CS205 C/C++ Program Design

PROJECT REPORT

Hand-Interactive System Based on yolov3

Student Name	Student ID
Wu Jingfu	12011437
Zhang Dechao	12011220
Zhuo Zhu	12011119

Semester: 2022 Spring
Lab Session: Friday 3-4 (Class 3)
Teacher: Zheng Feng, Liao Qimei
Student Assistants: He Zean, Liao Mingqian

Contents

Submission Date : 19/06/2022

1 Plan and Contributions

Table 1: Plan

Time	*Task*
Week12-Week13	1.OpenCV environment setting 2.References gathering
Week14	1.Raw structure design 2.Locate and configure the hand
Week15	1.Identify trajectories 2.Graphic fitting
Week18	1.Cursor controlling 2.Code combining 3.Presentation preparing

Table 2: Contributions

Member	Contributions	Ratio
Wu Jingfu	1. Trajectory recognition and graph fitting 2. GUI and Mode-switching design 3. Report typesetting	33%
Zhang Dechao	1. Implement controlling functions of cursor 2. Code combining 3. Report typesetting	33%
Zhuo Zhu	1. Locate and track the hand 2. Detect hand gesture 3. Code combining	33%

2 Introduction & Overview

2.1 Problem restatement

Given the serious negative impact caused by COVID-19 on daily life,developing a non-contact interactive system suitable for all countries to get rid of propagation is undoubtedly significant.

At the aim of avoiding directly touching screens and buttons on devices, this project is going to design a non-contact interactive system based on the OpenCV library, which uses an ordinary camera to detect the movement trajectory of hands in the field of view and controls the computer cursor as well as drawing graphs according to hand's gesture and trajectory.

2.2 Our work

2.2.1 Outline

The system consists of two modes, including a cursor controlling interface and a draw board, where entering and existing are detected by making a fist on specific areas according to guidance of GUI windows.

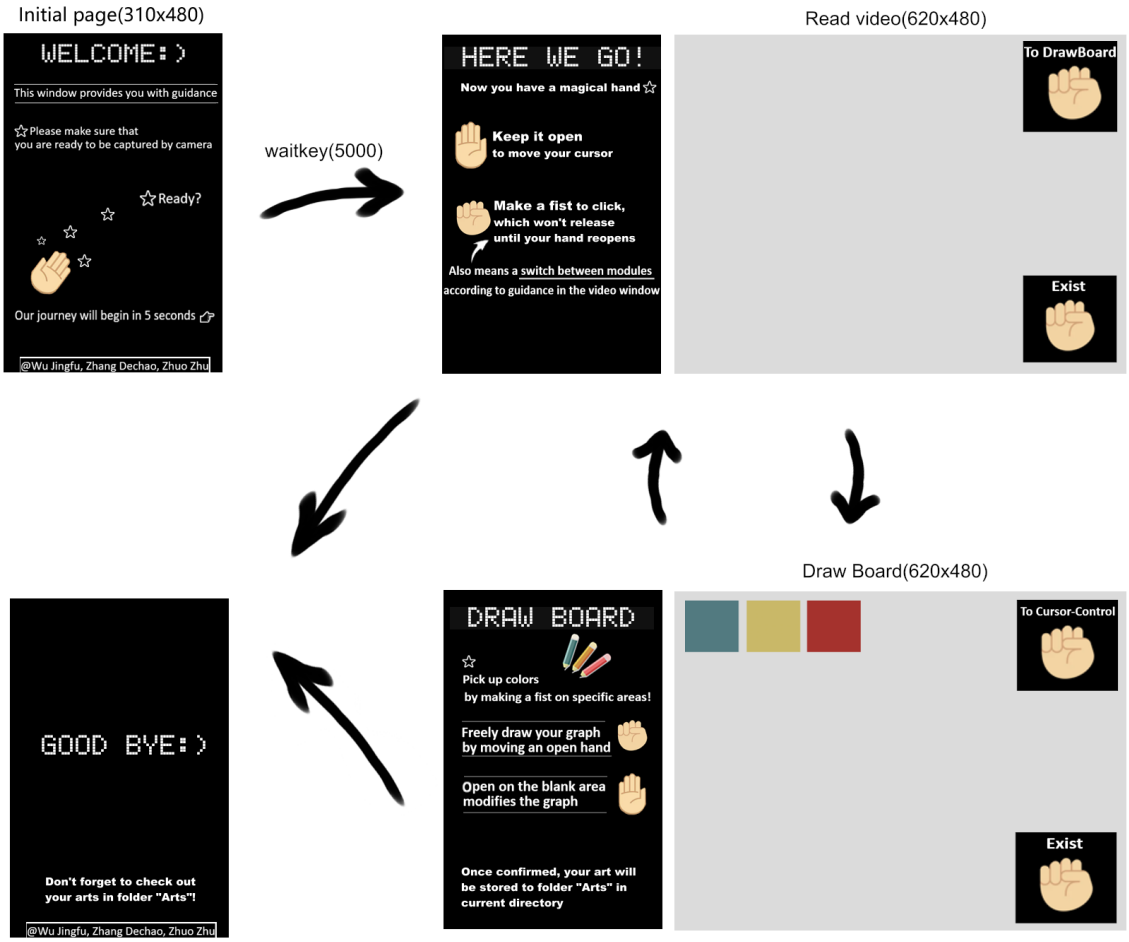
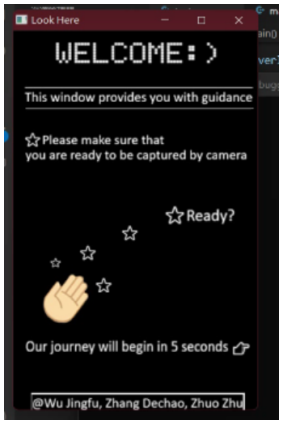
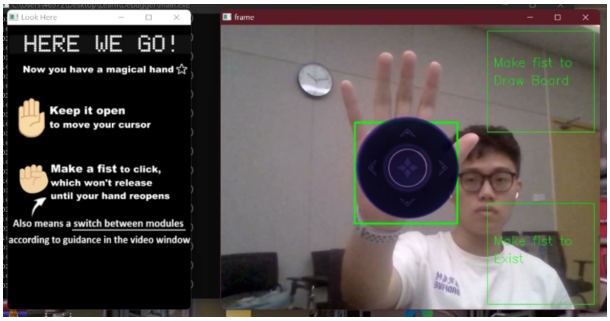


Figure 1: Mode-switching design

Initially, after five seconds of calling attention to video camera using,we enter the cursor controlling mode.



(a) Welcome page



(b) Detection output

Figure 2: Image Matching

The movement produced by user's open hand corresponds to the movement of cursor by remote sensing, and a fist means continuously pressing the left button on cursor.

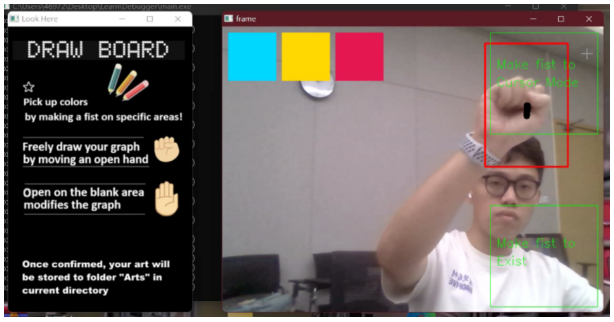


Figure 3: Mode-switching design

When making a fist in "Make fist to drawBoard" area, we will enter the drawBoard mode, where freely drawing as well as modified graphics are beautifully supported, which has a detailed specification in section 3.3.

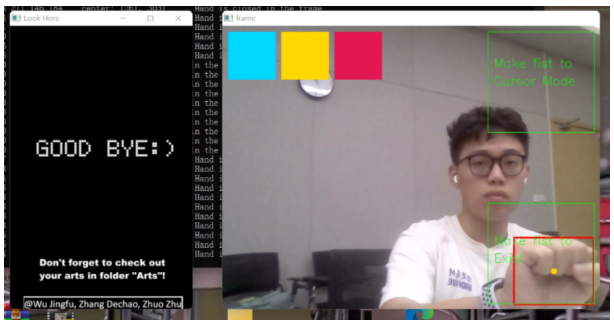


Figure 4: Mode-switching design

Making a fist in "Make fist to exist" area in any of the two modes immediately end the procedure and calls up the GoodBye GUI page.

All the above descriptions are included in the video demo HCI_group14_demo_video.mp4.

2.2.2 Highlights

- **Hand gesture detection(open and close)**.

  In addition to trajectory detection, this project obtains using hand gesture, which make it more elegant to implement functional requirements.

- **Humanized and Beautiful GUI Design**.

  Not only does this project encapsulate a complete interactive system, but it also has a detailed guidance window in the whole duration of the procedure, including a initial page calling attention to video camera using, mode usage guidance and a existing page.

- **Buffer to enhance mode performance**.

  Based on the unavoidable instability of hand detection mode, we clear the point set if and only if all locations of hand in buffer is invalid.

- **Remote sensing cursor**.

  Instead of directly locating cursor on hand, **remote sensing**(similar to video game controlling) is utilized to control the cursor movement.

- **Changeable color**

  The color of drawing is changeable upon the user's requirement.

- **Enhanced interaction**

  Whether modifying the graphic or not and when to save the graphic are decided by user's hand gesture.

3 Detailed Functional specifications

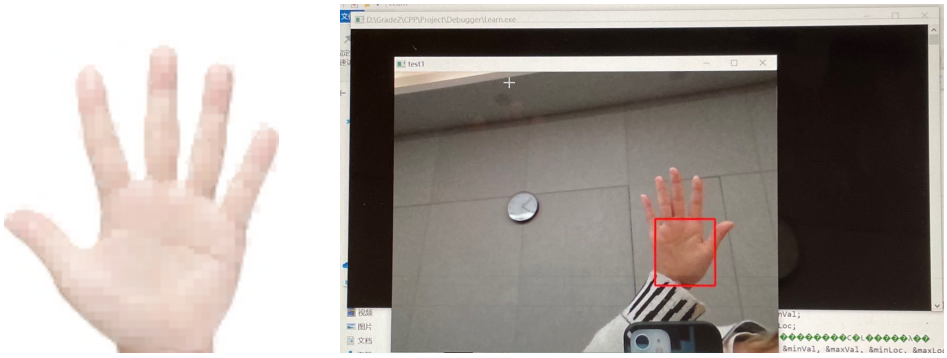
3.1 Locate & Track the hand

As the very basement part of this project, we've gone through multiple temps to finally achieve implementing this function in a satisfying way. The following are descriptions of our tempting procedure.

3.1.1 Image Matching

The first method we tried is to match the image from camera with a image that is the object we want. We use the function **cv::matchTemplate** provided by opencv, as is shown in *Learn.cpp*. This function is used to find the most matching (similar) part of an image to another template image [1].

It has four parameters:
(1) image: Input an image to be matched, support 8U or 32F.
(2) templ: Input a template image, the same type as image.
(3) result: output the matrix that holds the result, 32F type.
(4) method: The data comparison method to use.



(a) Template picture to be matched

(b) Detection output

Figure 5: Image Matching

But this method is too limited, it can identify only one object that we specify and the bounding box size is the template image size, thus making recognition effect not ideal.

3.1.2 Skin Color Recognition

The second method we explored is using color recognition by specifying the hand skin color and applying SVM classifier.

In the project uni_project_hand_detection, a first click on the skin of the user's hand with the left mouse button calls SVM classifier to learn its color at the aim of classifying it from the background color.

Then the finger count and state of the hand(open or close) will be shown in the main frame by detection of the finger tips. It can identify the hand and count the number of fingers [2].

But the background needs to have significantly different color from your hand, which means that the user's face can't appear in the camera view, making this method very unhumanized and limited.

Figure 6: Skin Color Recognition

3.1.3 Deep Neural Network

To detect hand gestures, we first have to detect the hand position in space. This pre-trained network is able to extract hands out of a '2D RGB' image, by using the YOLOv3 neural network.

The training set consists of the CMU Hand DB, the Egohands dataset and my own trained images (mainly from marathon runners), called cross-hands, which is free to access and download.

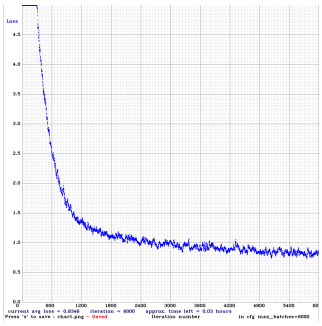


Figure 7: yolov3 tiny

We get the configuration and weights file of the well-trained neural network, then load the net using readNetFromDarknet. So we can do forward passing in the net.

The output of the darknet is a $6 \times n$ layers. The first four elements contains the x coordinate of the center, y coordinate of the center, the width and the height of the box that envelop the hand respectively. We multiply them by the size of the frame and then get the correct box that envelop the hand.

3.2 Cursor Controlling

To realize the funtion of controlling the cursor movement, we include the header file **windows.h**. We use the function **SetCursorPos** to move the cursor. It needs two parameters:

(1) X: the x coordinate of the screen coordinate system.

(2) Y: the y coordinate of the screen coordinate system.

((0, 0) is defined as the top right corner of the screen.)

We use the function **mouse_event** to control the left mouse button down and up. The function **mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, 0)** represents the left mouse button down and the function represents the left mouse button up. To open a folder, we need to double click the folder icon, which needs two times of left mouse button down and up.

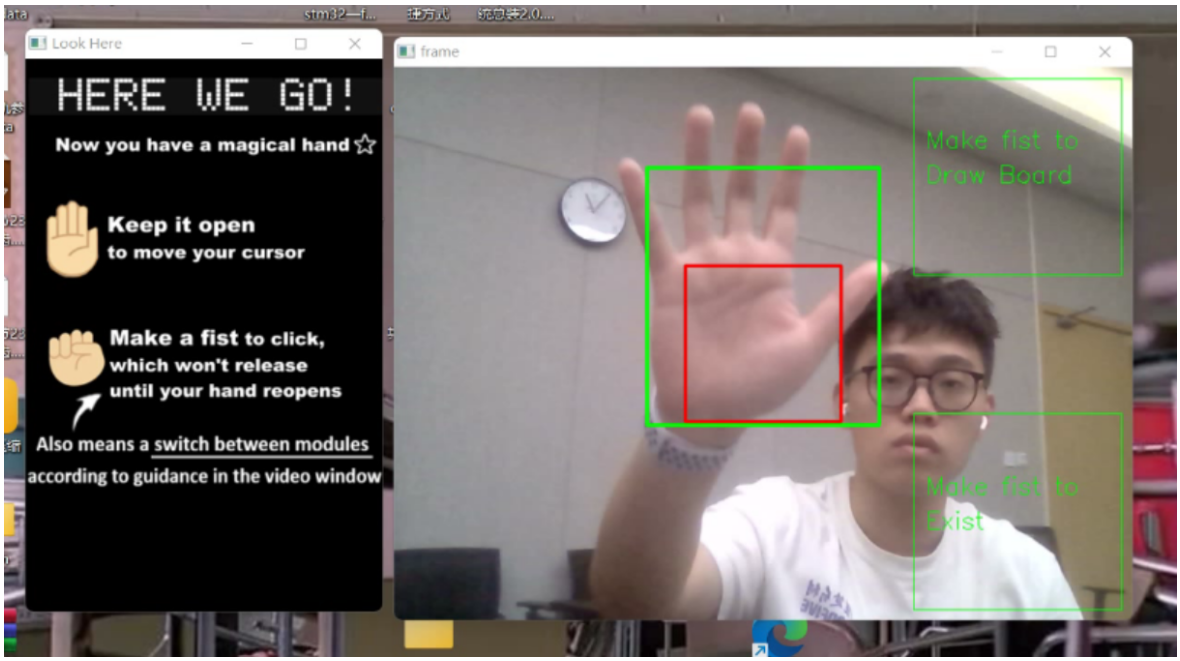


Figure 8: Cursor Controlling

To get stable mouse movement, we use a model like rockerif you put your hand in an area relative to the center, the cursor will move in that direction.

As for how to click the mouse, when you open your palm, the left mouse button will be disabled; when you make a fist, the left mouse button down will be enabled, as is shown in the guidance window.

3.3 Identify trajectory& Draw regulated geometry

Upon entering the drawBoard mode, trajectory of hand movement is recorded as **vector<Point>newPoints**, which will be cleared up when a gesture of fist is detected.

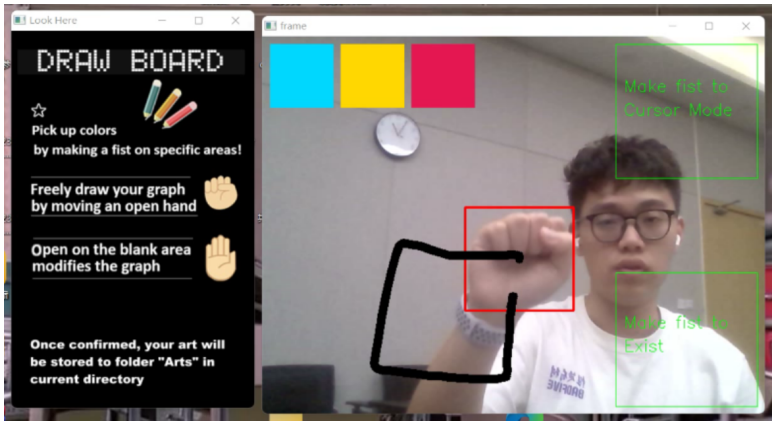
Function **cv::approxPolyDP** is used to regulate the trajectory, enabling this project to detect the shape's category by referring to the number of rows of **Mat result**. The parameters of this function are listed as bellow:

1) curve: Input vector of a 2D point stored in std::vector or Mat

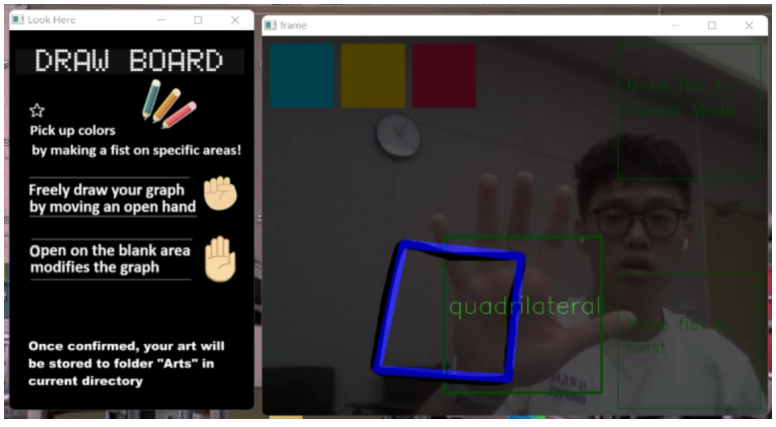
2) approxCurve: Result of the approximation. The type should match the type of the input curve.

3) epsilon: Parameter specifying the approximation accuracy. This is the maximum distance between the original curve and its approximation.

4) closed: If true, the approximated curve is closed (its first and last vertices are connected). Otherwise, it is not closed.



(a) Draw

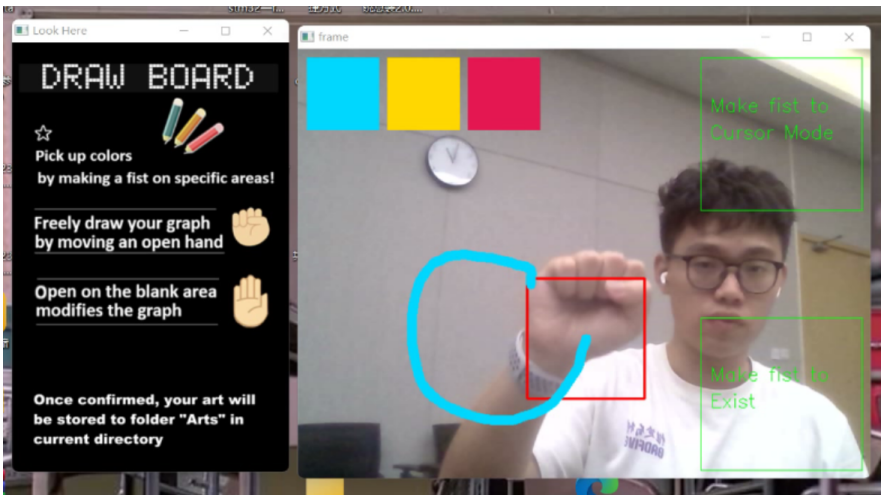


(b) Fit

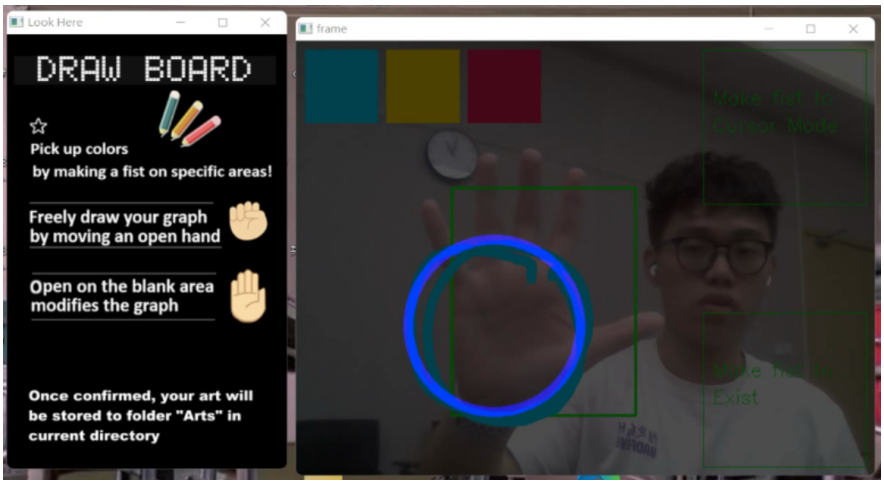
Figure 9: DrawBoard

The movement of user's fist will be real-time displayed as a colored trajectory, which will be modified and then stored immediately when the hand opens.

Specially, we derived an additional application that user can chose color of his/her trajectory by moving an open hand to one of the color space and make a fist to select.



(a) Draw



(b) Fit

Figure 10: DrawBoard

To exist this mode, the user can either chose to make a fist in the area "Make a fist to Cursor" or "Make a fist to exist", where the later one terminates the procedure and calls up the "GoodBye" page immediately. All of the above specifications have been briefly mentioned in the guidance window.

4 Problems and Solutions

- The frame number of video stream together with detection mode is **significantly effected by the computer performance**.

  Solution: To solve this unavoidable instability of hand detection mode, we came up with a buffer design to store a specific number of points, which clears the point set if and only if all locations of hand in buffer is invalid.

- The cursor function is limited by computer system, i.e. window.h is not supported in MAC system

  Solution: Set environment in windows system and transfer the codes.

- **namespace collision** namespcae cv and windows.h generates a collision.

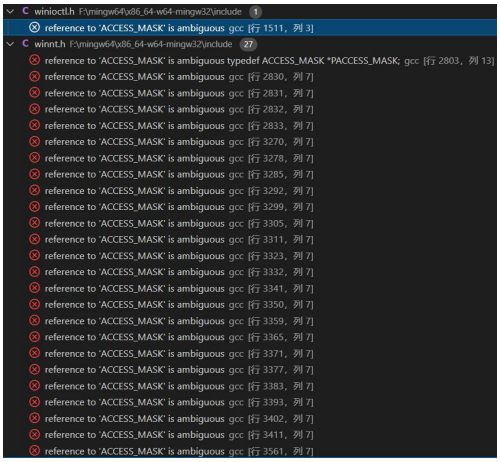


Figure 11: Error Message

  Solution: Delete the code "using namespace cv" and add the namespace to wherever it occurs.

- **gradual change failed:** We attempted to achieve gradually switch between raw graph and fitted graph via function **cv::addWeighted**, but it effects the continuous reading of video stream.

  Solution: We tried and gave up this functional implement.

- **Enclosed graphic**: Outliers(some points far away from the main part) occurred when fitting and draw the graphics.

  Solution: It turned out that the graphic needed to be enclosed, we obtained it by connecting newPoints[0] and newPoints[size-1].

- **Switching frequency** : Staying at the mode-switching area will make it frequently switch between modes.

  Solution: We use a counter to keep the switching frequency be at 10 frames/switch.

5 Conclusion

In this project, we have tried three methods to detect hands shown in the camera. But **Image Matching** and **Skin Color Recognition** have many limitations. At the end, we employ the **Deep Neural Network** method which can detect the hand ideally. About cursor control, we first try to implement the function in MacOS, but the library encapsulated is complex. Therefore, we use the well designed library in Windows to move the cursor. For the trajectory identification part, we use the library provided by opencv to fit the geometry.

After this project, we have learnt a lot about the opencv libraries and research how to write code and debug in a C++ project.

6 References

[1] OpenCV 2.4.13.7 documentation

https://docs.opencv.org/2.4/index.html.

[2] uni_project_hand_detection

https://github.com/ShtiliyanUzunov/uni_project_hand_detection.

[3] yolo-hand-detection

https://github.com/cansik/yolo-hand-detection/releases/tag/pretrained.

[4] Visual Paint

https://blog.csdn.net/mirrorboat/article/details/122945385

[5] Gradual graph switching

https://blog.csdn.net/youcans/article/details/121244153