CS3483 Assignment

Name: Chan Tsz Ngai

Student ID: 56213172

Eid: tnchan27

Date: 2/12/2022

CS3483
Multimodal Interface Design

1. A description of the design of the different sections of your program

In my design, the program can be divided into four main structures: the program setup, point drawing, function controllers and a specific Circle class for the circle drawing function.

Program Setup

Variable setup

```
1  let img, video;
2  let handpose, detections;
3  let pressed_f=false, pressed_v=false, pressed_c=false;
4  let history=[], circleHistory=[];
5  let c;
```

Different variables are set up. In the first line, img and video are set to store the image and the camera loaded in the setup function. The next line will be the variables handpose and detection for hand pose detection. Then, three true/false controllers are defined to check the status of the current running functions. In the fourth line, two arrays: history and circleHistory are set to store the trails and lastly will be the ‘c’ variable for getting a specific region of the image which will be used later.

Canvas setup

```
6
7  function setup() {
8    createCanvas(1170,390);
9    img=loadImage('images/town.jpg')
10   blurred=loadImage('images/town.jpg')
11
12   video=createCapture(VIDEO)
13   video.size(width/2, height)
14   video.hide()
15
16   handpose=ml5.handpose(video, modelReady);
17   handpose.on("hand", gotResults);
18 }
19
20 function modelReady(){console.log('model ready');}
21
22 function gotResults(results){detections=results;}
23
24 function draw() {
25   //img.width=1170, img.height=780
26   image(img,0,0,img.width/2, img.height/2)
27   image(video,width/2,0,video.width,video.height)
28
29   if(detections){
30     if(detections.length > 0){
31       drawKeypoints();
32     }
33   }
34 }
35
```

For the first function setup(), the canvas with a width of 1170 and a height of 390 will be created. Then, the chosen image will be loaded twice, one is for the original photo display and another one will be displayed in the view mode with a blur effect. Besides, the createCapture function uses and captures the video camera and it is sized half of the width of the canvas. The hand detection algorithms are based on the ml5.handpose function. To apply it, the script (<script src="https://unpkg.com/ml5@latest/dist/ml5.min.js"></script>) needs to be implemented in the index.html. Callback functions such as modelReady() and gotResults() (line20-22) to let the user know if the algorithm is ready to be used.

The draw() function will be loading the chosen image and combine it with the canvas of the video camera in a 1:1 ratio. If there is a detection of your hand, the drawKeypoints() function will be triggered.

Point drawing

Fingertips detection

```
53 function drawKeypoints(){
54   noStroke();
55   fill(255,0,0);
56
57   //fingertip of index finger
58   for(let i=0; i<detections.length; i++){
59     const detection = detections[i];
60     for (let j = 0;j<detection.landmarks.length; j++){
61         const keypoint = detection.landmarks[j];
62         if(j == 4 && pressed_c){
63           thumb_x=keypoint[0]
64           thumb_y=keypoint[1]
65           cam_thumb=thumb_x+video.width
66           ellipse(cam_thumb, thumb_y,30,30)
67         }
68
69         if (j == 8) {
70           pointerX = keypoint[0];
71           pointerY = keypoint[1];
72
73           //shown in image
74           if(pressed_c==false&&pressed_v==false){
75             ellipse(pointerX, pointerY, 30, 30);
76           }
77           //shown in camera
78           ellipse(pointerX+video.width, pointerY, 30, 30)
79
```

The drawKeypoints() function will be a large function that handles all the required functionality specified in the assignment. As there will be a pointer that detects the fingertip of the index finger, the pointer will be a circle that is set to red in color and

without a border. The nested for-loop from line 58 is to point out the specific fingertip among different pointers. For 62-67 will be pointing out both the thumb finger and index finger on the camera view after the key ‘c’ is pressed, which will be used in the circle drawing function and for the rest of the lines will be pointing out the fingertip of the index finger. It is set to display on camera view all the time and image view when it is not in circle drawing or view image mode, which will be explained later.

Freehand drawing

```
76          //Freehand drawing
77          if(pressed_f){
78            history.push(createVector(pointerX, pointerY))
79            stroke(255,0,0)
80            strokeWeight(8)
81            for(let i=0;i<history.length-1;i++){
82              let pos = history[i];
83              let nextpos=history[i+1]
84              line(pos.x, pos.y, nextpos.x, nextpos.y)
85            }
```

In freehand drawing mode, the pointer trails will be recorded and displayed. When the ‘f’ key is pressed, the x-coordinates and y-coordinates are recorded whenever the pointer moves and stored in the ‘history’ array. Lines will be created based on four coordinates, x and y-coordinates of the previous location and the x and y-coordinates of the next location. Therefore, trails of pointers can be displayed in the image view.

View image

```
87          //View image
88          } else if (pressed_v){
89            //get from not blurred image then replace blur image
90            c=get(pointerX,pointerY,100,100)
91            image(blurred,0,0,blurred.width/2, blurred.height/2)
92            image(c, pointerX, pointerY)
93            ellipse(pointerX+50, pointerY+50, 30,30)
94
```

In the view image mode, the blurred image will be the background of the image view and a rectangular area will be used to display the original and clear image of that area region. To achieve this, after clicking the ‘v’ key, the get() function will be used to get the clear region with width 100 and height 100. Then, the blurred image will be replacing the image view background. Next, the clear regions will be shown on top of the blurred background. The ellipse pointer will also be displayed on top and at the centre of the clear region.

CS3483
Multimodal Interface Design

Circle drawing

```
95          //Circle drawing
96          } else if (pressed_c){
97            let pxl = get(pointerX, pointerY)
98            //console.log(pxl)
99            let cdist = (sqrt(sq(thumb_x-pointerX)+sq(thumb_y-pointerY)))/2
100           circleHistory.push(new Circle(pointerX, pointerY, cdist,pxl))
101
102           for(let i=0;i<circleHistory.length;i++){
103             circleHistory[i].display();
104             //put it here to prevent storing in circelHistory
105             fill(255,0,0)
106             ellipse(pointerX, pointerY, 30)
107           }
108
109         }
110
```

In circle drawing mode, trails of circles will be displayed with specified pixel color, but varies in width and opacity. To achieve this, after clicking the ‘c’ key, the get() function will get the pixel color of the coordinates, ‘cdist’ will be calculating the distance between the pointers of the thumb finger and index finger. After gathering these information of the circles, we can use the Circle class to create different circles and stored in circleHistory array. Then, the circles can be displayed using the for-loop and the finger pointer is displayed back on top of these circles.

Circle class

```
170 class Circle{
171   constructor(x,y,rd,pxl){
172     this.x=x;
173     this.y=y;
174     this.rd=rd;
175     this.pxl=pxl;
176     this.op=random(0,255);
177   }
178
179   display(){
180     fill(this.pxl, this.op)
181     ellipse(this.x,this.y,this.rd)
182
183   }
184 }
```

The constructor of the Circle class is used for recording the characteristics (x-coordinates, y-coordinates, radius, pixel color and opacity) of different generated circle objects. Then, the display function will be used for circle display in the image view, which will be called repeatedly according to the length of the circleHistory array in the above function.

Function controllers

Pressed keys controllers

```
36  function keyPressed()
37  {
38    if(key=='v'){
39      applyBlur();
40    } else if(key=='e'){
41      exit();
42    } else if(key=='f'){
43      freehand();
44    } else if(key=='c'){
45      circleDrawing();
46    }
47  }
```

The keyPressed() function will be a controller function that directs to different functions based on the pressed keys. The function descriptions will be introduced in the following table:

Key	Function

'v'	applyBlur()
	`116 //blur the image`
	`117 function applyBlur(){`
	`118 //prevent draw trail when entering view image`
	`119 //(original drawing trail-> no drawing trail and turn blur)`
	`120 pressed_f=false`
	`121 pressed_c=false`
	`122 pressed_v=true`
	`123 empty_hist();`
	`124 blurred.filter(BLUR,10)`
	`125`
	`126 }`
	
	The function will blur the image and turn the boolean values of other key pressed to false and pressed_v to true. The trails of lines and circles are also cleared

'f'	freehand()
	`128 //freehand drawing`
	`129 function freehand(){`
	`130 pressed_v=false`
	`131 pressed_c=false`
	`132 pressed_f=true`
	`133 empty_hist();`
	`134 //prevent draw trail when it is blur (blur->original)`
	`135 loadOriginalImage()`
	`136 }`

	The function will turn the boolean values of other key pressed to false and pressed_f to true. The original clear image is loaded as the image view background. The trails of lines and circles are also cleared

‘c’	circleDrawing()
	`138 //circle drawing`
	`139 function circleDrawing(){`
	`140 pressed_f=false`
	`141 pressed_v=false`
	`142 pressed_c=true`
	`143 empty_hist();`
	`144 loadOriginalImage()`
	`145 }`
	
	Similar to the freehand(), but the boolean value of pressed_c is true and others are turned to false.

‘e’	exit()
	`152 function exit(){`
	`153 pressed_f=false`
	`154 pressed_v=false`
	`155 pressed_c=false`
	`156 empty_hist();`
	`157 loadOriginalImage();`
	`158 }`
	
	The ‘e’ key turns all the boolean values to false, which mean the program is back to the original mode (only a pointer is shown). The original clear image is loaded as the image view background. The trails of lines and circles are also cleared

The empty_hist() and loadOriginalImage() are minor functions that empty the trail and circles history and load the original clear image respectively.

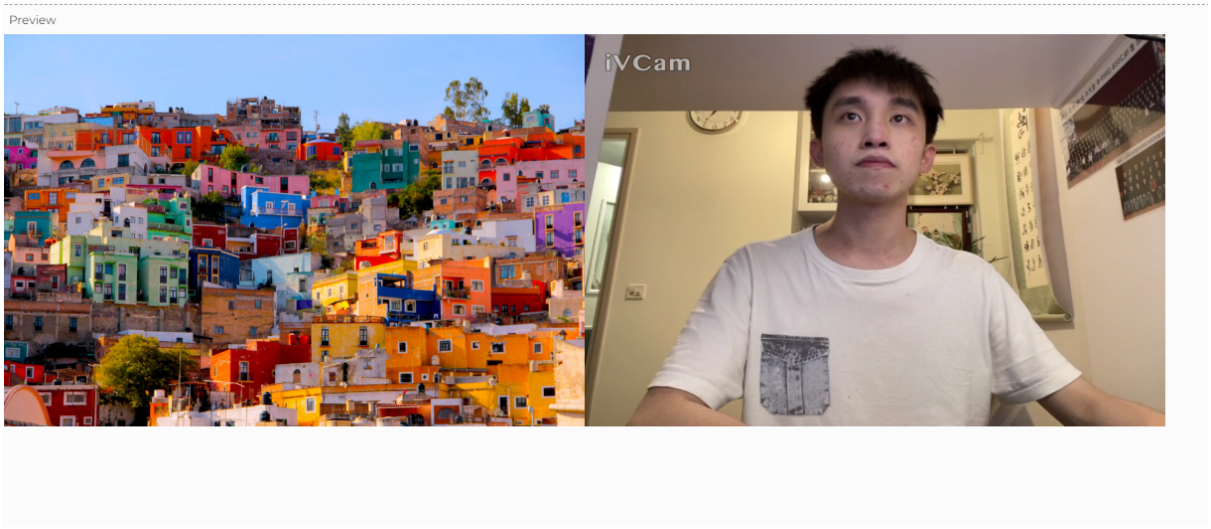
```
160 function empty_hist(){
161   history=[]
162   circleHistory=[]
163 }
164
165
166 function loadOriginalImage(){
167   img_copy=loadImage('images/town.jpg')
168   img=img_copy
169 }
170
```

CS3483
Multimodal Interface Design

2. Screen captures of the output

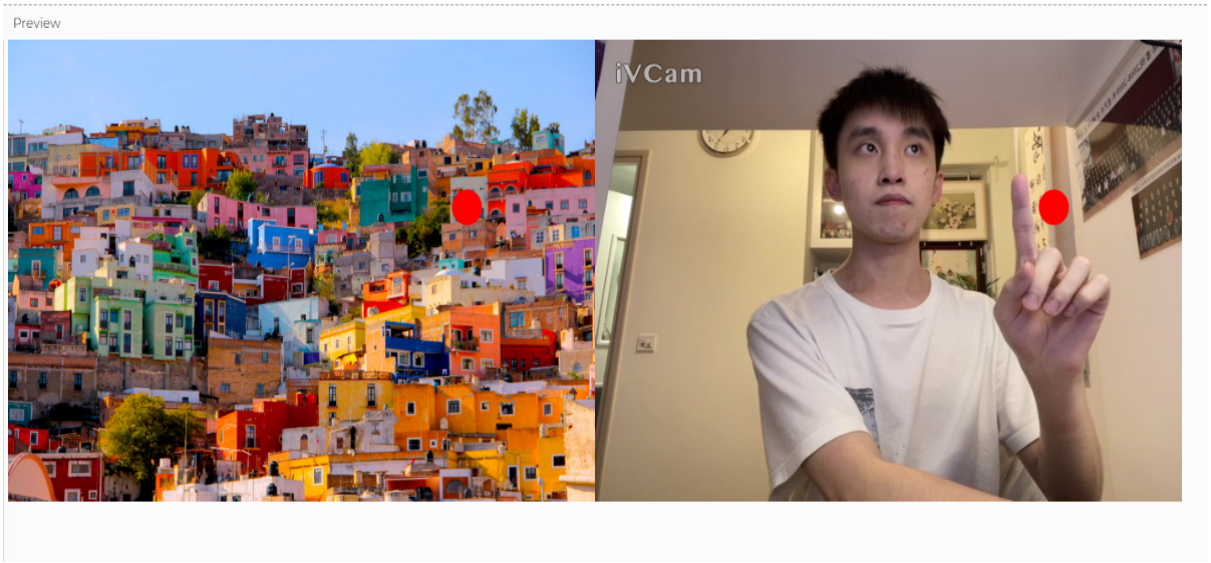
In this project, I have reflected the instant camera view and it is based on my actual pose. And the following pictures are all using my **right hand index fingertip**.

Initial setup

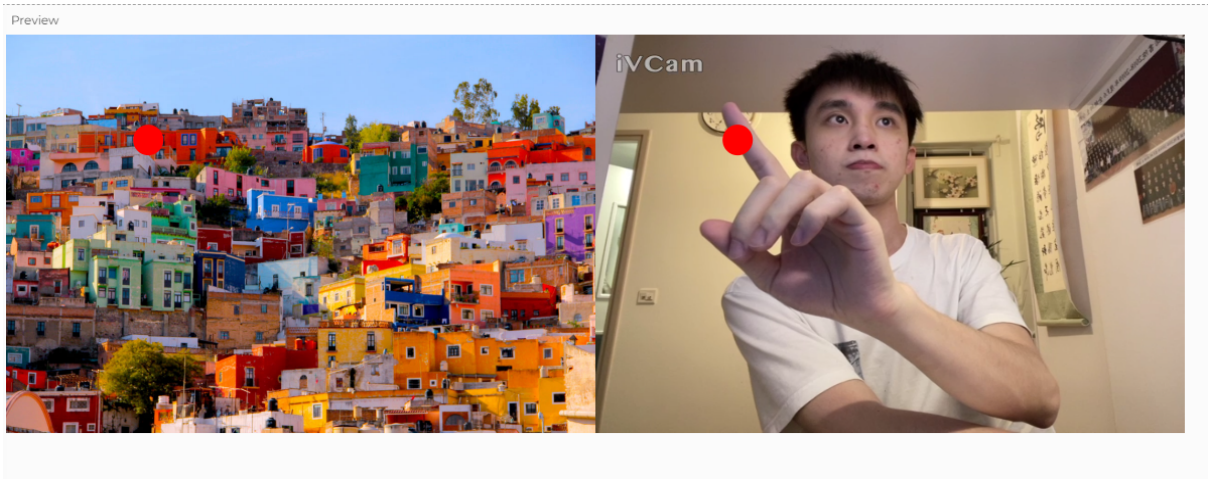


A picture of a colorful town is taken as the image view background and a canvas of width 1170 and height 390 is set. In the canvas, it is split into image view and instant camera view in a 1:1 ratio. Both views have the same size (width and height) and are displayed next to each other.

Index fingertip detection

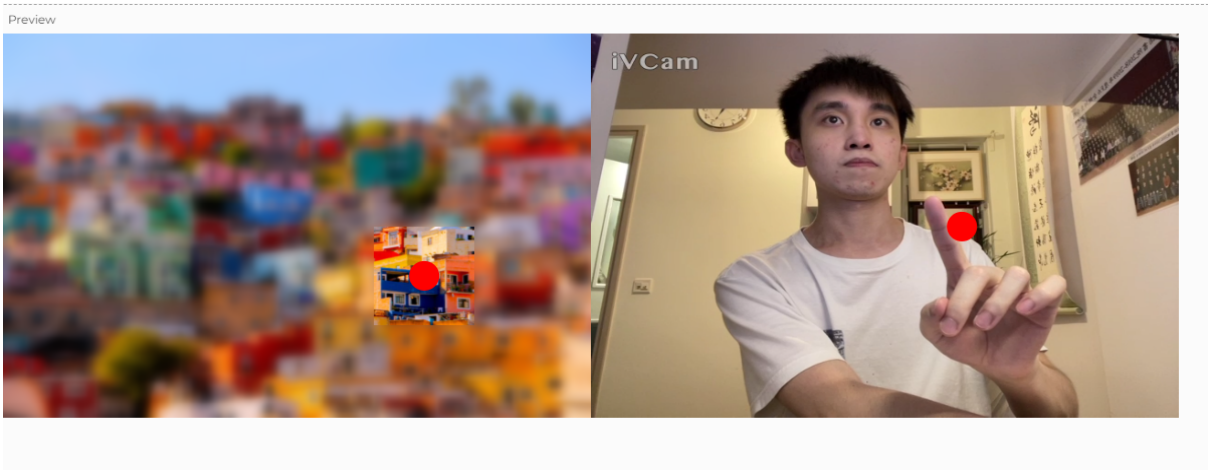


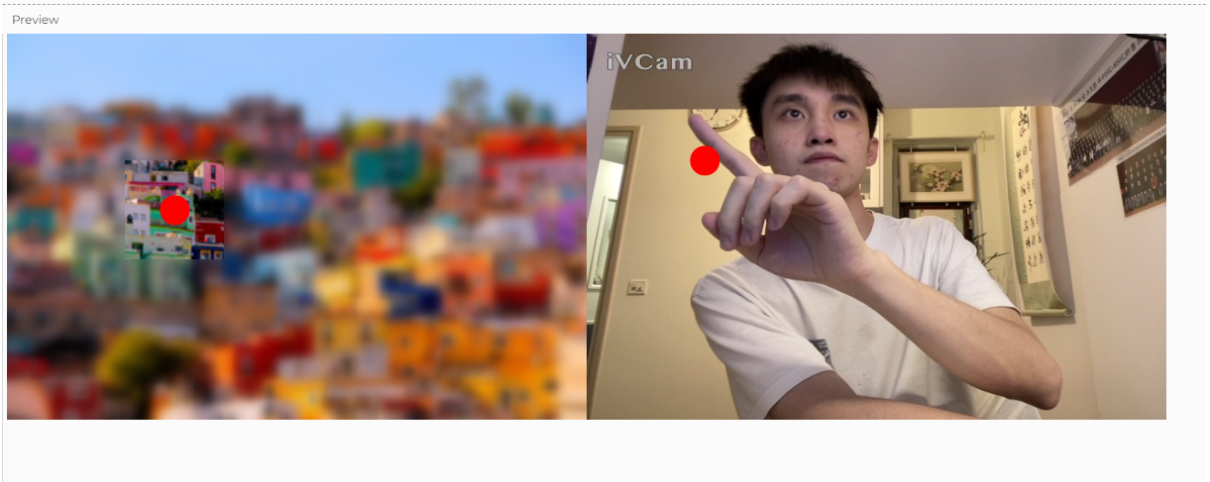
CS3483
Multimodal Interface Design



Two red circles are displayed on the screen, one of them indicates the position of the captured index fingertip and another one is the corresponding indicator shown on the image view that follows the movement of the fingertip. Both pictures can successfully detect the fingertip and display both circles.

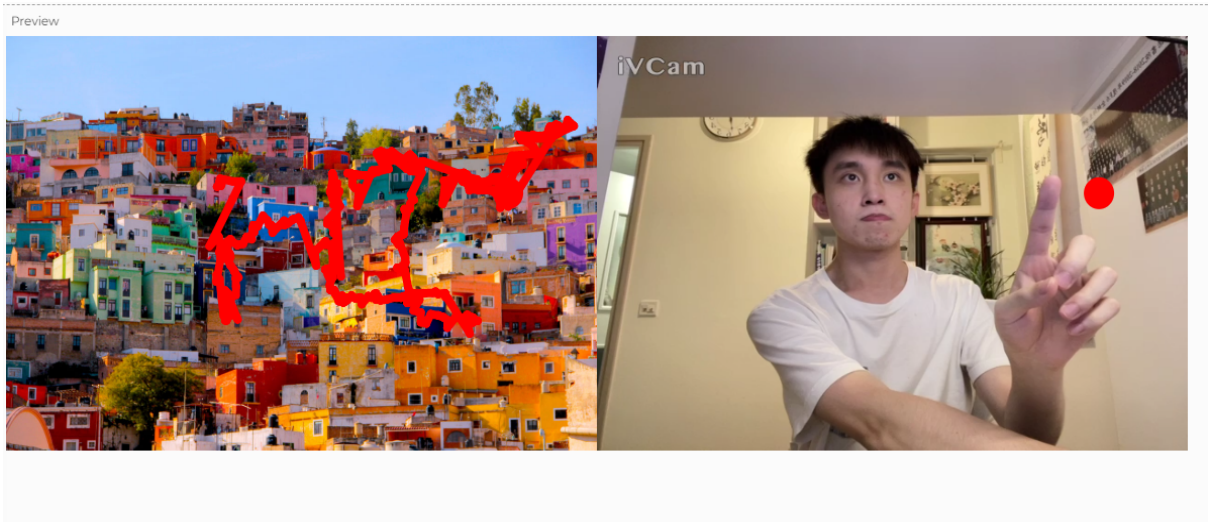
Viewing the image

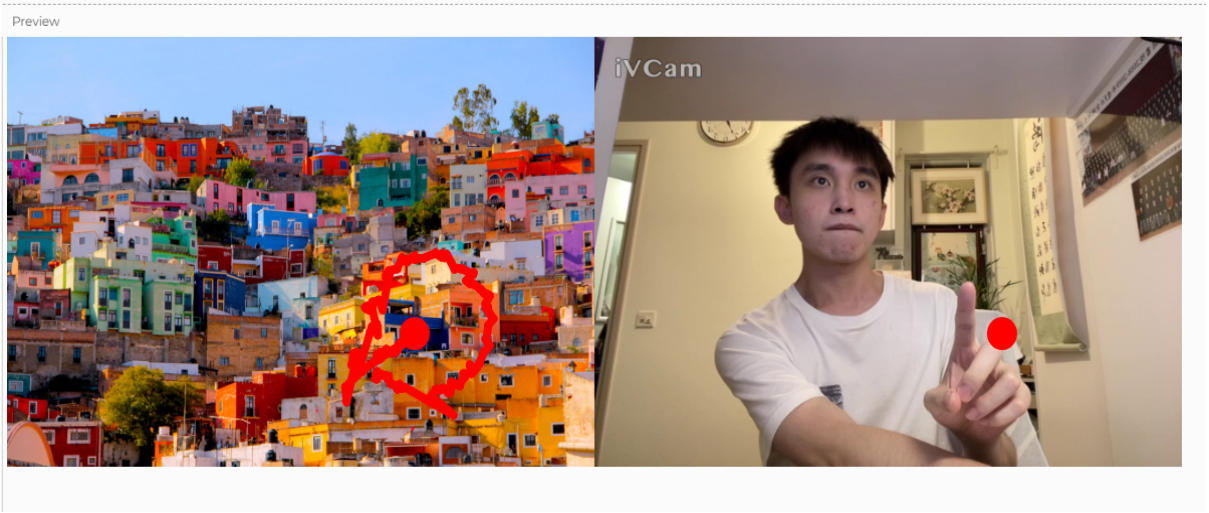




When the ‘v’ key is pressed and released, the blurred version of the image is replaced as the image view background. If the position indicator moves to a region of the image, that particular region will become clear and the remaining regions are still blurred. For example, when the indicators move to the blue house in the first picture, a clear blue house is shown, and when it moves away, the blue house becomes blurred and another region now becomes clear.

Freehand drawing on the image

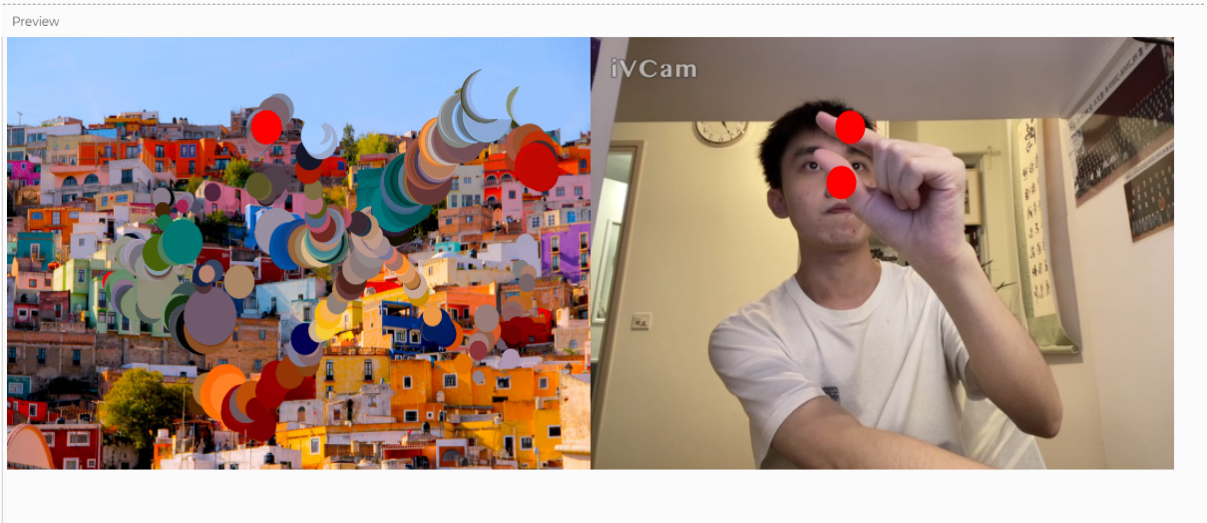




When the ‘f’ key is pressed and released, it enters the freehand drawing mode and a trace is drawn on the image view. In the first picture, the word ‘HI’ is drawn and for the second picture, a circle that circles the blue house is drawn. The trace is red in color with a weight of 8 pixels. The indicator does not vanish when entering the mode.

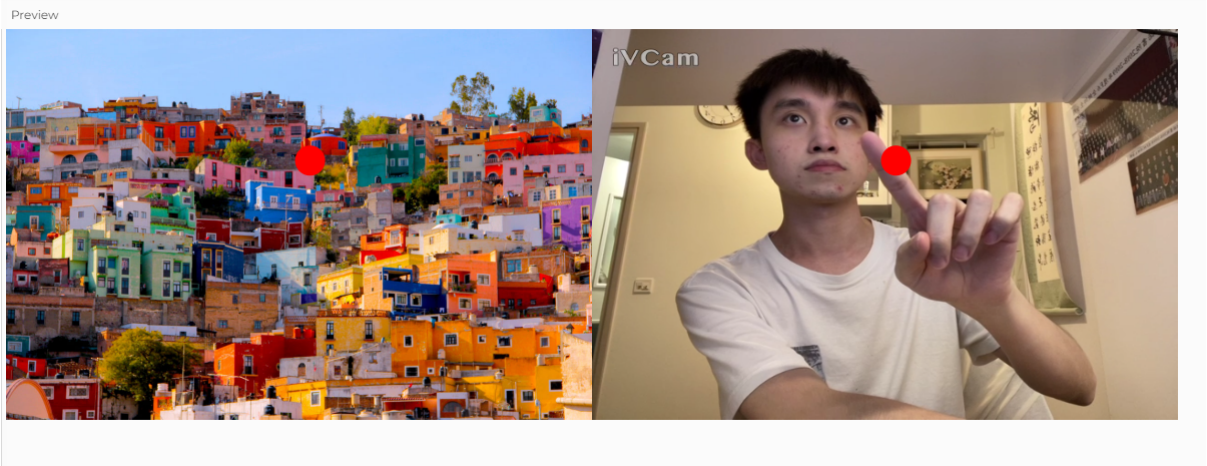
CS3483
Multimodal Interface Design

Drawing circles on the image



When the ‘c’ key is pressed and released, it enters the circle drawing mode. The original image is displayed. In the camera view, two circle indicators is shown that detects the thumb fingertip and index fingertip, while in the image view, lines of circles are drawn with different colors, size and opacity. The radius of the circles is adjusted based on the distance between two fingertips in proportion. The indicator also does not vanish when entering the mode.

Exiting the view image/freehand drawing/circle drawing mode



When the ‘e’ key is pressed and released, it exits the view image/freehand drawing/circle drawing mode and the original image is displayed back as the image view background. Besides, traces of circles or lines are also vanished.

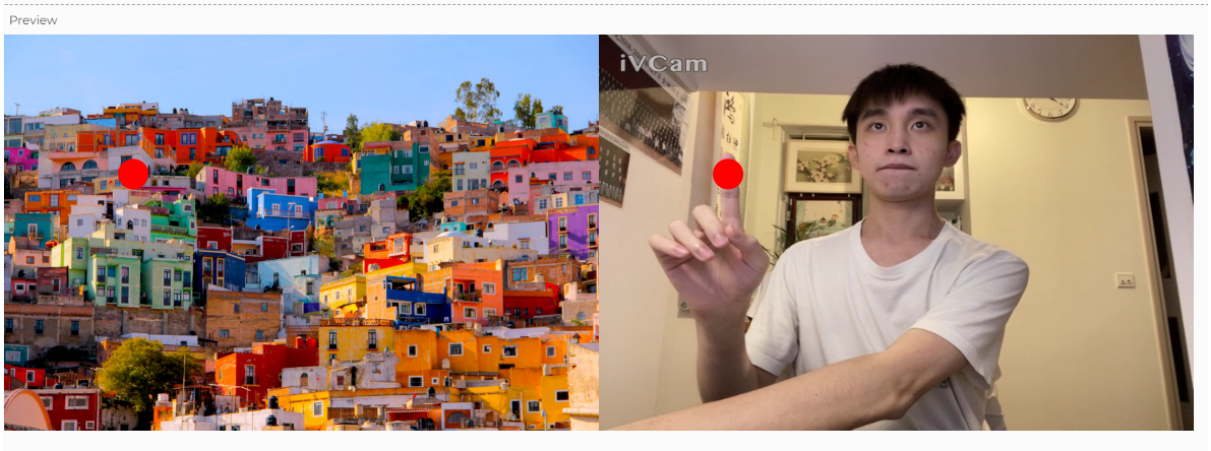
CS3483
Multimodal Interface Design

3. Limitations and possible improvements of the program

Limitation

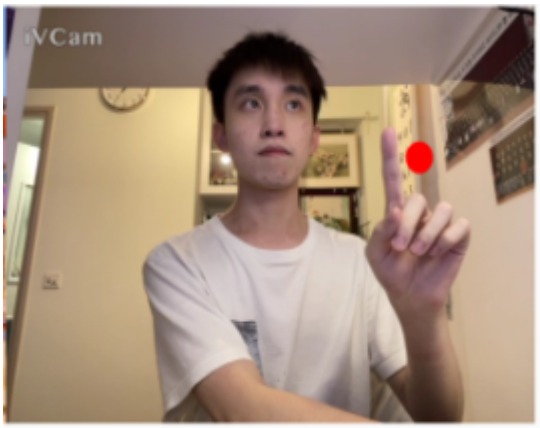
Assumption

First, the design has one assumption that the instant camera view should be showing the actual image, but not the reflected one, which is like the example below. In the below example, my right-hand index fingertip is up but it is shown on the left-hand side of the camera view. Besides, all the movement will be in the opposite direction due to the camera view reflection (my fingertip moves to the right, but the point indicator and trails are drawn to the left). Therefore, this is one of the limitations of the design.

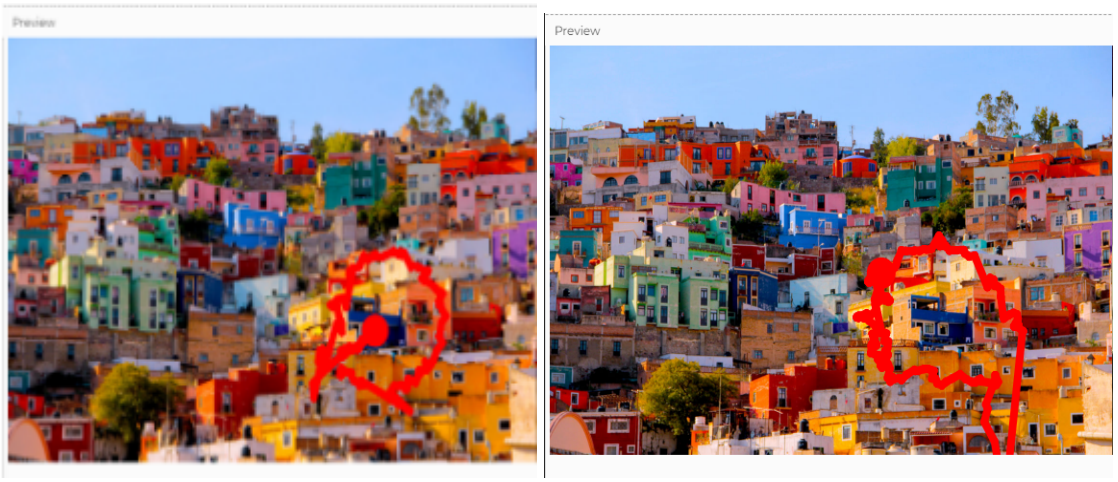


Hand detection algorithm

There are different limitations of the hand detection algorithm. First, the detection of the fingertip is not very accurate. In some rare scenarios, the pointer indicates other fingertips as the index fingertip. Even if it can correctly find and indicate the index fingertip, there may be a small gap between the actual fingertip and the indicator.

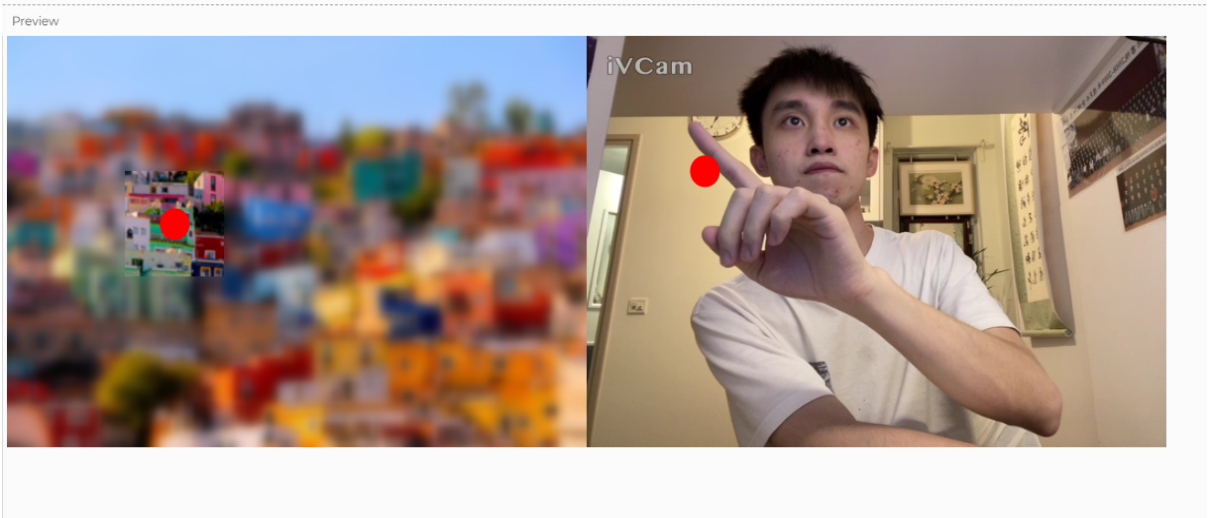


Second, the indicator is not that stable and sometimes the detection disappears. The indicator is quite shaky when I try to move in a direction. In the below, when a circle is drawn, there will be a sudden drop of trails as the detection of the hand hardly follow my movement. The slow and moving-down actions will also sometimes cause the indicators to disappear.



Problems of the design

One of the limitations of the design is in the view image mode, the correct corresponding indicator should be the upper left corner of the rectangular region, but to achieve the indicator pointing at the centre of the region. I have hidden the correct indicator, and a new indicator is drawn at the centre, but this may confuse the user as the indicators are not in the same corresponding position.



Possible improvements

- Regarding the limitations of the assumption and hand detection algorithm, I think there is not much work that can deal with these limitations because the algorithm is an external pre-trained tool. The research and adoption of other hand detection algorithms can be an improvement to the fingertip detection accuracy.

- For the issues of the design in the view image mode, a solution can be looking for methods that centre the image according to the pointing indicators. For example, the adoption of imageMode(CENTRE). However, I have tried this approach but this is not successful and causing more bugs. Therefore, I hope I can look for other methods to solve this problem.

- The canvas is set to perfectly fit the instant camera view and the image view. When after finishing the project, I want to add a text on top of the canvas that show the current running mode. However, all the parameters needs to be changed and causes different bugs such as positioning. In the future, a basic plan for all the features needs to be done.

- Another improvement can be trying different photos in the image view mode. At first, I think a colorful photo is better for this project for a better visual effect. However, when I try to draw circles in the circle drawing mode. I think the result is not that satisfactory as the photos become quite messy. A cleaner and less colored photo can be used for the project.

CS3483
Multimodal Interface Design