Pucong Han
Visual Interfaces to Computer COMS W4735
Professor John Kender
Assignment 1

# Section 1: Domain Engineering

System and Tool Configurations

Operating System: Mac OS 10.7
Programming Language: Python 2.7 (JPEG library + Python Imaging Library + OpenCV)

JPEG library

Installation Process

```
$ curl -O http://www.ijg.org/files/jpegsrc.v8c.tar.gz
$ tar zxvf jpegsrc.v8c.tar.gz
$ cd jpeg-8c/
$ ./conFigure
$ make
$ sudo make install
```

Python Imaging Library

Installation Process

```
$ curl -O -L http://effbot.org/downloads/Imaging-1.1.7.tar.gz
$ tar -xzf Imaging-1.1.7.tar.gz cd Imaging-1.1.7
$ python setup.py build
$ sudo python setup.py install
```

An alternative installing approach using pip:

```
$ sudo pip install PIL
```

OpenCV for Python

Installation Process

```
##Install numpy with Macports
$ sudo port install py27-numpy

##Install OpenCV with Python:
$ sudo port install opencv +python27

##Edit your ~/.bash_profile with:
$ open -t ~/.bash_profile

##Add the line:
export
PYTHONPATH=/opt/local/var/macports/software/opencv/2.2.0_0+pytho
n27/opt/local/Library/Frameworks/Python.framework/Versions/2.7/lib/pyt
hon2.7/site-packages:$PYTHONPATH
```

Programming Tools: Sublime Text

Camera Configuration

Camera Type: Macbook built-in iSight camera

Camera Position: The camera is at the top of my laptop screen. My laptop is on a desk. Professor Frederik C. M. Kjeldsen explained in chapter three of his paper *Visual Interpretation of Hand Gestures as a Practical Interface Modality*, "The camera [should place] at the bottom of the screen rather than the top because it provides a much better point of view for observing both location and pose of the user's hand." My camera is on top of my laptop. I adjust the height of the supporting table and make the camera points up to my hand. Such position provides a better point of view for observing both location and pose of my hand.

Environment: Professional video and audio editing room at Columbia Journalism School (Figure 1). The reason for me to use this editing room is because the background color of that room is black, which does not reflect lights from any light sources.

Lighting: Incandescent light bulb (Figure 2).
This light source allows the hue of blood to become clearer and more distinguishable to the camera. According to professor Kender's lecture, hue of blood is more important than intension of blood. In the domain-engineering step, I want my camera to capture picture with better representation of hue.

Background: Black background (Wall of the editing room).

Clothing: Dark color sweater.
This type of sweater does not reflect lights. I want my camera captures only the hue of my hand gesture, not any other parts of my body.


Figure 1: Professional Editing Room


Figure 2: Incandescent light source

Software: Photo Booth
Photo Format: JPEG RGB 640 × 426 pixels

Hand Gesture Images


Figure 3: Open palm.


Figure 4: Fist at the center.

As we can see from the images, the black background of the editing room absorbs light from the incandescent light bulb. My hand gestures become easily distinguishable from the images. These images are color 2D representations of 3D hand models.

## Section 2: Data Reduction Step

I implement this section in the data_reduction.py project.

Get Binary Images of Hand Gestures

Using binary images improves the accuracy of analyzing the body part in the color images. In order to get high quality binary images of the hand gestures, I take advantage of one useful function in the OpenCV library called threshold(). This function is available in Python.

The threshold() function provides the most common way to segment a region of an image. According to the OpenCV documentation, the threshold() function applies fixed-level thresholding to a single-channel array. The function is typically used to get a binary image out of a gray scale image or for removing a noise, that is, filtering out pixels with too small or too large values.

Procedures and Implementations for Generating Binary Images

Before using any of the functions, we have to import the following libraries:

```
import os
import glob
import Image
import cv2
```

The program needs to first read all image files from the gestures folder and open them using OpenCV:

```
for infile in glob.glob( os.path.join("gestures", '*.jpg') ):
        im = cv2.imread(infile)
```

Since the threshold() function requires gray scale image input, we need to transfer the colored image from the default BGR format to Gray:

```
imgray = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
```

Then, we call the threshold() function and pass the gray scale images:

```
ret, thresh = cv2.threshold(imgray,127,255,0)
```

My program saves the output binary images to the binary_image_gestures folder using the save() function provided by the Python Imaging Library:

```
binaryImage = Image.fromarray(thresh)
binaryImage.save("binary_image_" + infile)
```

Binary Output Images of Hand Gestures

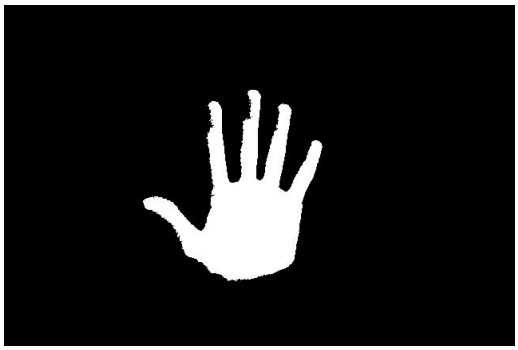


Figure 5: Open palm.

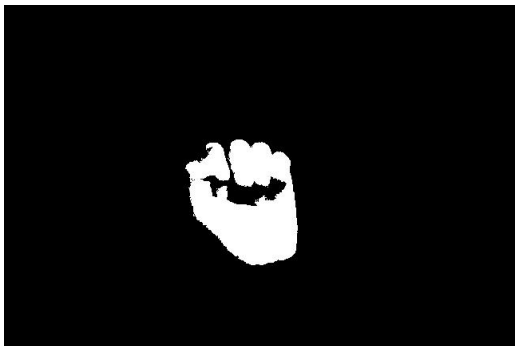


Figure 6: Fist at the center.

As we can see from these output images (Figure 5 and Figure 6), noise is avoided in the output binary files. In OpenCV, operations find white objects from the black background. The black background of the editing room allows the gestures to be distinguishable and makes an image-cleaning process unnecessary.

In order to find the boundaries of the image and the center of the gesture, I take advantage of another useful function in the OpenCV library called findContours(). This function is also available in Python.

According to the OpenCV documentation, the contours are a useful tool for shape analysis and object detection and recognition. This function allows the program to find the boundaries of objects from binary images.

Procedures and Implementations for Building Boundaries

Before using any of the functions, we have to import few additional libraries:

```
import numpy as np
import math
## This library enables printing colored text in console
## Make our results easily distinguishable
from termcolor import colored
import ImageDraw
```

Since I have produced binary images, my program directly calls the findContours() function and passes the binary images to the parameter of the function:

```
contours, hierarchy =
cv2.findContours(thresh,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
```

Each contour is stored as a vector of points. The output contour is a list of these boundary points in various color spaces. According to the OpenCV documentation, hierarchy variable is an optional output vector, containing information about the image topology.

To draw boundaries on the gray scale images, we use the drawContours() function. According to the OpenCV documentation, this function draws contour outlines in the image using the returned contours variable:

```
cv2.drawContours(im,contours,-1,(0,255,0),3)
```

The center of the gesture can be calculated by averaging the x axis values and y axis values:

```
count = 0;
xValue = 0;
yValue = 0;
for layers in contours:
        for demensions in layers:
                for values in demensions:
                        xValue += values[0]
                        yValue += values[1]
                        count += 1
xCenter = xValue / count
yCenter = yValue / count
```

To draw the center on the gray scale image, we call the circle() function provided by the OpenCV library:

```
cv2.circle(im,(xCenter, yCenter),5,(0,0,255),3)
```

My program saves the gray scale images with boundaries to the boundary_center _gestures folder using the save() function provided by the Python Imaging Library:

```
IplImage = Image.fromarray(im)
IplImage.save("boundary_center_gray_" + infile)
```

Gray Images of Hand Gestures with Boundaries and Center

Figure 7: Open palm.

Figure 8: Close palm.

The findContours() function detects and saves boundary points of hand gestures from each different color space. As we can see from the above images, boundaries of hand gestures get recognized from the binary images and saved in the contours variable, which contains boundary points in various color spaces.

Testing the Image

This portion of the program, using a qualitative approach, tests whether the gesture in the image is more like a fist or more like an open palm. In order to gather reliable data to test hand gestures in the image, the program calculates the convex hull – intersection of all convex sets – of hand gestures using the convexHull() function provided by Python OpenCV. According to the OpenCV documentation, the function finds the convex hull of a 2D point set using Sklansky's algorithm. The program also calculates the convexity defects of the hand gestures using the convexityDefects() function provided by Python OpenCV. According to the OpenCV documentation, the function finds all convexity defects of the input contour and returns an array of four values including start point, end point, farthest point and approximate distance to farthest point.

"The function worked well when the hand shape size was large, but many small defects were detected when the hand shape was small," according to the article Gesture Detection. "The number of existing defects allows us to locate the hand shape. In theory, by counting the number of existing defects, we could easily locate the hand shape and then recognize hand gestures." My testing results show that the number of defects was not under control. As shown in Figure 9 and Figure 10, the system picks up a large number of defects at the bottom and the side of the hand gestures. This noise will make my testing algorithms become less accurate.

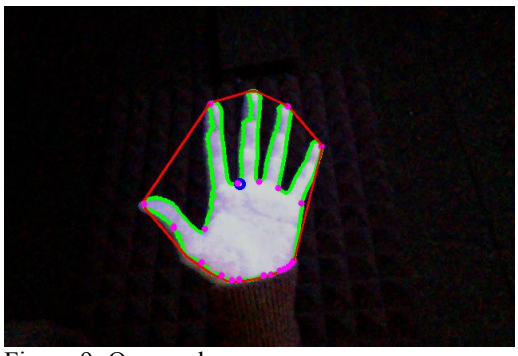


Figure 9: Open palm.

Figure 10: Fist at the center

I come up with a solution to eliminate noise by ignoring the ones with short distance between the points and the boundary of hand gestures. After eliminating defects with distance less than 1000, the results of the convexDefects() function improve. As shown in Figure 11 and Figure 12, the system prints the major defect points from memory:

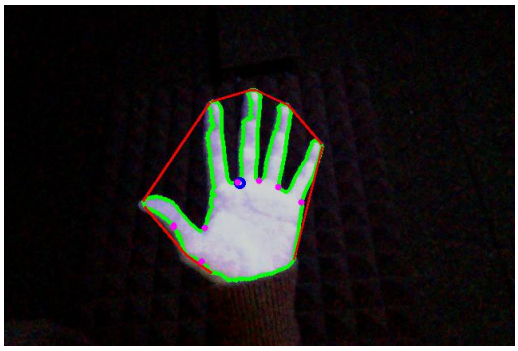


Figure 11: Open palm.

Figure 12: Fist at the center

Procedures and Implementations for Testing the Image

Before computing the convex hulls or convex defects of the hand gestures, we need to build a numpy array with contours from all layers.

We initialize a new numpy array outside the reading loop: boundaries = np.empty

After finding these contour blocks, we build a numpy array. The reason for me to build this separate array containing the contours is that the convexHull() function of this program requires contour inputs in such format. Here is the code to build the array:

```
##Build a numpy array with contours from all layers
boundaries = contours[0]
for layers in contours:
        if len(boundaries) != len(layers):
                boundaries = np.vstack((boundaries, layers))
```

The boundaries variable contains contours from all layers. This collection of contours allow us to compute and draw bounding boxes of hand gestures using the following code:

```
x,y,w,h = cv2.boundingRect(boundaries)
cv2.rectangle(im,(x,y),(x+w,y+h),(127,127,0),2)
boundingRect = w*h
cv2.circle(im,(x+w/2, y+h/2),4,[127,127,0],2)
```

The center of bounding boxes is different from the center of the contours. It depends on the shape of hand gestures in the image. As a result, the distance between these two centers is considered as one of the factors to test whether the gesture in the image is more like a fist or more like an open palm. Code to compute this distance:

```
distanceToCenter = math.sqrt(((x+w/2) - xCenter)*((x+w/2) - xCenter) + ((y+h/2) - yCenter)*((y+h/2) - yCenter))
```

By printing the distanceToCenter variable, we can have the following results in the console:

```
dyn-209-2-47-28:ph2369_assignment1 puconghan$ python data_reduction.py
Reading Image: gestures/close_palm.jpg
10.4403065089
Reading Image: gestures/cut_close.jpg
3.60555127546
Reading Image: gestures/cut_open.jpg
27.0739727414
Reading Image: gestures/first_finger.jpg
25.4558441227
Reading Image: gestures/fist_center.jpg
17.4642491966
Reading Image: gestures/fist_left_bottom.jpg
20.0
Reading Image: gestures/fist_left_up.jpg
21.0950231097
Reading Image: gestures/fist_right_bottom.jpg
13.9283882772
Reading Image: gestures/fist_right_up.jpg
13.3416640641
Reading Image: gestures/horizontal.jpg
10.6301458127
Reading Image: gestures/L.jpg
21.9317121995
Reading Image: gestures/middle_finger_ring_finger.jpg
6.40312423743
Reading Image: gestures/ok_sign.jpg
13.0384048104
Reading Image: gestures/open_palm.jpg
9.48683298051
Reading Image: gestures/thumb_and_first.jpg
9.8488578018
Reading Image: gestures/thumb_up.jpg
13.0384048104
```

Figure 13: Printed results for the distanceToCenter variable (A number of gestures in various locations).

As we can see, distance between the two centers of palm gestures is usually below 10, and distance between the two centers of fist gestures is usually above 15. This is one of the criterions for testing hand gestures.

After calculating the distance between the two centers, my program finds convex hulls and convexity defects for hand gestures using the yielded boundaries variable, which contains contours from all layers:

```
hull = cv2.convexHull(boundaries,returnPoints = False)
defects = cv2.convexityDefects(boundaries,hull)
```

The returned defects variable is an array of four values including start point, end point, farthest point and approximate distance to farthest point. Using this array variable, my program draws defect points with distance to associated contour greater than 1000. As I

explained in the above section, defect points with distance between the points and their associated contours lower than 1000 are considered as noises. While drawing the remaining defect points and contour boundaries, my program counts the number of remaining significant defects:

```
defectPointCount = 0
for i in range(defects.shape[0]):
   s,e,f,d = defects[i,0]
   if d >= 1000:
            start = tuple(boundaries[s][0])
            end = tuple(boundaries[e][0])
            far = tuple(boundaries[f][0])
            cv2.line(im,start,end,[255,0,0],2)
            cv2.circle(im,far,4,[255,0,255],-1)
            if d >= 10000:
                  defectPointCount += 1
```

My program considers defect points with distance between the center of the defects and their associated contours greater than 10000 to be significant. Only such defect points are counted and remained in my program. By printing the defectPointCount variable, we have the following results:

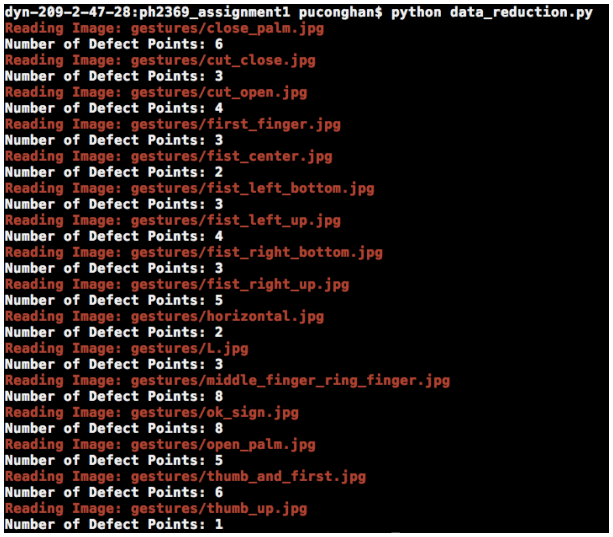


Figure 14: Printed results for the defectPointCount variable.

As we can see from Figure 14, number of significant defects of open palm gestures is usually greater than and equal to 5, and number of significant defects of fist gestures is usually less than and equal 5. This is another criterion for determining hand gestures. In the testing portion of my program, hand gestures with the number of significant defects greater than and equal to 5 are considered as open palm. Hand gestures with the number of significant defects less than 5 are considered as fist.

After drawing defect points, contour boundaries and the bounding boxes, we have the following images (I only select two image from the folder, a representative open palm gesture and a representative fist gesture):

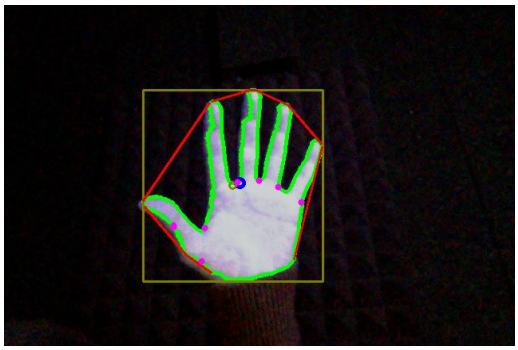


Figure 15: Open palm.

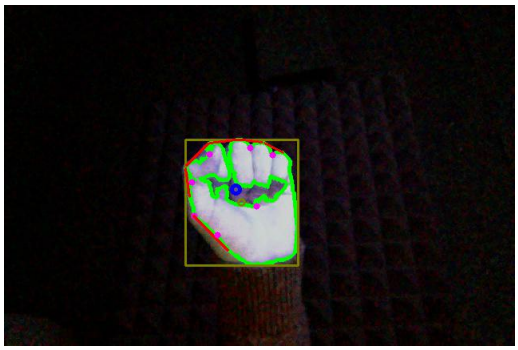


Figure 16: Fist at the center

These analyzed images are saved in the analyzed_gestures folder. As we can see from Figure 15 and Figure 16, the number of significant defects in open palm gesture is different from the number in fist gesture, and the distance between the center of bounding box and the center of contour for open palm gesture is different from the distance for fist gesture.

To make my testing process accurate, I introduce one additional criterion, the size of bounding boxes, to my program. This variable is denoted by boundingRect, which is yielded before drawing the bounding box of hand gestures. Recall previous codes of computing and drawing bounding boxes of hand gestures:

```
x,y,w,h = cv2.boundingRect(boundaries)
cv2.rectangle(im,(x,y),(x+w,y+h),(127,127,0),2)
## The following line computes the size of the bounding box
boundingRect = w*h
cv2.circle(im,(x+w/2, y+h/2),4,[127,127,0],2)
```

By printing the boundingRect variables for each hand gesture, we can have the following results in console:

```
dyn-209-2-47-28:ph2369_assignment1 puconghan$ python data_reduction.py
Reading Image: gestures/close_palm.jpg
Reading Image: gestures/cut_close.jpg
Reading Image: gestures/cut_open.jpg
Reading Image: gestures/first_finger.jpg
Reading Image: gestures/fist_center.jpg
Size of Bounding Rectangle: 22137
Size of Bounding Rectangle: 22137
Size of Bounding Rectangle: 22137
Size of Bounding Rectangle: 22137
Size of Bounding Rectangle: 22137
Size of Bounding Rectangle: 22137
Size of Bounding Rectangle: 22137
Reading Image: gestures/fist_left_bottom.jpg
Size of Bounding Rectangle: 32752
Size of Bounding Rectangle: 32752
Size of Bounding Rectangle: 32752
Size of Bounding Rectangle: 32752
Size of Bounding Rectangle: 32752
Size of Bounding Rectangle: 32752
Size of Bounding Rectangle: 32752
Size of Bounding Rectangle: 32752
Size of Bounding Rectangle: 32752
Size of Bounding Rectangle: 32752
Reading Image: gestures/fist_left_up.jpg
Size of Bounding Rectangle: 31504
Size of Bounding Rectangle: 31504
Size of Bounding Rectangle: 31504
Size of Bounding Rectangle: 31504
Size of Bounding Rectangle: 31504
Size of Bounding Rectangle: 31504
Size of Bounding Rectangle: 31504
Size of Bounding Rectangle: 31504
Size of Bounding Rectangle: 31504
Size of Bounding Rectangle: 31504
Reading Image: gestures/fist_right_bottom.jpg
Size of Bounding Rectangle: 27360
Size of Bounding Rectangle: 27360
Size of Bounding Rectangle: 27360
Size of Bounding Rectangle: 27360
Size of Bounding Rectangle: 27360
Size of Bounding Rectangle: 27360
Size of Bounding Rectangle: 27360
Size of Bounding Rectangle: 27360
Size of Bounding Rectangle: 27360
Reading Image: gestures/fist_right_up.jpg
Size of Bounding Rectangle: 22140
Size of Bounding Rectangle: 22140
Size of Bounding Rectangle: 22140
Size of Bounding Rectangle: 22140
Size of Bounding Rectangle: 22140
Size of Bounding Rectangle: 22140
Size of Bounding Rectangle: 22140
Reading Image: gestures/horizontal.jpg
Reading Image: gestures/L.jpg
Reading Image: gestures/middle_finger_ring_finger.jpg
Reading Image: gestures/ok_sign.jpg
Reading Image: gestures/open_palm.jpg
Size of Bounding Rectangle: 53775
Size of Bounding Rectangle: 53775
Size of Bounding Rectangle: 53775
Size of Bounding Rectangle: 53775
Size of Bounding Rectangle: 53775
Size of Bounding Rectangle: 53775
Size of Bounding Rectangle: 53775
Size of Bounding Rectangle: 53775
Reading Image: gestures/thumb_and_first.jpg
Reading Image: gestures/thumb_up.jpg
```

Figure 17: Printed results for the d variable and the boundingRect variable.

As we can see from Figure 17, the size of bounding box of open palm gestures is usually greater than 50000. The size of bounding box of fist gestures is usually less than 35000. This is another criterion for determining hand gestures.

My program can test hand gestures using the three criterions that I explained above. The variable that I need to evaluate for testing hand gestures are defectPointCount,

boundingRect, and distanceToCenter. Here is my implementation for testing hand gestures:

```
        if defectPointCount >= 5 and boundingRect > 50000 and distanceToCenter < 10:
                print colored("[RESULT] Hand Gesture is more like a --> Open Palm",
"blue")
        if defectPointCount <= 5 and boundingRect < 35000 and distanceToCenter > 12:
                print colored("[Result] Hand Gesture is more like a --> Fist", "blue")
```

As we can see from Figure 18, my testing results for the type of gestures in the image are fairly accurate:

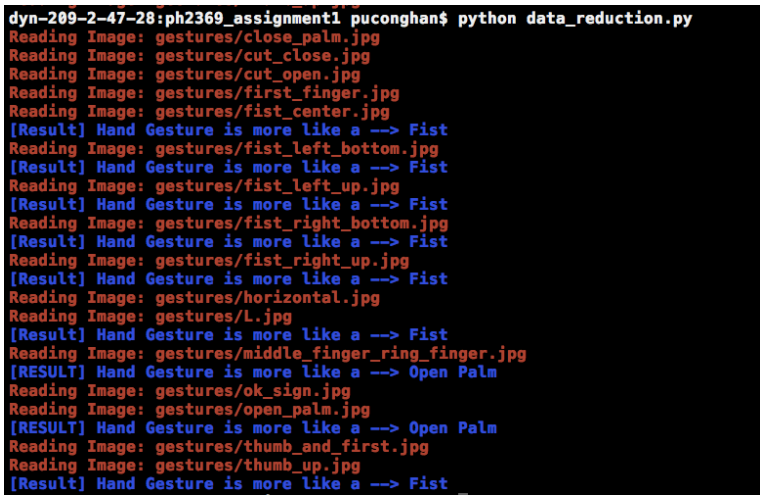


Figure 18: Result of the testing process.

The five fist gestures and the open palm gesture are tested in my system. In addition, the L hand gesture and thumb up hand gesture are suggested like a fist. The middle finger with ring finger hand gesture is suggested like an open palm. Hand gestures in these images are captured by my testing criterions. In order to make my system recognizes more types of hand gestures, I need to come up with superior algorithms to testify these qualitative variables.

# Section 3: Parsing and Performance Step

I implement the parsing and performance in a new application parsing_and_performance.py.

Screen Position Specification

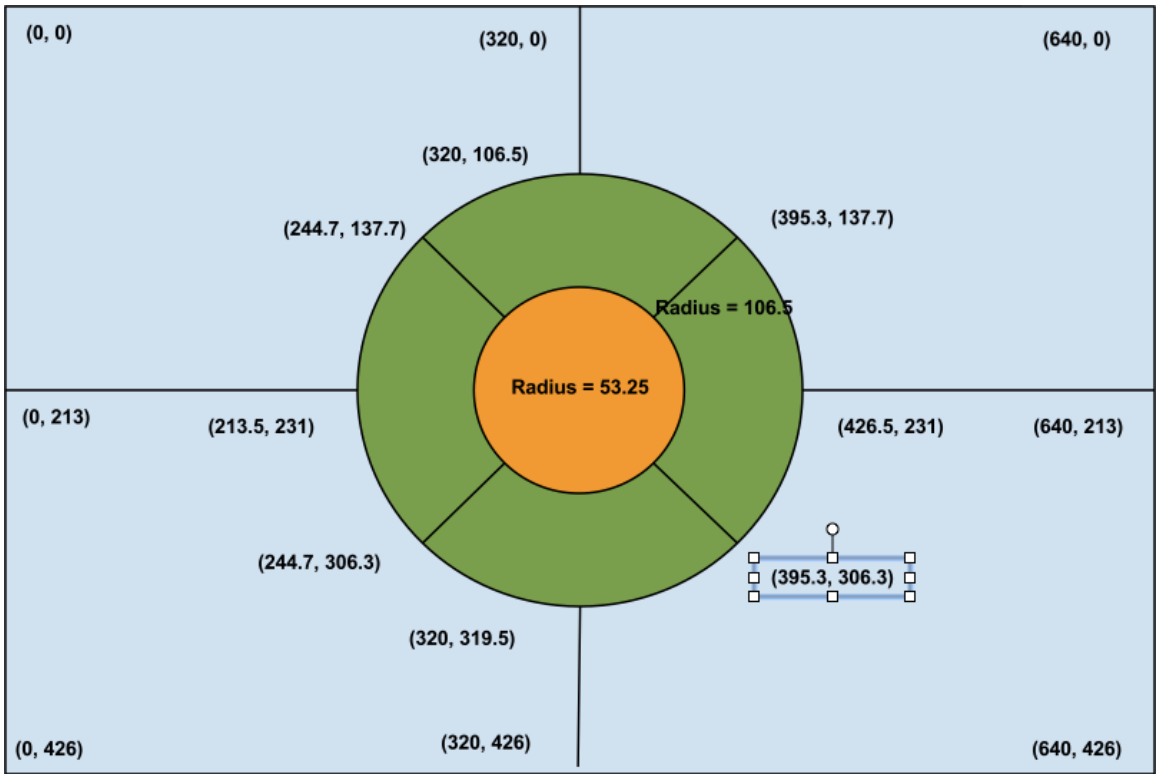



Figure 19: Screen position specification

In my program, I divide the screen into nine areas. The orange circle with radius equal to 53.25 pixels, 1/8 of the screen height, locates in the center of the screen. Surrounding the orange circle is a green concentric circle with a radius equal to 106.5 pixels (from the center of the image to the boarder of the green concentric circle), one quarter of the screen height. The green concentric circle is divided into four equal sections, upper section, right section, lower section and left section. The four blue rectangles in the back with length equal to 320 pixels and height equal to 213 pixels, divide the screen into four equal areas. Locations of all intersection points are marked in Figure 19.

Grammar (definition) of the nine divisions

> Note that, in my grammar, hand gestures fall into the nine areas control the movements of a cursor. The cursor locates at the center of the screen on default.

To keep my grammar clean and simple, only gestures fall into the center area are tested for open palm or fist (open palm means print and fist means reset). In other eight areas, both open palm gestures and fist gestures share the same grammar.

Upper Left Corner (The blue area in the upper left corner)

This area is located at the upper left of the screen. It implicitly gives users an impression of that direction.

**Definition = for both fist and open palm, move cursor to the upper left corner by one unit (Decrease both the x-axis of the cursor and y-axis of the cursor by one unit).**

Lower Left Corner (The blue area in the lower left corner)

This area is located at the lower left of the screen. It implicitly gives users an impression of that direction.

**Definition = for both fist and open palm, move cursor to the lower left corner by one unit (Decrease the x-axis of the cursor by one unit and increase the y-axis of the cursor by one unit).**

Upper Right Corner (The blue area in the upper right corner)

This area is located at the upper right of the screen. It implicitly gives users an impression of that direction.

**Definition = for both fist and open palm, move cursor to the upper right corner by one unit (Increase the x-axis of the cursor by one unit and decrease the y-axis of the cursor by one unit).**

Lower Right Corner (The blue area in the lower right corner)

This area is located at the lower right of the screen. It implicitly gives users an impression of that direction.

**Definition = for both fist and open palm, move cursor to the lower right corner by one unit (Increase both the x-axis of the cursor and y-axis of the cursor by one unit).**

Middle Left Area (green section in the left)

This area is located at the left of the screen. It implicitly gives users an impression of that direction.

**Definition = for both fist and open palm, move the cursor to the left by one unit (Decrease x-axis of the cursor by one unit).**

Middle Right Area (green section in the right)

This area is located at the right of the screen. It implicitly gives users an impression of that direction.

**Definition = for both fist and open palm, move the cursor to the right by one unit (Increase x-axis of the cursor by one unit).**

Middle Up Area (green section on the top)

This area is located on top of the screen. It implicitly gives users an impression of that direction.
**Definition = for both fist and open palm, move the cursor up by one unit (Decrease y-axis of the cursor by one unit).**

Middle Down Area (green section at the bottom)

This area is located at the bottom of the screen. It implicitly gives users an impression of that direction.
**Definition = for both fist and open palm, move the cursor down by one unit (Increase y-axis of the cursor by one unit).**

Center Area (The orange circle in the middle)

This area is located at the center of the screen. It implicitly gives users an impression of triggering actions. In my program, it triggers a print action if an open palm gesture is detected or a reset action if a fist gesture is detected in the center.
**Definition = Reset the location of the cursor if a fist gesture is detected in this area. Print the location of the cursor if an open palm gesture is detected in this area.**

Implementations for Parsing and Performance

Compared with my previous application, I did not make any major changes up to the contours computation. This new program reads sequence of images from ten default folders. The first portion of the code reads these sequences of images from the ten folders and load into the program, as shown in Figure 20.

```
import os
import glob
import Image
import cv2
import numpy as np
import math
from termcolor import colored
import ImageDraw

counterX = 320
counterY = 213

sequence_list = ["sequence_1", "sequence_2", "sequence_3", "sequence_4", "sequence_5_failed", "sequence_6_failed", "sequence_7", "sequence_8", "sequence_9", "sequence_10_failed

for folder in sequence_list:
    print "\n"
    print colored("Reading Folder: " + folder, 'green')
    boundaries = np.empty
    for infile in glob.glob( os.path.join("sequences_of_hand_gesture_images/" + folder, '*.jpg') ):
        print colored("Reading Image: " + infile, 'red')
        im = cv2.imread(infile)
        imgray = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)

        ##Create binary images
        ret, thresh = cv2.threshold(imgray,127,255,0)

        ##Find boundaries of objects from binary images
        contours, hierarchy = cv2.findContours(thresh,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
        cv2.drawContours(im,contours,-1,(0,255,0),3)

        ##Average the contours and calculate the center
        count = 0;
        xValue = 0;
        yValue = 0;
        for layers in contours:
            for demensions in layers:
                for values in demensions:
                    xValue += values[0]
                    yValue += values[1]
                    count += 1
        xCenter = xValue / count
        yCenter = yValue / count
```

Figure 20: Implementation up to the contours computation for section 3.

Similar to the previous project, this new project converts the imported images to binary image file using the following approach:

```
## Convert colored image to gray scale image
imgray = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
## Using the threshold() function to generate binary image and saved in thresh
ret, thresh = cv2.threshold(imgray,127,255,0)
```

Using the findContours() function provided by OpenCV, my application computes contours for binary images:

```
contours, hierarchy =
cv2.findContours(thresh,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
```

Again, the output contours store a list of boundary points in various color spaces or layers. The center of gestures can be calculated by averaging the x-axis values and y-axis values using the following approach:

```
count = 0;
xValue = 0;
yValue = 0;
for layers in contours:
        for demensions in layers:
                for values in demensions:
                        xValue += values[0]
                        yValue += values[1]
                        count += 1
xCenter = xValue / count
yCenter = yValue / count
```

Up to this point, my program obtains coordinates of the center of hand gestures. Using this value, my program can test which area does the center of a hand gesture is located. Here is my implementation for the grammar (Figure 21):

```
if (math.sqrt((320 - xCenter)*(320 - xCenter) + (213 - yCenter)*(213 - yCenter)) <= 53.25):
    ##Build a numpy array with contours from all layers
    boundaries = contours[0]
    for layers in contours:
        if len(boundaries) != len(layers):
            boundaries = np.vstack((boundaries, layers))

    ##Draw bounding Box
    #hull = cv2.convexHull(boundaries)
    x,y,w,h = cv2.boundingRect(boundaries)
    boundingRect = w*h


    ##Distance from the center of bounding box to the center of coutour
    distanceToCenter = math.sqrt(((x+w/2) - xCenter)*((x+w/2) - xCenter) + ((y+h/2) - yCenter)*((y+h/2) - yCenter))

    ##Find convex hull and convexity defects for boundaries of objects
    hull = cv2.convexHull(boundaries,returnPoints = False)
    defects = cv2.convexityDefects(boundaries,hull)

    defectPointCount = 0
    for i in range(defects.shape[0]):
        s,e,f,d = defects[i,0]
        if d >= 10000:
            defectPointCount += 1

    if defectPointCount >= 5 and boundingRect >= 45000:
        print colored("Location of Counter", "blue")
        print colored("Counter X:" + str(counterX), "red")
        print colored("Counter Y:" + str(counterY), "red")
        number_print = number_print + "print"
    elif defectPointCount <= 5 and boundingRect <= 35000:
        print "Reset X -> 320 and Y -> 213"
        counterX = 320
        counterY = 213
    else:
        print "Opps I don't understand this gesture"
```

```
elif (53.25 < math.sqrt((320 - xCenter)*(320 - xCenter) + (213 - yCenter)*(213 - yCenter)) <= 106.5):
    if xCenter < 320 and yCenter < 213:
        if math.sqrt((320 - xCenter)*(320 - xCenter)) > math.sqrt((213 - yCenter)*(213 - yCenter)):
            print "Left (X - 1)"
            counterX -= 1
        else:
            print "Up (Y - 1)"
            counterY -= 1
    elif xCenter > 320 and yCenter < 213:
        if math.sqrt((320 - xCenter)*(320 - xCenter)) > math.sqrt((213 - yCenter)*(213 - yCenter)):
            print "Right (X + 1)"
            counterX += 1
        else:
            print "Up (Y - 1)"
            counterY -= 1
    elif xCenter < 320 and yCenter > 213:
        if math.sqrt((320 - xCenter)*(320 - xCenter)) > math.sqrt((213 - yCenter)*(213 - yCenter)):
            print "Left (X - 1)"
            counterX -= 1
        else:
            print "Down (Y + 1)"
            counterY += 1
    elif xCenter > 320 and yCenter > 213:
        if math.sqrt((320 - xCenter)*(320 - xCenter)) > math.sqrt((213 - yCenter)*(213 - yCenter)):
            print "Right (X + 1)"
            counterX += 1
        else:
            print "Down (Y + 1)"
            counterY += 1
elif (math.sqrt((320 - xCenter)*(320 - xCenter) + (213 - yCenter)*(213 - yCenter)) > 106.5):
    if xCenter < 320 and yCenter < 213:
        print "Left and Up (X - 1 and Y - 1)"
        counterX -= 1
        counterY -= 1
    if xCenter > 320 and yCenter < 213:
        print "Right and Up (X + 1 and Y - 1)"
        counterX += 1
        counterY -= 1
    if xCenter < 320 and yCenter > 213:
        print "Left and Down (X - 1 and Y + 1)"
        counterX -= 1
        counterY += 1
    if xCenter > 320 and yCenter > 213:
        print "Right and Down (X + 1 and Y + 1)"
        counterX += 1
        counterY += 1
```

Figure 21: Implementation for the grammar for section 3.

The math.sqrt((320 - xCenter)*(320 - xCenter) + (213 - yCenter)*(213 - yCenter)) clause computes the distance between the center of a gesture and the center of the image where the gesture comes from. This distance allows our program to determine if the center falls into the orange center of the image, the four green concentric circle sections or the four blue corner areas.

The first if-statement captures the case that the center of a gesture falls into the orange center of the image. The radius of the orange center is 53.25px. The condition of this if-statement is that the distance between the center of a gesture and the center of the image is less than or equal to 53.25 (we want to include the case that the center of the gesture is on the boarder):

if (math.sqrt((320 - xCenter)*(320 - xCenter) + (213 - yCenter)*(213 - yCenter)) <= 53.25):

Inside this if-statement, my program builds the numpy array of the contours and computes the boundary rectangles, the convex hulls and defects:

```
##Build a numpy array with contours from all layers
boundaries = contours[0]
for layers in contours:
        if len(boundaries) != len(layers):
                boundaries = np.vstack((boundaries, layers))

##Generate bounding box
x,y,w,h = cv2.boundingRect(boundaries)
boundingRect = w*h
```

```
##Find convex hull and convexity defects for boundaries of objects
hull = cv2.convexHull(boundaries,returnPoints = False)
defects = cv2.convexityDefects(boundaries,hull)

##Count the number of defect point
defectPointCount = 0
for i in range(defects.shape[0]):
        s,e,f,d = defects[i,0]
        if d >= 10000:
                defectPointCount += 1
```

My grammar in this new program only tests the two gestures (open palm or fist) in the center area. To improve the tolerance of testing the two hand gestures, I change the criterions for testing open palm and fist from the previous implementation. I remove the condition for testing the distance between the two centers since it might become subtle in some gestures. Here is my implementation for the new criterions:

```
if defectPointCount >= 5 and boundingRect >= 45000:
        print colored("Location of Counter", "blue")
        print colored("Counter X:" + str(counterX), "red")
        print colored("Counter Y:" + str(counterY), "red")
elif defectPointCount <= 5 and boundingRect <= 35000:
        print "Reset X -> 320 and Y -> 213"
        counterX = 320
        counterY = 213
else:
        print "Opps I don't understand this gesture"
```

The size of the bounding box of hand gestures depends on the distance between the camera and the hand. The condition of the first if-statement will accept more open palm gestures with the size of the bounding box greater than 45000 and the number of defect points greater than 5. If this condition is fulfilled, the program will print the location of the counter. The condition of the second if-statement will accept more fist gestures with the size of the bounding box less than 35000 and the number of defect points less than 5. If this condition is fulfilled, the program will reset the location of the counter to (320, 213).

The radius of the green concentric circle (from the center of the image to the boarder of the green concentric circle) is 106.5px. The radius of hand gestures fall into these four green areas is between 53.25px and 106.5px. Radius of any points in this range is considered as in one of the four concentric zones. Here is my implementation for the second if-statement:

```
elif (53.25 < math.sqrt((320 - xCenter)*(320 - xCenter) + (213 - yCenter)*(213 - yCenter)) <= 106.5):
```

Within this if-statement, there are four child if-statements to test which side does the center of the gestures fall into (upper left, upper right, lower left and lower right). For each of these four if-statements, my program has two additional if-statements to test which of the four green sectors does the center of the gestures fall into (up, right, bottom and left):

```
if xCenter < 320 and yCenter < 213:
        if math.sqrt((320 - xCenter)*(320 - xCenter)) > math.sqrt((213 -
        yCenter)*(213 - yCenter)):
                print "Left (X - 1)"
                counterX -= 1
        else:
                print "Up (Y - 1)"
                counterY -= 1
elif xCenter > 320 and yCenter < 213:
        if math.sqrt((320 - xCenter)*(320 - xCenter)) > math.sqrt((213 -
        yCenter)*(213 - yCenter)):
                print "Right (X + 1)"
                counterX += 1
        else:
                print "Up (Y - 1)"
                counterY -= 1
elif xCenter < 320 and yCenter > 213:
        if math.sqrt((320 - xCenter)*(320 - xCenter)) > math.sqrt((213 -
        yCenter)*(213 - yCenter)):
                print "Left (X - 1)"
                counterX -= 1
        else:
                print "Down (Y + 1)"
                counterY += 1
elif xCenter > 320 and yCenter > 213:
        if math.sqrt((320 - xCenter)*(320 - xCenter)) > math.sqrt((213 -
        yCenter)*(213 - yCenter)):
                print "Right (X + 1)"
                counterX += 1
        else:
                print "Down (Y + 1)"
                counterY += 1
```

Explanations of the conditions for these four child if-statements (Logics):

If x value of the center of hand gestures is less than 320 (half of the x value of the image), the gesture falls into the left of the screen.

If x value of the center of hand gestures is greater than 320 (half of the x value of the image), the gesture falls into the right of the screen.

If y value of the center of hand gesture is less than 213 (half of the y value of the image), the gesture falls into the top of the screen.

If y value of the center of hand gesture is greater than 213 (half of the y value of the image), the gesture falls into the bottom of the screen.

If the distance between x value of the center of a hand gesture and 320 (half of the x value of the image) is greater than the distance between y value of the center of a hand gesture and 213 (half of the y value of the image), the center of hand gestures is at the left or right of the green concentric circle. Otherwise, the center of hand gestures is at the top or bottom of the green concentric circle.

Combining these conditions into a list of if-statements, my application can test which green concentric section does the center of a hand gesture locates.

The third if-statement block checks whether or not a gesture falls into the four blue areas in the back. As we can see from the conditions of this if-statement, if the distance between the center of a hand gesture and the center of the image is greater than 106.5px (falls into the four blue areas), the statements under this if-statement will be executed:

```
elif (math.sqrt((320 - xCenter)*(320 - xCenter) + (213 - yCenter)*(213 - yCenter)) > 106.5):
```

Within this if-statement, I have four branch if-statements to check which particular blue section does the center located:

```
if xCenter < 320 and yCenter < 213:
        print "Left and Up (X - 1 and Y - 1)"
        counterX -= 1
        counterY -= 1
if xCenter > 320 and yCenter < 213:
        print "Right and Up (X + 1 and Y - 1)"
        counterX += 1
        counterY -= 1
if xCenter < 320 and yCenter > 213:
        print "Left and Down (X - 1 and Y + 1)"
        counterX -= 1
        counterY += 1
if xCenter > 320 and yCenter > 213:
        print "Right and Down (X + 1 and Y + 1)"
        counterX += 1
        counterY += 1
```

The logics and conditions of these branch if-statements are fairly simple. If x value of the center of a hand gesture is greater than 320, the hand gesture falls into the right of the image. Otherwise, the gesture falls into the left of the image. If y value of the center of a gesture is greater than 213, the gesture falls into the bottom of the image. Otherwise, the gesture falls into the top of the image.

In this project, I designed ten sequences of images stored in the sequences_of_hand_gesture_images folder. Ten sequences of hand gesture images are carefully organized under this folder (These images are saved in ten branch folders name sequence_1 to sequence_10). This new application yields the following results:

## Results of Section 3

```
dyn-209-2-224-179:ph2369_assignment1 puconghan$ python parsing_and_performance.py


Reading Folder: sequence_1
Reading Image: sequences_of_hand_gesture_images/sequence_1/fist_center.jpg
Reset X -> 320 and Y -> 213
Reading Image: sequences_of_hand_gesture_images/sequence_1/fist_left_bottom.jpg
Left and Down (X - 1 and Y + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_1/fist_left_up.jpg
Left and Up (X - 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_1/fist_right_bottom.jpg
Right and Down (X + 1 and Y + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_1/fist_right_up.jpg
Right and Up (X + 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_1/open_palm.jpg
Location of Counter
Counter X:320
Counter Y:213
The Sequence of Actions is Successfully Processed


Reading Folder: sequence_2
Reading Image: sequences_of_hand_gesture_images/sequence_2/001_open_palm_right_bottom.jpg
Right and Down (X + 1 and Y + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_2/002_fist_right_bottom.jpg
Right and Down (X + 1 and Y + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_2/003_fist_left_bottom.jpg
Left and Down (X - 1 and Y + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_2/004_fist_middle_right.jpg
Right (X + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_2/005_open_palm_center.jpg
Location of Counter
Counter X:322
Counter Y:216
The Sequence of Actions is Successfully Processed


Reading Folder: sequence_3
Reading Image: sequences_of_hand_gesture_images/sequence_3/001_fist_left.jpg
Right (X + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_3/002_open_palm_right_up.jpg
Right and Up (X + 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_3/003_open_plam_left_up.jpg
Left and Up (X - 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_3/004_fist_center.jpg
Reset X -> 320 and Y -> 213
Reading Image: sequences_of_hand_gesture_images/sequence_3/005_fist_right.jpg
Right (X + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_3/006_open_palm_center.jpg
Location of Counter
Counter X:321
Counter Y:213
The Sequence of Actions is Successfully Processed


Reading Folder: sequence_4
Reading Image: sequences_of_hand_gesture_images/sequence_4/001_fist_right.jpg
Right (X + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_4/002_fist_left_bottom.jpg
Left and Down (X - 1 and Y + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_4/003_fist_left_up.jpg
Left and Up (X - 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_4/004_open_palm_right_up.jpg
Right and Up (X + 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_4/005_fist_left_bottom.jpg
Left and Down (X - 1 and Y + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_4/006_open_palm_center.jpg
Location of Counter
Counter X:320
Counter Y:213
The Sequence of Actions is Successfully Processed


Reading Folder: sequence_5_failed
Reading Image: sequences_of_hand_gesture_images/sequence_5_failed/001_fist_left_up.jpg
Left and Up (X - 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_5_failed/002_fist_center.jpg
Reset X -> 320 and Y -> 213
Reading Image: sequences_of_hand_gesture_images/sequence_5_failed/003_fist_left_up.jpg
Left and Up (X - 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_5_failed/004_fist_right_up.jpg
Right and Up (X + 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_5_failed/005_open_palm_center.jpg
Up (Y - 1)
The Sequence of Actions is Failed


Reading Folder: sequence_6_failed
Reading Image: sequences_of_hand_gesture_images/sequence_6_failed/001_open_palm_right.jpg
Up (Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_6_failed/002_open_palm_right_bottom.jpg
Right and Down (X + 1 and Y + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_6_failed/003_fist_right_up.jpg
Right and Up (X + 1 and Y - 1)
```

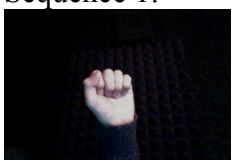
```
Reading Image: sequences_of_hand_gesture_images/sequence_6_failed/004_fist_right.jpg
Reset X -> 320 and Y -> 213
Reading Image: sequences_of_hand_gesture_images/sequence_6_failed/005_open_palm_center.jpg
Location of Counter
Counter X:320
Counter Y:213
The Sequence of Actions is Failed


Reading Folder: sequence_7
Reading Image: sequences_of_hand_gesture_images/sequence_7/001_open_palm_right_bottom.jpg
Right and Down (X + 1 and Y + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_7/002_open_palm_left_bottom.jpg
Left and Down (X - 1 and Y + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_7/003_fist_right_up.jpg
Right and Up (X + 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_7/004_open_palm_right_bottom.jpg
Right and Down (X + 1 and Y + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_7/005_fist_center.jpg
Reset X -> 320 and Y -> 213
Reading Image: sequences_of_hand_gesture_images/sequence_7/006_open_palm_center.jpg
Location of Counter
Counter X:320
Counter Y:213
The Sequence of Actions is Successfully Processed


Reading Folder: sequence_8
Reading Image: sequences_of_hand_gesture_images/sequence_8/001_fist_right_up.jpg
Right and Up (X + 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_8/002_open_palm_right_up.jpg
Right and Up (X + 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_8/003_open_palm_left_up.jpg
Left and Up (X - 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_8/004_fist_center.jpg
Reset X -> 320 and Y -> 213
Reading Image: sequences_of_hand_gesture_images/sequence_8/005_fist_left_up.jpg
Left and Up (X - 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_8/006_open_palm_center.jpg
Location of Counter
Counter X:319
Counter Y:212
The Sequence of Actions is Successfully Processed


Reading Folder: sequence_9
Reading Image: sequences_of_hand_gesture_images/sequence_9/001_fist_bottom.jpg
Down (Y + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_9/002_fist_center.jpg
Reset X -> 320 and Y -> 213
Reading Image: sequences_of_hand_gesture_images/sequence_9/003_fist_right_up.jpg
Right and Up (X + 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_9/004_open_palm_left.jpg
Left (X - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_9/005_open_palm_up.jpg
Up (Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_9/006_open_palm_center.jpg
Location of Counter
Counter X:320
Counter Y:211
The Sequence of Actions is Successfully Processed


Reading Folder: sequence_10_failed
Reading Image: sequences_of_hand_gesture_images/sequence_10_failed/001_fist_right.jpg
Right (X + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_10_failed/002_fist_bottom.jpg
Reset X -> 320 and Y -> 213
Reading Image: sequences_of_hand_gesture_images/sequence_10_failed/003_fist_bottom.jpg
Reset X -> 320 and Y -> 213
Reading Image: sequences_of_hand_gesture_images/sequence_10_failed/004_fist_center.jpg
Reset X -> 320 and Y -> 213
Reading Image: sequences_of_hand_gesture_images/sequence_10_failed/005_palm_left.jpg
Left (X - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_10_failed/006_open_palm_left_up.jpg
Left and Up (X - 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_10_failed/007_open_palm_right_up.jpg
Up (Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_10_failed/008_open_palm_bottom.jpg
Location of Counter
Counter X:318
Counter Y:211
Reading Image: sequences_of_hand_gesture_images/sequence_10_failed/009_open_palm_center.jpg
The Sequence of Actions is Failed
```

Figure 22: Results of Section 3

As we can see from Figure 22, seven of the sequences are correctly processed. Three sequences are failed.

Details of the Inputs and Outputs

Sequence 1:



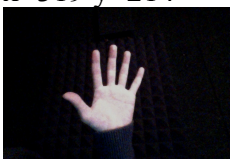
1. Reset
x=320 y=213

2. Move left & down
x=319 y=214

3. Move left & up
x=318 y=213

4. Move right & down
x=319 y=214

5. Move right & up

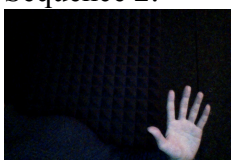


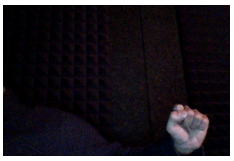
6. Print

x=320 y=213

As we can see from Figure 22, the system successfully processed this sequence and returned the location of the counter at (320, 213)

Sequence 2:


1. Move left & down
x=321 y=214


2. Move left & down
x=322 y=215

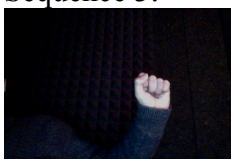

3. Move right & down
x=321 y=216


4.Move right
x=322 y=216


5. Print

As we can see from Figure 22, the system successfully processed this sequence and returned the location of the counter at (322, 216)

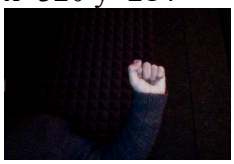
Sequence 3:

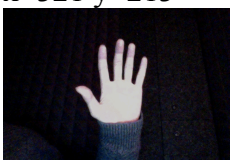

1. Move right
x=320 y=214


2. Move right & up
x=321 y=213


3. Move left & up
x=320 y=212


4. Reset
x=320 y=213


5. Move right
x=321 y=213


6. Print

As we can see from Figure 22, the system successfully processed this sequence and returned the location of the counter at (321, 213)

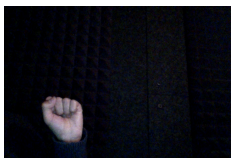
Sequence 4:


1. Move right
x=322 y=213


2. Move left & down
x=320 y=214


3. Move left & up
x=319 y=213


4. Move right & up
x=320 y=212




5. Move left & down x=319 y=213

6. Print

Originally, this sequence was not passed, as shown in Figure 23. By printing the value of the counter, I found that the counter was not correctly set to (320, 213). To fix this problem, I added two statements before reading images from a new sequence folder:

counterX = 320
counterY = 213

These two lines of code will reset the counter location for every new sequence. This sequence helps me to capture a bug of my system☺

```
Reading Folder: sequence_4
Counter X:321
Counter Y:213
Reading Image: sequences_of_hand_gesture_images/sequence_4/001_fist_right.jpg
Right (X + 1)
Counter X:322
Counter Y:213
Reading Image: sequences_of_hand_gesture_images/sequence_4/002_fist_left_bottom.jpg
Left and Down (X - 1 and Y + 1)
Counter X:321
Counter Y:214
Reading Image: sequences_of_hand_gesture_images/sequence_4/003_fist_left_up.jpg
Left and Up (X - 1 and Y - 1)
Counter X:320
Counter Y:213
Reading Image: sequences_of_hand_gesture_images/sequence_4/004_open_palm_right_up.jpg
Right and Up (X + 1 and Y - 1)
Counter X:321
Counter Y:212
Reading Image: sequences_of_hand_gesture_images/sequence_4/005_fist_left_bottom.jpg
Left and Down (X - 1 and Y + 1)
Counter X:320
Counter Y:213
Reading Image: sequences_of_hand_gesture_images/sequence_4/006_open_palm_center.jpg
Location of Counter
Counter X:320
Counter Y:213
The Sequence of Actions is Failed
```

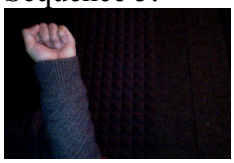
Figure 23: Failure result.

```
Reading Folder: sequence_4
Counter X:320
Counter Y:213
Reading Image: sequences_of_hand_gesture_images/sequence_4/001_fist_right.jpg
Right (X + 1)
Counter X:321
Counter Y:213
Reading Image: sequences_of_hand_gesture_images/sequence_4/002_fist_left_bottom.jpg
Left and Down (X - 1 and Y + 1)
Counter X:320
Counter Y:214
Reading Image: sequences_of_hand_gesture_images/sequence_4/003_fist_left_up.jpg
Left and Up (X - 1 and Y - 1)
Counter X:319
Counter Y:213
Reading Image: sequences_of_hand_gesture_images/sequence_4/004_open_palm_right_up.jpg
Right and Up (X + 1 and Y - 1)
Counter X:320
Counter Y:212
Reading Image: sequences_of_hand_gesture_images/sequence_4/005_fist_left_bottom.jpg
Left and Down (X - 1 and Y + 1)
Counter X:319
Counter Y:213
Reading Image: sequences_of_hand_gesture_images/sequence_4/006_open_palm_center.jpg
Location of Counter
Counter X:319
Counter Y:213
The Sequence of Actions is Successfully Processed
```

Figure 24: New result after fixing the problem.

As we can see from Figure 24, after adding these two lines of code, the system successfully processed this sequence and returned the location of the counter at (321, 213).

Sequence 5:

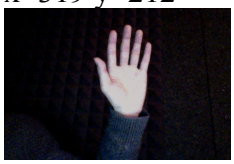


1. Move left & up
x=319 y=212

2. Reset
x=320 y=213

3. Move left & up
x=319 y=212

4. Move right & up
x=320 y=211



5. Print

As we can see from Figure 25, this particular sequence was failed.

```
Reading Folder: sequence_5_failed
Reading Image: sequences_of_hand_gesture_images/sequence_5_failed/001_fist_left_up.jpg
Left and Up (X - 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_5_failed/002_fist_center.jpg
Reset X -> 320 and Y -> 213
Reading Image: sequences_of_hand_gesture_images/sequence_5_failed/003_fist_left_up.jpg
Left and Up (X - 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_5_failed/004_fist_right_up.jpg
Right and Up (X + 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_5_failed/005_open_palm_center.jpg
Up (Y - 1)
The Sequence of Actions is Failed
Current Location of Counter is:
Counter X:320
Counter Y:210
```

Figure 25: Failure results

Instead of printing (320, 211) for the location of the counter, the current counter moved to (320, 210). From the result, we can see that the last image was not correctly tested. It should be an open palm at the center of the screen, which indicates to print the result. However, the last one was tested to be an open palm on the top, which indicates to move the counter up by one unit. As a result, the y-axis of the counter moves one additional unit up.

Figure 26: Open palm analyzed image.

As we can see from Figure 26, the center of the open palm was slightly above the center of the image. It falls into the green middle section (move up). If my hand was open enough while taking the picture, the center of the image will have a lower position. Figure 27 shows an example of what the center of open palm hand gesture should locate relative to the detected hand gesture.

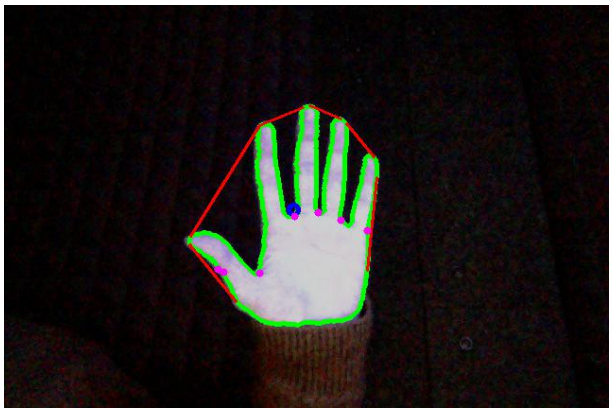
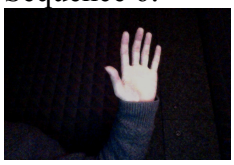


Figure 27: Open palm analyzed image.

Sequence 6:

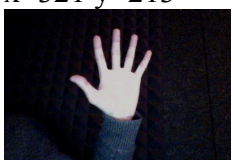


1. Move right x=321 y=213

2. Move right & down x=322 y=214

3. Move left & up x=323 y=213

4. Move right x=324 y=213



5. Print

As we can see from Figure 28, this particular sequence was failed.

```
Reading Folder: sequence_6_failed
Reading Image: sequences_of_hand_gesture_images/sequence_6_failed/001_open_palm_right.jpg
Up (Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_6_failed/002_open_palm_right_bottom.jpg
Right and Down (X + 1 and Y + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_6_failed/003_fist_right_up.jpg
Right and Up (X + 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_6_failed/004_fist_right.jpg
Reset X -> 320 and Y -> 213
Reading Image: sequences_of_hand_gesture_images/sequence_6_failed/005_open_palm_center.jpg
Location of Counter
Counter X:320
Counter Y:213
The Sequence of Actions is Failed
Current Location of Counter is:
Counter X:320
Counter Y:213
```

Figure 28: Failure result

Instead of printing (324, 213) for the location of the counter, the current counter moved to (320, 213). From the result, we can see that the first image and fourth image were not correctly tested.

The first image should be an open palm at right, which indicates to move the counter right by one unit. However, the first one was tested to be an open palm at the top, which indicates to move the counter up by one unit. Instead of adding one unit to the x-axis, the program adds one unit to the y-axis of the counter.

Figure 29: Open palm analyzed image.

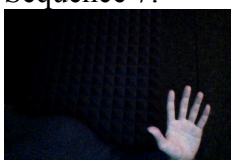
As we can see from Figure 29, the center of the open palm was slightly above the center of the image. Compared with the open palm gesture in Figure 15, fingers in that picture are not open enough. The gesture falls into the upper green concentric section indicating move the counter up by one unit.

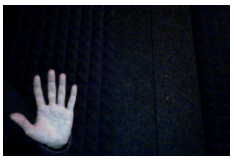
The fourth image should be a fist at right, which indicates move the counter right by one unit. However, the fourth one was tested to be a fist at the center, which indicate to reset the counter to the center of the image. Instead of adding one unit to the x-axis, the program reset the (x, y) to the center of the screen at (320, 213).

Figure 30: Fist analyzed image.

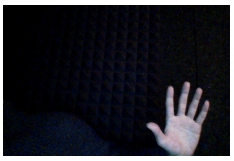
As we can see from Figure 30, the center of the fist was slightly to the left of the gesture. It should fall into the right green section, which indicates moving the counter to the right by one unit. Because the green section is relatively small and next to the center of the image, the center of the fist accidentally fell into the orange center of the image, which indicates to reset the counter.

Sequence 7:

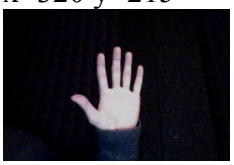

1. Move right & down
x=321 y=214


2. Move left & down
x=320 y=215


3. Move right & up
x=321 y=214


4. Move right & down
x=322 y=215


5. Reset
x=320 y=213

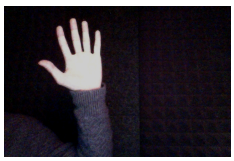

6. Print

As we can see from Figure 22, the system successfully processed this sequence and returned the location of the counter at (320, 213)

Sequence 8:

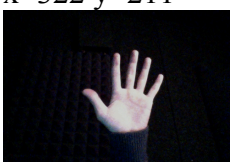

1. Move right & up
x=321 y=212


2. Move right & up
x=322 y=211


3. Move left & up
x=321 y=210

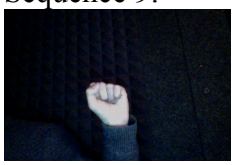

4. Reset
x=320 y=213


5. Move left & up
x=319 y=212


6. Print

As we can see from Figure 22, the system successfully processed this sequence and returned the location of the counter at (319, 212)

Sequence 9:


1. Move down
x=320 y=214


2. Reset
x=320 y=213


3. Move right & up
x=321 y=212


4. Move left
x=320 y=212

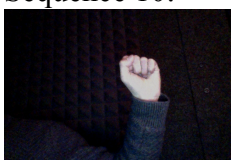

5. Move up


6. Print

x=320 y=211

As we can see from Figure 22, the system successfully processed this sequence and returned the location of the counter at (320, 211)

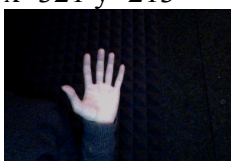
Sequence 10:

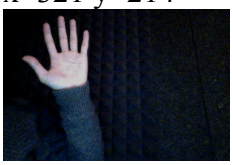

1. Move right
x=321 y=213

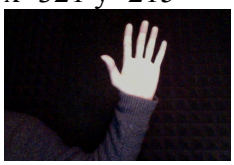

2. Move down
x=321 y=214

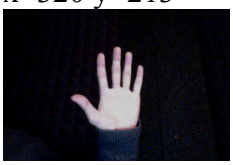

3. Move down
x=321 y=215


4. Reset
x=320 y=213


5. Move left
x=319 y=213


6. Move left & up
x=318 y=212


7. Move right & up
x=319 y=211


8. Move down
x=319 y=212


9. Print

As we can see from Figure 31, this particular sequence was failed.

```
Reading Folder: sequence_10_failed
Reading Image: sequences_of_hand_gesture_images/sequence_10_failed/001_fist_right.jpg
Right (X + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_10_failed/002_fist_bottom.jpg
Reset X -> 320 and Y -> 213
Reading Image: sequences_of_hand_gesture_images/sequence_10_failed/003_fist_bottom.jpg
Reset X -> 320 and Y -> 213
Reading Image: sequences_of_hand_gesture_images/sequence_10_failed/004_fist_center.jpg
Reset X -> 320 and Y -> 213
Reading Image: sequences_of_hand_gesture_images/sequence_10_failed/005_palm_left.jpg
Left (X - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_10_failed/006_open_palm_left_up.jpg
Left and Up (X - 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_10_failed/007_open_palm_right_up.jpg
Up (Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_10_failed/008_open_palm_bottom.jpg
Location of Counter
Counter X:318
Counter Y:211
Reading Image: sequences_of_hand_gesture_images/sequence_10_failed/009_open_palm_center.jpg
Opps I don't understand this gesture
The Sequence of Actions is Failed
Current Location of Counter is:
Counter X:318
Counter Y:211
```

Figure 31: Failure results.

Instead of printing (319, 212) for the location of the counter, the current counter moved to (318, 211). From the result, we can see that the second image, the third image, the seventh, the eighth and ninth were not correctly tested.

The second image should be a fist at the bottom, which indicates to move the counter down by one unit. However, the first one was tested to be a fist at the center, which indicates to reset the counter back to (320, 213).


Figure 32: Fist analyzed image.

As we can see from Figure 32, the center of the fist, located at the middle finger, falls into the orange center of the image. The location of this gesture was not like what I expected at the bottom of the image center.

The third image should be a fist at the bottom, which indicates to move the counter down by one unit. However, this gesture was tested to be a fist at the center, which indicates to reset the counter back to (320, 213).


Figure 33: Fist analyzed image.

As we can see from Figure 33, the center of the fist, located at the middle finger, falls into the orange center of the image. Similar to the gesture in the second image, the location of this gesture was not like what I expected at the bottom of the image center.

The seventh image should be an open palm at the right, which indicates to move the counter right by one unit. However, the center of this open palm was tested to be at the top of the image, which indicates to move the counter up by one unit.


Figure 34: Open palm analyzed image.

As we can see from Figure 34, the center of the open palm, located at the bottom of the middle finger, falls into the upper green concentric section of the image. The location of this gesture was not like what I expected at the upper right of the image. The center of the open palm gesture is too close to the boarder between the upper green concentric section and the upper right blue area.

The eighth image should be an open palm at the bottom, which indicates to move the counter down by one unit. However, this open palm was tested to be at the center of the image, which indicates to print the result.

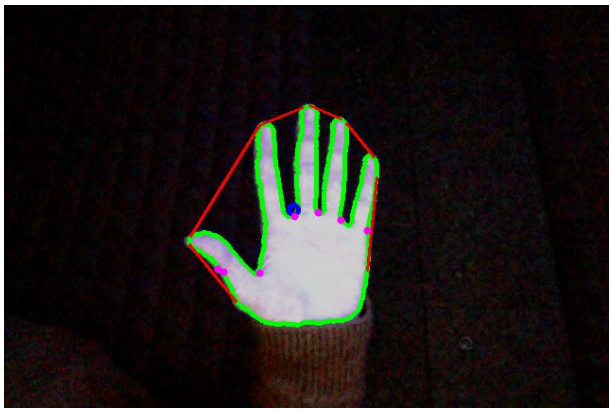


Figure 35: Open palm analyzed image.

As we can see from Figure 35, the center of the open palm, located between the bottom of the middle finger and the bottom of the index finger, falls into the orange center of the image. The location of this gesture was not like what I expected at the bottom of the image. Although the body of the gesture is at the bottom of the image, the center of the open palm gesture is too close to the boarder between the orange center and the lower green concentric section. It accidentally falls into the center of the image.

The ninth image should be an open palm at the center, which indicates to print the counter. However, the system prints that it did not understand the gesture.


Figure 36: Open palm analyzed image.

As we can see from Figure 36, although the center of the open palm falls into the center of the image, the fingers of this open palm gesture are not open enough. The condition of the third if-statement for testing open palm hand gestures is not fulfilled. As a result, the system does not recognize this gesture.

Step three shows that the visual interfaces are a measurable science. Seven of these sequences run correctly, in which the system gives proper answer. For the failures, some false positive and some false negative. Better algorithms and processing procedures are required to advance the performance of my current system.

# Section 4: Creativity Step

Implemented in creativity.py application
Following the suggestions from Professor Kender, I decided to make my system recognizes two more hand gestures, thumb up pose and horizontal palm pose.

**Thumb Up Gesture**

Following is the procedure of testing and adding the thumb up pose. As shown in Figure 37, the detected thumb up pose is quite different from the detected open palm gesture and the detected fist gesture.

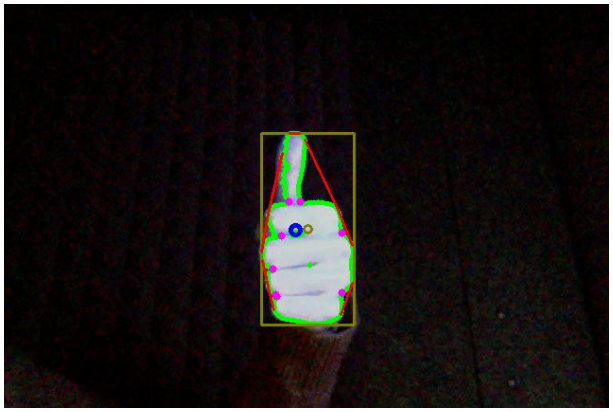


Figure 37: Analyzed thumb up gesture.

This new gesture has a smaller bounding box. Distinguishing it from open palm gestures and fist gestures require further analyses of its properties. By looking at the figure of thumb up gestures, the most significant difference between it and fist gestures is the size of bounding box. In my new application, I printed sizes of bounding box for all fist gestures, and I got the following results:

```
Reading Image: sequences_of_hand_gesture_images/sequence_8/004_fist_center.jpg
11375
Reading Image: sequences_of_hand_gesture_images/sequence_7/005_fist_center.jpg
15470
Reading Image: sequences_of_hand_gesture_images/sequence_6_failed/004_fist_right.jpg
13110
Reading Image: sequences_of_hand_gesture_images/sequence_5_failed/002_fist_center.jpg
10788
Reading Image: sequences_of_hand_gesture_images/sequence_3/004_fist_center.jpg
10788
```

Figure 38: Printed results for size of bounding box.

As we can see from figure 38, the sizes of bounding box for fist gestures are below 16,000 pixels. Compared with the size of bounding box for thumb up gestures, which is about 19,000 pixels, this difference is a significant. I changed my code to:

```
##[TESTING RESULT]Print the analyzing results
if defectPointCount >= 5 and boundingRect >= 45000:
        print colored("[RESULT] Hand Gesture is more like a --> Open Palm", "blue")
elif defectPointCount <= 5 and 18000 <= boundingRect < 35000:
        print colored("[Result] Hand Gesture is more like a --> Fist", "blue")
elif defectPointCount <= 5 and boundingRect < 18000:
        print colored("[Result] Hand Gesture is more like a --> Thumb Up", "blue")
else
        print "I don't recognized it"
```

The result is fairly accurate, as shown in Figure 39:

```
dyn-160-39-29-91:ph2369_assignment1 puconghan$ python data_reduction.py
Reading Image: gestures/fist_center.jpg
[Result] Hand Gesture is more like a --> Fist
Reading Image: gestures/open_palm.jpg
[RESULT] Hand Gesture is more like a --> Open Palm
Reading Image: gestures/thumb_up.jpg
[Result] Hand Gesture is more like a --> Thumb Up
```

Figure 39: Testing results.

Now, I added this new rule to my grammar in step 3:

**New grammar for thumb up: If a thumb up gesture is detected in the center of the screen, my system will accelerate the speed of the counter movement by one unit. In other words, once a thumb up gesture is detected in the center of the screen, further movements of counter will be increased by one (i.e. instead of moving the counter by one unit, we will move it by two unit).**

```
if defectPointCount >= 5 and boundingRect >= 45000:
    print colored("Location of Counter", "blue")
    print colored("Counter X:" + str(counterX), "red")
    print colored("Counter Y:" + str(counterY), "red")
    number_print = number_print + "print"
elif defectPointCount <= 5 and 18000 <= boundingRect < 35000:
    accelerator += 1
    print "Accelerator is increased to: " + str(accelerator)
elif defectPointCount <= 5 and boundingRect < 18000:
    print "Reset X -> 320 and Y -> 213"
    counterX = 320
    counterY = 213
else:
    print "Opps I don't understand this gesture"
```

Figure 40: Implementation for testing thumb up and fist.

As shown in Figure 40, I also declared a new accelerator variable at the beginning of the program. Once a thumb up gesture is detected, this accelerator variable is increased by one. As shown in Figure 41, I made associated changes to add this new variable to my previous works:

```
    elif (53.25 < math.sqrt((320 - xCenter)*(320 - xCenter) + (213 - yCenter)*(213 - yCenter)) <= 106.5):
        if xCenter < 320 and yCenter < 213:
            if math.sqrt((320 - xCenter)*(320 - xCenter)) > math.sqrt((213 - yCenter)*(213 - yCenter)):
                print "Left (X - " + str(accelerator) + ")"
                counterX -= accelerator
            else:
                print "Up (Y - " + str(accelerator) + ")"
                counterY -= accelerator
        elif xCenter > 320 and yCenter < 213:
            if math.sqrt((320 - xCenter)*(320 - xCenter)) > math.sqrt((213 - yCenter)*(213 - yCenter)):
                print "Right (X + " + str(accelerator) + ")"
                counterX += accelerator
            else:
                print "Up (Y - " + str(accelerator) + ")"
                counterY -= accelerator
        elif xCenter < 320 and yCenter > 213:
            if math.sqrt((320 - xCenter)*(320 - xCenter)) > math.sqrt((213 - yCenter)*(213 - yCenter)):
                print "Left (X - " + str(accelerator) + ")"
                counterX -= accelerator
            else:
                print "Down (Y + " + str(accelerator) + ")"
                counterY += accelerator
        elif xCenter > 320 and yCenter > 213:
            if math.sqrt((320 - xCenter)*(320 - xCenter)) > math.sqrt((213 - yCenter)*(213 - yCenter)):
                print "Right (X + " + str(accelerator) + ")"
                counterX += accelerator
            else:
                print "Down (Y + " + str(accelerator) + ")"
                counterY += accelerator
    elif (math.sqrt((320 - xCenter)*(320 - xCenter) + (213 - yCenter)*(213 - yCenter)) > 106.5):
        if xCenter < 320 and yCenter < 213:
            print "Left and Up (X - "+ str(accelerator) + " and Y - " + str(accelerator) + ")"
            counterX -= accelerator
            counterY -= accelerator
        if xCenter > 320 and yCenter < 213:
            print "Right and Up (X + "+ str(accelerator) + " and Y - " + str(accelerator) + ")"
            counterX += accelerator
            counterY -= accelerator
        if xCenter < 320 and yCenter > 213:
            print "Left and Down (X - "+ str(accelerator) + " and Y + " + str(accelerator) + ")"
            counterX -= accelerator
            counterY += accelerator
        if xCenter > 320 and yCenter > 213:
            print "Right and Down (X + "+ str(accelerator) + " and Y + " + str(accelerator) + ")"
            counterX += accelerator
            counterY += accelerator
```

Figure 41: New implementation with accelerator.

After making these changes my previous passed results still hold. I test the previous ten sequences after adding the second additional hand gesture, as shown in figure 49. Here, I test one additional sequence with my new thumb up gesture:

Additional sequence:


1. Accelerator = 2


2. Move left & down x=318 y=215


3. Print

I included this new sequence to the sequence_11 folder. After running my new program, this new sequence gets passed, as shown in figure 42:

```
Reading Folder: sequence_11
Reading Image: sequences_of_hand_gesture_images/sequence_11/001_thumb_up.jpg
Accelerator is increased to: 2
Reading Image: sequences_of_hand_gesture_images/sequence_11/002_fist_left_bottom.jpg
Left and Down (X - 2 and Y + 2)
Reading Image: sequences_of_hand_gesture_images/sequence_11/003_open_palm.jpg
Location of Counter
Counter X:318
Counter Y:215
The Sequence of Actions is Successfully Processed
dyn-160-39-29-91:ph2369_assignment1 puconghan$
```

Figure 42: Result for the new sequence.

Now, I have a new feature, accelerator added to my program. A new thumb up gesture triggers this new feature. I have showed that at least seven sequences plus one additional sequence pass the test.

## Horizontal Palm Gesture

Following is the procedure of testing and adding the horizontal palm pose as shown in Figure 14. As shown in Figure 43, the detected thumb up pose is quite different from the detected open palm gesture, the detected fist gesture and the thumb up gesture.

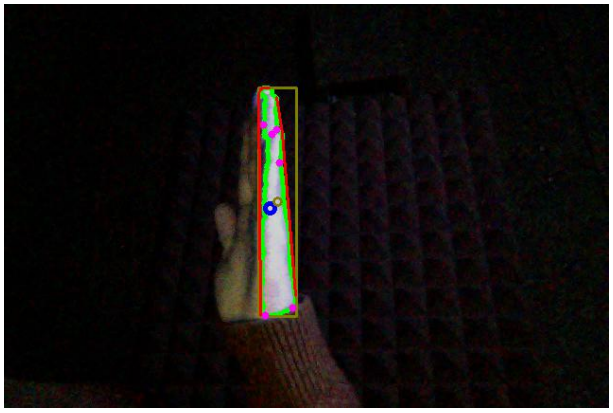


Figure 43: Analyzed side of open palm gesture.

This new gesture has a smaller horizontal bounding box. Distinguishing it from open palm gestures and fist gestures require further analyses of its properties. Using codes from section two, I tested and printed two variables, number of significant defects and size of bounding box.

Here are my testing codes for the new gesture:

```
##[TEST]Print the number of defect points
print "Number of Defect Points: " + str(defectPointCount)
        ##[TEST]Print the distance from defect point to the contour and the size of the bounding rectangle
if infile == "gestures/fist_center.jpg":
    print "Size of Bounding Rectangle: " + str(boundingRect)
if infile == "gestures/horizontal.jpg":
    print "Size of Bounding Rectangle: " + str(boundingRect)
if infile == "gestures/open_palm.jpg":
    print "Size of Bounding Rectangle: " + str(boundingRect)
if infile == "gestures/thumb_up.jpg":
    print "Size of Bounding Rectangle: " + str(boundingRect)
```

Figure 44: Printing methods

```
##[TESTING RESULT]Print the analyzing results
if defectPointCount >= 5 and boundingRect >= 45000:
    print colored("[RESULT] Hand Gesture is more like a --> Open Palm", "blue")
elif defectPointCount <= 5 and 19000 <= boundingRect < 35000:
    print colored("[RESULT] Hand Gesture is more like a --> Thumb Up", "blue")
elif defectPointCount <= 5 and 10000 <= boundingRect < 19000:
    print colored("[Result] Hand Gesture is more like a --> Fist", "blue")
elif defectPointCount <= 5 and boundingRect < 10000:
    print colored("[Result] Hand Gesture is more like a --> Horizontal Palm", "blue")
else:
    print "Opps I don't understand this gesture"
```

Figure 45: Testing Methods

The printed result is:

```
dyn-160-39-29-91:ph2369_assignment1 puconghan$ python data_reduction.py
Reading Image: gestures/fist_center.jpg
7.21110255093
Number of Defect Points: 1
Size of Bounding Rectangle: 10788
[Result] Hand Gesture is more like a --> Fist
Reading Image: gestures/horizontal.jpg
10.6301458127
Number of Defect Points: 2
Size of Bounding Rectangle: 9640
[Result] Hand Gesture is more like a --> Horizontal Palm
Reading Image: gestures/open_palm.jpg
9.48683298051
Number of Defect Points: 5
Size of Bounding Rectangle: 53775
[RESULT] Hand Gesture is more like a --> Open Palm
Reading Image: gestures/thumb_up.jpg
13.0384048104
Number of Defect Points: 1
Size of Bounding Rectangle: 19796
[RESULT] Hand Gesture is more like a --> Thumb Up
dyn-160-39-29-91:ph2369_assignment1 puconghan$
```

Figure 46: Result of the testing.

As we can see from Figure 46, horizontal palm gesture has a smaller bounding rectangle, which is less than 10,000. This is quite significant compared to the bounding rectangle of fist, open palm and thumb up gestures. My new testing rules in Figure 45 captures these four gestures and distinguishes horizontal palm from the four gestures, as shown in Figure 46. After finishing this testing step, I am ready to expand my grammar.

Now, I added this new rule to my grammar in step 3:

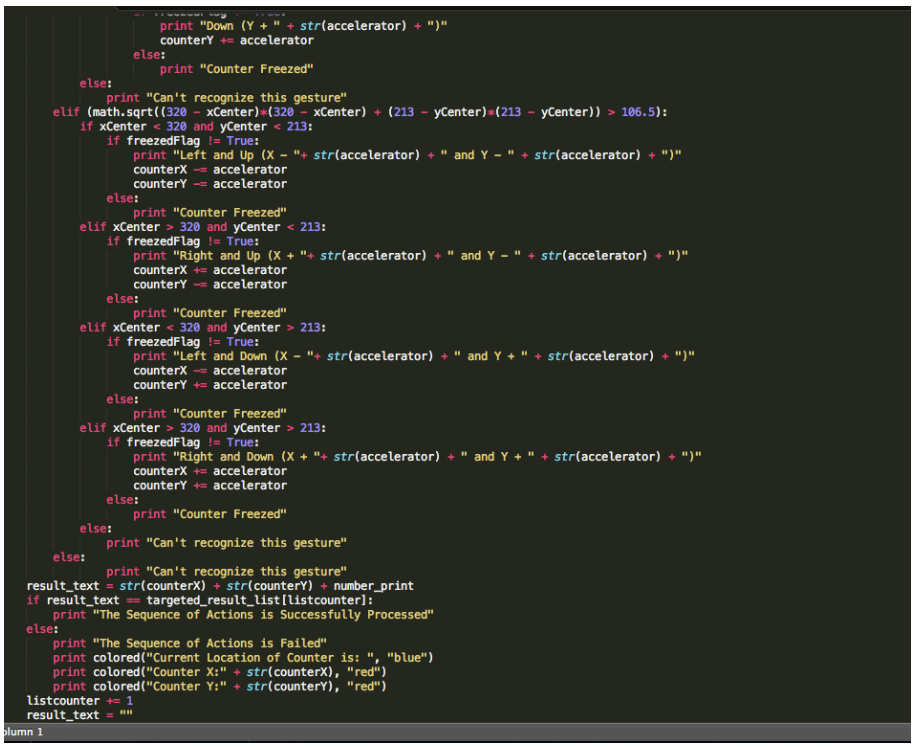
**New grammar for horizontal palm: If a horizontal palm gesture is detected in the center of the screen, my system will freeze the movement of counter to the current location. In other words, once a horizontal palm gesture is detected in the center of the screen, further movements of counter will be eliminated.**

Here is my new implementation after expanding my grammar to recognize two additional gestures:

```
if defectPointCount >= 5 and boundingRect >= 45000:
    print colored("Location of Counter", "blue")
    print colored("Counter X:" + str(counterX), "red")
    print colored("Counter Y:" + str(counterY), "red")
    number_print = number_print + "print"
elif defectPointCount <= 5 and 19000 <= boundingRect < 35000:
    accelerator += 1
    print "Accelerator is increased to: " + str(accelerator)
elif defectPointCount <= 5 and 10000 <= boundingRect < 19000:
    if freezedFlag != True:
        print "Reset X -> 320 and Y -> 213"
        counterX = 320
        counterY = 213
    else:
        print "Counter Freezed"
elif defectPointCount <= 5 and boundingRect < 10000:
    print "Counter Freezed. No further movements are allowed"
    freezedFlag = False
else:
    print "Opps I don't understand this gesture"
```

Figure 47: Code to testing the gesture at center region of the screen.

```
elif (53.25 < math.sqrt((320 - xCenter)*(320 - xCenter) + (213 - yCenter)*(213 - yCenter)) <= 106.5):
    if xCenter < 320 and yCenter < 213:
        if math.sqrt((320 - xCenter)*(320 - xCenter)) > math.sqrt((213 - yCenter)*(213 - yCenter)):
            if freezedFlag != True:
                print "Left (X - " + str(accelerator) + ")"
                counterX -= accelerator
            else:
                print "Counter Freezed"
        else:
            if freezedFlag != True:
                print "Up (Y - " + str(accelerator) + ")"
                counterY -= accelerator
            else:
                print "Counter Freezed"
    elif xCenter > 320 and yCenter < 213:
        if math.sqrt((320 - xCenter)*(320 - xCenter)) > math.sqrt((213 - yCenter)*(213 - yCenter)):
            if freezedFlag != True:
                print "Right (X + " + str(accelerator) + ")"
                counterX += accelerator
            else:
                print "Counter Freezed"
        else:
            if freezedFlag != True:
                print "Up (Y - " + str(accelerator) + ")"
                counterY -= accelerator
            else:
                print "Counter Freezed"
    elif xCenter < 320 and yCenter > 213:
        if math.sqrt((320 - xCenter)*(320 - xCenter)) > math.sqrt((213 - yCenter)*(213 - yCenter)):
            if freezedFlag != True:
                print "Left (X - " + str(accelerator) + ")"
                counterX -= accelerator
            else:
                print "Counter Freezed"
        else:
            if freezedFlag != True:
                print "Down (Y + " + str(accelerator) + ")"
                counterY += accelerator
            else:
                print "Counter Freezed"
    elif xCenter > 320 and yCenter > 213:
        if math.sqrt((320 - xCenter)*(320 - xCenter)) > math.sqrt((213 - yCenter)*(213 - yCenter)):
            if freezedFlag != True:
                print "Right (X + " + str(accelerator) + ")"
                counterX += accelerator
            else:
                print "Counter Freezed"
        else:
            if freezedFlag != True:
```



```
                print "Down (Y + " + str(accelerator) + ")"
                counterY += accelerator
            else:
                print "Counter Freezed"
        else:
            print "Can't recognize this gesture"
    elif (math.sqrt((320 - xCenter)*(320 - xCenter) + (213 - yCenter)*(213 - yCenter)) > 106.5):
        if xCenter < 320 and yCenter < 213:
            if freezedFlag != True:
                print "Left and Up (X - "+ str(accelerator) + " and Y - " + str(accelerator) + ")"
                counterX -= accelerator
                counterY -= accelerator
            else:
                print "Counter Freezed"
        elif xCenter > 320 and yCenter < 213:
            if freezedFlag != True:
                print "Right and Up (X + "+ str(accelerator) + " and Y - " + str(accelerator) + ")"
                counterX += accelerator
                counterY -= accelerator
            else:
                print "Counter Freezed"
        elif xCenter < 320 and yCenter > 213:
            if freezedFlag != True:
                print "Left and Down (X - "+ str(accelerator) + " and Y + " + str(accelerator) + ")"
                counterX -= accelerator
                counterY += accelerator
            else:
                print "Counter Freezed"
        elif xCenter > 320 and yCenter > 213:
            if freezedFlag != True:
                print "Right and Down (X + "+ str(accelerator) + " and Y + " + str(accelerator) + ")"
                counterX += accelerator
                counterY += accelerator
            else:
                print "Counter Freezed"
        else:
            print "Can't recognize this gesture"
    else:
            print "Can't recognize this gesture"
result_text = str(counterX) + str(counterY) + number_print
if result_text == targeted_result_list[listcounter]:
    print "The Sequence of Actions is Successfully Processed"
else:
    print "The Sequence of Actions is Failed"
    print colored("Current Location of Counter is: ", "blue")
    print colored("Counter X:" + str(counterX), "red")
    print colored("Counter Y:" + str(counterY), "red")
listcounter += 1
result_text = ""
olumn 1
```

Figure 48: New implementation with accelerator and freezer.

Once a hand gesture is detected in the center of the screen, the testing code in Figure 47 will be executed. In my new testing code, I added one additional if-statement to capture the horizontal palm gesture in which the area of bounding box is less than 10,000 pixels. Before adding this new if-statement, I adjusted the other three if-statements to clearly distinguish the four hand gestures:

Size of bounding box of thumb up gesture is between 19,000 pixels (included) and 35,000 pixels.

Size of bounding box of fist gesture is between 10,000 pixels (included) and 19,000 pixels.

Size of bounding box of horizontal palm gesture is less than 10,000 pixels.

After making these changes my previous passed results still hold, as shown in Figure 49:

```
dyn-160-39-29-91:ph2369_assignment1 puconghan$ python creativity.py


Reading Folder: sequence_1
Reading Image: sequences_of_hand_gesture_images/sequence_1/fist_center.jpg
Reset X -> 320 and Y -> 213
Reading Image: sequences_of_hand_gesture_images/sequence_1/fist_left_bottom.jpg
Left and Down (X - 1 and Y + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_1/fist_left_up.jpg
Left and Up (X - 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_1/fist_right_bottom.jpg
Right and Down (X + 1 and Y + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_1/fist_right_up.jpg
Right and Up (X + 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_1/open_palm.jpg
Location of Counter
Counter X:320
Counter Y:213
The Sequence of Actions is Successfully Processed


Reading Folder: sequence_2
Reading Image: sequences_of_hand_gesture_images/sequence_2/001_open_palm_right_bottom.jpg
Right and Down (X + 1 and Y + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_2/002_fist_right_bottom.jpg
Right and Down (X + 1 and Y + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_2/003_fist_left_bottom.jpg
Left and Down (X - 1 and Y + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_2/004_fist_middle_right.jpg
Right (X + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_2/005_open_palm_canter.jpg
Location of Counter
Counter X:322
Counter Y:216
The Sequence of Actions is Successfully Processed


Reading Folder: sequence_3
Reading Image: sequences_of_hand_gesture_images/sequence_3/001_fist_left.jpg
Right (X + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_3/002_open_palm_right_up.jpg
Right and Up (X + 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_3/003_open_plam_left_up.jpg
Left and Up (X - 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_3/004_fist_center.jpg
Reset X -> 320 and Y -> 213
Reading Image: sequences_of_hand_gesture_images/sequence_3/005_fist_right.jpg
Right (X + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_3/006_open_palm_center.jpg
Location of Counter
Counter X:321
Counter Y:213
The Sequence of Actions is Successfully Processed


Reading Folder: sequence_4
Reading Image: sequences_of_hand_gesture_images/sequence_4/001_fist_right.jpg
Right (X + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_4/002_fist_left_bottom.jpg
Left and Down (X - 1 and Y + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_4/003_fist_left_up.jpg
Left and Up (X - 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_4/004_open_palm_right_up.jpg
Right and Up (X + 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_4/005_fist_left_bottom.jpg
Left and Down (X - 1 and Y + 1)
```

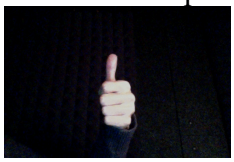
```
Reading Image: sequences_of_hand_gesture_images/sequence_4/006_open_palm_center.jpg
Location of Counter
Counter X:319
Counter Y:213
The Sequence of Actions is Successfully Processed


Reading Folder: sequence_5_failed
Reading Image: sequences_of_hand_gesture_images/sequence_5_failed/001_fist_left_up.jpg
Left and Up (X - 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_5_failed/002_fist_center.jpg
Reset X -> 320 and Y -> 213
Reading Image: sequences_of_hand_gesture_images/sequence_5_failed/003_fist_left_up.jpg
Left and Up (X - 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_5_failed/004_fist_right_up.jpg
Right and Up (X + 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_5_failed/005_open_palm_center.jpg
Up (Y - 1)
The Sequence of Actions is Failed
Current Location of Counter is:
Counter X:320
Counter Y:210


Reading Folder: sequence_6_failed
Reading Image: sequences_of_hand_gesture_images/sequence_6_failed/001_open_palm_right.jpg
Up (Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_6_failed/002_open_palm_right_bottom.jpg
Right and Down (X + 1 and Y + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_6_failed/003_fist_right_up.jpg
Right and Up (X + 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_6_failed/004_fist_right.jpg
Reset X -> 320 and Y -> 213
Reading Image: sequences_of_hand_gesture_images/sequence_6_failed/005_open_palm_center.jpg
Location of Counter
Counter X:320
Counter Y:213
The Sequence of Actions is Failed
Current Location of Counter is:
Counter X:320
Counter Y:213


Reading Folder: sequence_7
Reading Image: sequences_of_hand_gesture_images/sequence_7/001_open_palm_right_bottom.jpg
Right and Down (X + 1 and Y + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_7/002_open_palm_left_bottom.jpg
Left and Down (X - 1 and Y + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_7/003_fist_right_up.jpg
Right and Up (X + 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_7/004_open_palm_right_bottom.jpg
Right and Down (X + 1 and Y + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_7/005_fist_center.jpg
Reset X -> 320 and Y -> 213
Reading Image: sequences_of_hand_gesture_images/sequence_7/006_open_palm_center.jpg
Location of Counter
Counter X:320
Counter Y:213
The Sequence of Actions is Successfully Processed


Reading Folder: sequence_8
Reading Image: sequences_of_hand_gesture_images/sequence_8/001_fist_right_up.jpg
Right and Up (X + 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_8/002_open_palm_right_up.jpg
Right and Up (X + 1 and Y - 1)
```

```
Reading Image: sequences_of_hand_gesture_images/sequence_8/003_open_palm_left_up.jpg
Left and Up (X - 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_8/004_fist_center.jpg
Reset X -> 320 and Y -> 213
Reading Image: sequences_of_hand_gesture_images/sequence_8/005_fist_left_up.jpg
Left and Up (X - 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_8/006_open_palm_center.jpg
Location of Counter
Counter X:319
Counter Y:212
The Sequence of Actions is Successfully Processed


Reading Folder: sequence_9
Reading Image: sequences_of_hand_gesture_images/sequence_9/001_fist_bottom.jpg
Down (Y + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_9/002_fist_center.jpg
Reset X -> 320 and Y -> 213
Reading Image: sequences_of_hand_gesture_images/sequence_9/003_fist_right_up.jpg
Right and Up (X + 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_9/004_open_palm_left.jpg
Left (X - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_9/005_open_palm_up.jpg
Up (Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_9/006_open_palm_center.jpg
Location of Counter
Counter X:320
Counter Y:211
The Sequence of Actions is Successfully Processed


Reading Folder: sequence_10_failed
Reading Image: sequences_of_hand_gesture_images/sequence_10_failed/001_fist_right.jpg
Right (X + 1)
Reading Image: sequences_of_hand_gesture_images/sequence_10_failed/002_fist_bottom.jpg
Reset X -> 320 and Y -> 213
Reading Image: sequences_of_hand_gesture_images/sequence_10_failed/003_fist_bottom.jpg
Reset X -> 320 and Y -> 213
Reading Image: sequences_of_hand_gesture_images/sequence_10_failed/004_fist_center.jpg
Reset X -> 320 and Y -> 213
Reading Image: sequences_of_hand_gesture_images/sequence_10_failed/005_palm_left.jpg
Left (X - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_10_failed/006_open_palm_left_up.jpg
Left and Up (X - 1 and Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_10_failed/007_open_palm_right_up.jpg
Up (Y - 1)
Reading Image: sequences_of_hand_gesture_images/sequence_10_failed/008_open_palm_bottom.jpg
Location of Counter
Counter X:318
Counter Y:211
Reading Image: sequences_of_hand_gesture_images/sequence_10_failed/009_open_palm_center.jpg
Opps I don't understand this gesture
The Sequence of Actions is Failed
Current Location of Counter is:
Counter X:318
Counter Y:211


Reading Folder: sequence_11
Reading Image: sequences_of_hand_gesture_images/sequence_11/001_thumb_up.jpg
Accelerator is increased to: 2
Reading Image: sequences_of_hand_gesture_images/sequence_11/002_fist_left_bottom.jpg
Left and Down (X - 2 and Y + 2)
Reading Image: sequences_of_hand_gesture_images/sequence_11/003_open_palm.jpg
Location of Counter

Counter X:318
Counter Y:215
The Sequence of Actions is Successfully Processed


Reading Folder: sequence_12
Reading Image: sequences_of_hand_gesture_images/sequence_12/001_thumb_up.jpg
Accelerator is increased to: 2
Reading Image: sequences_of_hand_gesture_images/sequence_12/002_horizontal.jpg
Counter Freezed. No further movements are allowed
Reading Image: sequences_of_hand_gesture_images/sequence_12/003_fist_left_bottom.jpg
Counter Freezed
Reading Image: sequences_of_hand_gesture_images/sequence_12/004_open_palm.jpg
Location of Counter
Counter X:320
Counter Y:213
The Sequence of Actions is Successfully Processed
dyn-160-39-29-91:ph2369_assignment1 puconghan$
```

Figure 49: Results from the creativity part.

This proved that at least seven of my previous sequences are working. Now, I just need to test one additional sequence with both my new thumb up gesture and horizontal palm gesture:

Additional sequence:



1. Accelerator = 2

2. Freeze counter

3. Move left & down x=320 y=213

4. Print

I included this new sequence to the sequence_12 folder. After running my new program, this new sequence gets passed, as shown in figure 50:

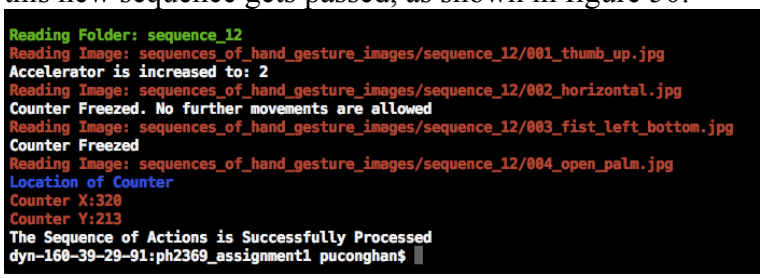



Figure 50: Result of my new sequence.

Now, I have two new features, both accelerators and freezers are added to my program. A new horizontal palm gesture triggers the freezer. I have showed that at least seven sequences plus one additional sequence pass the test.

Work Cited

"Gesture Detection." *Camera Based Interactive Wall Display With Hand Detection Using LED Lights*. Google Sites, n. d. Web. Web. 3 Feb. 2013. <https://sites.google.com/site/tkp98099>.

Kjeldsen, Frederik C. M. . "Visual Interpretation of Hand Gestures as a Practical Interface Modality." COLUMBIA UNIVERSITY. (1997).

"OpenCV Python Tutorials." *OpenCV Python*. Google Blogger. Web. 3 Feb 2013. <http://opencvpython.blogspot.com/2012/06/hi-this-article-is-tutorial-which-try.html>.

"OpenCV 2.4.3 documentation." *OpenCV Documentation*. OpenCV Developer Team, 26 Dec 2012. Web. 3 Feb 2013. <http://docs.opencv.org/index.html>.