1.2.7 Perspective Transformation

Perspective transformation is also called projection transformation. The purpose of perspective transformation is to transform objects that are line in reality but may appear as oblique lines on the picture into line by perspective transformation.

Perspective transformation can map rectangles to arbitrary quadrilaterals. We will use this technique when our robot Autopilot lessons.

Perspective transformation function,

dst = cv2. warpPerspective(src, M, dsize[,flag, [,borderMode[,borderValue]]])

- dst: the output picture after perspective transformation, dsize determines the actual size of the output picture .
- src: source image (Original picture)
- M: 3x3 transformation matrix
- dsize: size of output image
- Flags: interpolation method, the default INTER_LINEAR (bilinear interpolation), when it is WARP_INVERSE_MAP, it means that M is an inverse transformation, which can realize the inverse transformation from the target dst to src.
- borderMode: edge type. The default is BORDER_CONSTANT. When the value is BORDER_TRANSPARENT, it means that the values in the target image are not changed, and these values correspond to the abnormal values in the original image.
- borderValue: border value, the default is 0.

OpenCV provides a function to provide a transformation matrix for perspective transformation.

The function is as follows:

matAffine = cv2.getPerspectiveTransform(matSrc, matDst)

- matSrc: the coordinates of the four vertices of the input picture.
- matDst: the coordinates of the four vertices of the output picture.

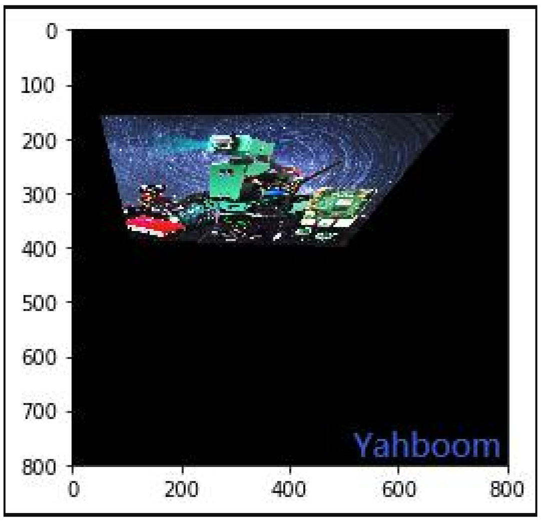
Code path:

/home/pi/Yahboom_Project/Raspbot/1.OpenCV_course/02Geometric_transformation/ 07_Perspective_transformation.ipynb

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread('yahboom.jpg',1)

imgInfo = img.shape
height = imgInfo[0]
width = imgInfo[1]
```

```
#src 4->dst 4 (Upper left corner, Lower left corner, Upper right corner)
matSrc = np.float32([[200,100],[200,400],[600,100],[width-1,height-1]])
matDst = np.float32([[200,200],[200,300],[500,200],[500,400]])

#Combine
matAffine = cv2.getPerspectiveTransform(matSrc,matDst)# mat 1 src 2 dst
dst = cv2.warpPerspective(img,matAffine,(width,height))
img_bgr2rgb = cv2.cvtColor(dst, cv2.COLOR_BGR2RGB)
plt.imshow(img_bgr2rgb)
    plt.show()
```

After running the following program, we can see the picture after the perspective transformation, as shown below.


0
100
200
300
400
500
600
700
800
0
200
400
600
800
Yahboom