目　　录

1 剪刀石头布（手势跟踪识别应用）

1.1 任务提出

基于 OpenCV 的动态手势跟踪识别，根据分类识别手势，并依据“剪刀石头布”规则瞬间做出赢人类玩家的反应的程序。

就是一个机器人跟你玩剪刀石头布，当你快出来的时候，机器识别你的手势，迅速给出反应。

1.2 功能（步骤）说明

1. 滤波去噪

由于边缘检测容易受到噪音影响，所以第一步是使用高斯滤波器去除噪声。

2. 去除背景，提取手部的轮廓

2.1 方法一：肤色检测手部区域

利用肤色来检测手部是手部检测最直接的方法。皮肤颜色稳定，不轻易受到缩放、平移和旋转的影响，且对图像的尺寸、拍摄方向和角度的依赖性较小

颜色空间主要有两种，分别是 YCbCr 颜色空间和 HSV 颜色空间，根据 Enamin D. Zarit 等人对肤色在这些彩色空间分布的研究，以及在检测中性能的分析，本文通过对手部皮肤颜色进行特征分析(选取黄种人的肤色),选用HSV颜色空间进行手部区域检测。

RGB 颜色空间和 YCbCr 颜色空间的混合肤色检测器。像素值满足如下条件：

$$\begin{cases} R > G \wedge R > B \\ (G \geq B \wedge 5R - 12G + 7B \geq 0) \vee (G < B \wedge 5R + 7G - 12B \geq 0) \\ C_r \in (135, 180) \wedge C_b \in (85, 135) \wedge Y > 80. \end{cases}$$

然而，利用 python 编程的结果很卡顿。

结论分析：有点卡顿，这与 python 计算能力有关，若用 c++，则没有此现象，故使用方法二。

此方法也有一个技术难点就是生成新的二值图像时，需要注意图像格式的问题。后来我使用了深复制的思想，完整地实现了。

2.2 方法二：灰度图像，阈值分割：

图像阈值化分割是一种传统的最常用的图像分割方法，因其实现简单、计算量小、性能较稳定而成为图像分割中最基本和应用最广泛的分割技术。它特别适用于目标和背

景占据不同灰度级范围的图像。难点在于如何选择一个合适的阈值实现较好的分割。

我们将 RGB 图像转变为灰度图像，便于后续处理。

2.3　方法三：k-means 分割

Kmeans 是最简单的聚类算法之一，应用十分广泛，Kmeans 以距离作为相似性的评价指标，其基本思想是按照距离将样本聚成不同的簇，两个点的距离越近，其相似度就越大，以得到紧凑且独立的簇作为聚类目标。

缺点：易受光源影响，参数需根据光源修改。

3. 找出轮廓：

利用 Opencv 中通过使用 findContours 函数，简单几个的步骤就可以检测出物体的轮廓。

4. 凸包

凸包(凸壳)是指如果在集合 A 内连接任意两个点的直线段都在 A 的内部，则称集合 A 是凸形的。简单点理解，就是一个多边型，没有凹的地方。凸包(凸壳)能包含点集中所有的点

5. 求 moment，后求质心（用 mean shift 的方法求质心）

计算手指的个数，来识别手势特征并进行跟踪。首先要提取手掌轮廓，计算轮廓形状特征有：轮廓的质心、轮廓的最短最长径长、轮廓的外接圆(圆心和半径)、轮廓的周长和面积、轮廓在图像中的矩形框位置、轮廓的外包络点集合、轮廓的点集合、轮廓的各阶矩、轮廓的有效的特征向量的提取、手指指尖的定位。手的位置特征是指手掌的质心位置，针对二维图像，质心位置是可以通过计算零阶距和 X、Y 的一阶距得到的。假设二值化之后的图像为 I(X,Y)，质心(x_c,y_c)计算公式如下：

$$M_{00} = \sum_x \sum_y I(x,y) \quad (6)$$

$$M_{10} = \sum_x \sum_y xI(x,y) \quad (7)$$

$$M_{01} = \sum_x \sum_y yI(x,y) \quad (8)$$

$$x_c = \frac{M_{10}}{M_{00}}, y_c = \frac{M_{01}}{M_{00}} \quad (9)$$

6. 标出手指和手掌

质心处即为手掌，手指求法如下：

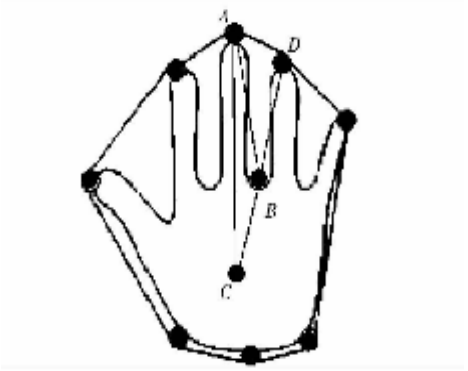
经过实验测试具体的指尖筛除条件如下:

(1) 当 $AC > BC$ 时 $0.1R \leqslant BC \leqslant 1.3R$; 当 $AC < BC$ 时 $0.1R \leqslant AC \leqslant 1.3R$;

(2) $MIN(BC, AC) / MAX(BC, AC) \leqslant 0.8$;

(3) $AB \geqslant 10$, $BD \geqslant 10$, $MAX(AB, BD) / MIN(AD, BD) \geqslant 0.8$。

将满足以上条件的指尖点存储到点集 A_p 中,然后以 A_p 中的点 P_i 为原点,在手部轮廓中各取 P_i 之前和之后的 j 个点,并计算这 $2j$ 个点的 K 曲率值,即向量 A_iA_{i-j} 与 A_iA_{i+j} 之间的夹角的余弦值。若点 P_i 的 K 曲率值小于 60°,并且是 $2j$ 个点中 K 曲率值最小的点,则其为准确的指尖点。



7. 判断手势和形状

7.1 方法一：特征点

把提取的特征点和手势字典进行对比，然后判断手势和形状

7.2 方法二：手指的个数

根据图像中凹凸点中的 (开始点, 结束点, 远点)的坐标, 利用余弦定理计算两根手指之间的夹角, 其必为锐角, 根据锐角的个数判别手势。

7.3 方法三：轮廓的匹配程度

函数 cv2.matchShape() 可以帮我们比较两个形状或轮廓的相似度。如果返回值越小，匹配越好。它是根据 Hu 矩来计算的。

8. 输出结果：

动态输出克制当前输出的手势（依据角刀石头布规则）.

1.3 实验结果分析

结论分析：用 python 写肤色检测程序的话，效果有点卡顿，这与 python 计算能力有关，若用 c++，则没有此现象，故使用阈值分割的方法代替。最终效果没有延迟，检测效果,能够迅速检测出手势,并且计算出手指的个数和手掌质心位置,与结果想匹配。此程序系统鲁棒性很好。

附件 A 程序框图

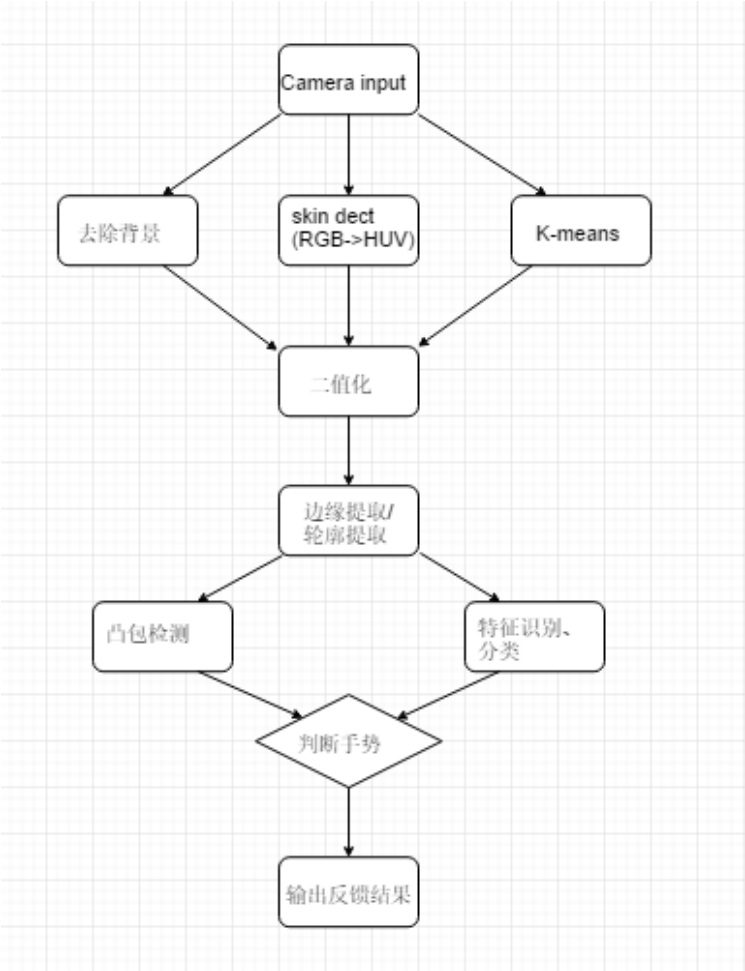


图 9 程序框图

附件 B 程序源码

以下第一份代码是使用了灰度图像，阈值分割和手指个数的方法。

```
1. import cv2
2. import numpy as np
3. import math
4. ##输入结果库
5. pic_1 = cv2.imread('v1.png')
6. pic_2 = cv2.imread('v2.png')
7. pic_3 = cv2.imread('v3.png')
8. ##摄像机输入
9. cap = cv2.VideoCapture(0)
```

```
while( cap.isOpened() ) :
        ret,img = cap.read()
        gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray,(5,5),0)
##阈值分割
        ret,thresh1 = cv2.threshold(blur,70,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)

aa,contours, hierarchy = cv2.findContours(thresh1,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
##深复制
        drawing = np.zeros(img.shape,np.uint8)

        max_area=0
       ##找轮廓
        for i in range(len(contours)):
                cnt=contours[i]
                area = cv2.contourArea(cnt)
                if(area>max_area):
                        max_area=area
                        ci=i
        cnt=contours[ci]
        hull = cv2.convexHull(cnt)#0621
        ##meanshift 求质心
        moments = cv2.moments(cnt)
        #print len(cnt)
        #print hull
       ##求质心公式
        if moments['m00']!=0:
                        cx = int(moments['m10']/moments['m00']) # cx = M10/M00
                        cy = int(moments['m01']/moments['m00']) # cy = M01/M00

        centr=(cx,cy)
        cv2.circle(img,centr,5,[0,0,255],2)
        #cv2.circle(img,centr,5,[0,255,255],2)#0621
        #cv2.rectangle(original, p1, p2, (77, 255, 9), 1, 1)#0621

        cv2.drawContours(drawing,[cnt],0,(255,255,0),2)
        #cv2.drawContours(drawing,[hull],0,(0,0,255),2)

        cnt = cv2.approxPolyDP(cnt,0.01*cv2.arcLength(cnt,True),True)
        hull = cv2.convexHull(cnt,returnPoints = False)

ndefects = 0 #0622 for counting finger_number
```

```
52. ###根据图像中凹凸点中的 (开始点, 结束点, 远点)的坐标, 可利用余弦定理计算两根手指之间的夹角,
53.         if(1):
54.                         defects = cv2.convexityDefects(cnt,hull)
55.                         #mind=0
56.                         #maxd=0
57.                         for i in range(defects.shape[0]):
58.                                 s,e,f,d = defects[i,0]
59.                                 start = tuple(cnt[s][0])
60.                                 end = tuple(cnt[e][0])
61.                                 far = tuple(cnt[f][0])
62.                                 #dist = cv2.pointPolygonTest(cnt,centr,True)
63.                 a = np.sqrt(np.square(start[0]-end[0])+np.square(start[1]-end[1]))#0622
64.                 b = np.sqrt(np.square(start[0]-far[0])+np.square(start[1]-far[1]))#0622
65.                 c = np.sqrt(np.square(end[0]-far[0])+np.square(end[1]-far[1]))#0622
66.
67.
68.                                 angle      = math.acos((b ** 2 + c ** 2 - a ** 2) / (2 * b * c)) # * 57#0622
69.         ##其必为锐角, 根据锐角的个数判别手势
70.                                 if angle <= math.pi/2 :#90:#0622
71.                                         ndefects = ndefects + 1#0622
72.
73.                                 #cv2.line(img,start,end,[0,255,255],2)
74.                                 cv2.line(img,start,centr,[0,255,255],2)
75.                                 cv2.circle(img,start,20,[0,255,255],4)
76.                                 #cv2.circle(img,end,20,[0,255,0],4)
77.                                 cv2.circle(img,far,5,[0,0,255],-1)
78.                         #print(i)
79.                 print ndefects
80.                         i=0
81.                 if ndefects == 0:
82.                 print 07
83.                 cv2.imshow("RESULT",pic_3)
84.                 else:
85.             if ndefects == 1:
86.                 print 27
87.                 cv2.imshow("RESULT",pic_2)
88.             else:
89.                 if ndefects == 4:
90.                     print 57
91.                     cv2.imshow("RESULT",pic_1)
92.                 else:
93.                     print 0000
```

```
94.
95.         cv2.imshow('output',drawing)
96.         cv2.imshow('input',img)
97.
98.         k = cv2.waitKey(10)
99.  #Esc
100.        if k == 27:
101.            break
102.
```

以下这份代码是用了肤色检测（将 RGB 空间转换到 Ycbcr 空间）的方法。

```
1   .import cv2
2   import numpy as np
3
4   cap = cv2.VideoCapture(0)
5   while( cap.isOpened() ) :
6    ret,img = cap.read()
7    # load an original image
8    #img = cv2.imread(imgFile)
9    rows,cols,channels = img.shape
10   # convert color space from rgb to ycbcr
11   imgYcc = cv2.cvtColor(img, cv2.COLOR_BGR2YCR_CB)
12
13   # convert color space from bgr to rgb
14   img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
15
16   # prepare an empty image space
17   imgSkin = np.zeros(img.shape, np.uint8)
18   # copy original image/深复制
19   imgSkin = img.copy()
20   for r in range(rows):
21         for c in range(cols):
22
23        # non-skin area if skin equals 0, skin area otherwise
24        skin = 0
25        # get values from rgb color space
26        R = img.item(r,c,0)
27        G = img.item(r,c,1)
28        B = img.item(r,c,2)
29
30        # get values from ycbcr color space
31        Y = imgYcc.item(r,c,0)
32        Cr = imgYcc.item(r,c,1)
33        Cb = imgYcc.item(r,c,2)
34
```

```
35
36        # skin color detection
37
38        if R > G and R > B:
39                    if (133 <= Cr and Cr <= 173) and (77 <= Cb and Cb <= 127):
40                          skin = 1
41                          # print 'Skin detected!'
42
43        if 0 == skin:
44              imgSkin.itemset((r,c,0),0)
45              imgSkin.itemset((r,c,1),0)
46              imgSkin.itemset((r,c,2),0)
47        if 1 == skin:
48              imgSkin.itemset((r,c,0),255)
49              imgSkin.itemset((r,c,1),255)
50              imgSkin.itemset((r,c,2),255)
51
52  cv2.imshow("RESULT",imgSkin)
53
54  k = cv2.waitKey(10)
55  if k == 27:
56        break
57
```