Image Analysis 1st Assignment report

Date:2019-02-06

1. Identification of person and assignment:

* Group member:

Yukuan Hao, Liming Yang, Liang Gu

* Assignment:

1. recognition of objects: find the connected components from the input image.
2. Calculate the features of objects:

a). Counting the number of objects.

b). Some features of objects, like area, perimeter, circularity, second moment and so forth.

c). Classify objects: put the similar object into one group.

1. Brief problems definition:
   1. How to change a grayscale image into a binary image
   2. How to use morphological filters to enhance the image
   3. How to find the object and count the object
   4. How to calculate the features of the object
   5. How to classify the objects
2. Summary of choices made for the solution:

Our group choose C++ and OpenCV library to solve the problems, we create two programs to solve the problems, one is using OpenCV library to solve the problems, the other one we use some basic function in OpenCV library and create some functions to solve the problems.

For the first problem we mentioned in section II, we use the OpenCV library function “threshold” to create a binary image. We also try to use “adaptive threshold” function, but the result is not good, it contains many noises on the image, so we have to choose a threshold value to make a binary image, we design a slider to allow the user to choose the threshold value. At first, we plan to create a histogram of the image and using Otsu’s method to set the threshold to create the binary, but after we create the histogram image, we have no idea to statistics the information on the histogram, so we give up.

For the second problem, we use the OpenCV library “dilate” and “erode” function to make closing objects.

For the third problem, we use two solutions; the first one is using OpenCV library “Canny” and “findContours” function to get each object’s contour, using vector and 2D vector to store the different objects’ feature(like moments, object’s center point) and points and using “drawContours” function to draw the objects in a different colour to represent. Also, we count the object by calling the parameter “m00” in “moment” class in the OpenCV library and count the 2D vector’s size to get the number of the object. In the second solution, we write a function by ourselves by following the pseudocode on the textbook(Computer Vision, Linda G. Shapiro, George C. Stockman, Prentice Hall, 2001, page 71,algorithm3. Recursive connected component). And we count the object by changing the pixel value in different object, we set the pixel of all the points belonging to No.i object to i. (but the number of objects must less than 254) Another method is using the formula”(I-E)/4” to count the number of objects, but if the object contains holes it won’t get the correct answer.

For the fourth problem, we also use two solutions, the first one we use the OpenCV library class “moment”, ”arclength’ function, in “moment” class, it contains many parameters and we can use those parameters to calculate the area of object, the second moment of object, and the axis of least inertia. Using “arclength” function, we can get the perimeter of the object; then we can use perimeter and area to calculate the circularity of the object. In the second solution, we create a function to traverse the entire image and count how many pixels in different objects, at the same time, we also collect the x, and y coordinates information to calculate the second moment and the axis of least inertia. (we failed to create a function to calculate the perimeter of the object because of the formula to count the perimeter in the ppt is not correct )

Before we solve the fifth problem, we use “approxPolyDP,” “ boundRect,” and “minEnclosingCircle ” function to draw the accurate curve, bounding box, and minimum enclosing circle on the image. (Those codes are from <https://docs.opencv.org/2.4/doc/tutorials/imgproc/shapedescriptors/bounding_rects_circles/bounding_rects_circles.html> , we also site this website in our program as a comment)

For the fifth problem, We try to create a function to classify the object, but we do not have enough knowledge to do that. Then, we attempt to use the “matchShapes” function to calculate the similarity of different objects. ( matchShapes function return a value, if this value trend to zero, the two objects are more similar) But if the image has a large number of the object, this function will execute many times, and sometimes we can not see the entire result from the console windows, so we put that code as a comment.

1. 2example images and results of running the program on them:

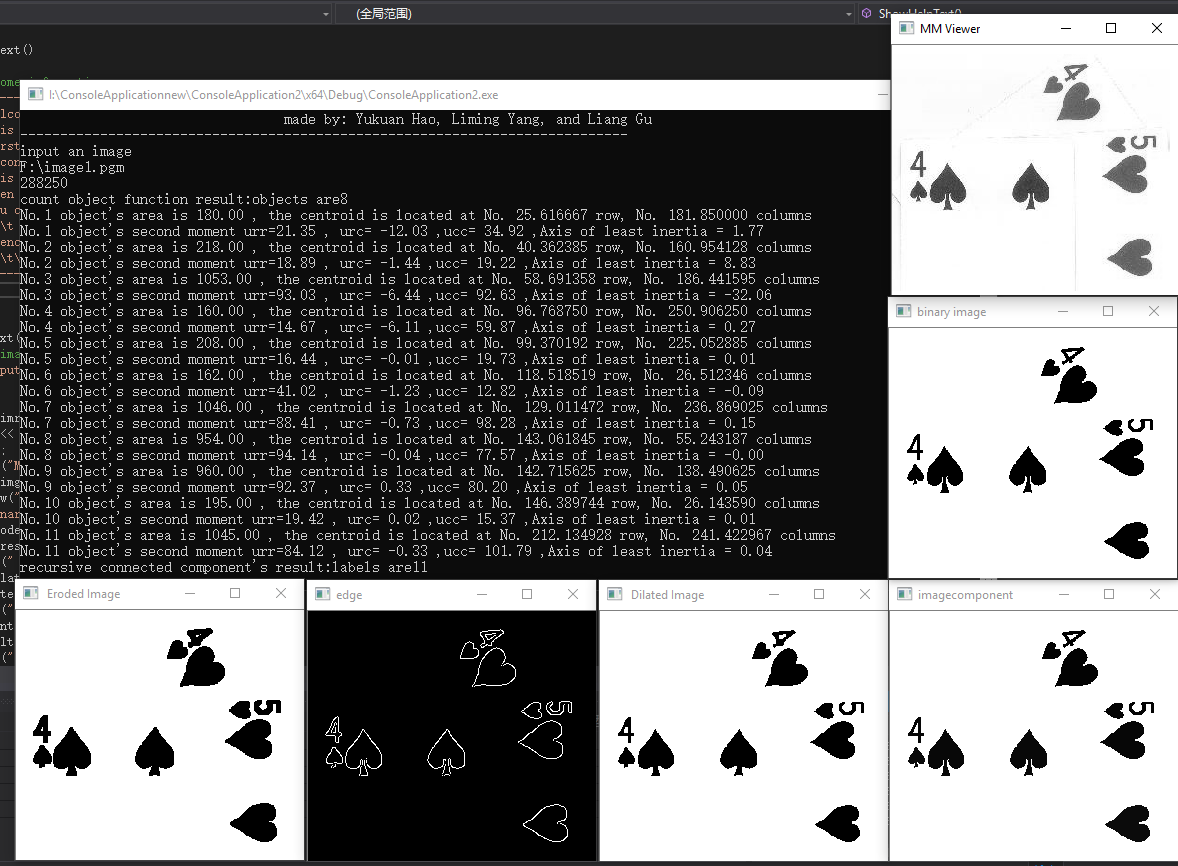
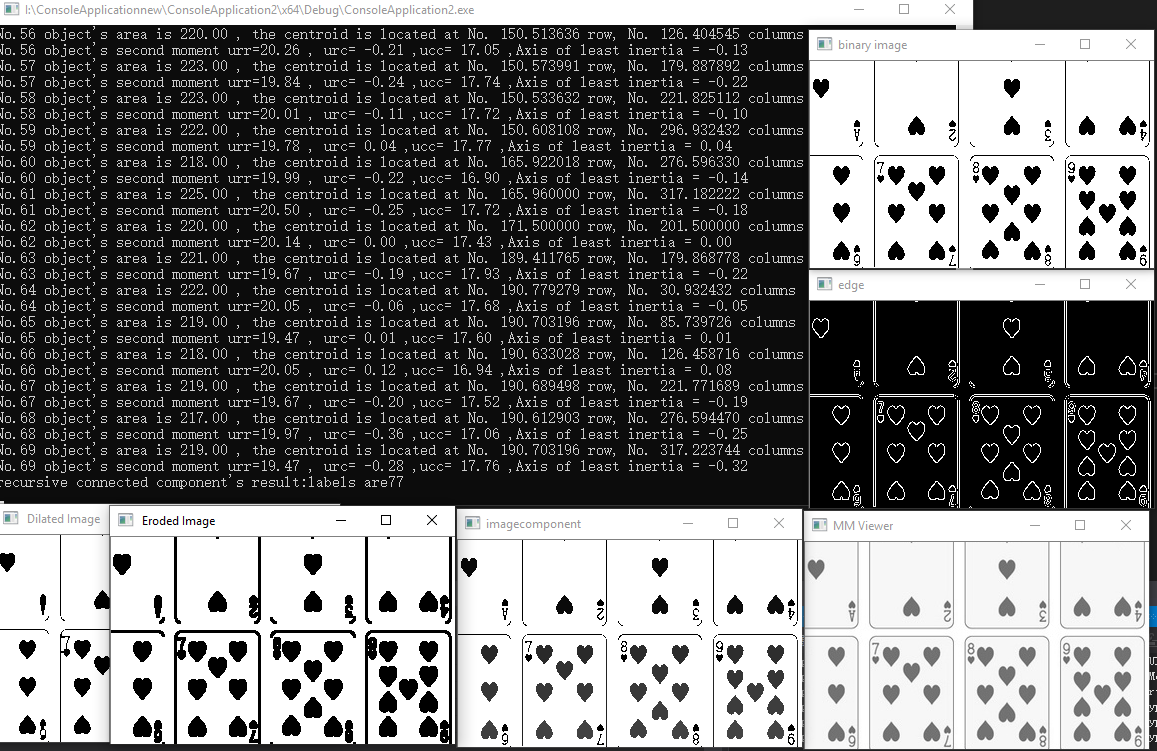
 

Fig.1, and Fig.2 the program which creates functions by ourself to analysis image1.pgm and image3.pgm (although the object colour is black, the pixel from different objects are slightly different)

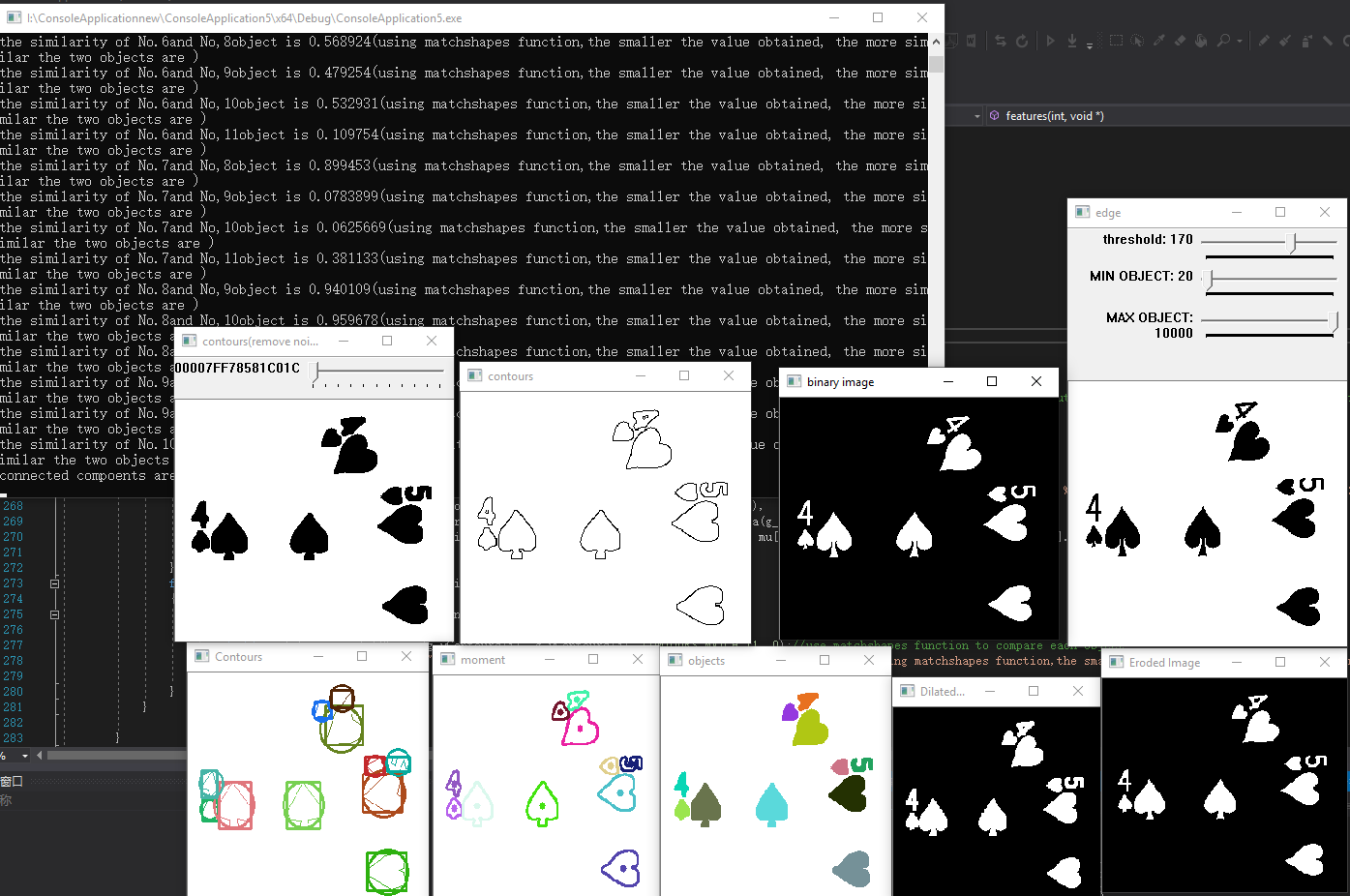
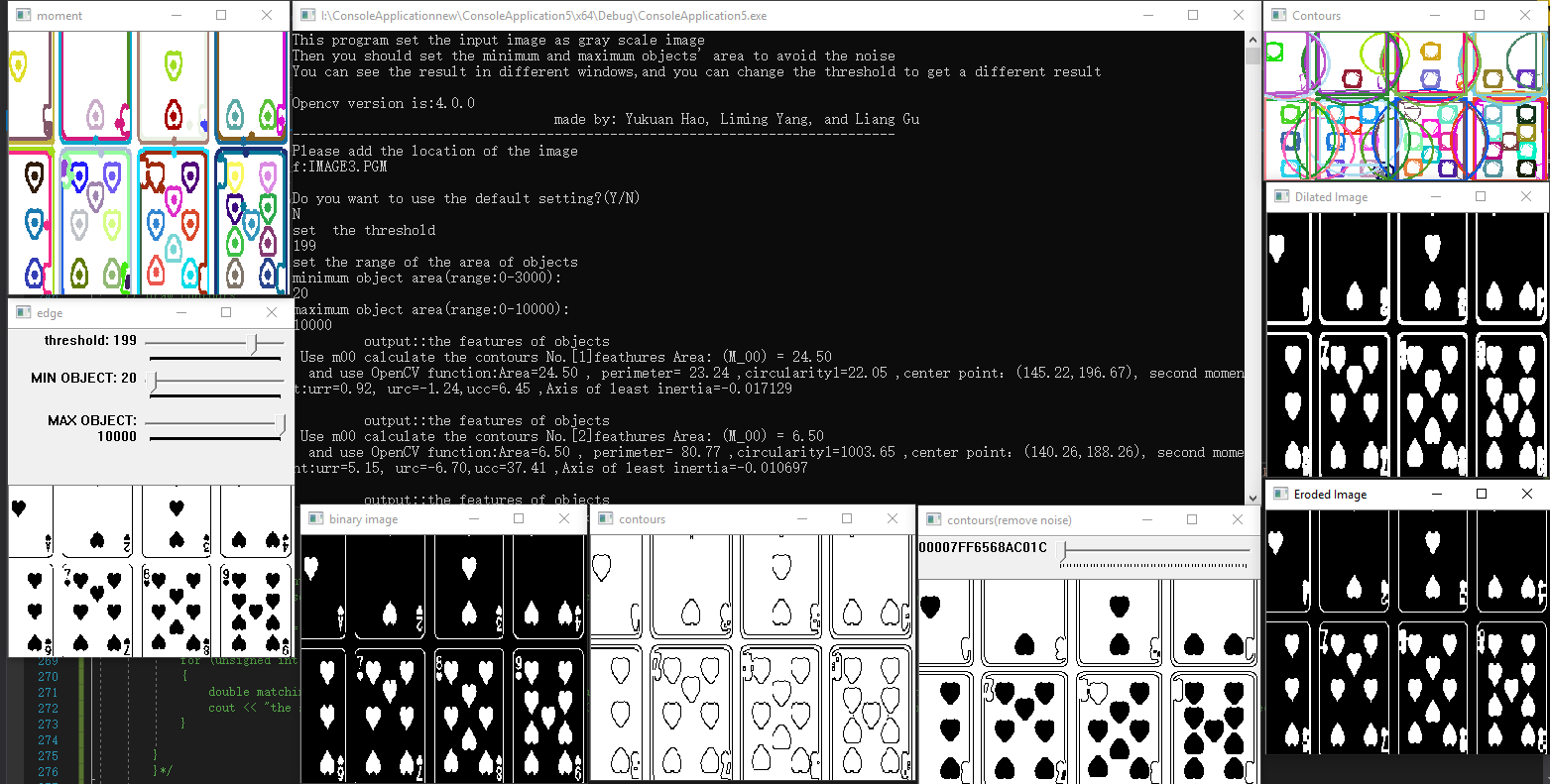
 

Fig.3 and Fig.4 the program which we use the OpenCV library function to analysis the image1.pgm and image3.pgm

1. A brief discussion of the result:

During the experiment, we use image1.pgm as a sample to analyze the image, so when we use image1.pgm as the input image, we get the best result, but other images we do not get a perfect result.

For our first program, when we use image2.pgm as the input image, the program does not work, because of stack overflow. For other images, the program can not detect all the object; sometimes the program recognizes the shadow as an object, sometimes it can not remove the card edges completely. We should improve the preprocess to image next time, like using a different mask to dilate and erode the image to get a better closing object, then optimize the algorithm to avoid stack overflow. In other aspects, this program has some simple interactive with the user, print all the objects’ features on console windows, and there are different windows show the result of different processing.

For our second program, although it can process all the images, it also suffers the problem which recognizes the shadow as an object, and it can not remove the card edges completely. Besides, we use the default dilate and erode function to process the image, but sometime after those operations, some object still not closing, when we fill the contours, some object shows the contour, not the whole object. In other aspects, this program is better than the first one, we design some function to give user chance to input the location of the image if they input the wrong location of image, and also set a default setting for different threshold value, and if you want to set your threshold value, you input your threshold value. Besides, we design some slider to control the process of the image so that this program can provide dynamically output the result.

By comparing the result of two programs, from result images, we can see there are some difference, and the features also have different result, but the results are similar, so we estimate the result is correct.

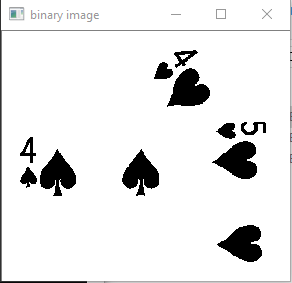
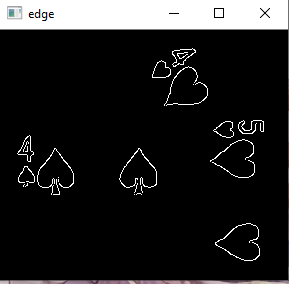
1. Summary

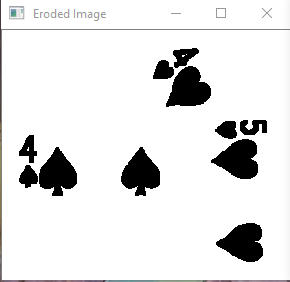
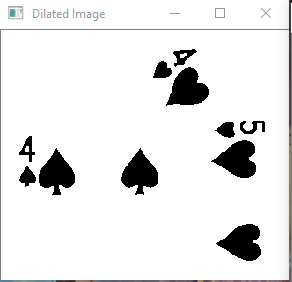
From this assignment, our group spent a lot of time to write those two programs, during this period, we learn how to write C++ code and how to use the function from OpenCV library. Each time we add a new function, we got some error. Fortunately, we can get help from website, the OpenCV documentation website help us a lot, ( <https://docs.opencv.org/2.4/modules/refman.html> ) it contain many sample code, that teaches us how to use those code, from that website, we know the data type in C++ and OpenCV library is very important, more than half of the errors are caused by data type.

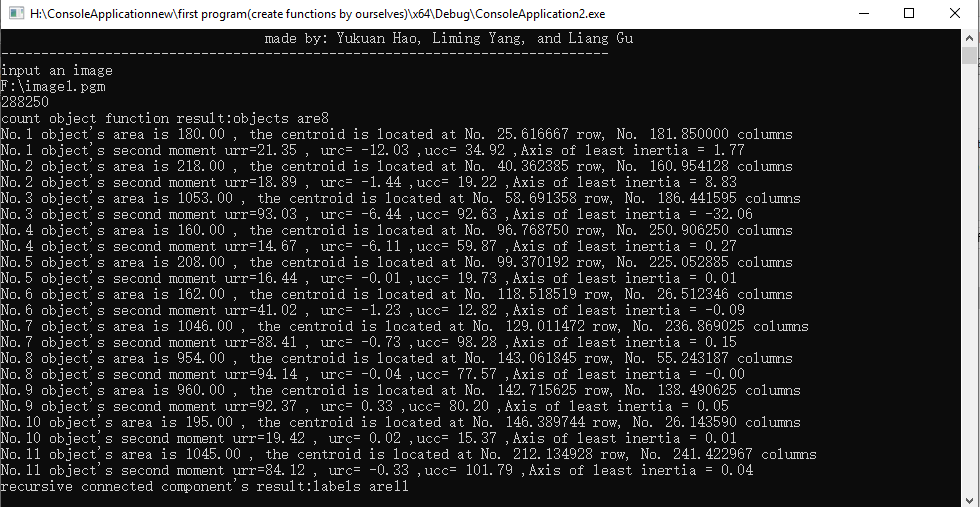
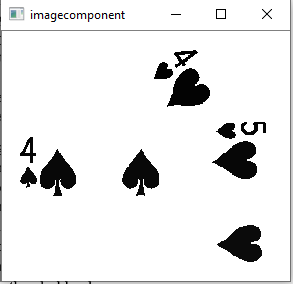
All in all, although we have not applied all of the learned content to this assignment, our programming skill has improved.

Program run screenshot

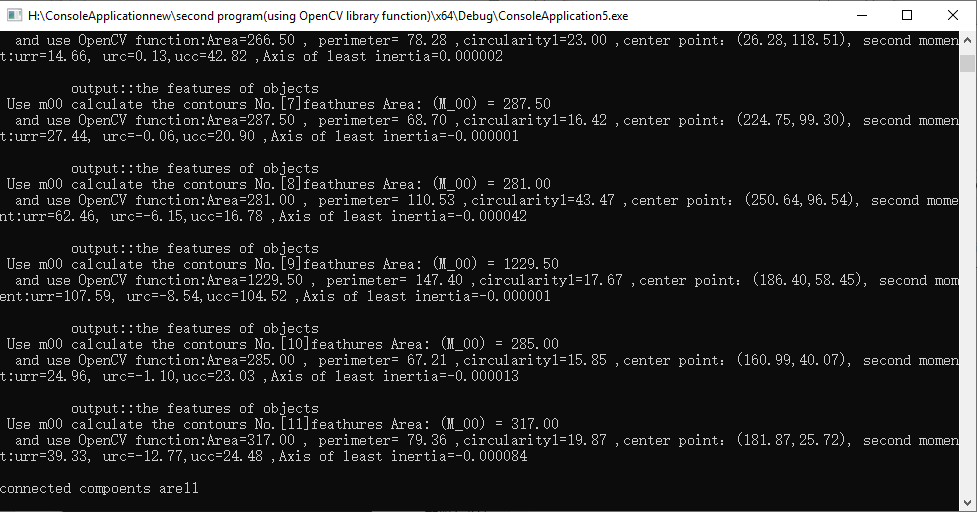
First program (create function by ourself)

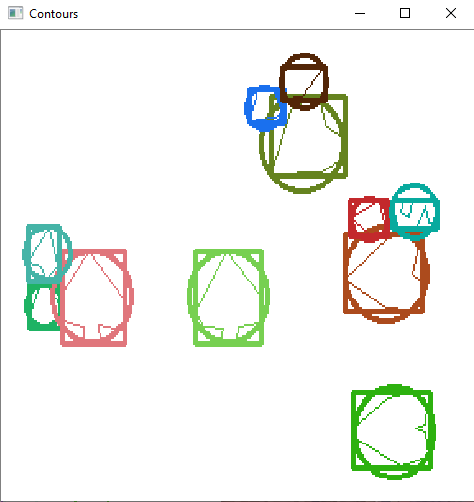
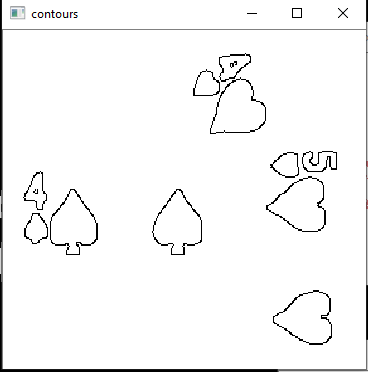


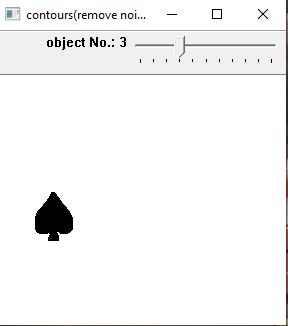
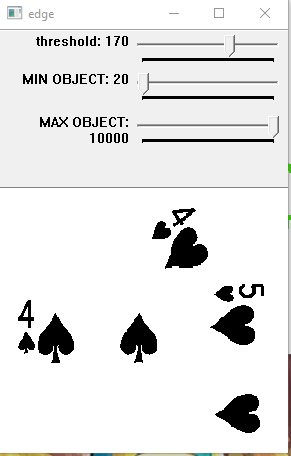
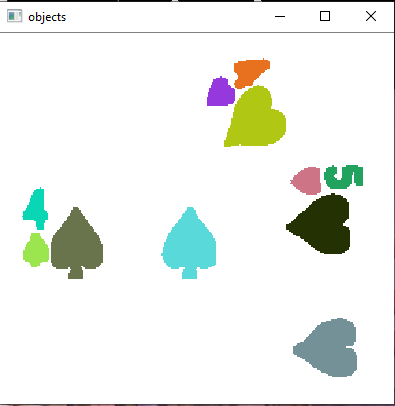
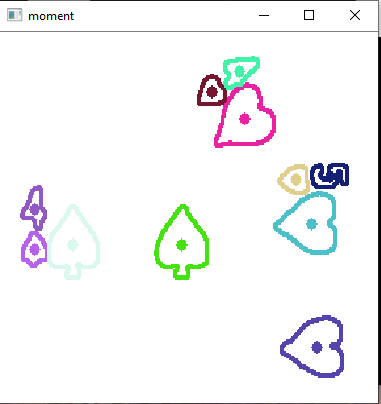




Second program (using OpenCV library)







Reference

Shapiro, L. G., & Stockman, G. C. (2001). Computer vision. Upper Saddle River, NJ: Prentice-Hall.

OpenCV API Reference. (n.d.). Retrieved from https://docs.opencv.org/2.4/modules/refman.html