基于SIFT之CPU方向辨識

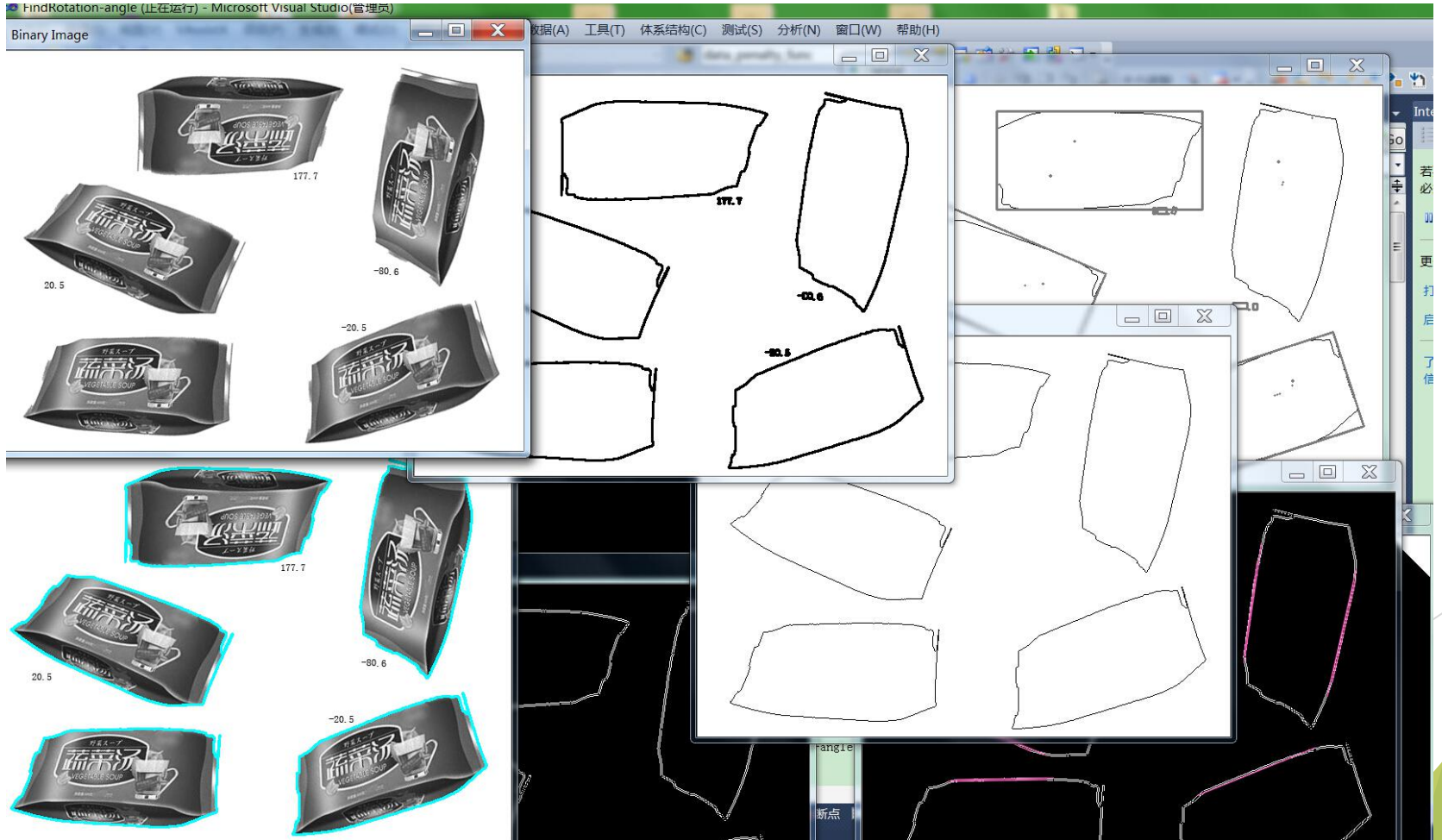
張軍斌 溫韻筑

2021年01月25日

CPS LAB

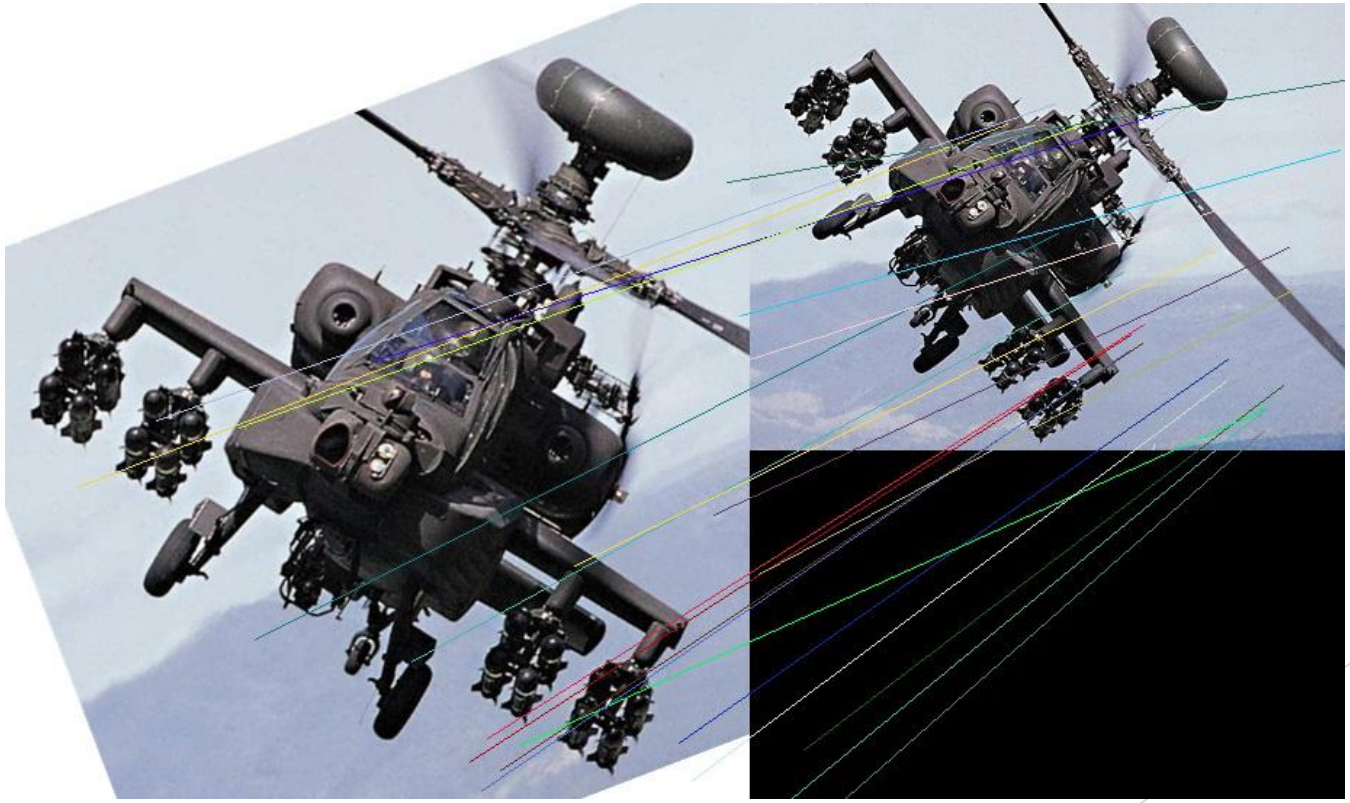
根據模板，如何找到圖片的旋轉角度？

- 用離散傅里葉變換加霍夫變換，求圖像的旋轉角度。這種方案對圖片有比較高的要求，圖像中必須有行列規整的物體，例如文本是一個很典型的例子，如果是像你給出的這種圖像，效果不太好，當然你也可以試試加一些約束和判斷或許可以改進。



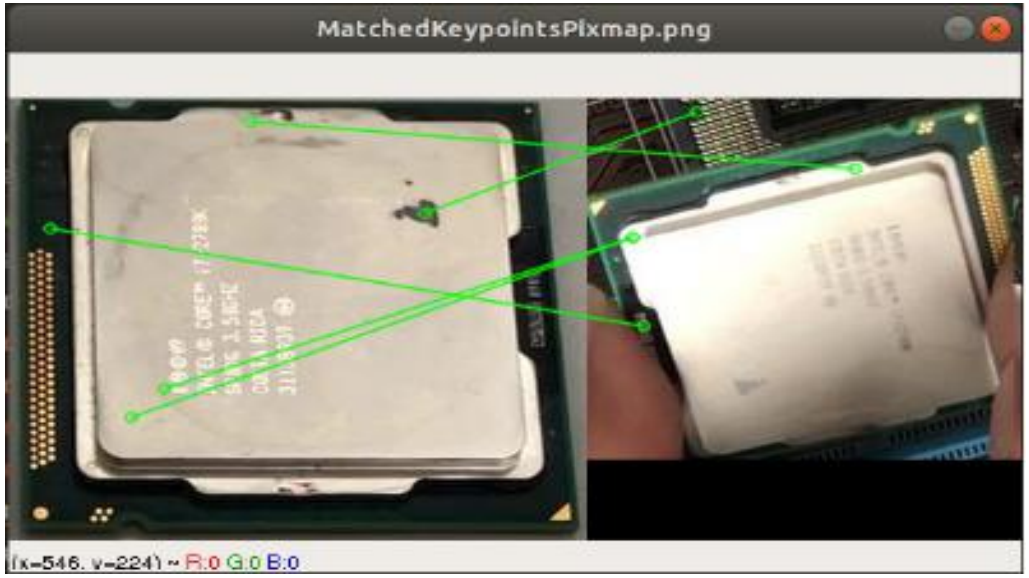
根據模板，如何找到圖片的旋轉角度？

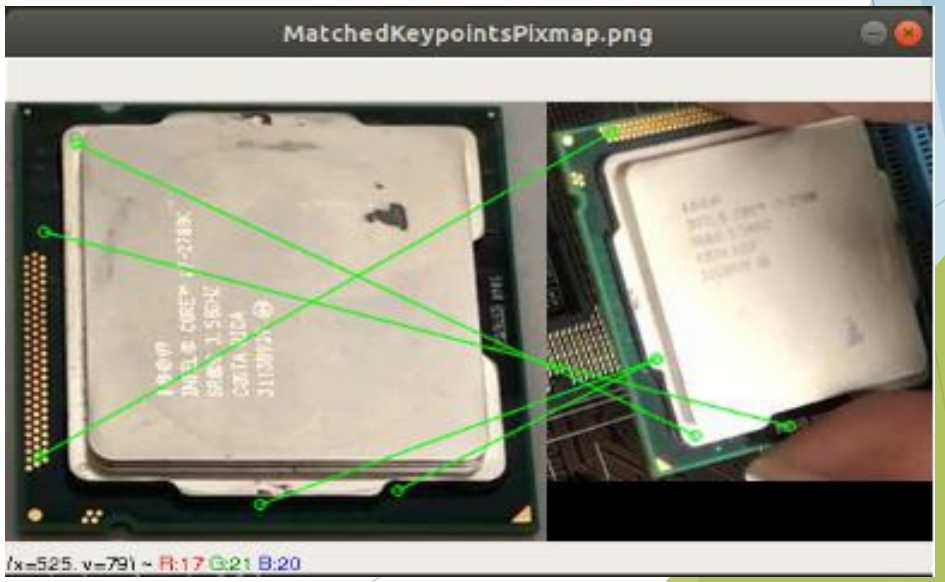
- 用SIFT或者SURF求圖片的描述算子，然後用findHomography求得H矩陣，然後用opencv的decomposeHomographyMat分解出旋轉矩陣和平移矩陣，再用opencv的Rodrigues將旋轉矩陣轉為歐拉角。第二種方法有一個難點，那就是decomposeHomographyMat的結果往往不唯一，因為homography的矩陣分解是與相機矩陣相關的



CPS LAB

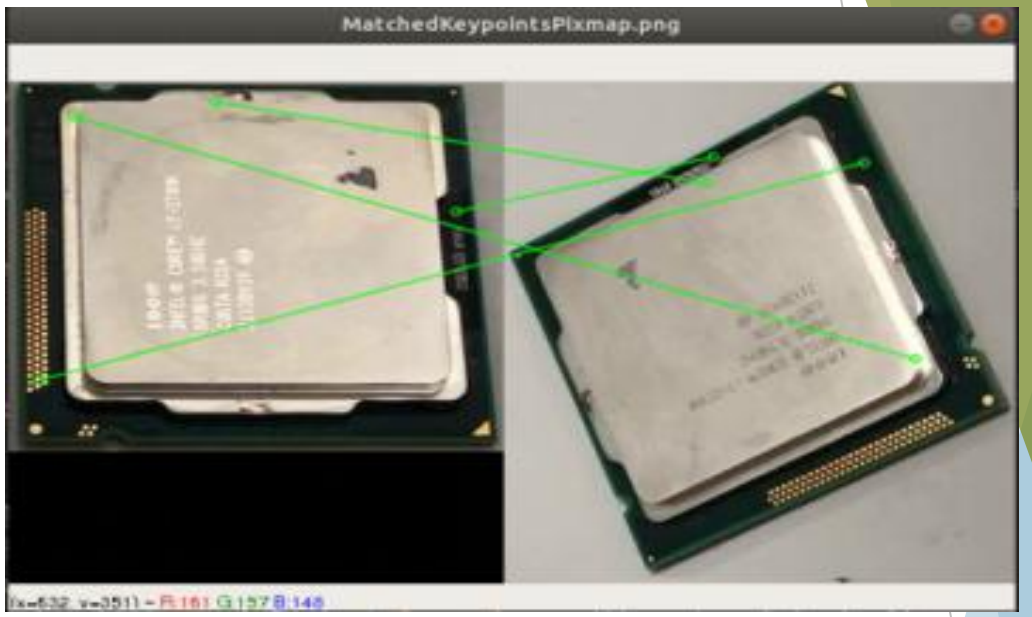
CPU1 特徵點





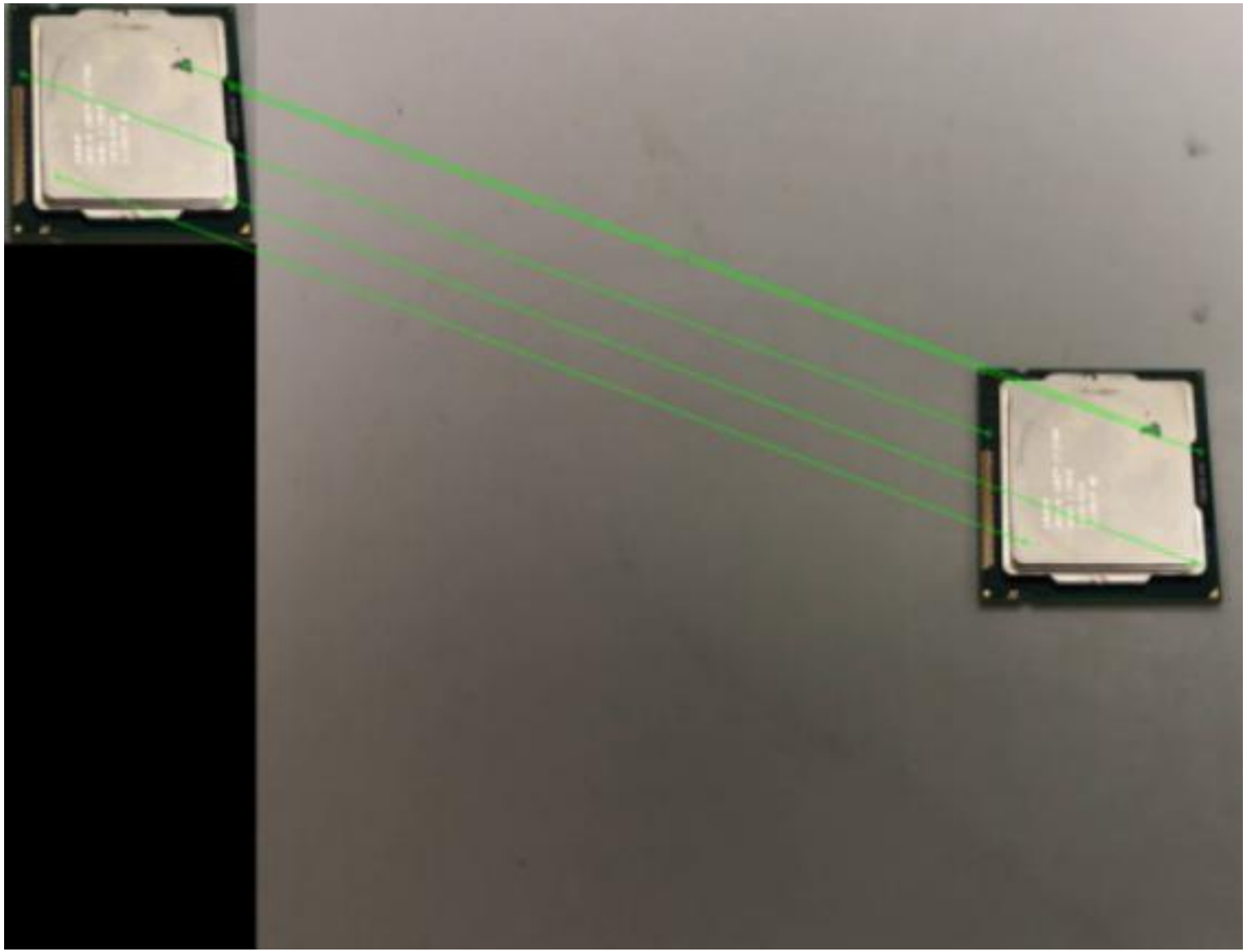
CPS LAB

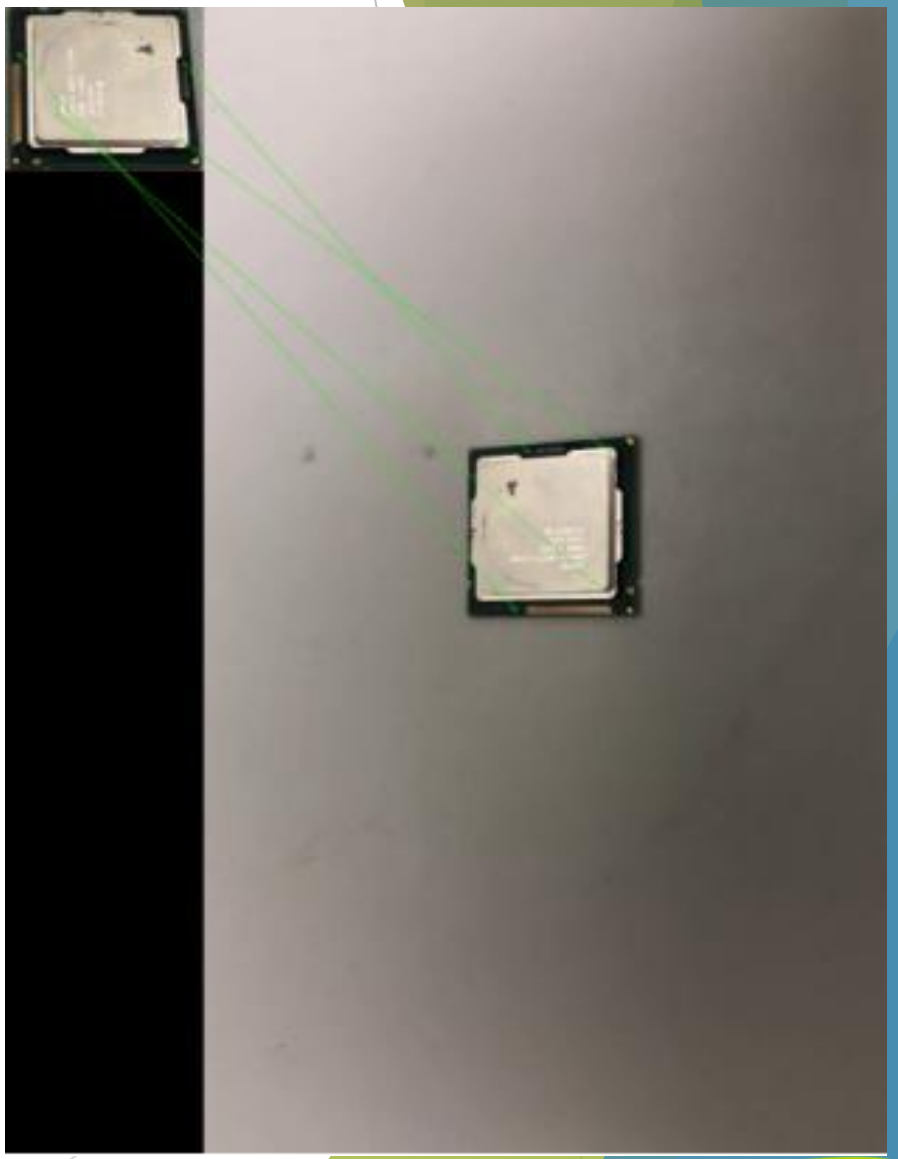
CPU2 特徵點



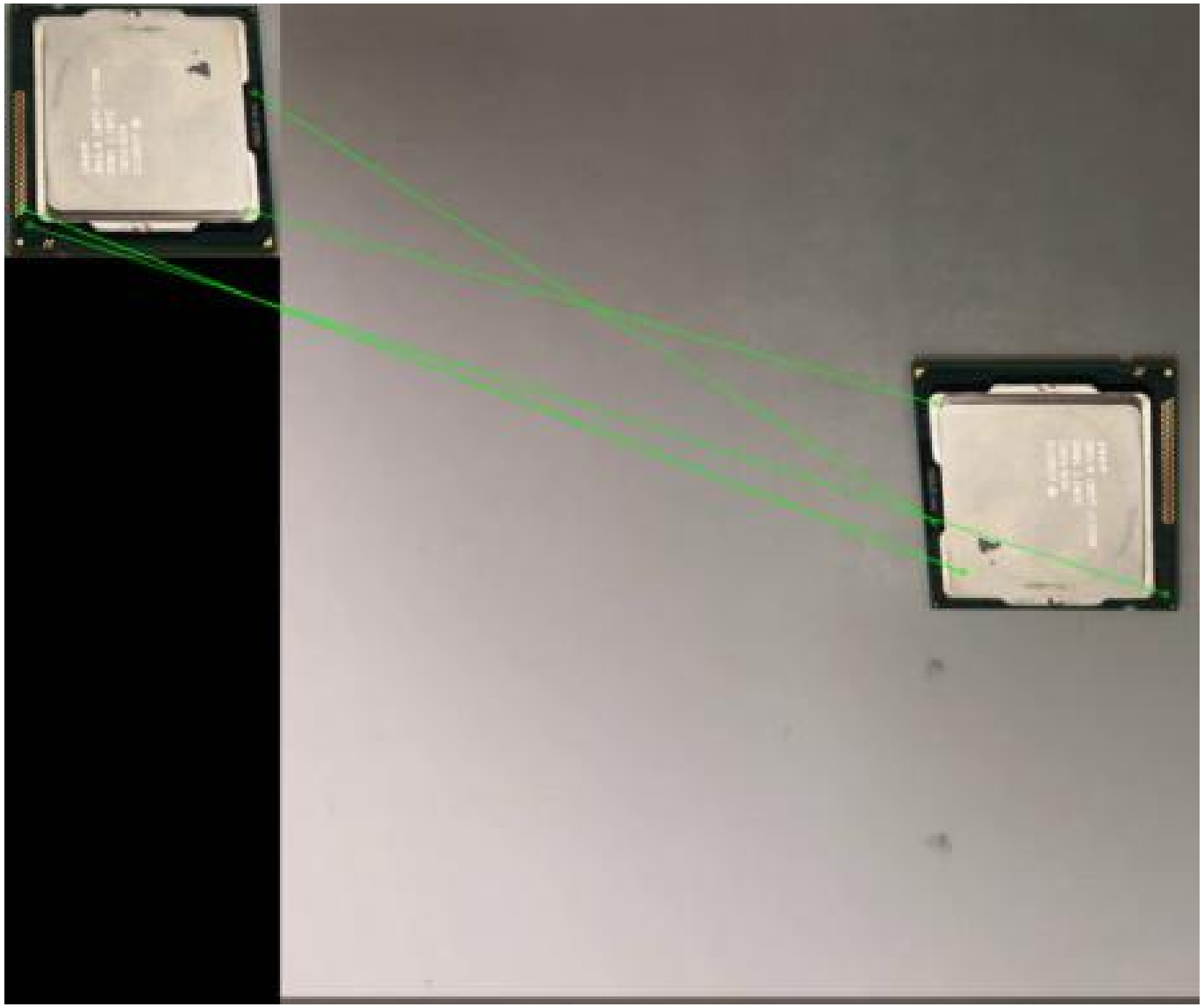
CPS LAB

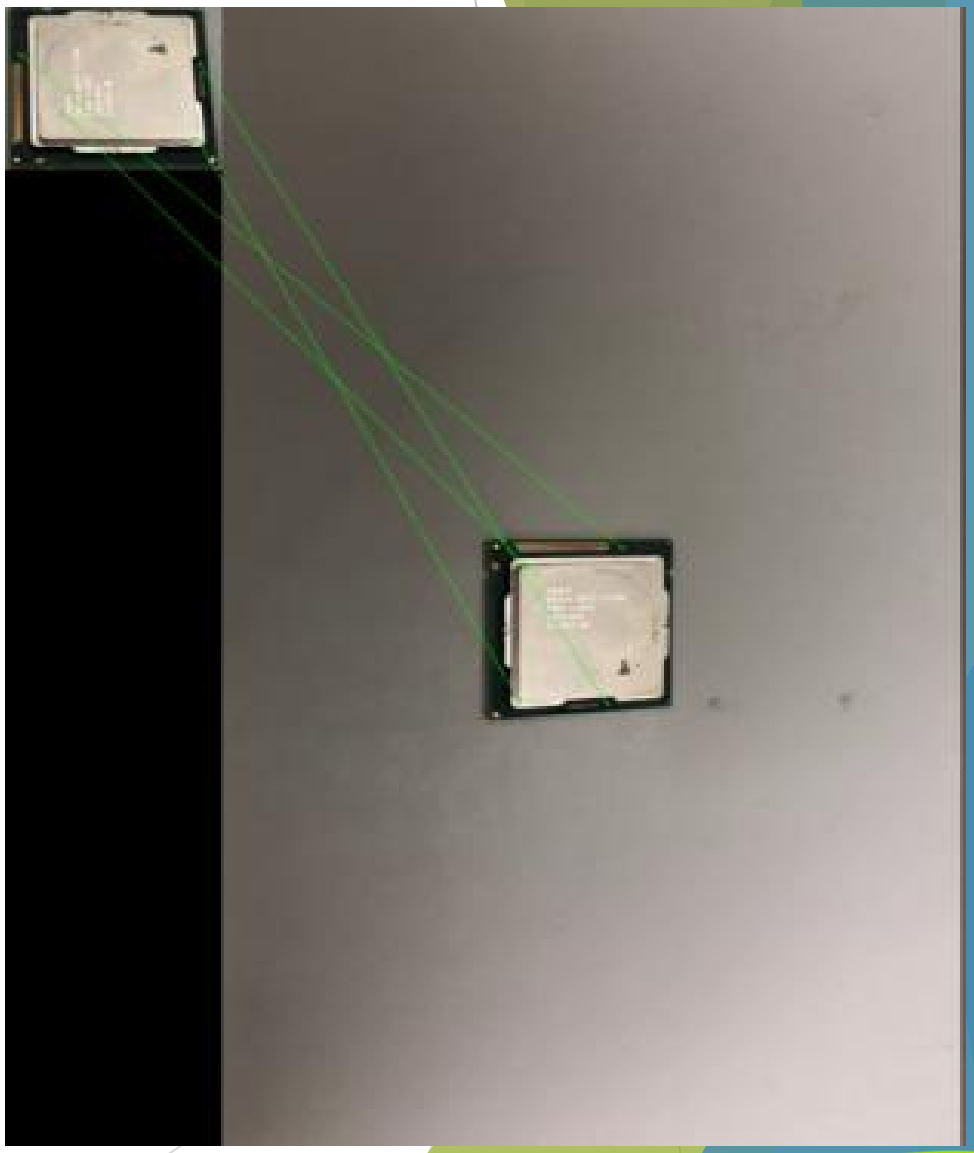
CPU1 特徵點





CPU1 特徵點



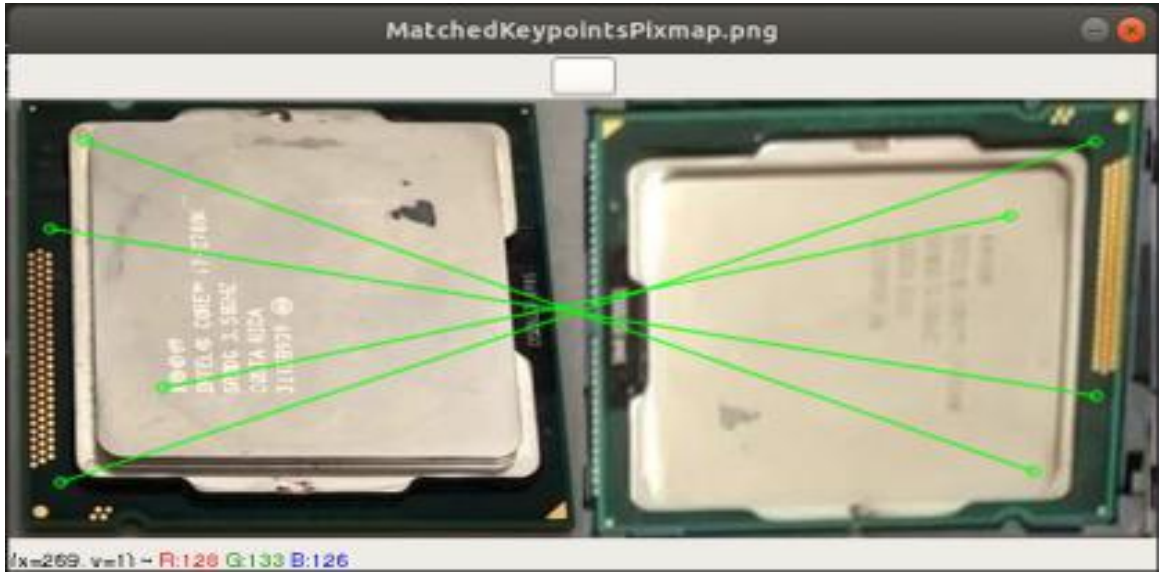


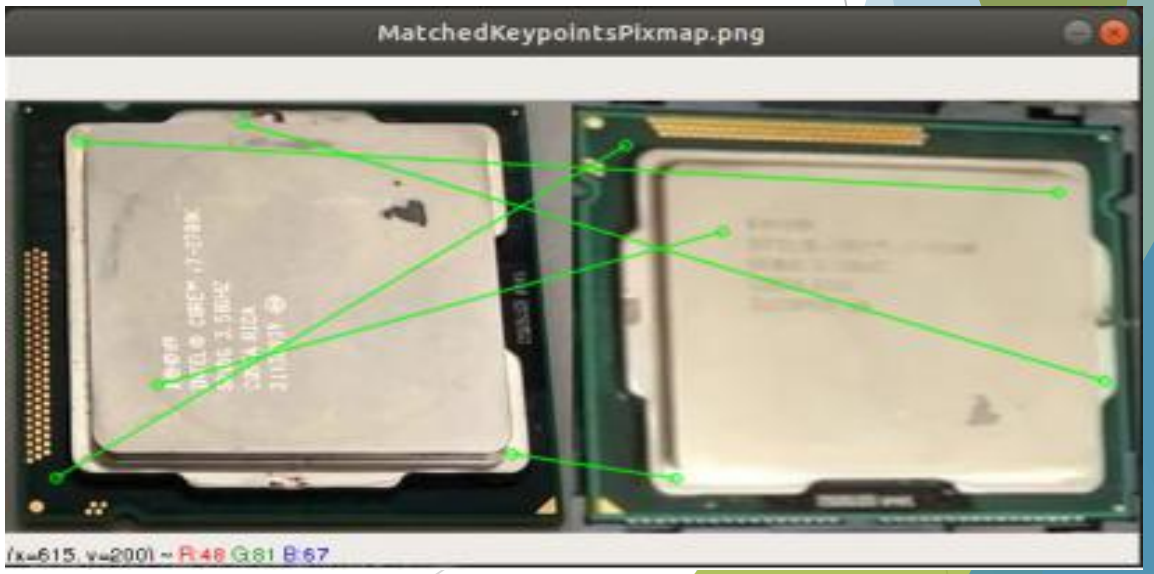
圖片整理成一樣大小

```
178  def Read_Image():
179      # 讀取一張圖作爲匹配模板
180      Template_image = cv2.imread('/home/wolf/桌面/CPU/Template_image/frame00294_1.jpg')
181      # cv2.imshow('Template_image', Template_image)
182      # 讀取一張測試圖
183      Target_image = cv2.imread('/home/wolf/桌面/CPU/Target_image/frame00015_1.jpg')
184      width = Template_image.shape[1]
185      height = Template_image.shape[0]
186      # 將測試圖的大小resize成與模板圖一樣
187      Target_image = cv2.resize(Target_image, (width, height))
188      # cv2.imshow('Target_image', Target_image)
189      return Template_image, Target_image
190
191
192  if __name__ == "__main__":
193      Template_image, Target_image = Read_Image()
194      # 提取關鍵特徵點
195      Keypoints(Template_image, Target_image)
196      # 特徵點匹配，計算角度
197      Matched_Keypoints(Template_image, Target_image)
198
```

CPS LAB

CPU1 特徵點





CPS LAB

SIFT提取特徵點，並排序

```python
def Keypoints_click(Template_image, Target_image):
    # 用SIFT提取特徵點
    sift = cv2.xfeatures2d.SIFT_create()
    # 模板圖特徵點
    kp1 = sift.detect(Template_image, None)
    # 測試圖特徵點
    kp2 = sift.detect(Target_image, None)
    # 排序，取前20個
    tmp_kp_sort1 = sorted(kp1, key=lambda x: x.size, reverse=True)[:20]
    tmp_kp_sort2 = sorted(kp2, key=lambda x: x.size, reverse=True)[:20]
    # 存儲
    keypoints_Template_image, descriptor_Template_image = sift.compute(Template_image, tmp_kp_sort1)
    keypoints.append(keypoints_Template_image)
    descriptors.append(descriptor_Template_image)
    keypoints_Target_image, descriptor_Target_image = sift.compute(Target_image, tmp_kp_sort2)
    keypoints.append(keypoints_Target_image)
    descriptors.append(descriptor_Target_image)
```

CPS LAB

SIFT特徵的匹配

```
79      MIN_MATCH_COUNT = 1
80      # 特征点匹配用的是BFMatcher，brute force暴力匹配，就是选取几个最近的
81      flann = cv2.BFMatcher(cv2.NORM_L2)
82      # 使用KNN算法匹配
83      matches = flann.knnMatch(descriptors[0], descriptors[1], k=2)
84      # 去除错误匹配
85      good = []
86      for m, n in matches:
87          if m.distance < 0.95 * n.distance:
88              good.append(m)
89      # 单应性
90      if len(good) > MIN_MATCH_COUNT:
91          # 改变数组的表现形式，不改变数据内容，数据内容是每个关键点的坐标位置
92          src_pts = np.float32([keypoints[0][m.queryIdx].pt for m in good]).reshape(-1, 1, 2)
93          dst_pts = np.float32([keypoints[1][m.trainIdx].pt for m in good]).reshape(-1, 1, 2)
94          # findHomography 函数是计算变换矩阵
95          # 参数cv2.RANSAC是使用RANSAC算法寻找一个最佳单应性矩阵H，即返回值M
96          # 返回值：M 为变换矩阵，mask是掩模
97          M, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC, 5.0)
98          # ravel方法将数据降维处理，最后并转换成列表格式
99          matchesMask = mask.ravel().tolist()
100     else:
101         print("Not enough matches are found - %d/%d") % (len(good), MIN_MATCH_COUNT)
102         matchesMask = None
103     # 显示匹配结果
104     draw_params = dict(matchColor=(0, 255, 0),  # draw matches in green color
105                        singlePointColor=None,
106                        matchesMask=matchesMask,  # draw only inliers
107                        flags=2)
108     # 畫匹配的特徵點
109     image_Matches = cv2.drawMatches(Template_image, keypoints[0], Target_image, keypoints[1], good, None, **draw_params)
110     cv2.imwrite('/home/wolf/桌面/11111111/MatchedKeypoints.png', image_Matches)
111     cv2.imshow('MatchedKeypoints.png', image_Matches)
```

CPS LAB

SIFT提取匹配特徵點

```
113         pop = []
114         # 將模板圖和測試圖匹配的特徵點提取出來
115         for i in range(len(matchesMask)):
116             if matchesMask[i] == 1:
117                 pop.append([src_pts[i][0][0], src_pts[i][0][1], dst_pts[i][0][0], dst_pts[i][0][1]])
118             i += 1
```

CPS LAB

計算特徵點之間的距離

```
119     # 匹配的特徵點排序，以x從小到大
120     # pop.sort(key=takeSecond)
121     i = 0
122     j = 0
123     k = 0
124     # 指示是否有找到滿足的4個點
125     FLAGE = 0
126     t = True
127     while t:
128         for i in range(0, len(pop)):
129             for j in range(0, len(pop)):
130                 if i == j:
131                     continue
132                 # 根據兩點座標計算直線距離
133                 # 來源於模板圖的兩個點
134                 AB_distance = np.sqrt(np.sum((np.array([pop[i][0], pop[i][1]]) - np.array([pop[j][0], pop[j][1]])) ** 2))
135                 # 來源於測試圖的兩個點
136                 CD_distance = np.sqrt(np.sum((np.array([pop[i][2], pop[i][3]]) - np.array([pop[j][2], pop[j][3]])) ** 2))
137                 # 距離在閾值內，判斷點相似
138                 if abs(AB_distance - CD_distance) <= 15:
139                     k = 1
140                     break
141                 else:
142                     k = 0
143             if k == 1:
144                 t = False
145                 break
146             if i == len(pop) - 1:
147                 t = False
148                 # 沒有有找到滿足的4個點
149                 FLAGE = 1
150                 break
```

CPS LAB

計算角度

```
151         # 直線的點座標矩陣
152         AB = [pop[i][0], pop[i][1], pop[j][0], pop[j][1]]
153         CD = [pop[i][2], pop[i][3], pop[j][2], pop[j][3]]
154         # 直線的方向始終由右往左
155         if pop[i][0] < pop[j][0]:
156             temp = i
157             i = j
158             j = temp
159         # 兩直線的夾角
160         angle_ABCD = angle(AB, CD)
161         # 畫直線
162         angle_Template_image = cv2.arrowedLine(Template_image, tuple(pop[i][0:2]), tuple(pop[j][0:2]), (0, 255, 0), 2)
163         angle_Target_image = cv2.arrowedLine(Target_image, tuple(pop[i][2:4]), tuple(pop[j][2:4]), (0, 0, 255), 2)
164         # 模板圖和測試圖合併爲一張
165         image = np.concatenate((angle_Template_image, angle_Target_image), axis=1)
166         # 夾角在閾值內，且找到了４個點，說明方向相同
167         if angle_ABCD < 30 and FLAGE == 0:
168             text = 'OK'
169         else:
170             text = 'No'
171         text = "{:.3f}    :    {}".format(angle_ABCD, text)
172         # 圖片上顯示結果
173         cv2.putText(image, text, (5, 10), cv2.FONT_HERSHEY_COMPLEX, fontScale=0.3, color=(0, 255, 255), thickness=1)
174         cv2.imshow(text, image)
175         cv2.imwrite('/home/wolf/桌面/11111111/image.png', image)
176         cv2.waitKey(0)
```

S LAB

CPU1方向


MatchedKeypointsPixmap.png
1.598 : OK
MatchedKeypointsPixmap.png
98.278 : No

CPU1方向


MatchedKeypointsPixmap.png
114.769 : No
MatchedKeypointsPixmap.png
91.939 : No

CPU2方向

CPS LAB

CPU2方向


MatchedKeypointsPixmap.png
166.278 : No
MatchedKeypointsPixmap.png
59.578 : No

CPU3方向

CPS LAB

CPU3方向

CPS LAB