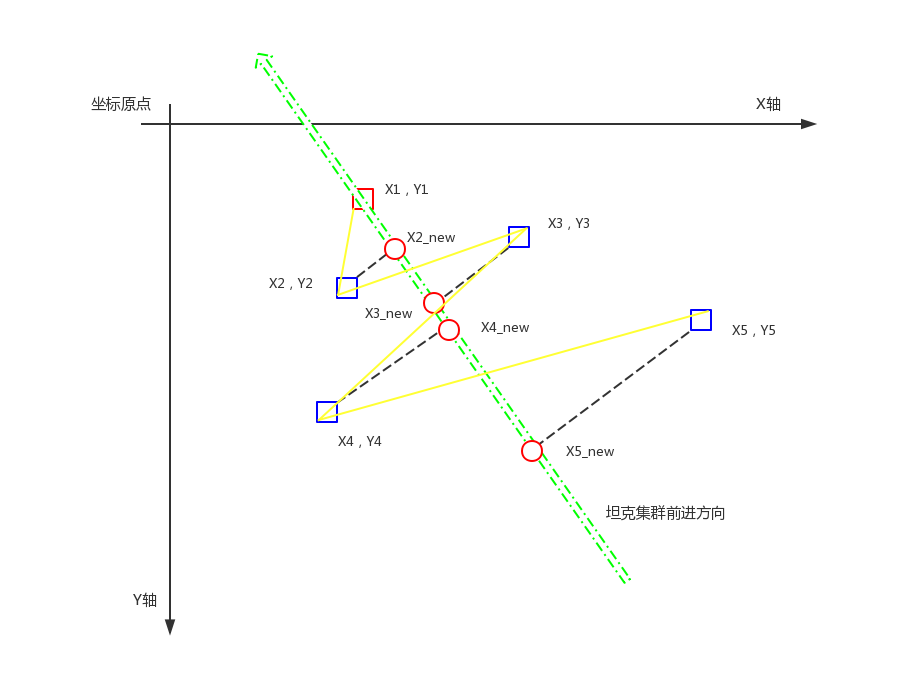
关于简单目标识别与意图分析的机器学习实战研究（连线算法）

根据这个课题的要求，在识别意图的方法上需要好好的想一想，因为是作战坦克，从简单的方面入手，我们假设它有三种意图，行军、突袭和作战（我看了下资料，好像很少涉及防守一说，好多论文研究的都是这三个方面，例如这篇：<http://cdmd.cnki.com.cn/Article/CDMD-10487-1017833083.htm>）。对于深度学习大量的运算，我看了看自己的小笔记本，这根本带不动，所以机智的我依然决然的选择了一个简化算法，一张图片信息量太多，那一条线呢？没错，我就是要把一张图转换成一条线，用这条线来代替坦克阵型，因为我们最后要做的就是根据坦克阵型来判别坦克的作战意图。

又因为坦克的意图多样，所以这条连线必须体现出坦克阵型的所有信息，在模板匹配中，可以通过cv2.minMaxLoc函数得到最佳匹配区域，并返回它在图片上的坐标值。既然每辆坦克的坐标点有了，那么现在要考虑的就是如何连线，这个问题说难不难，说简单也不简单，在图片做完之后，下一步要做的就是视频上的分析，所以坦克的行进方向是一个重要的信息，连线顺序就是以行进方向为准，将所有点投影到这条直线上，而后按照投影之后点的坐标顺序，从横坐标从小到大，或纵坐标从小到大依次连线即可。



连线算法示意图

上面那条黄色的线就是我们最终简化的结果，坦克行进的方向不同，这条线的朝向也会不同，但是这些点内部的顺序是确定的。而后面通进行深度学习时只训练这样的特征连线就可以了，这样便达成了我们减小运算量，却不失信息量的目的。

连线制作简化模型的思路就是这样，下面开始具体编写代码，我第一次写这个连线算法的代码是这个样子的（刚开始没找到什么简便算法，就固定五辆坦克，一个点一个点的算距离，然后在排序，最后才连线，因为在排序、运算的过程中要一直确保每辆坦克一直都有标识）：

1. s = "ZZZ"
2. c = "Z"
4. s\_content = "What direction? (U[上下] / L[左右] / 任意键进入角度选择)"
5. c\_content = "请输入您想要在哪个角度上进行匹配连线（角度）："
7. s = raw\_input(s\_content.decode("utf-8").encode("gbk"))
8. **if** (s != "ZZZ" **and** s != "U" **and** s != "L"):
9. c = raw\_input(c\_content.decode("utf-8").encode("gbk"))
10. red = (0, 0, 255)
11. col = red
12. thickness = 20
13. **print** tl
14. tl1 = []
16. #c = tl[1][0] - tl[2][0]
17. #print c
18. #s = [x[0] for x in tl][0]
19. #print s
20. **if** c != "Z":
21. **for** g **in** range(0,5):
22. tl1.append( l\_get(tl[g],c))
23. **print** tl1
24. **for** j **in** range(0,5):
25. **for** k **in** range(0,5):
26. **if** j != k:
27. **if** k == 0:
28. **for** j **in** range(0,5):
29. **for** k **in** range(0,5):
30. **if** j != k:
31. **if** k == 0:
32. **if** j == 1:
33. **if** (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[2])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[3])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[4])):
34. cv.line(target, tl[j], tl[k], col, thickness)
35. **if** j == 2:
36. **if** (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[1])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[3])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[4])):
37. cv.line(target, tl[j], tl[k], col, thickness)
38. **if** j == 3:
39. **if** (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[1])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[2])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[4])):
40. cv.line(target, tl[j], tl[k], col, thickness)
41. **if** j == 4:
42. **if** (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[1])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[2])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[3])):
43. cv.line(target, tl[j], tl[k], col, thickness)
45. **if** k == 1:
46. **if** j == 0:
47. **if** (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[2])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[3])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[4])):
48. cv.line(target, tl[j], tl[k], col, thickness)
49. **if** j == 2:
50. **if** (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[0])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[3])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[4])):
51. cv.line(target, tl[j], tl[k], col, thickness)
52. **if** j == 3:
53. **if** (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[0])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[2])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[4])):
54. cv.line(target, tl[j], tl[k], col, thickness)
55. **if** j == 4:
56. **if** (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[0])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[2])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[3])):
57. cv.line(target, tl[j], tl[k], col, thickness)
59. **if** k == 2:
60. **if** j == 0:
61. **if** (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[1])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[3])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[4])):
62. cv.line(target, tl[j], tl[k], col, thickness)
63. **if** j == 1:
64. **if** (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[0])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[3])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[4])):
65. cv.line(target, tl[j], tl[k], col, thickness)
66. **if** j == 3:
67. **if** (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[0])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[1])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[4])):
68. cv.line(target, tl[j], tl[k], col, thickness)
69. **if** j == 4:
70. **if** (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[0])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[1])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[3])):
71. cv.line(target, tl[j], tl[k], col, thickness)
73. **if** k == 3:
74. **if** j == 0:
75. **if** (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[1])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[2])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[4])):
76. cv.line(target, tl[j], tl[k], col, thickness)
77. **if** j == 1:
78. **if** (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[0])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[2])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[4])):
79. cv.line(target, tl[j], tl[k], col, thickness)
80. **if** j == 2:
81. **if** (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[0])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[1])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[4])):
82. cv.line(target, tl[j], tl[k], col, thickness)
83. **if** j == 4:
84. **if** (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[0])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[1])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[2])):
85. cv.line(target, tl[j], tl[k], col, thickness)
87. **if** k == 4:
88. **if** j == 0:
89. **if** (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[1])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[2])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[3])):
90. cv.line(target, tl[j], tl[k], col, thickness)
91. **if** j == 1:
92. **if** (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[0])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[2])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[3])):
93. cv.line(target, tl[j], tl[k], col, thickness)
94. **if** j == 2:
95. **if** (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[0])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[1])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[3])):
96. cv.line(target, tl[j], tl[k], col, thickness)
97. **if** j == 3:
98. **if** (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[0])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[1])) & (l\_z(tl1[j],tl1[k]) >= l\_z(tl1[j] , tl1[2])):
99. cv.line(target, tl[j], tl[k], col, thickness)
101. cv.namedWindow("match-m-" + str(j), cv.WINDOW\_NORMAL)
102. cv.imshow("match-m-" + str(j), target)
104. **if**(s == "L"):
105. **for** j **in** range(0,5):
106. **for** k **in** range(0,5):
107. **if** j != k:
108. **if** k == 0:
109. **if** j == 1:
110. **if** (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[2])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[3])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[4])):
111. cv.line(target, tl[j], tl[k], col, thickness)
112. **if** j == 2:
113. **if** (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[1])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[3])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[4])):
114. cv.line(target, tl[j], tl[k], col, thickness)
115. **if** j == 3:
116. **if** (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[1])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[2])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[4])):
117. cv.line(target, tl[j], tl[k], col, thickness)
118. **if** j == 4:
119. **if** (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[1])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[2])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[3])):
120. cv.line(target, tl[j], tl[k], col, thickness)
122. **if** k == 1:
123. **if** j == 0:
124. **if** (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[2])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[3])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[4])):
125. cv.line(target, tl[j], tl[k], col, thickness)
126. **if** j == 2:
127. **if** (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[0])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[3])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[4])):
128. cv.line(target, tl[j], tl[k], col, thickness)
129. **if** j == 3:
130. **if** (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[0])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[2])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[4])):
131. cv.line(target, tl[j], tl[k], col, thickness)
132. **if** j == 4:
133. **if** (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[0])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[2])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[3])):
134. cv.line(target, tl[j], tl[k], col, thickness)
136. **if** k == 2:
137. **if** j == 0:
138. **if** (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[1])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[3])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[4])):
139. cv.line(target, tl[j], tl[k], col, thickness)
140. **if** j == 1:
141. **if** (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[0])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[3])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[4])):
142. cv.line(target, tl[j], tl[k], col, thickness)
143. **if** j == 3:
144. **if** (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[0])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[1])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[4])):
145. cv.line(target, tl[j], tl[k], col, thickness)
146. **if** j == 4:
147. **if** (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[0])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[1])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[3])):
148. cv.line(target, tl[j], tl[k], col, thickness)
150. **if** k == 3:
151. **if** j == 0:
152. **if** (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[1])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[2])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[4])):
153. cv.line(target, tl[j], tl[k], col, thickness)
154. **if** j == 1:
155. **if** (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[0])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[2])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[4])):
156. cv.line(target, tl[j], tl[k], col, thickness)
157. **if** j == 2:
158. **if** (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[0])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[1])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[4])):
159. cv.line(target, tl[j], tl[k], col, thickness)
160. **if** j == 4:
161. **if** (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[0])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[1])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[2])):
162. cv.line(target, tl[j], tl[k], col, thickness)
164. **if** k == 4:
165. **if** j == 0:
166. **if** (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[1])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[2])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[3])):
167. cv.line(target, tl[j], tl[k], col, thickness)
168. **if** j == 1:
169. **if** (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[0])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[2])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[3])):
170. cv.line(target, tl[j], tl[k], col, thickness)
171. **if** j == 2:
172. **if** (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[0])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[1])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[3])):
173. cv.line(target, tl[j], tl[k], col, thickness)
174. **if** j == 3:
175. **if** (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[0])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[1])) & (l\_x(tl[j],tl[k]) <= l\_x(tl[j] , tl[2])):
176. cv.line(target, tl[j], tl[k], col, thickness)
178. cv.namedWindow("match-m-" + str(j), cv.WINDOW\_NORMAL)
179. cv.imshow("match-m-" + str(j), target)


183. **if**(s == "U"):
184. **for** j **in** range(0,5):
185. **for** k **in** range(0,5):
186. **if** j != k:
187. **if** k == 0:
188. **if** j == 1:
189. **if** (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[2])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[3])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[4])):
190. cv.line(target, tl[j], tl[k], col, thickness)
191. **if** j == 2:
192. **if** (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[1])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[3])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[4])):
193. cv.line(target, tl[j], tl[k], col, thickness)
194. **if** j == 3:
195. **if** (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[1])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[2])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[4])):
196. cv.line(target, tl[j], tl[k], col, thickness)
197. **if** j == 4:
198. **if** (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[1])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[2])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[3])):
199. cv.line(target, tl[j], tl[k], col, thickness)
201. **if** k == 1:
202. **if** j == 0:
203. **if** (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[2])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[3])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[4])):
204. cv.line(target, tl[j], tl[k], col, thickness)
205. **if** j == 2:
206. **if** (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[0])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[3])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[4])):
207. cv.line(target, tl[j], tl[k], col, thickness)
208. **if** j == 3:
209. **if** (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[0])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[2])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[4])):
210. cv.line(target, tl[j], tl[k], col, thickness)
211. **if** j == 4:
212. **if** (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[0])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[2])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[3])):
213. cv.line(target, tl[j], tl[k], col, thickness)
215. **if** k == 2:
216. **if** j == 0:
217. **if** (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[1])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[3])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[4])):
218. cv.line(target, tl[j], tl[k], col, thickness)
219. **if** j == 1:
220. **if** (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[0])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[3])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[4])):
221. cv.line(target, tl[j], tl[k], col, thickness)
222. **if** j == 3:
223. **if** (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[0])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[1])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[4])):
224. cv.line(target, tl[j], tl[k], col, thickness)
225. **if** j == 4:
226. **if** (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[0])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[1])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[3])):
227. cv.line(target, tl[j], tl[k], col, thickness)
229. **if** k == 3:
230. **if** j == 0:
231. **if** (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[1])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[2])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[4])):
232. cv.line(target, tl[j], tl[k], col, thickness)
233. **if** j == 1:
234. **if** (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[0])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[2])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[4])):
235. cv.line(target, tl[j], tl[k], col, thickness)
236. **if** j == 2:
237. **if** (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[0])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[1])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[4])):
238. cv.line(target, tl[j], tl[k], col, thickness)
239. **if** j == 4:
240. **if** (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[0])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[1])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[2])):
241. cv.line(target, tl[j], tl[k], col, thickness)
243. **if** k == 4:
244. **if** j == 0:
245. **if** (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[1])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[2])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[3])):
246. cv.line(target, tl[j], tl[k], col, thickness)
247. **if** j == 1:
248. **if** (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[0])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[2])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[3])):
249. cv.line(target, tl[j], tl[k], col, thickness)
250. **if** j == 2:
251. **if** (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[0])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[1])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[3])):
252. cv.line(target, tl[j], tl[k], col, thickness)
253. **if** j == 3:
254. **if** (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[0])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[1])) & (l\_y(tl[j],tl[k]) <= l\_y(tl[j] , tl[2])):
255. cv.line(target, tl[j], tl[k], col, thickness)
256. cv.namedWindow("match-m-" + str(j), cv.WINDOW\_NORMAL)
257. cv.imshow("match-m-" + str(j), target)



不仅繁琐，而且计算量又上去了，不过这里为执行方便，除了任意角度（坦克方向）外，还直接编写了按横、纵坐标连线的两个方法，这样就不用执行过长的代码，但是代码仍需精简，具体精简的方面是：

1.从直接进行坐标间距离的计算变成先投影再计算；

2.多用函数，避免重复编写代码，造成代码执行效率低的问题；

3.多用数组以减少运算量。

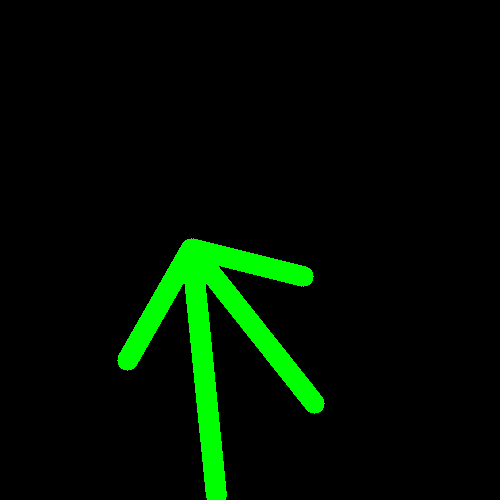
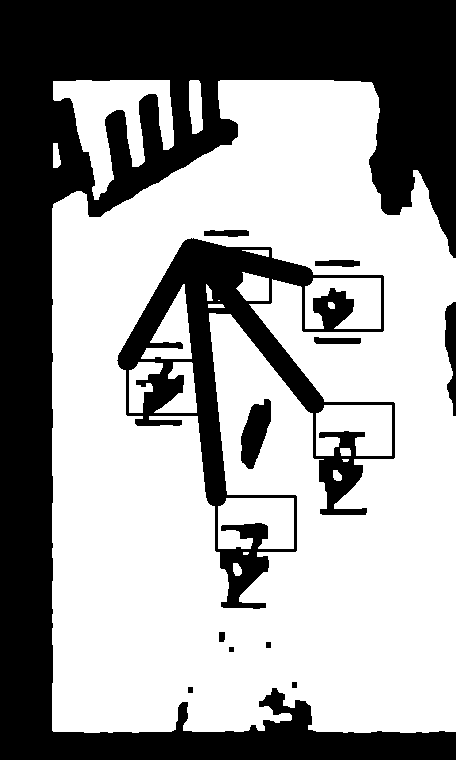
更改好的代码是这个样子的（也算是一次代码修复过程吧）：

1. **def** l\_get(a,c):
2. #在c方向上对坐标进行投影变换，a为坐标
3. #对于C方向直线方程：y = tan(c) \* x
4. c1 = int(c) \* math.pi / 180
5. k = math.tan(c1)
6. a1 = (k \* a[1] + a[0]) / (k \*\* 2 + 1)
7. b1 = k \* a1
8. **return** (a1,b1)
10. #这个函数的意思是 对 nums从下标p一直到q的全排列。
11. **def** permutation(nums, p, q , tl5):
12. **if** p == q:#这里使用 list(nums)是因为如果直接添加 添加的都是指向nums所存数组的地址 nums变化了 tl5里面的数组内容也会跟着变化。
13. tl5.append(list(nums))
14. **else**:
15. **for** i **in** range(p, q):
16. nums[i], nums[p] = nums[p], nums[i]
17. permutation(nums, p+1, q,tl5)
18. nums[i], nums[p] = nums[p], nums[i]
20. **def** line\_matching(t,h):
21. **while** len(t) != 1:
22. **for** b **in** t:
23. **for** j **in** range(0,num - 1):
24. **if** b[j][h] >  b[j + 1][h]:
25. t.remove(b)
26. **break**
27. **else**:
28. **pass**

31. **if** c != "Z":
32. tl5 = []
33. permutation(tl, 0, len(tl),tl5)
34. **for** f **in** tl5:
35. **for** g **in** range(0,num):
36. f[g] = (f[g],l\_get(f[g],c))
37. **while** len(tl5) != 1:
38. **for** b **in** tl5:
39. **for** j **in** range(0,num - 1):
40. **if** b[j][1][1] >  b[j + 1][1][1]:
41. tl5.remove(b)
42. **break**
43. **else**:
44. **pass**
46. **for** z **in** tl5:
47. **for** y **in** range(0,num - 1):
48. cv.line(dst, tl5[0][y][0], tl5[0][y + 1][0], col, thickness)
49. cv.line(openedT, tl5[0][y][0], tl5[0][y + 1][0], col, thickness)
50. cv.line(img, tl5[0][y][0], tl5[0][y + 1][0], green, thickness)#绿色，20个像素宽度
52. **if**(s == "U"):
53. tl5 = []
54. permutation(tl, 0, len(tl),tl5)
55. line\_matching(tl5,1)
56. **for** z **in** tl5:
57. **for** y **in** range(0,num - 1):
58. cv.line(dst, tl5[0][y], tl5[0][y + 1], col, thickness)
59. cv.line(openedT, tl5[0][y], tl5[0][y + 1], col, thickness)
60. cv.line(img, tl5[0][y], tl5[0][y + 1], green, thickness)#绿色，20个像素宽度
62. **if**(s == "L"):
63. tl5 = []
64. permutation(tl, 0, len(tl),tl5)
65. line\_matching(tl5,0)
66. **for** z **in** tl5:
67. **for** y **in** range(0,num - 1):
68. cv.line(dst, tl5[0][y], tl5[0][y + 1], col, thickness)
69. cv.line(openedT, tl5[0][y], tl5[0][y + 1], col, thickness)
70. cv.line(img, tl5[0][y], tl5[0][y + 1], green, thickness)#绿色，20个像素宽度
72. cv.imwrite('FinalMatch.png',openedT)
73. cv.imwrite('ContrastLine.png',img)

效果如图所示：





好的，这个就是连线构造简图的具体方法，这边刚刚做出简化（连线）模型，同学那边传来了一个消息了。