关于简单目标识别与意图分析的机器学习实战研究（透视变换）

这边刚刚解决了一个连线算法问题，同学阶段性汇报回来，说老师让他将在空中看坦克群的视角方向考虑进去，也就是说在不同视角下看同一类型的坦克要尽量保证一样，这样在训练数据的时候也能降低误差，值得一提的是，他的老师肯定了我连线算法的可行性，可以作为一个创新点来汇报。

我仔细一寻思，老师说的没错，在空中侦查坦克的视角是随机的，这一点不解决确实会影坦克意图的判别，怎样解决呢？看来还是要从图片处理的角度入手，我发现在多种图片体系变换中，透视变换是指利用透视中心、像点、目标点三点共线的条件，按透视旋转定律使透视面绕透视轴旋转某一角度，破坏原有的投影光束，仍能保持透视面上投影几何图形不变的变换。看来只要实现了透视变换就可以解决不同视角问题。

在Python中可以先用numpy库将图片转换为矩阵格式，而后使用opencv对矩阵进行透视变换，最后重新生成图片，这样得到的图片就是被拉伸到统一视角下的状态，其实际做法是在目标图片中首先通过模板匹配大致找到目标，然后在目标周围确定四个点（这些点要距离目标稍远一些，否则会造成透视变换之后的目标再次匹配时出现找不到目标的情况），而后将这四个点以及四点连线中间的所有像素点一起拉伸到同一平面，即可完成透视变换。

根据上述思路，直接上代码：

1. ###原图
2. tpl ="D://PythonPicTemplate/binarization/tankTemplate3.jpg"
3. target = "D://PythonPicTemplate/binarization/tanks4.jpg"
4. tpl = Img\_read(tpl)
5. target = Img\_read(target)
7. ###透视变换
8. methods = []
9. **for** o **in** range(0,num):
10. methods.append(cv.TM\_CCORR\_NORMED)
11. th, tw = tpl.shape[:2]
12. rows, cols = target.shape[:2]
14. i =0
15. tl0 = []
16. br0 = []
17. **for** md **in** methods:
18. #print(md)
19. result = cv.matchTemplate(target, tpl, md)
20. min\_val, max\_val, min\_loc, max\_loc = cv.minMaxLoc(result)
22. **if** md == cv.TM\_SQDIFF\_NORMED:
23. tl0.append(min\_loc)
24. **else**:
25. tl0.append(max\_loc)
27. br0.append((tl0[i][0]+tw, tl0[i][1]+th))
28. cv.rectangle(target, tl0[i], br0[i], (0, 0, 255),2)
29. i += 1
31. tl2 = []
32. tl3 = []
33. tl4 = []
34. **for** q **in** tl0:
35. tl3.append(q[0] - tw)
36. tl4.append(q[1] - th)
37. **for** p **in** br0:
38. tl3.append(p[0])
39. tl4.append(p[1])
40. min\_x = min(tl3)
41. min\_y = min(tl4)
42. max\_x = max(tl3)
43. max\_y = max(tl4)
45. tl2.append([min\_x - 2 \* tw,min\_y - 2 \* th])
46. tl2.append([min\_x - 2 \* tw,max\_y + 2 \* th])
47. tl2.append([max\_x + 2 \* tw,min\_y - 2 \* th])
48. tl2.append([max\_x + 2 \* tw,max\_y + 2 \* th])
50. # 原图中已经识别的四个角点
51. pts1 = np.float32(tl2)
52. # 变换后分别在左上、右上、左下、右下四个点
53. #pts2 = np.float32([[0, 0],[0, 500], [500, 0],[500, 500]])
54. pts2 = np.float32([[0, 0],[0, cols], [rows, 0],[rows, cols]])
55. #rows, cols
56. # 生成透视变换矩阵
57. M = cv.getPerspectiveTransform(pts1, pts2)
58. # 进行透视变换
59. dst = cv.warpPerspective(target, M, (rows, cols))
60. cv.namedWindow("match-PerspectiveTransformation", cv.WINDOW\_NORMAL)
61. cv.imshow("match-PerspectiveTransformation", dst)

结果如下图所示：



