Affine transformation

January 29, 2020

[1]:
```
import cv2
import numpy as np
```

1 Read source image and it size

[2]:
```
mainImg = cv2.imread("homewithpoints.jpg")
rows, cols, ch = mainImg.shape
```

2 identify the two triangles

[3]:
```
FirstMatrix = np.float32([[375.0, 102.0, 1.0,0,0,0],[244.0, 193.0, 1.
↪0,0,0,0],[503.0, 193.0, 1.0,0,0,0],[0,0,0,375.0, 102.0, 1.0]
                          ,[0,0,0,244.0, 193.0, 1.0],[0,0,0,503.0, 193.0, 1.0]])
SecondMatrix = np.float32([188.0,290.0,210.0,207.0,200.0,284.0])
```

3 finding the transformation Matrix

[4]:
```
transformMatrix = np.float32(np.linalg.inv(FirstMatrix).dot(SecondMatrix))
```

4 rearrange transformation Matrix

[5]:
```
transformMatrixArranged = np.
↪float32([[transformMatrix[0],transformMatrix[1],transformMatrix[2]],
                    [transformMatrix[3],transformMatrix[4],transformMatrix[5]],
                    [0,0,1]])
```

5 finding the inverse of the transformation matrix

[6]:
```
invsOfTransMatrix = np.array(np.linalg.inv(transformMatrixArranged))
```

6 find the size of the destination image

[7]:
```
def findSize(transMatrix,max_x,max_y):
    for i in range(0,rows):
        for j in range(0,cols):
            y = i * transMatrix[0] + j * transMatrix[1] + transMatrix[2]
            x = i * transMatrix[3] + j * transMatrix[4] + transMatrix[5]
            if x > max_x:
                xresult = x
            if y >max_y:
                yresult = y

    return xresult,yresult


max_x,max_y = findSize(transformMatrix,0,0)
```

7 initialize the destination image with the size we found

[8]:
```
result_img = np.zeros((int(max_x),int(max_y), 3))
height, width, c = result_img.shape
```

8 finding the corresponding coordinates

[9]:
```
def findingCoords(u, v, invMatrixCopy):
    x = v * invMatrixCopy.item(3) + u * invMatrixCopy.item(4) + invMatrixCopy.
 ↪item(5)
    y = v * invMatrixCopy.item(0) + u * invMatrixCopy.item(1) + invMatrixCopy.
 ↪item(2)
    return x, y
```

9 check if its inside the limits or not

[10]:
```
def insideTheLimits(cor1, cor2 , nRows, nCols):
    return (cor1 >= 0 and cor1 < nRows and cor2 >= 0 and cor2 < nCols)
```

10 loop through destination image and copy colors

[11]:
```
for specH in range(0, height):
    for specW in range(0, width):
        x, y = findingCoords(specH, specW, invsOfTransMatrix)
        if insideTheLimits(x, y, rows,cols):
            result_img[specH, specW, 0] = mainImg[int(x), int(y), 0]
```

```
            result_img[specH, specW, 1] = mainImg[int(x), int(y), 1]
            result_img[specH, specW, 2] = mainImg[int(x), int(y), 2]
```

11 save the result

[12]:
```
cv2.imwrite('resultedPic.png', result_img)
```

[12]:
```
True
```

Original

Affine transformation