关于简单目标识别与意图分析的机器学习实战研究（颜色识别）

上一节我们说到要加强目标识别方法，怎么做到这点呢，我翻了翻OpenCV的官方文档，发现除了模板匹配之外，颜色匹配是它的另一个目标识别方法，一般的思路是这样的：

首先使用OpenCV截取图像，然后将图像内RGB转换为HSV色彩空间进行判断，统计判断HSV和RGB的范围对应关系，最后输出RGB值

然后将目标图片进行处理，将RGB颜色转为HSV；使用cv2中inRange()函数过滤背景颜色；将过滤后的颜色进行二值化处理，得到双色彩图片（其中有一种是我们目标的颜色）；再使用dilate()函数对双色彩图片进行形态学腐蚀膨胀；最后统计白色区域面积。

这里为什么要将RGB转换成HSV色彩空间呢？

主要是因为HSV色彩空间可以提高识别的准确率，较RGB而言更适合于目标识别这一领域，当前很多出色的颜色识别项目，都是在这个空间下进行的识别。这里我想到了一个增加精度的方法，在测试前对目标图片做一次高斯模糊的效果会更好一些，实验也证明确实是这样。下面就具体举一个例子，来看看颜色识别的效果吧，一言不合就上代码：

1. #coding:utf-8
2. **import** cv2 as cv
3. **import** numpy as np
4. **import** collections
5. **import** math
6. **import** ctypes,sys
8. **def** getColorList():
9. dict = collections.defaultdict(list)
11. # 黑色
12. lower\_black = np.array([0, 0, 0])
13. upper\_black = np.array([180, 255, 46])
14. color\_list = []
15. color\_list.append(lower\_black)
16. color\_list.append(upper\_black)
17. dict['black'] = color\_list
19. # 白色
20. lower\_white = np.array([0, 0, 221])
21. upper\_white = np.array([180, 30, 255])
22. color\_list = []
23. color\_list.append(lower\_white)
24. color\_list.append(upper\_white)
25. dict['white'] = color\_list
27. #红色
28. lower\_red = np.array([156, 43, 46])
29. upper\_red = np.array([180, 255, 255])
30. color\_list = []
31. color\_list.append(lower\_red)
32. color\_list.append(upper\_red)
33. dict['red']=color\_list
35. # 红色2
36. lower\_red = np.array([0, 43, 46])
37. upper\_red = np.array([5, 255, 255])
38. color\_list = []
39. color\_list.append(lower\_red)
40. color\_list.append(upper\_red)
41. dict['red2'] = color\_list
43. #橙色
44. lower\_orange = np.array([11, 43, 46])
45. upper\_orange = np.array([25, 255, 255])
46. color\_list = []
47. color\_list.append(lower\_orange)
48. color\_list.append(upper\_orange)
49. dict['orange'] = color\_list
51. #黄色
52. lower\_yellow = np.array([15, 55, 55])
53. upper\_yellow = np.array([50, 255, 255])
54. color\_list = []
55. color\_list.append(lower\_yellow)
56. color\_list.append(upper\_yellow)
57. dict['yellow'] = color\_list
59. #绿色
60. lower\_green = np.array([35, 43, 46])
61. upper\_green = np.array([77, 255, 255])
62. color\_list = []
63. color\_list.append(lower\_green)
64. color\_list.append(upper\_green)
65. dict['green'] = color\_list
67. #青色
68. lower\_cyan = np.array([78, 43, 46])
69. upper\_cyan = np.array([99, 255, 255])
70. color\_list = []
71. color\_list.append(lower\_cyan)
72. color\_list.append(upper\_cyan)
73. dict['cyan'] = color\_list
75. #蓝色
76. lower\_blue = np.array([50, 15, 15])
77. upper\_blue = np.array([130, 255, 255])
78. color\_list = []
79. color\_list.append(lower\_blue)
80. color\_list.append(upper\_blue)
81. dict['blue'] = color\_list
83. # 紫色
84. lower\_purple = np.array([125, 43, 46])
85. upper\_purple = np.array([155, 255, 255])
86. color\_list = []
87. color\_list.append(lower\_purple)
88. color\_list.append(upper\_purple)
89. dict['purple'] = color\_list
91. **return** dict

94. **def** get\_color(frame):
95. **print**('go in get\_color')
96. hsv = cv.cvtColor(frame,cv.COLOR\_BGR2HSV\_FULL)
97. maxsum = -120
98. color = None
99. color\_dict = getColorList()
100. **for** d **in** color\_dict:
101. mask = cv.inRange(hsv,color\_dict[d][0],color\_dict[d][1])
102. #cv.imwrite(d+'.jpg',mask)
103. binary = cv.threshold(mask, 90, 255, cv.THRESH\_BINARY)[1]
104. binary = cv.dilate(binary,None,iterations=2)
105. img, cnts, hiera = cv.findContours(binary.copy(),cv.RETR\_EXTERNAL,cv.CHAIN\_APPROX\_SIMPLE)
106. sum = 0
107. **for** c **in** cnts:
108. sum+=cv.contourArea(c)
109. **if** sum > maxsum :
110. maxsum = sum
111. color = d
113. **return** color
115. **def** separate\_color(frame,color):
116. hsv = cv.cvtColor(frame, cv.COLOR\_BGR2HSV)                              #色彩空间转换为hsv，便于分离
117. cv.imshow("f",hsv)
118. lower\_hsv = np.array(color[0])                                                  #提取颜色的低值
119. high\_hsv = np.array(color[1])                                                   #提取颜色的高值
120. mask = cv.inRange(hsv, lowerb = lower\_hsv, upperb = high\_hsv)           #下面详细介绍
121. cv.namedWindow("ColorFilter", cv.WINDOW\_NORMAL)
122. cv.imshow("ColorFilter", mask)

125. **if** \_\_name\_\_ == '\_\_main\_\_':
127. ###高斯模糊
128. filename='D://PythonPicTemplate/colorCutting/tankTemplate1.jpg'
129. targetname = "D://PythonPicTemplate/colorCutting/tanks1.jpg"
130. kernel\_size = (15, 15)
131. sigma = 1.5
132. imgC = cv.imread(filename)
133. imgColor = cv.GaussianBlur(imgC, kernel\_size, sigma)
135. #提取模板颜色
136. imghsv = cv.cvtColor(imgColor, cv.COLOR\_BGR2HSV)                   #色彩空间转换为hsv，便于分离
137. cv.imshow("f0",imghsv)
139. content = "正在提取模板颜色HAV矩阵向量......"
140. printGreen(content.decode("utf-8").encode("gbk"))

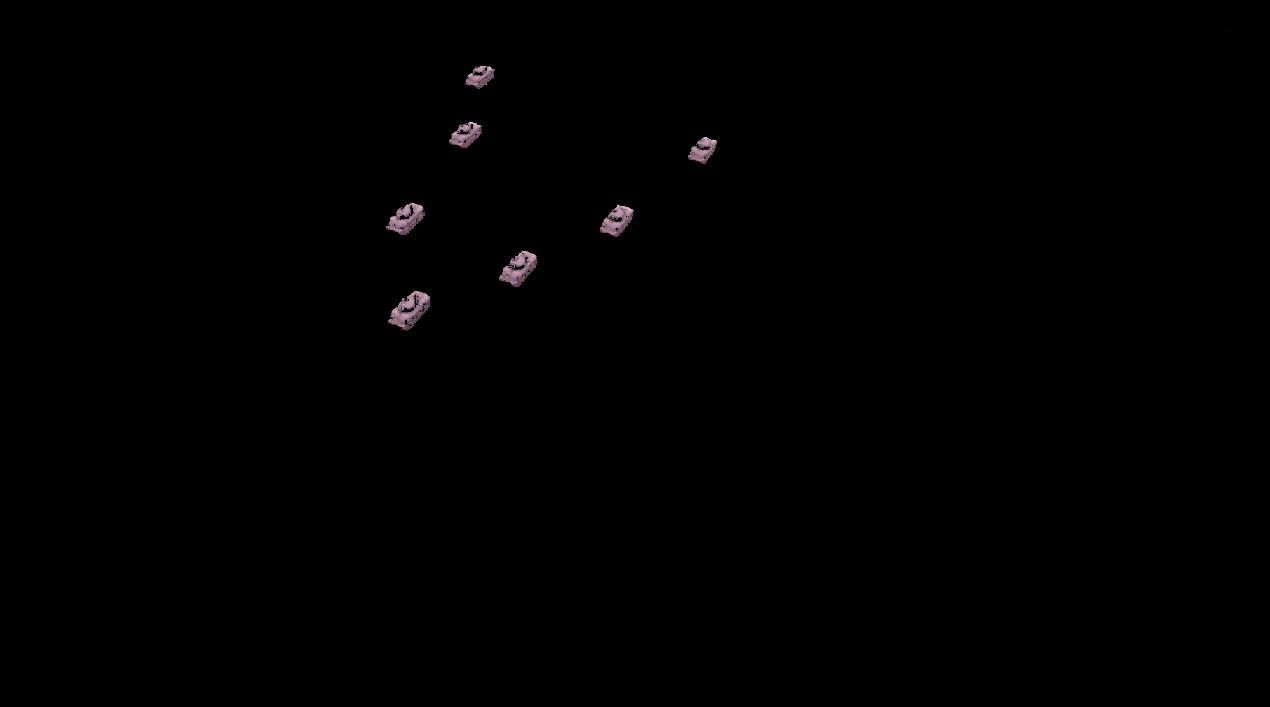
143. color\_dict = getColorList()
144. **print**(get\_color(imgColor))
145. tplColor = color\_dict[get\_color(imghsv)]
147. #目标颜色追踪
148. target = cv.imread(targetname)
149. HSV = cv.cvtColor(target, cv.COLOR\_BGR2HSV)
150. #H, S, V = cv.split(HSV)
151. LowerBlue = np.array(tplColor[0])
152. UpperBlue = np.array(tplColor[1])
154. content0 = "正在对目标进行颜色分割处理......"
155. printGreen(content0.decode("utf-8").encode("gbk"))

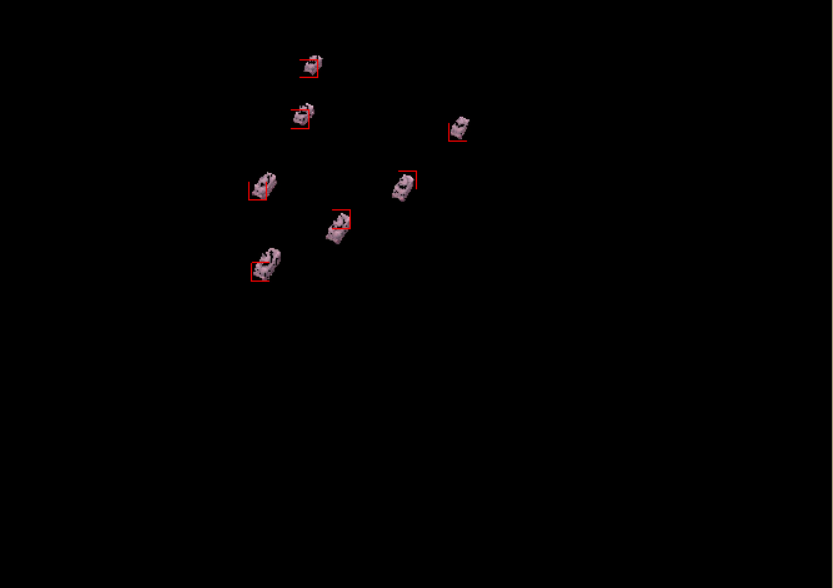
158. mask = cv.inRange(HSV, LowerBlue, UpperBlue)
159. targetCutting0 = cv.bitwise\_and(target, target, mask=mask)
160. cv.imwrite('targetCutting.jpg',targetCutting0)
161. cv.namedWindow('targetCutting0', cv.WINDOW\_NORMAL)
162. cv.imshow('targetCutting0',targetCutting0)
163. cv.waitKey(0)
164. cv.destroyAllWindows()
166. ###初始化处理过的目标
167. filename2='D://PythonPicTemplate/colorCutting/tankTemplate2.jpg'
168. targetname2 = "D://PythonPicTemplate/colorCutting/targetCutting.jpg"
169. imgC = cv.imread(filename2)
170. targetCutting0 = cv.imread(targetname2)
171. imgC = cv.GaussianBlur(imgC, kernel\_size, sigma)
173. HSV2 = cv.cvtColor(imgC, cv.COLOR\_BGR2HSV)
174. LowerBlue = np.array(tplColor[0])
175. UpperBlue = np.array(tplColor[1])
176. mask2 = cv.inRange(HSV2, LowerBlue, UpperBlue)
177. imgC= cv.bitwise\_and(imgC, imgC, mask=mask2)

注：代码中对颜色范围是从网上找的，自己只实验了下灰白和紫色。

测试结果如下：







从结果中我们大概就可以看出，颜色识别这一方法优点是在目标坦克颜色与地貌、环境不符时，可以排除其他颜色下的干扰因素，然后便可进行模板匹配，增加识别的精度。缺点仍然有很多，例如：

1.该方法十分依赖RGB空间范围和HSV空间的对应关系，但网上没有试验将两者的对应关系很好的展现出来，在一些论文研究的时候也是作者自己测试了部分他所需要的数据而已，而网上的数据集十分不精确。

2.颜色匹配要求目标坦克的颜色要在图片中区别于其他障碍物，但实际中坦克外面是已经做过伪装措施的，而且针对不同地域进行了不同的伪装，其中就包括颜色的改变，所以这对本来数据集就少的颜色层次区分有了进一步的要求。

这里大家可能会问为什么一开始不直接使用机器学习的方法训练数据集来达到目标识别的目的，其实这个在后面的文章中会讲，主要原因一个是机器学习进行训练的时候，训练图片内容越复杂，训练起来越费时费力，我试过了好几种目标检测方法，效果都不如各种方法展示的那么好。而且我用的是笔记本，实在带不动太多的图片数据。