold for an optimal value, it may vary depending on the image.

img[dst > 0.01 \* dst.max()] = [0, 120, 255]

X, Y = np.nonzero(dst > 0.01 \* dst.max())

points = np.array([[x, y] for x, y in zip(X, Y)], dtype=np.float32)

# define criteria and apply kmeans()

criteria = (cv2.TERM\_CRITERIA\_EPS + cv2.TERM\_CRITERIA\_MAX\_ITER, 10, 1.0)

ret, label, center = cv2.kmeans(points, 3, None, criteria, 10, cv2.KMEANS\_RANDOM\_CENTERS)

# Now separate the data, Note the flatten()

A = points[label.ravel() == 0]

B = points[label.ravel() == 1]

C = points[label.ravel() == 2]

for p in A:

img[int(p[0]), int(p[1])] = [255, 0, 0]

for p in B:

img[int(p[0]), int(p[1])] = [0, 255, 0]

for p in C:

img[int(p[0]), int(p[1])] = [0, 0, 255]

cv2.imwrite('result2.jpg', img)

if cv2.waitKey(0) & 0xff == 27:

cv2.destroyAllWindows()

Output:



1. Write a program to draw a bounding box for each cluster of the data points of part 2.

• Display the original image with the bounding boxes on it (use different colors).

import cv2

import numpy as np

filename = 'im.jpg'

img = cv2.imread(filename)

gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

gray = np.float32(gray)

dst = cv2.cornerHarris(gray,2,3,0.06)

# Threshold for an optimal value, it may vary depending on the image.

#img[dst>0.01\*dst.max()]=[0,120,255]

X, Y = np.nonzero(dst>0.01\*dst.max())

points=np.array([[x,y]for x,y in zip(X,Y)], dtype=np.float32)

# define criteria and apply kmeans()

criteria = (cv2.TERM\_CRITERIA\_EPS + cv2.TERM\_CRITERIA\_MAX\_ITER, 10, 1.0)

ret,label,center=cv2.kmeans(points,3,None,criteria,10,cv2.KMEANS\_RANDOM\_CENTERS)

# Now separate the data, Note the flatten()

A = points[label.ravel()==0]

B = points[label.ravel()==1]

C = points[label.ravel()==2]

for p in A:

img[int(p[0]), int(p[1])] = [255,0,0]

for p in B:

img[int(p[0]), int(p[1])] = [0,255,0]

for p in C:

img[int(p[0]), int(p[1])] = [0,0,255]

y = np.min(A[:,0])

x = np.min(A[:,1])

h = np.max(A[:,0]) - x

w = np.max(A[:,1]) - y

# draw a red rectangle to visualize the bounding rect

cv2.rectangle(img, (x, y), (x+y, w+h), (255, 0, 0), 2)

y = np.min(B[:,0])

x = np.min(B[:,1])

h = np.max(B[:,1]) - x

w = np.max(B[:,0]) - y

# draw a blue rectangle to visualize the bounding rect

cv2.rectangle(img, (x, y), (x+y, w+h), (0, 255, 0), 2)

y = np.min(C[:,0])

x = np.min(C[:,1])

h = np.max(C[:,1]) - x

w = np.max(C[:,0]) - y

# draw a green rectangle to visualize the bounding rect

cv2.rectangle(img, (x, y), (x+y, w+h), (0, 0, 255), 2)

cv2.imwrite('result3.jpg',img)

if cv2.waitKey(0) & 0xff == 27:

cv2.destroyAllWindows()

Output:



**Another Method**

import os

import cv2

import numpy as np

img = cv2.imread("im.jpg")

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

gray = np.float32(gray)

dst = cv2.cornerHarris(gray, 2, 3, 0.04)

rows = dst.shape[0]

cols = dst.shape[1]

reshape\_dst = np.reshape(dst, dst.shape[0] \* dst.shape[1])

ind = np.argsort(reshape\_dst)

top\_thirty = np.zeros((rows, cols))

for i in range(300):

top\_thirty[int(ind[-i - 1] / (cols)), int(ind[-i - 1] % (cols))] = 1

# Threshold for an optimal value, it may vary depending on the image.

img2 = np.copy(img)

img3 = np.copy(img)

img4 = np.copy(img)

corners = dst > 0.02 \* dst.max()

img[corners] = [0, 0, 255]

img2[top\_thirty > 0] = [0, 0, 255]

x = []

y = []

for row in range(rows):

for col in range(cols):

if corners[row, col] == True:

x.append(col)

y.append(row)

coords = np.column\_stack((x, y))

print(coords.shape)

kmeans = KMeans(n\_clusters=3, random\_state=0).fit(coords)

print(kmeans.labels\_)

col\_clust1 = [2000, 0]

row\_clust1 = [2000, 0]

col\_clust2 = [2000, 0]

row\_clust2 = [2000, 0]

col\_clust3 = [2000, 0]

row\_clust3 = [2000, 0]

for i in range(coords.shape[0]):

if kmeans.labels\_[i] == 0:

img3[coords[i][1], coords[i][0]] = [0, 0, 255]

if coords[i][0] < col\_clust1[0]:

col\_clust1[0] = coords[i][0]

elif coords[i][0] > col\_clust1[1]:

col\_clust1[1] = coords[i][0]

if coords[i][1] < row\_clust1[0]:

row\_clust1[0] = coords[i][1]

elif coords[i][1] > row\_clust1[1]:

row\_clust1[1] = coords[it][1]

elif kmeans.labels\_[i] == 1:

img3[coords[i][1], coords[i][0]] = [0, 255, 0]

if coords[i][0] < col\_clust2[0]:

col\_clust2[0] = coords[i][0]

elif coords[i][0] > col\_clust2[1]:

col\_clust2[1] = coords[i][0]

if coords[i][1] < row\_clust2[0]:

row\_clust2[0] = coords[i][1]

elif coords[i][1] > row\_clust2[1]:

row\_clust2[1] = coords[i][1]

else:

img3[coords[i][1], coords[i][0]] = [255, 0, 0]

if coords[i][0] < col\_clust3[0]:

col\_clust3[0] = coords[i][0]

elif coords[i][0] > col\_clust3[1]:

col\_clust3[1] = coords[i][0]

if coords[i][1] < row\_clust3[0]:

row\_clust3[0] = coords[i][1]

elif coords[i][1] > row\_clust3[1]:

row\_clust3[1] = coords[i][1]

cv2.rectangle(img4, (col\_clust1[0], row\_clust1[0]), (col\_clust1[1], row\_clust1[1]), (0, 0, 255), thickness=2,

lineType=8)

cv2.rectangle(img4, (col\_clust2[0], row\_clust2[0]), (col\_clust2[1], row\_clust2[1]), (0, 255, 0), thickness=2,

lineType=8)

cv2.rectangle(img4, (col\_clust3[0], row\_clust3[0]), (col\_clust3[1], row\_clust3[1]), (255, 0, 0), thickness=2,

lineType=8)

cv2.imshow('harris\_corners\_points.jpg', img)

cv2.imshow('top\_thirty\_features.jpg', img2)

cv2.imshow('kmeans\_clustering', img3)

cv2.imshow('bounding\_boxes', img4)

if cv2.waitKey(0) & 0xff == 27:

cv2.destroyAllWindows()