

1.

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, 0.01, 10)

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)



1\_1.

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, 0.01, 10)

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)



2.

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()



3.

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()

