

## 7. Perspective transformation

---

Perspective transformation is also called projection transformation. The affine transformation we often call is a special case of perspective transformation. The purpose of perspective transformation is to convert an object that is a straight line in reality into a straight line through perspective transformation, which may appear as a diagonal line in the picture. Perspective transformation can map a rectangle to an arbitrary quadrilateral. This technology will be used later when our robot drives autonomously.

Perspective transformation via function:

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

I dst: Output image after perspective transformation, dsize determines the actual size of the output.

I src: source image

I M: 3X3 transformation matrix

I dsize: Output image size.

I flags: Interpolation method, the default is 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.

I borderMode: edge type. Default is BORDER\_CONSTANT. When the value is BORDER\_TRANSPARENT, the values in the target image are not changed. These values correspond to the outliers in the original image.

I borderWidth: border value, default is 0.

Like affine transformation, OpenCV will still provide a function `cv2.getPerspectiveTransform()` to provide the transformation matrix above.

The function is as follows:

```
matAffine = cv2.getPerspectiveTransform(matSrc, matDst)
```

I matSrc: input the four vertex coordinates of the image.

I matDst: output the four vertex coordinates of the image.

### The code was run on jupyterlab

```
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 lower 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]])

#combination

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()

```

