Visual Combination Lock

1.1 Step 1 (5 points) Domain engineering step

1. Capture Image:
   - Used iPhone's camera.
   - Used a bright light source directly above the hands to reduces shadows.
   - Each image should contain only one hand.
   - Used a black background to maximize contrast to skin.
2. Environment:
   - Hardware: Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz, RAM 8GB
   - OS: Windows 10
   - Language: Python 3.9
   - Packages: OpenCV, NumPy
3. Library:
   - There are 38 pictures in total. The role of the picture is mainly to test and complete the lock.
4. Images:
   - The image will be displayed in the later step

1.2 Step 2 (5 points) Data reduction step.

1. Design

1.1 Detect Skin(binary image):

Used YCrCb color space to detect skin.

If the skin is mapped to YCrCb space, the pixel points of the skin in YCrCb space are approximated as an elliptical distribution. So if we get an ellipse of CrCb, for a point with coordinates (Cr, Cb), we only need to determine whether it is inside the ellipse (including the boundary) to know whether it is a skin-colored point or not.

```
def detectSkin(img):
    YCrCb = cv2.cvtColor(img, cv2.COLOR_BGR2YCR_CB)
    (y, cr, cb) = cv2.split(YCrCb)
    cr = cv2.GaussianBlur(cr, (5,5), 0)
    _, skin = cv2.threshold(cr, 0, 255, cv2.THRESH_BINARY +
cv2.THRESH_OTSU)
    return skin
```

1.2 Determine "fist" or "splay" or "unknown":

Use the ratio of the convex hull area to the actual area of the hand to distinguish between fist and splay

If the ratio is between 0.65 and 0.85, then it is splay. If the ratio is larger than or equal to 0.85, it is fist. Otherwise it is unknown.

```
def detectGesture(contours, hull, img):
    handArea = cv2.contourArea(contours)
```

```
    hullArea = cv2.contourArea(hull)
    areaRatio = handArea / hullArea

    if areaRatio < 0.85 and areaRatio > 0.65:
        gesture = "splay"
    elif   areaRatio >= 0.85:
        gesture = "fist"
    else:
        return "unknown"
    return gesture
```

1.3 Detect Position:

Divided the picture into 9 parts.

Used the center coordinates of the hand to determine it' s position.

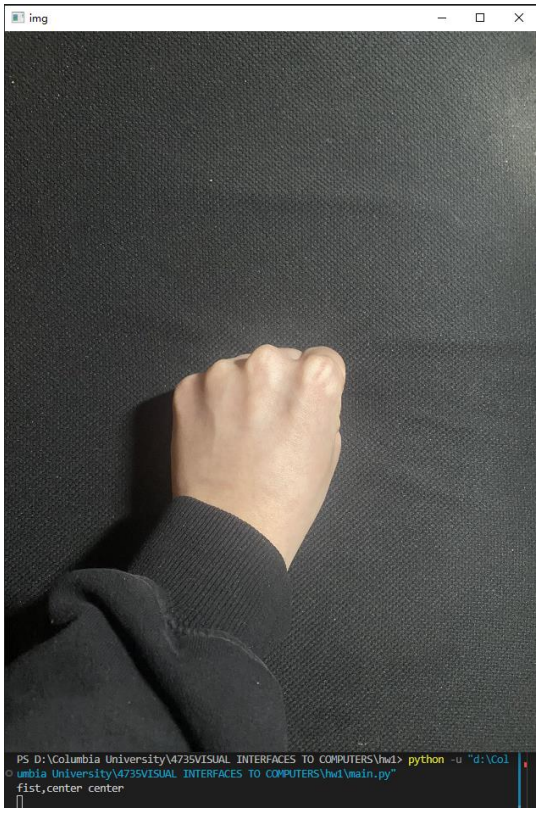
The coordinates at the boundary are considered to be unknown.

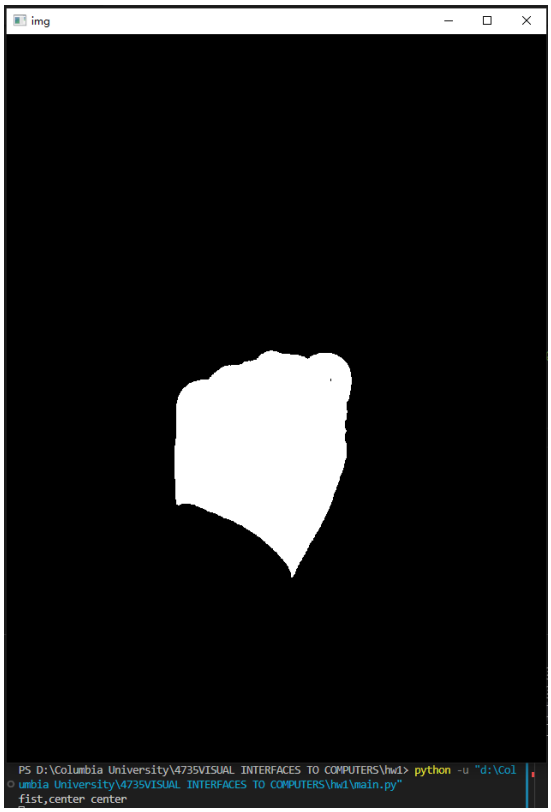
```
def detectPosition(contours, height, width):
    x, y, w, h = cv2.boundingRect(contours)

    centerX = x + w / 2
    centerY = y + h / 2

    heightLine = height / 3 #567
    widthLine = width / 3 #425

    lineY = 7 * heightLine / 8 #425
    lineX = 7 * widthLine / 8 #319

    if centerX <= lineX:
        if centerY <= lineY:
            return "upper left"
        elif centerY <= heightLine + lineY and centerY >= 2 * heightLine - lineY:
            return "center left"
        elif centerY >= 3 * heightLine - lineY:
            return "lower left"
        else:
            return "unknown"
    elif centerX <= widthLine + lineX and centerX >= 2 * widthLine - lineX:
        if centerY <= lineY:
            return "upper center"
        elif centerY <= heightLine + lineY and centerY >= 2 * heightLine - lineY:
```

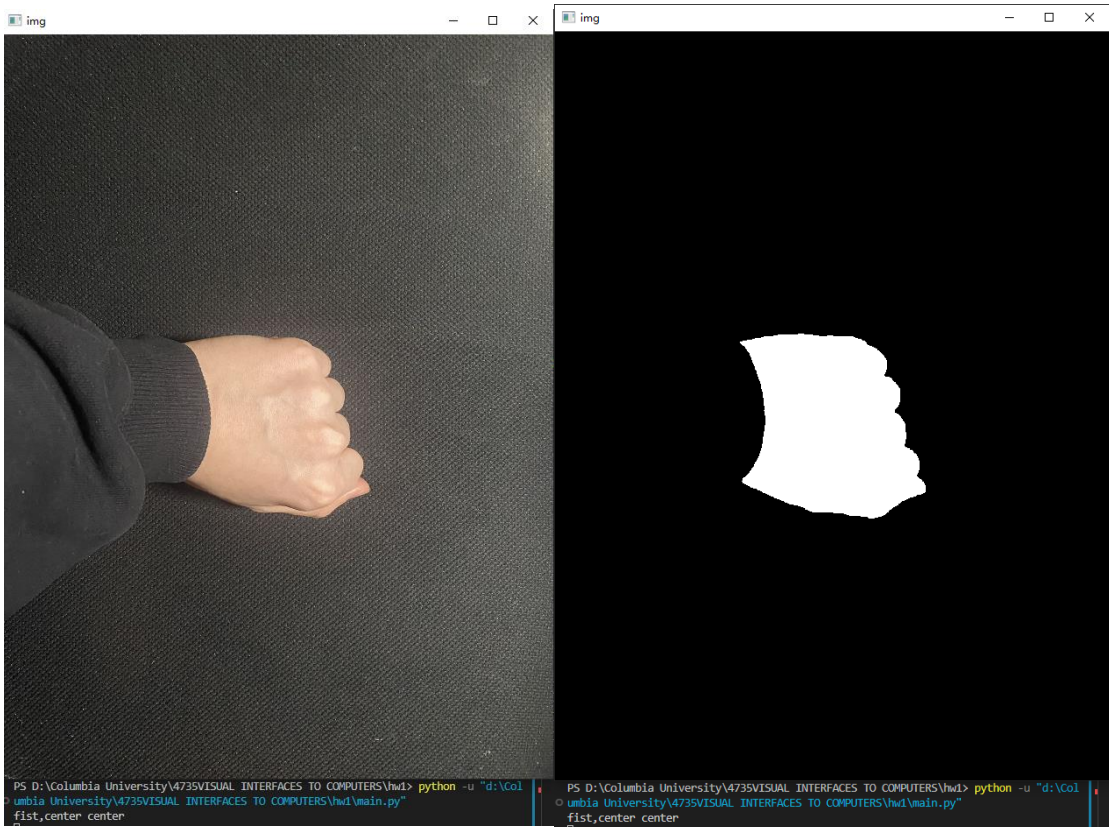
```
            return "center center"
        elif centerY > 3 * heightLine - lineY:
            return "lower center"
        else:
            return "unknown"
    elif centerX >= 3 * widthLine - lineX:
        if centerY <= lineY:
            return "upper right"
        elif centerY <= heightLine + lineY and centerY >= 2 * heightLine - lineY:
            return "center right"
        elif centerY >= 3 * heightLine - lineY:
            return "lower right"
        else:
            return "unknown"
    else:
        return "unknown"
```

2. Test

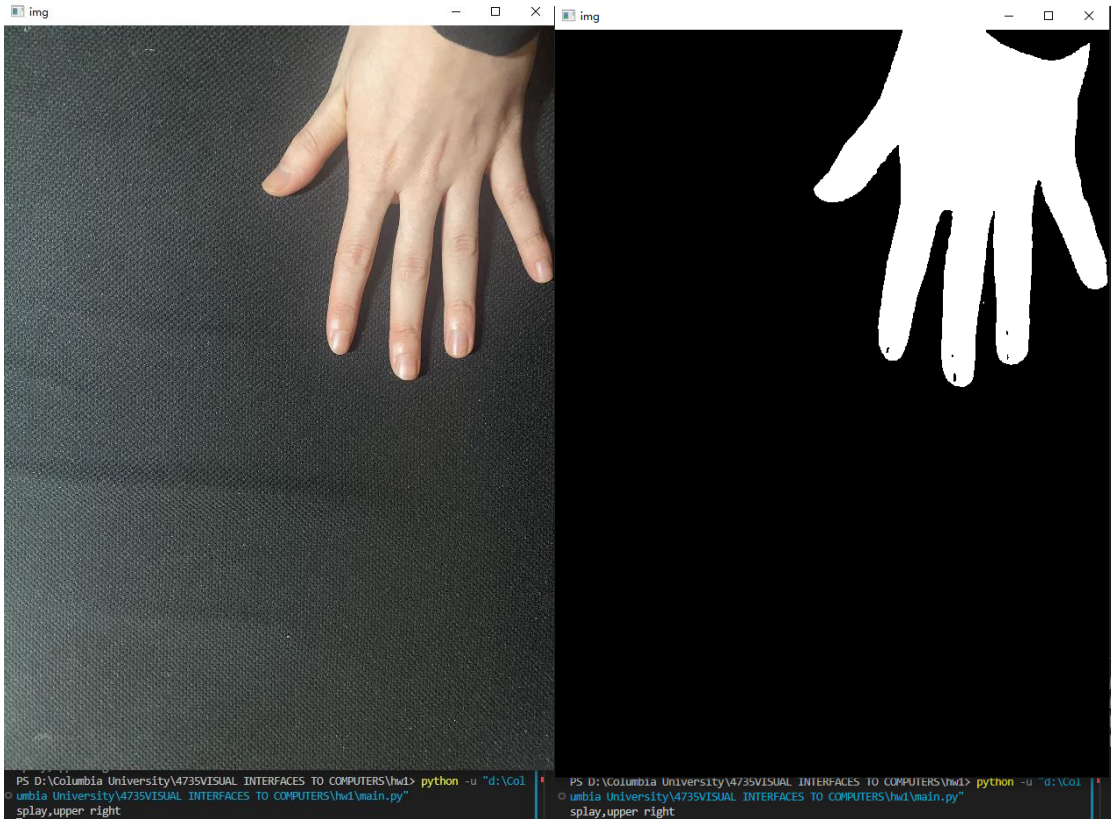
2.1 Images of “fist, center”:

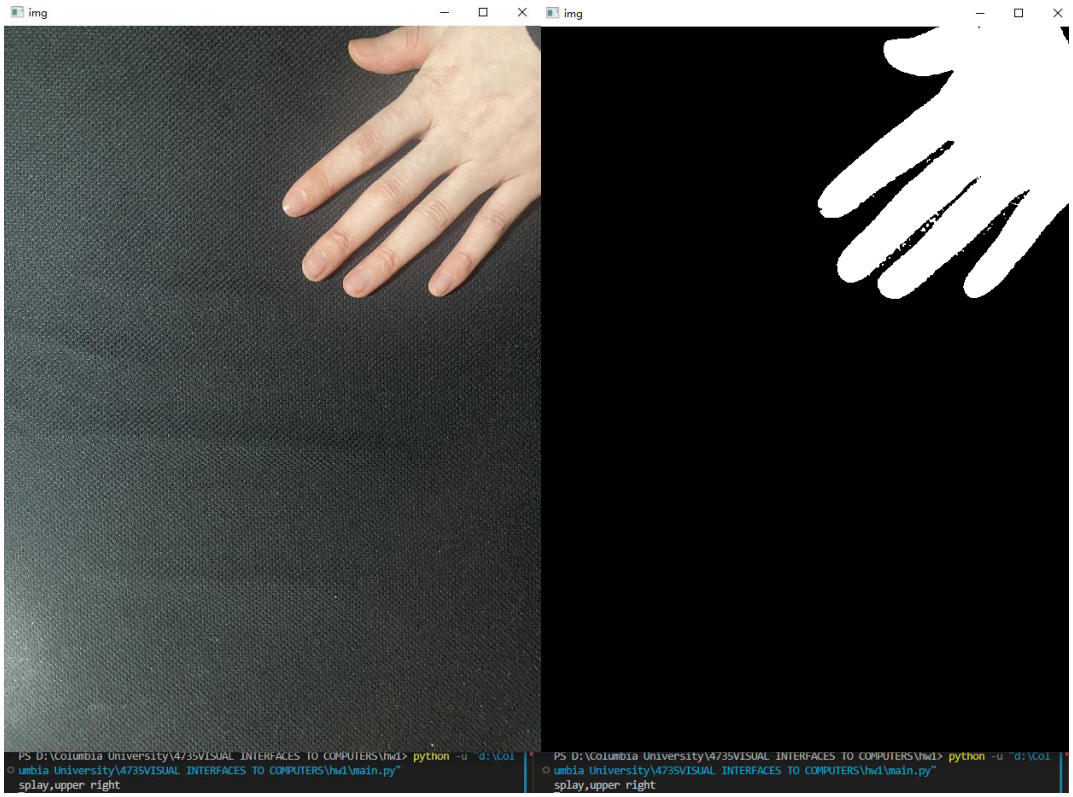






2.2 Images of “splay, upper right”:



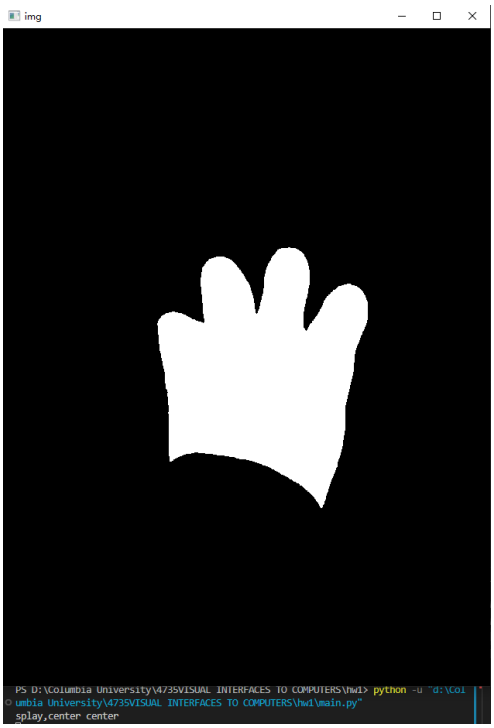


1.3 Step 3 (5 points) Edge cases and extension.

First:

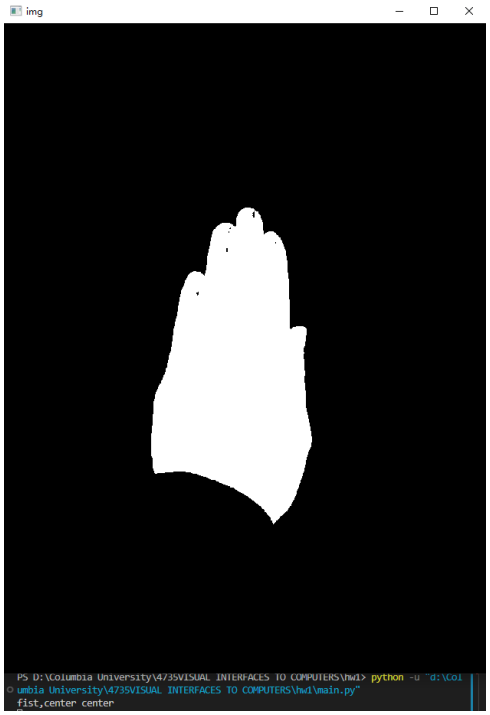
1. One image of a centered fist that is not recognized as “fist, center” (false negative)

Since the fist is not held tightly, there is a large gap between the fingers. So it causes the area ratio to become smaller and is considered to be splay.



2. One image that is incorrectly recognized as “fist, center” (false positive)

Since the area of the convex hull of the fist and palm(unknown) is almost the same as the actual area of itself. So the palm(unknown) is mistaken for the fist.

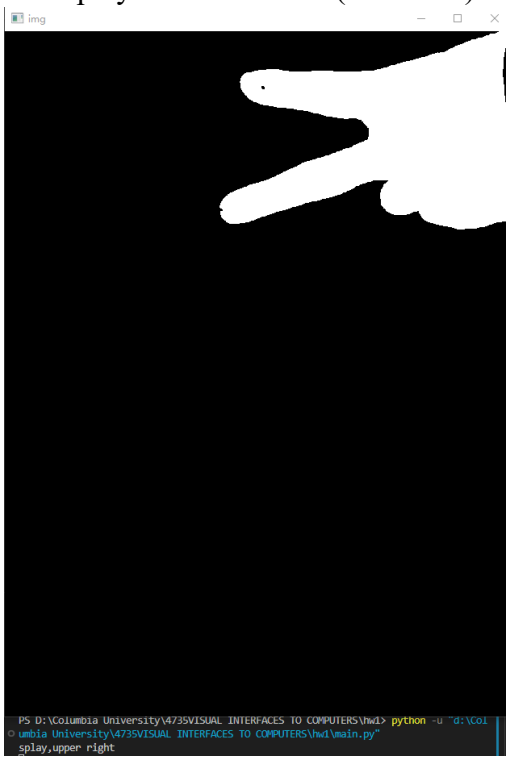


3. One image of a cornered splay that is incorrectly recognized as “splay, upper right” (false negative)

Since the gap between the fingers is small, the area of the convex hull of the splay is almost the same as the actual area of itself. So the splay is mistaken for the fist.

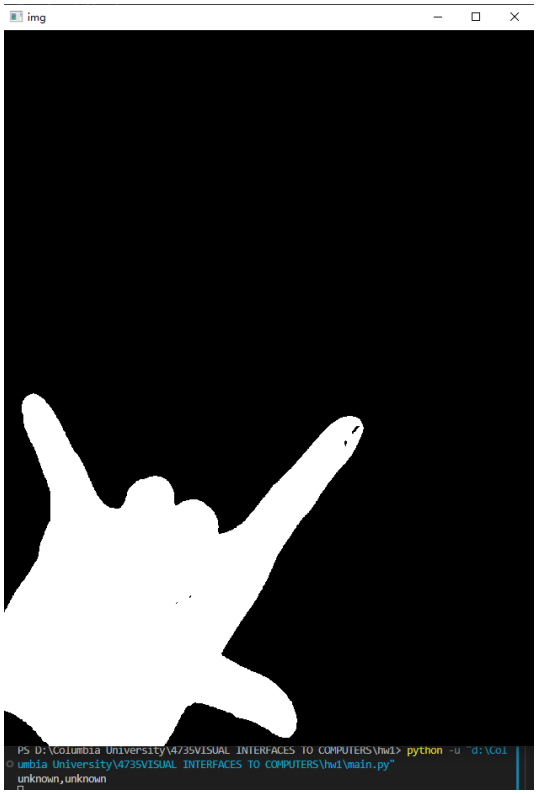
4. One image that is incorrectly recognized as “splay, upper right” (false positive)

Since the area of the convex hull of scissors is much larger than the actual area, which situation is similar to splay. So the scissors(unknown) is mistaken for the splay.

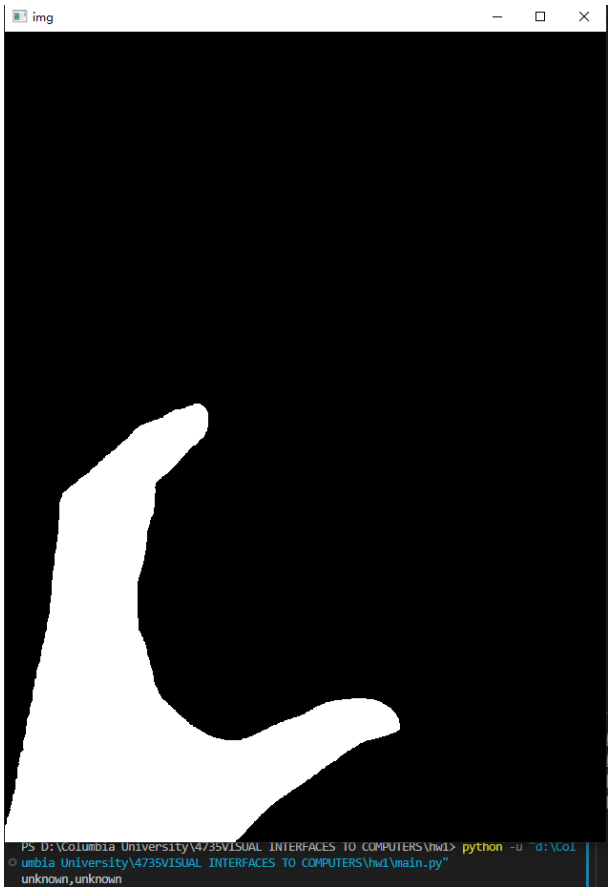


5. Two images that are accurately recognized as “unknown”, that is, neither “fist” nor “splay”(true negative)

5.1 In this gesture, since the area of the convex hull is much larger than the actual area and exceeds the ratio of the splay. So it is recognized as “unknown”



5.2 In this gesture, since the area of the convex hull is much larger than the actual area and exceeds the ratio of the splay. So it is recognized as “unknown”



Second: Extend the “what” vocabulary to include “palm”.

Determine “fist” or “splay” or “palm” or “unknown”:

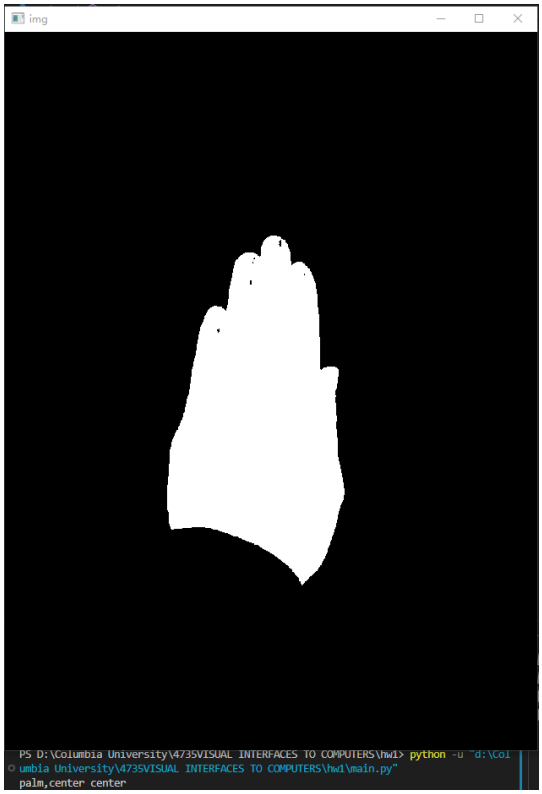
Use the ratio of the convex hull area to the actual area of the hand to distinguish between splay and fist, palm. The area ratio of splay should between 0.65 and 0.85. The area ratio of fist or palm should between 0.85 and 1.0

Use aspect ratio to distinguish between fist and palm. The aspect ratio of fist should larger than 0.7. The aspect ratio of palm should between 0.4 and 0.7.

```
def detectGesture(contours, hull, img):
    handArea = cv2.contourArea(contours)
    hullArea = cv2.contourArea(hull)
    areaRatio = handArea / hullArea

    x, y, w, h = cv2.boundingRect(contours)
    aspectRatio = float(w) / h

    if areaRatio < 0.85 and areaRatio > 0.65:
        if aspectRatio > 0.6:
            gesture = "splay"
```

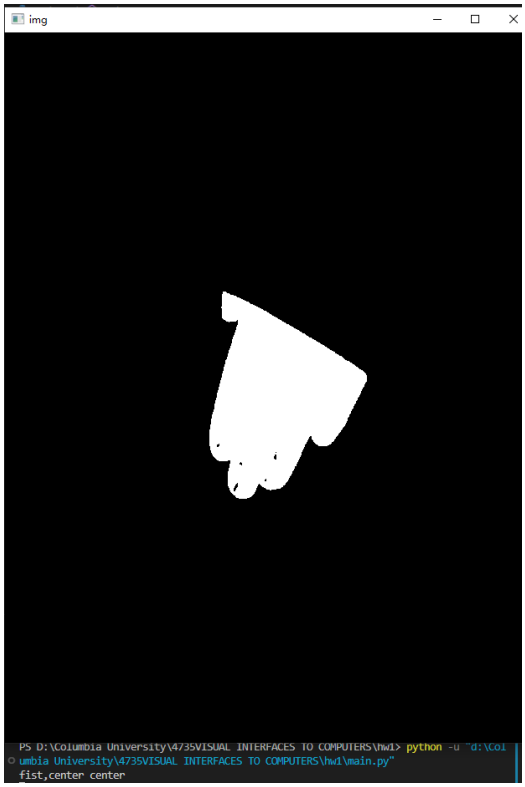
```
        else:
            gesture = "unknown"
    elif areaRatio >= 0.85 and areaRatio < 1.0:
        if aspectRatio > 0.7:
            gesture = "fist"
        elif aspectRatio > 0.4:
            gesture = "palm"
        else:
            gesture = "unknown"
    else:
        gesture = "unknown"
    return gesture
```

1. Two images of a palm that are correctly recognized as “palm”. (true positive)



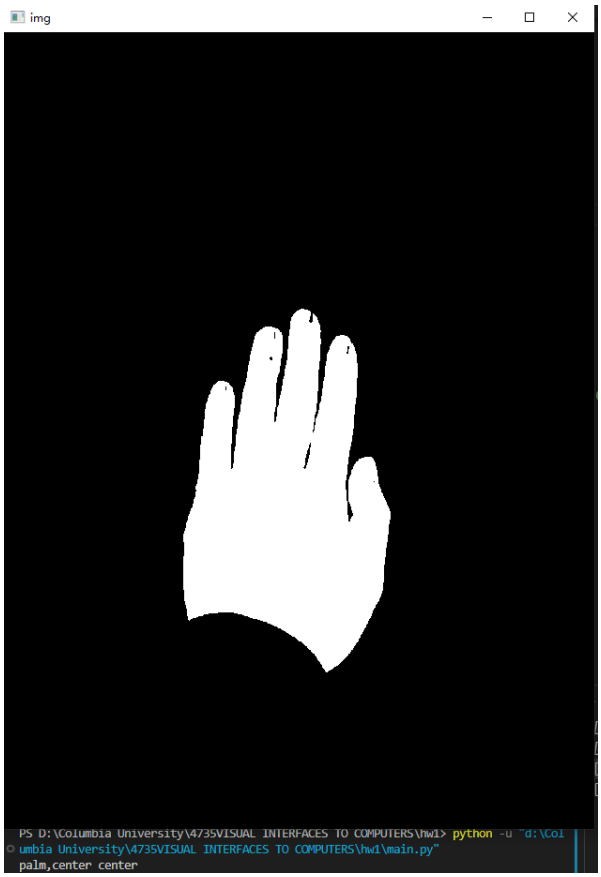
2. One image of a palm that is incorrectly recognized as “fist” (false negative)

Since only the fingers of the palm are exposed, the aspect ratio of palm is similar to fist. And the area ratio of palm is similar to fist. So the palm is mistaken for the fist.



3. One image of “splay” that is incorrectly recognized as “palm” (false positive)

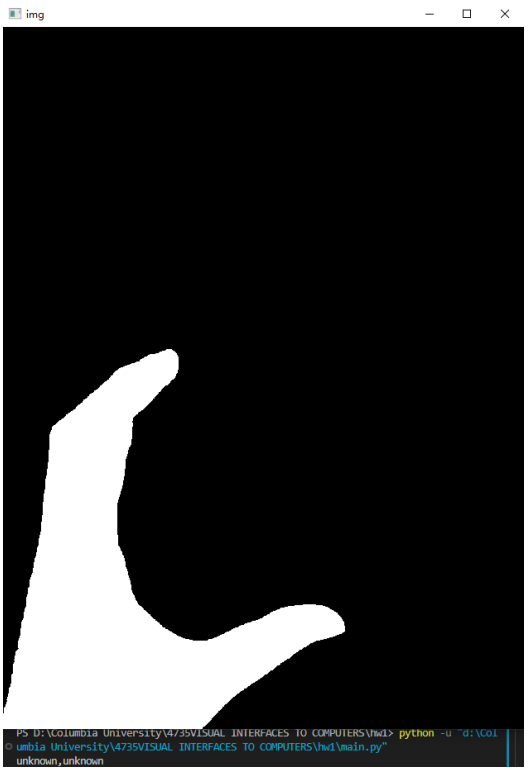
Since the gap between the fingers is small, the area of the convex hull of the splay is almost the same as the actual area of itself. And the aspect ratio of splay is similar to palm. So the splay is mistaken for the palm.



4. Two images that are accurately recognized as “unknown”, that is, neither “fist” nor “splay” nor “palm” (true negative)

4.1 In this gesture, since the area of the convex hull is much larger than the actual area and the area ratio is not between fist, palm and splay. So it is recognized as “unknown”

4.2 In this gesture, since the area ratio is similar to splay, but the aspect ratio is bigger than splay. So it is recognized as “unknown”



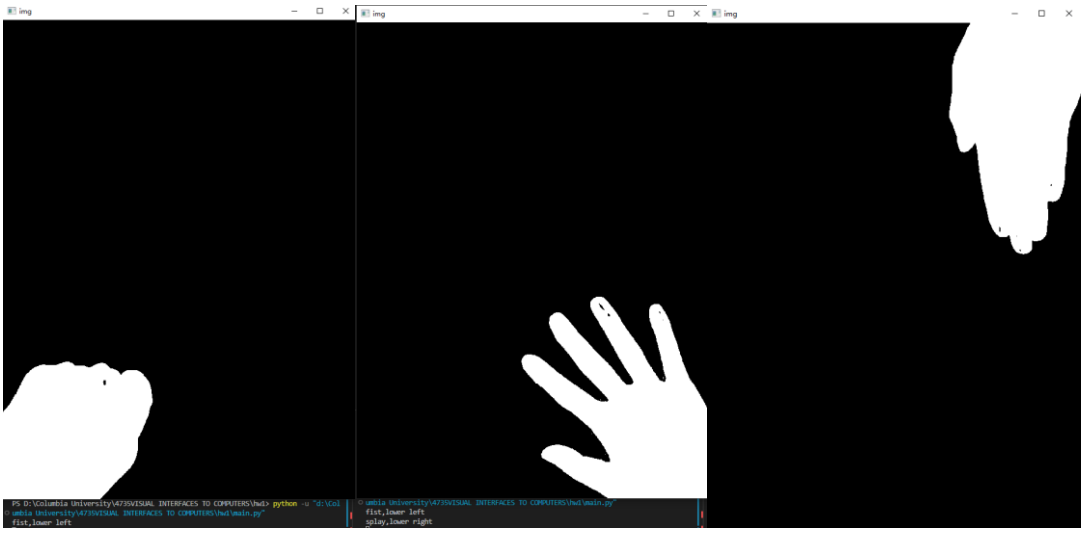
1.4 Step 4 (5 points) Evaluation step

1. Explain how you determined the three sequences

1.1 Easy Sequence

The system can distinguish between standard splay, fists, and palm. So, these three gestures will be used. In addition, can better distinguish between four positions: upper right, upper left, lower left and lower right. So, three of these positions will be used.

lock = [["fist","lower left"],["splay","lower right"],["palm","upper right"]]



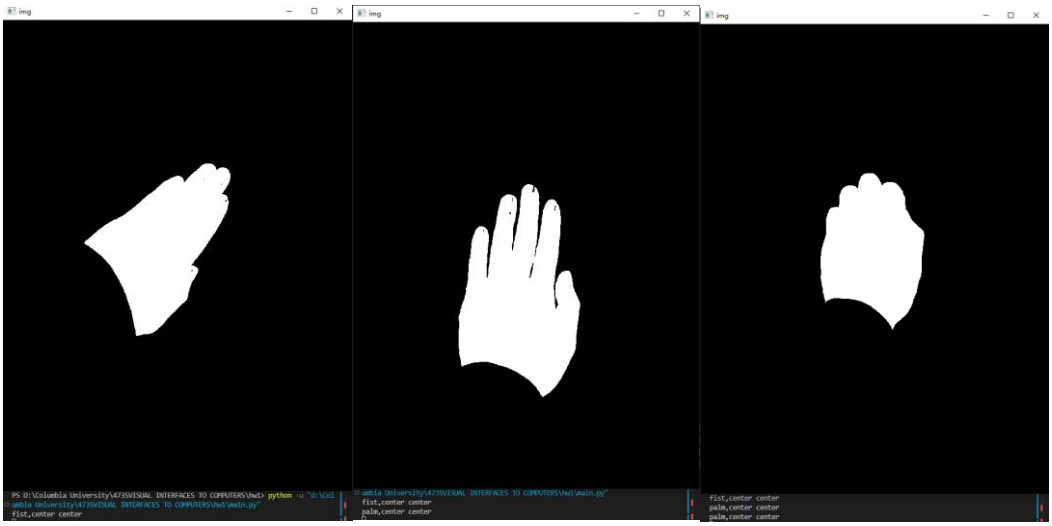
```
PS D:\Columbia University\4735VISUAL INTERFACES TO COMPUTERS\hw1> python -u "d:\Col
umbia University\4735VISUAL INTERFACES TO COMPUTERS\hw1\main.py"
fist,lower left
splay,lower right
palm,upper right
Can open the lock
```

1.2 Difficult Sequence

- The main method to distinguish between palm and fist is aspect ratio. I used Cv2.boundingRect function to calculate the minimum rectangle containing the hand. However, if the gesture is palm and placed diagonally, it will make the aspect ratio of palm similar to fist.
- The main method to distinguish between palm and splay is area ratio. If the gesture is palm, but the fingers are not tightened, it will make the area ratio of palm similar to spaly.
- There are many different ways to make a fist. One way of making a fist has a shape that similar to palm in binary image.
- “center center” is the most difficult position to detect. Because there are eight Areas around the “center center”, it is easy to misjudge.

So, I will use the following lock and unlock it with a non-standard gesture.

lock = [["palm","center center"],["splay","center center"],["fist","center center"]]

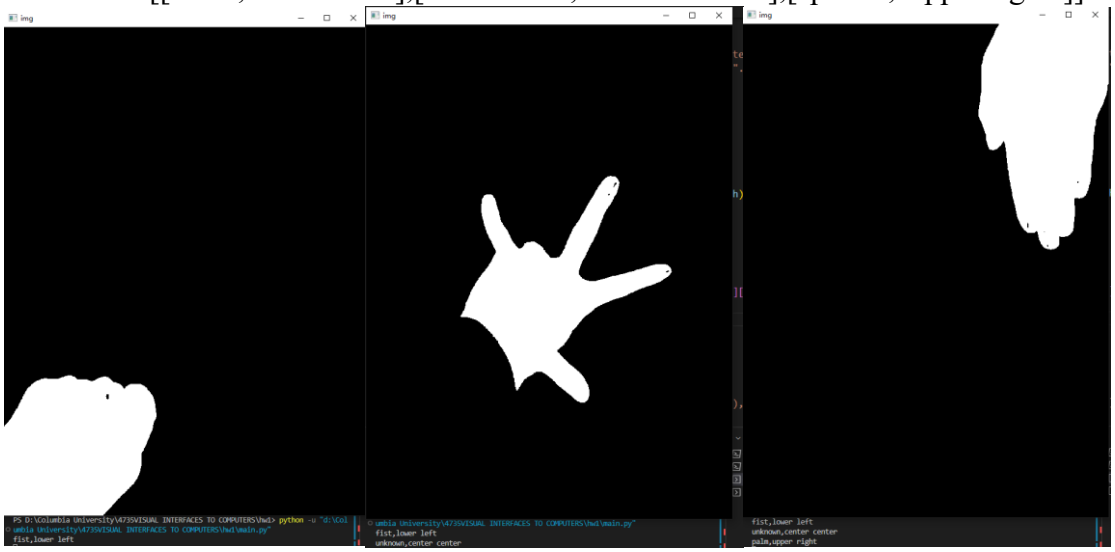


```
PS D:\Columbia University\4735VISUAL INTERFACES TO COMPUTERS\hw1> python -u "d:\Col
umbia University\4735VISUAL INTERFACES TO COMPUTERS\hw1\main.py"
fist,center center
palm,center center
palm,center center
Can't open the lock
```

1.3 Good Sequence

To add interest, I will add an "unknown, center center" lock to the easy sequence. This case is the case where the lock holds the success rate and the highest complexity.

lock = [["fist","lower left"],["unknown","center center"],["palm","upper right"]]

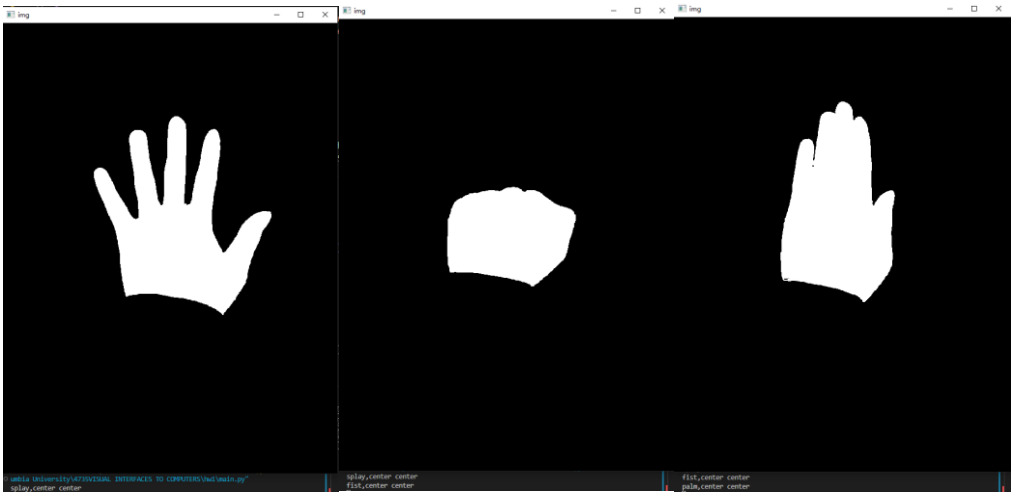


```
PS D:\Columbia University\4735VISUAL INTERFACES TO COMPUTERS\hw1> python -u "d:\Col
umbia University\4735VISUAL INTERFACES TO COMPUTERS\hw1\main.py"
fist,lower left
unknown,center center
palm,upper right
Can open the lock
```

2. Display the (new) reduced resolution binary intermediates you gathered from your friend

2.1 Easy Sequence

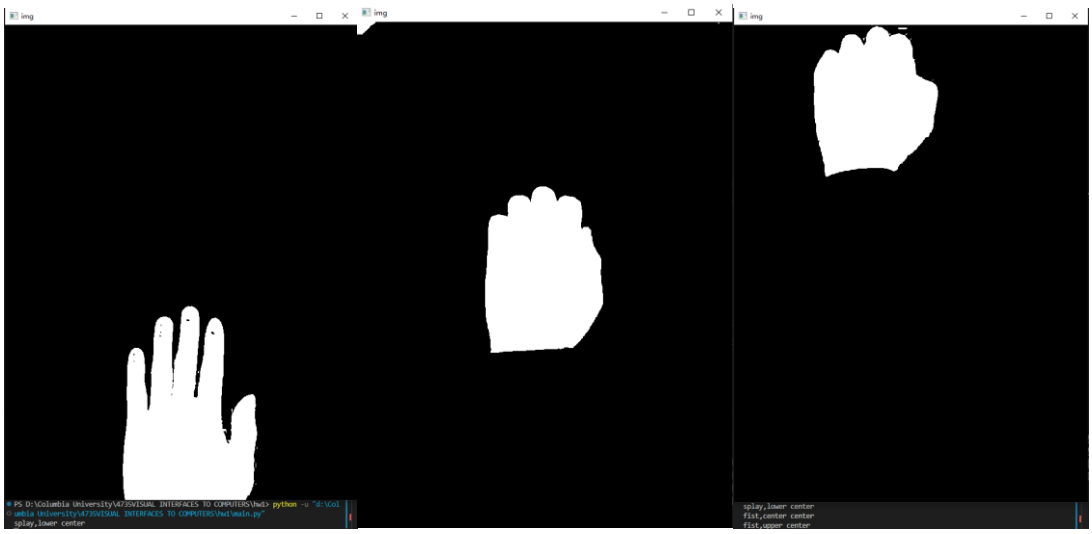
lock = [["splay","center center"],["fist","center center"],["palm","center center"]]



```
PS D:\Columbia University\4735VISUAL INTERFACES TO COMPUTERS\hw1> python -u "d:\Col
umbia University\4735VISUAL INTERFACES TO COMPUTERS\hw1\main.py"
splay,center center
fist,center center
palm,center center
Can open the lock
```

2.2 Difficult Sequence

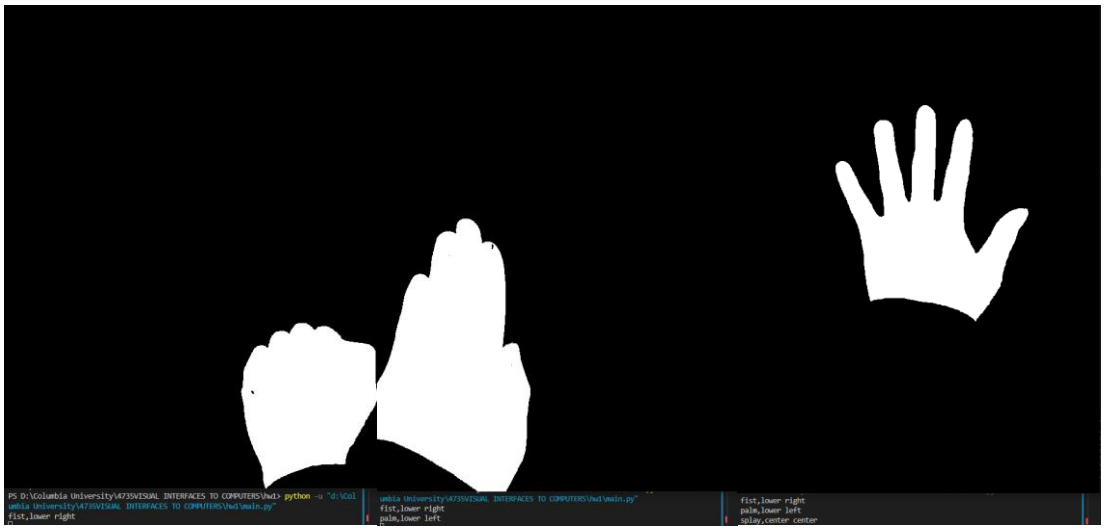
lock = [["palm","lower center"],["fist","center center"],["fist","unknown"]]



```
PS D:\Columbia University\4735VISUAL INTERFACES TO COMPUTERS\hw1> python -u "d:\Col
umbia University\4735VISUAL INTERFACES TO COMPUTERS\hw1\main.py"
splay,lower center
fist,center center
fist,upper center
Can't open the lock
```

2.3 Good Sequence

lock = [["fist","lower right"],["palm","lower left"],["splay","center center"]]



```
PS D:\Columbia University\4735VISUAL INTERFACES TO COMPUTERS\hw1> python -u "d:\Col
umbia University\4735VISUAL INTERFACES TO COMPUTERS\hw1\main.py"
fist,lower right
palm,lower left
splay,center center
Can open the lock
```

3. Report any feedback from your friend about the system: ease of use, confusability of "what", and of "where", and any other comments that could be used to improve the system

When I introduced this system to my friend, the first question he asked me was is this system real-time? Real-time system is a direction of improvement.

In addition, he said that he has never seen a gesture lock in daily life, but he thinks the complexity of the gesture lock is relatively high. I speculate that the reason for not using the gesture lock is that the gesture lock is easy for others to obtain the password.

4. Given your experiences, summarize how you would approach System 2.0: what you would keep, what you would change, etc.

For System 2.0, I will keep the skin detection algorithm and location detection algorithm. But I will improve the gesture detection algorithm, so that the system can accurately distinguish more gestures. In addition, the system needs to be changed to distinguish gestures in real-time.