Valdo Joao Assignment: 3D Pointer and Simple Stroke Recognition

The assignment can be divided into three main parts which are:

I. Create an application that isolates the hand in the scene

There are many possible approaches to solve this problem, each with different complexity and accuracy, few of them are:

1. **Shape Analysis**: taking advantage of landmarks in order to find the mean shape of hands
2. **Segmentation techniques:**
   - **a)** Segmentation by skin color: the idea is quite simple, find the color of the user's skin and use it as a threshold to binarize the image.
   - **b)** Segmentation by distance: Assuming that the hands are in front of the body, this technique calculates a threshold of the closest object in front of the camera (hands in our case).
3. **Model**: Train a deep learning model or haar cascade classifier in order to find the hand. This can be seen as a binary class classification problem.

Given the characteristics of the dataset I decided to apply the Segmentation by distance technique which proved to be reliable in the majority of the test cases.

II. Compute a pointer located in that hand (in depth-image space)

In order to solve this problem there a two possible approaches:

a) A manual approach such that for each frame we calculate the center of the hand by dividing the image into smaller parts and keeping the one with the bigger mass in it. The smallest window will have the biggest probability to have the hand segmented inside.

b) An automated approach such that for each frame we calculate the center of the mass utilizing OpenCV available resources.

Approach a) might have a slightly better precision but at the cost of higher latency, so I decided to apply the approach b).

III. Uses the sequence of positions to classify the stroke

This problem can be solved as Deep Learning mutually exclusive Multi-Class classification problem where we have a sequence of points as input and a final decision as output indicating to which Stroke class the points belong to.

Another option is to utilize the suggested $1 Unistroke Recognizer, a 2-D single-stroke recognizer designed for rapid prototyping of gesture-based user interfaces.

I decided to apply the $1 Unistroke Recognizer because of its simplicity in classifying a stroke.

Assumptions

In order to deal with the gestures performed with two hands, I divided the screen in two vertical parts. Each part saves the points of one hand then each hand stroke is classified singularly. Of course this is not an optimal implementation as there is some ambiguity when the hands swap positions.

The biggest challenge is to distinguish between two hands, let's say we have hands A and B moving on (x, y) axis. Ideally we should be able to track both hands and store their points on corresponding vector A and vector B.

Right now the algorithm is able to track both hands A and B but is not able to say which is hand A and which is hand B. So if the hands swap positions, it might happen than some coordinates of hand A goes into vector B and vice versa.

The algorithm was able to identify precisely the two hands or one hand situation. The algorithm was able to predict precisely five gestures (X), was able to grab the most frequent gestures of the open/closed hand sequence (X) and misclassified four gestures (X).

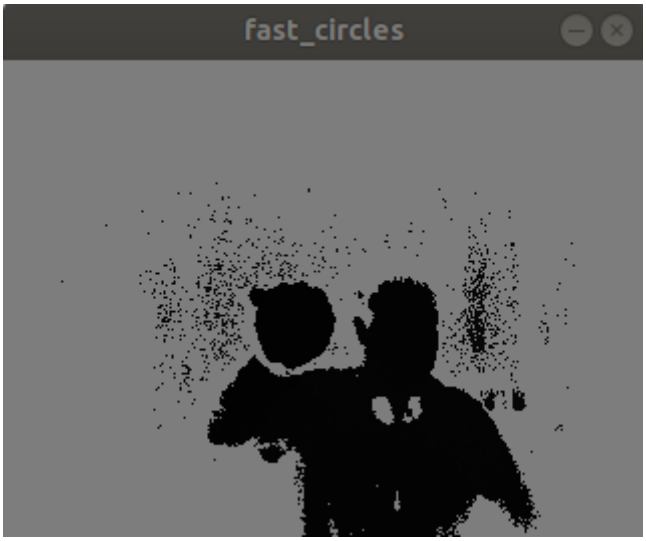
In the case where the two hands are performing same actions (multiple_shapes_1), the algorithm recognized the same gesture for both of them with almost the same prediction score.

The algorithm was able to eliminate false positives up to 95%

Stroke	Recognized gesture	1$ Prediction Score	Hand	
fast_circles	fast_circles	0.668237	One hand	X
gestures_two_hands	fast_circles	0.450969	Left-hand	
	rectangle_cw	0.700298	Right-hand	
gestures_two_hands_swap	circle_ccw_far	0.556355	Left-hand	
	fast_circles	0.487673	Right-hand	
sequence_closed_hand	rectangle_ccw	0.378731	One hand	X
sequence_open_hand	circle_sequence	0.428114	One hand	X
sequence_small_shapes	sequence_small_shapes	0.603275	One hand	X
circle_ccw	circle_ccw	0.97696	One hand	X
circle_ccw_far	circle_ccw_far	0.982804	One hand	X
circle_sequence	circle_sequence	0.692057	One hand	X
multiple_shapes_1	fast_circles	0.493951	Left-hand	
	fast_circles	0.469132	Right-hand	
rectangle_ccw	circle_sequence	0.432314	One hand	X
rectangle_cw	fast_circles	0.463554	One hand	X
star	circle_sequence	0.5217	One hand	X
zigzag	circle_ccw_far	0.608705	One hand	X

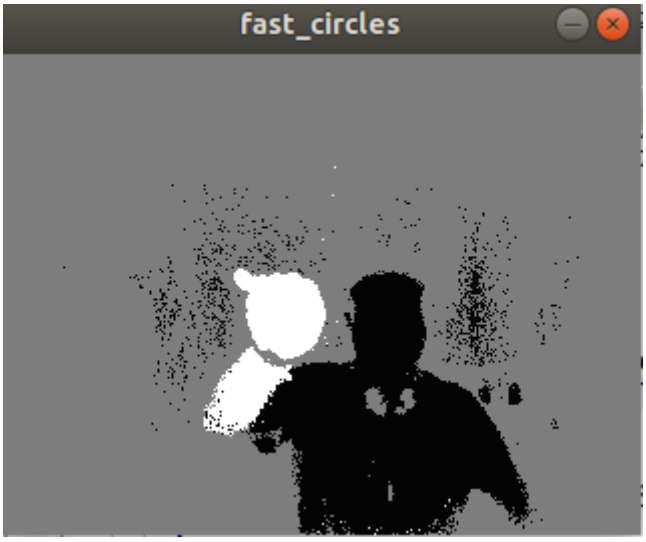
Interaction Between Classes And Algorithm Overview

1. **Class MultiStrokeRec.cpp:** Main Class in order to Run the Project
2. **DirAccess.cpp:** class to load the and simulate a video stream in the application



3. **ImageProcessing.cpp:** The core of the project

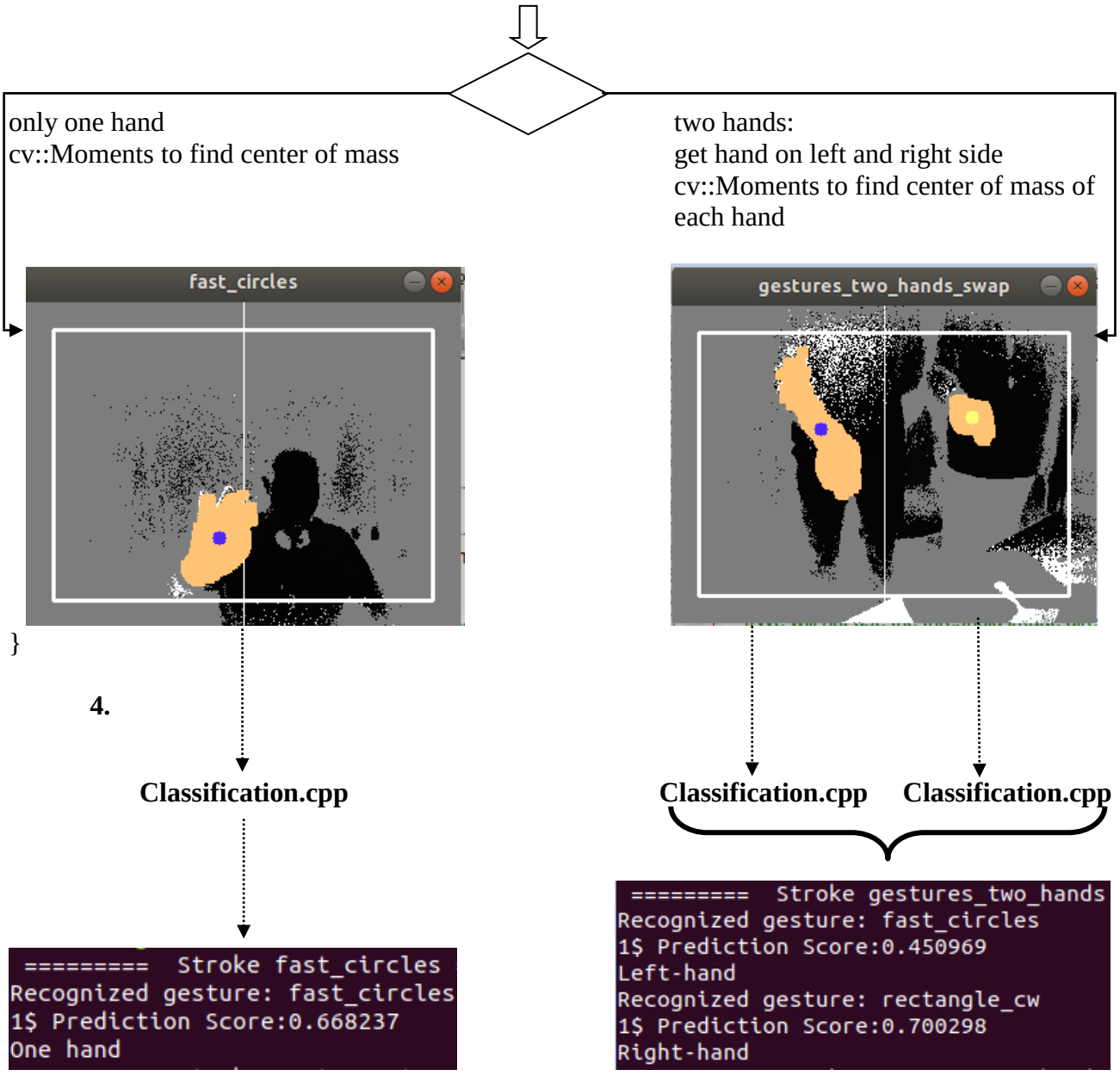
```
//Calculates optimized threshold using histogram equalization by iterations
void ImageProc::HistEqu(cv::Mat* ImgSrc, int32_t* Thres, uint16_t bins)
{
  this->Hist(ImgSrc, &Equhist, bins)              //Calculate histogram
  this->CumHist(&Equhist, &EquCumHist)      //Calculate cumulative histogram
  ...
}

void ImageProc::Thres(cv::Mat& ImgSrc, cv::Mat& ImgDest, int32_t* Thres)
{
    //Segment the hand based on a threshold
```




```
}
```

//Calculates the center of the mass utilizing OpenCV available resources
void ImageProc::BlobAnalysis(cv::Mat& ImgSrc, cv::Mat& ImgDest)
{

a) Calculate the ROI
b) Divide the screen in two vertical parts
c) cv::inRange detects object based on range of pixel values that fall under the threshold
d) cv::erode and cv::dilate to clean image and remove false positives. It also makes the threshold object more visible
e) cv::findContours to join continuous points with same intensity
f) get the biggest contour and the second biggest (if exists)
g) Verify how many hands were found:



Classification.cpp uses the sequence of positions to classify the stroke with $1 Unistroke Recognizer.

Implementation Guide

Dependencies

OpenCV https://docs.opencv.org/3.3.1/d7/d9f/tutorial_linux_install.html

Go to the project dir

open CMakeLists.txt and adjust the path to your OpenCV dir

in command line cd **<root project dir>**

Build

cmake CMakeLists.txt

make

Run

./bin/MultistrokeRec