Lesson 15 Image Processing---Corner Detection

1. Corner Introduction

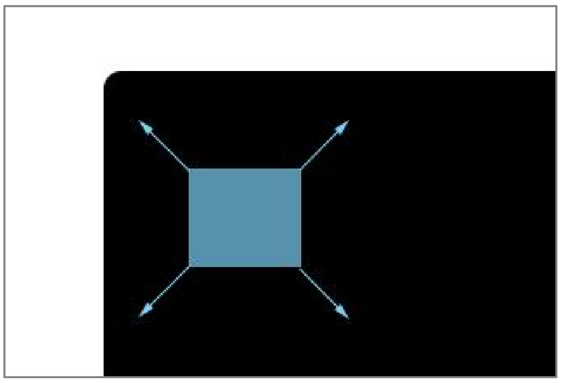
1.1 Corner Definition

The corner is defined as the intersection of two edges, or a feature with two main directions in the neighborhood. In general, corner is the point on the edge curve with maximum curvature, or the point with large variation in intensity in the image.

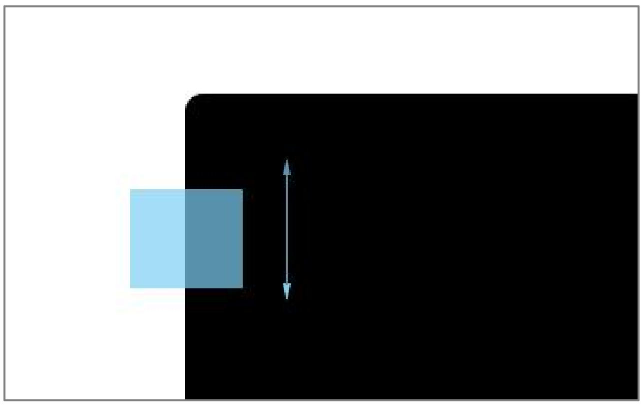
1.2 Detection Idea

Define a tiny local window in the image, then move this window in all directions, which will leads to three results, including flat areas, edges and corners.

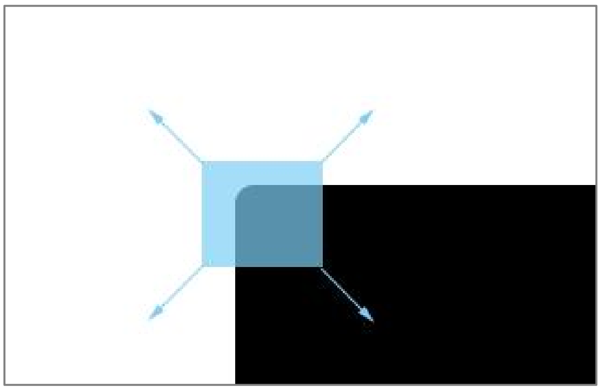
1) If the image intensity of window doesn't change as the window is moving in all directions, inside the window are all flat areas, and corner isn't involved.



2) If the image intensity of window change greatly as the window is moving in one (some) direction(s), but keeps still in other directions, there maybe "edges" inside the window.



3) f the image intensity of window change greatly in all direction, there maybe “corners” inside the window.



1.3 Harris Corner Detection Formula

$$R = \det(M) - k(\operatorname{trace}(M))^2$$

And：

1) **det(M)= λ1λ2**。

2) **trace(M)= λ1+ λ2**。

3) **λ1 and λ2 are the feature values of the moment M.**

The region type can be judged based on these features:

When |R| is small, which happens when λ1 and λ2 are small, the region is flat.

When R<0, which happens when λ1>>λ2 or vice versa, the region is edge.

When R is large, which happens when λ1 and λ2 are large and λ1~ λ2, the region is a corner.

2. Operation Steps

Next, detect the corners of the image through Harris corner detection.

Note：

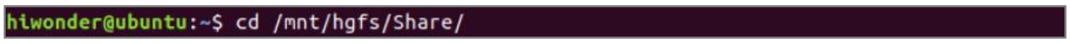
1)Before operation, please copy the routine “corners_demo.py” and sample pictures “test.jpg” in “4.OpenCV->Lesson 15 Image Processing --- Corner Detection->Routine Code” to the shared folder.

2)For how to configure the shared folder, please refer to the file in “2. Linux Basic Lesson->Lesson 3 Linux Installation and Source Replacement”.

3)The input command should be case sensitive and the keywords can be complemented by “Tab” key.

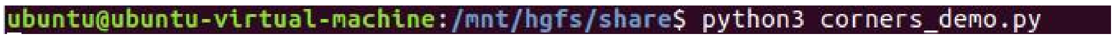
1） Open virtual machine and start the system. Click “⋮⋮⋮”, and then “>_” or press “**Ctrl+Alt+T**” to open command line terminal.

2） Input command “**cd /mnt/hgfs/share/**” and press Enter to enter the shared folder.



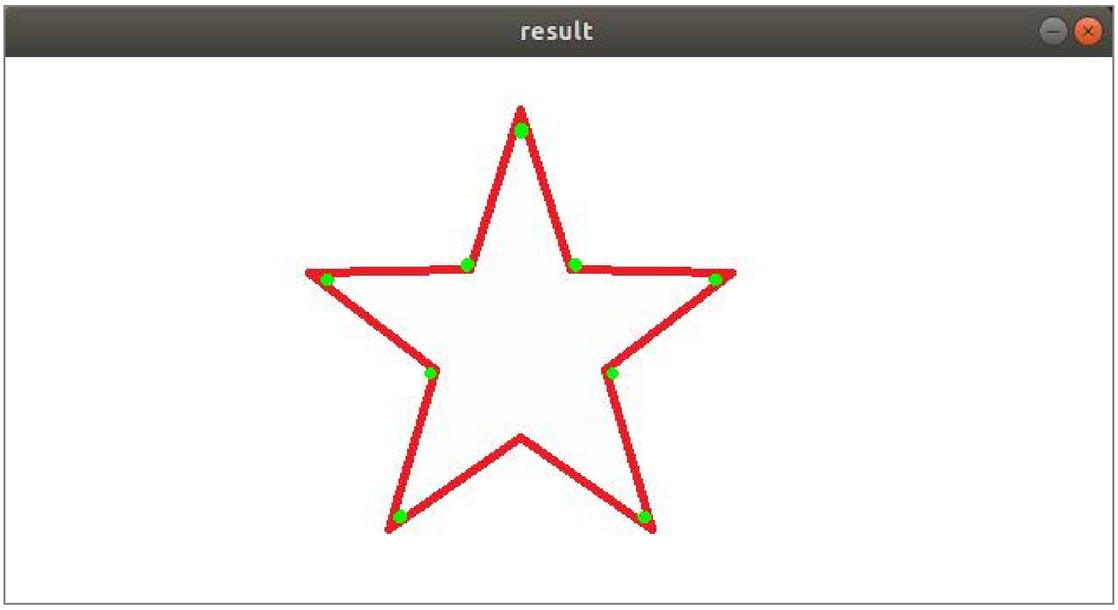
```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/
```

3） Input command “**python3 bf_demo.py**” and press Enter to run the routine.



```
ubuntu@ubuntu-virtual-machine:/mnt/hgfs/share$ python3 corners_demo.py
```

3. Program Outcome



4. Code Analysis

The routine “**corners_demo.py**” can be found in “**4.OpenCV->Lesson 15 Image Processing --- Corner Detection->Routine Code**”

```python
import numpy as np
import cv2 as cv


def harris(image):
    # Detector parameters
    blockSize = 2
    apertureSize = 3
    k = 0.04
    # Detecting corners
    gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
    dst = cv.cornerHarris(gray, blockSize, apertureSize, k)
    # Normalizing
    print(dst)
    dst_norm = np.empty(dst.shape, dtype=np.float32)
    cv.normalize(dst, dst_norm, alpha=0, beta=255, norm_type=cv.NORM_MINMAX)
    # Drawing a circle around corners
    for i in range(dst_norm.shape[0]):
        for j in range(dst_norm.shape[1]):
            if int(dst_norm[i, j]) > 120:
                cv.circle(image, (j, i), 2, (0, 255, 0), 2)
    # output
    return image


src = cv.imread("test.jpg")
result = harris(src)
cv.imshow('result', result)
cv.waitKey(0)
cv.destroyAllWindows()
```

1) **Import Module:** Import cv2 and nmupy module.

```
gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
```

2) **Create corner detection function：harris(image)：the parameter in the bracket is the image that needs to process**

3) **color space conversion：cvtColor(src,mode)**

The first parameter “**src**” refers to the image to convert.

The second parameter “**mode**” is the conversion mode of the color space.

```
dst = cv.cornerHarris(gray, blockSize, apertureSize, k)
```

4) Corner Detection：**cornerHarris(src,blockSize,apertureSize,k)**

The first parameter “**src**” is the image for detection.

The second parameter “**blockSize**” is the size of the domain pixel.

The third parameter “**apertureSize**” is the size of the used window.

The fourth parameter “**k**” is a free parameter ranging [0.04，0.06].

```
dst_norm = np.empty(dst.shape, dtype=np.float32)
```

5) **Acquire the array of the same type：empty(shape,dtype)**

The first parameter “**shape**” represents the shape of the returned array defined by the integer or a tuple of integers.

The second parameter “dtype” refers to the data type defining the type of the returned array.

```
cv.normalize(dst, dst_norm, alpha=0, beta=255, norm_type=cv.NORM_MINMAX)
```

6) **Normalization:** Normalization is to process the data and then limit within the required range. The size of the output image of Harris corner

detection after normalization is the same as the original image. Pixel value of each point corresponds to the probability that the point is corner. The greater the value, the more likely it is a corner.

Function format: normalize(src,dst,alpha,beta,normType)

The first parameter "**src**" is the input array.

The second parameter "**dst**" is the output array after processing.

The third parameter "**alpha**" is the minimum value of normalization

The fourth parameter "**beta**" is the maximum value of normalization

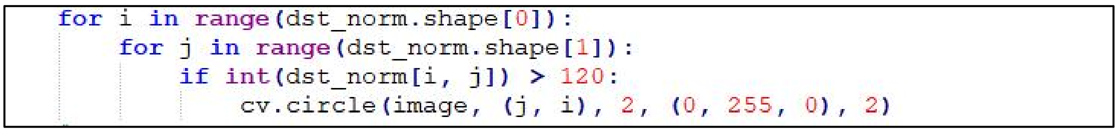
The fifth parameter "**normType**" indicates the types of normalization as below.

1) NORM_MINMAX: The value of the array is translated or scaled to a specified range. Linearly normalized is commonly used.

2) NORM_INF: it is the C-norm of the normalized array (the maximum value of the absolute value)

3) NORM_L1: L1-norm of the normalized array (sum of absolute values)

4) NORM_L2: (Euclidean) L2-norm of the normalized array



```
for i in range(dst_norm.shape[0]):
    for j in range(dst_norm.shape[1]):
        if int(dst_norm[i, j]) > 120:
            cv.circle(image, (j, i), 2, (0, 255, 0), 2)
```

7) Circle the corner： use a loop to traverse the normalized image array and draw a circle in the corner area.

Function format：circle(src,point,radius,color,thickness)

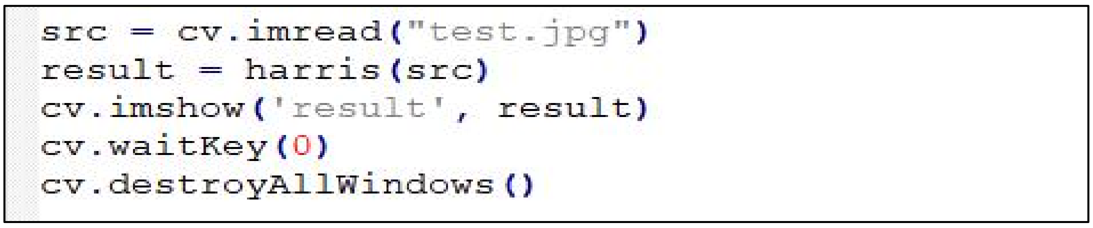
The first parameter "**src**" is the image to be drawn.

The second parameter "**point**" is the center of the drawn circle

The third parameter "**radius**" refers to the radius of the circle

The fourth parameter “**color**” is the set color

The fifth parameter “**thickness**” refers to the line thickness of the circle. When it is negative number, it is solid circle.



```
src = cv.imread("test.jpg")
result = harris(src)
cv.imshow('result', result)
cv.waitKey(0)
cv.destroyAllWindows()
```

8) **Read image, process image, display image and close the window:**

Read image: Use imread(image) function to read the image, and the parameter in the bracket is the name of the picture

Process image： call harris(image) function to process the image and the image parameters will be passed in.

Display image: employ **imshow(title,src)** function to display the output image. The parameters in the bracket are the title of window and the displayed image.

Close window： Adopt waitKey function to wait for the key to be pressed, and then call **destroyAllWindows** function to close the display window.