

WebCam 활용 화상회의 음성채팅 서비스

1726052 최신희
1826089 황해성
2026009 김민석

TEAM OPENPY

1

INDEX

- 개발 동기
- 개발 목표
- 시스템 구조도
- 소스 코드
- 프로젝트 일정
- 프로젝트 결과
- 역할
- 출처

2

개발 동기



왜? 화상회의 서비스인가

현재는 코로나 바이러스 백신이 보편적으로 대부분의 사람들에게 제공되고 있으나, 여전히 코로나19의 위협으로부터 완전히 벗어났다고는 말할 수 없습니다. 아직도 이 바이러스로 인하여 죽는 사람이 발생하고 있고, 우리는 실내에서 모두 마스크를 착용하여 생활하고 있습니다.

이러한 상황에서 줌(Zoom)이라는 화상회의의 프로그램에 대하여 상당한 수요가 발생하였는데, 그 이유는 각자 자택에서 화상회의를 하게 되면 접점이 없기 때문에 코로나19로부터 안전하기 때문입니다.

그리고 워라밸을 중요시 여기는 저희 세대의 공통적인 관심사를 존중하는 기업 문화의 변화 또한 앞으로 화상회의의 서비스의 수요가 더 증가될 것이라는 것을 유추할 수 있게 합니다. 워라밸을 비롯한 개인의 생활을 존중하고자 하는 이러한 사회의 분위기는 앞으로 재택근무의 사례가 더 증가함에 따라 화상회의의 서비스에 많은 관심이 돌리게 될 것입니다.

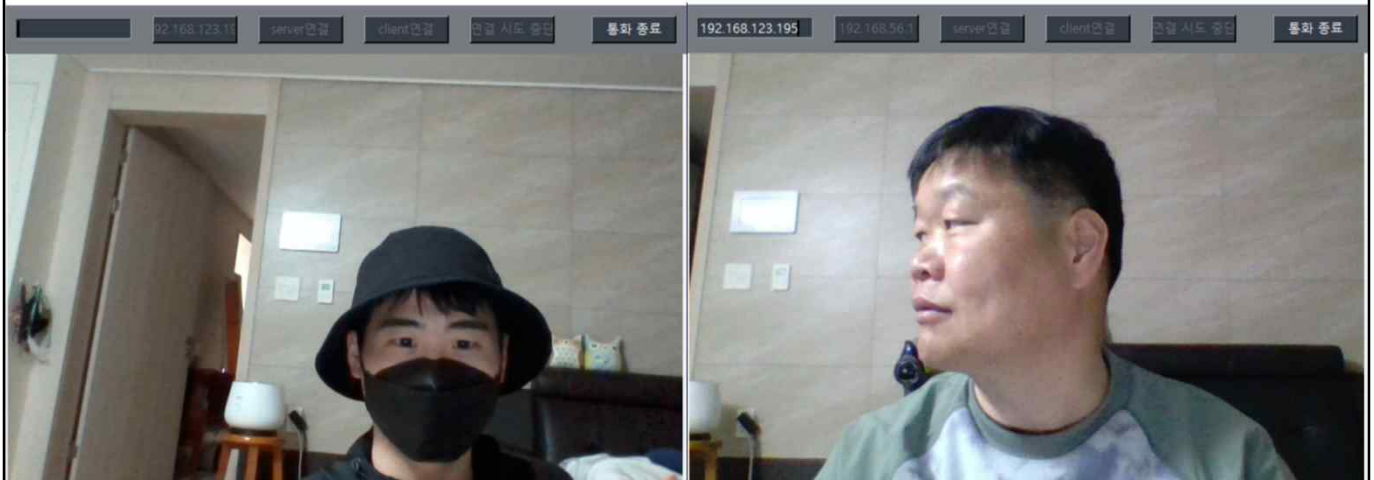


3

개발 목표

OpenCV 라이브러리를 사용하여 개인용 컴퓨터에 탑재되어 있는 WebCam을 통하여 서비스 이용자의 얼굴을 화면에 나타냅니다. 서비스 이용자의 전화를 수신하는 수신자 또한 화면상에 나타납니다.

수신자와 발신자 모두 WebCam을 지원하여야 합니다. 또한 PyAudio 라이브러리를 사용하여 음성채팅 서비스를 구현하고자 합니다. 따라서 수신자와 발신자는 전화에 사용하고 있는 컴퓨터에 마이크가 내장되어 있는 지 미리 확인해야 합니다. 만약 내장되어 있지 않다면 외부 입력장치로써 연결하여 사용하여야도 괜찮습니다. 음성으로 채팅을 주고 받기 때문에 필수적입니다. 스피커 또한 필요합니다. 마이크가 음성을 입력하는 장치라면, 스피커는 음성을 출력하기 위한 장치입니다.



4

시스템 구조도



인터넷을 사용하여 두 명의 사용자가 1대1로 연결되는 방식이며, 주소를 사용하여 연결한다.

각 사용자는 PC에 카메라가 내장되어 있는 지, 음성 입출력을 위한 입출력 장치가 연결되어 있는 지를 확인해야 한다.

5

사용된 주요 모듈

tkinter

인터페이스 구현
화상통화 프로그램의 GUI를 설계할 때 사용되는 모듈이다.

cv2

영상 처리
OpenCV라는 이름으로 더 많이 불리는 파이썬의 영상처리 모듈이다. 노트북의 웹캠을 사용할 수 있게 된다.

pyaudio

음성 데이터 처리
화상통화 간에 음성 데이터가 정상적으로 오갈 수 있도록 사용한 모듈이다.

socket

네트워크 연결
두 사용자가 1대1로 연결할 수 있도록 네트워크를 연결할 때 사용하는 모듈입니다.

사용 언어는 파이썬입니다.

6

App 클래스

```
def __init__(self, window, window_title, video_source=0):
    self.window=window
    self.window.title(window_title)
    self.video_source=video_source

    self.vid= MyVideoCapture(self.video_source)
    self.canvas=tkinter.Canvas(window, width=self.vid.width, height = self.vid.height)
    self.canvas.pack()

    btn_frame=tkinter.Frame(window, background=self.from_rgb((117, 123, 129)))
    btn_frame.place(x=0,y=0, anchor="nw", width=self.vid.width+4)

    self.btn_input=tkinter.Entry(btn_frame, width=15,bg=self.from_rgb((52, 61, 70)), fg="white")
    self.btn_input.pack(side="left", padx=10, pady=10)

    self.btn_myip=tkinter.Button(btn_frame, text=socket.gethostname(socket.gethostname()),width=10, command=None, bg=self.from_rgb((52, 61, 70)), fg="white")
    self.btn_myip.pack(side="left", padx=10, pady=10)
    self.btn_myip['state'] = tkinter.DISABLED
```

생성자

App 클래스의 생성자는 GUI를 세팅하거나 클래스의 멤버 변수를 초기화하는 데 사용되었습니다.

7

App 클래스

```
# 서버로서 연결 시도하는 버튼
def btn_server_callback():
    if DEBUG==True: print('btn_server_callback')
    if self.check_ip(self.btn_input.get()):
        self.btn_server['state'] = tkinter.DISABLED # 연결버튼 비활성화
        self.btn_client['state'] = tkinter.DISABLED # 연결버튼 비활성화
        self.btn_kill['state'] = tkinter.NORMAL # 연결시도 중단 버튼 활성화
        self.waiting_or_connected_event.set()
        threading.Thread(target=self.main_server).start() # main_server 함수로
self.btn_server=tkinter.Button(btn_frame, text="server연결", width=10, command=btn_server_callback, bg=self.from_rgb((52, 61, 70)), fg="white")
self.btn_server.pack(side="left", padx=10, pady=10)

# 클라이언트로서 연결 시도하는 버튼
def btn_client_callback():
    if DEBUG==True: print('btn_client_callback')
    if self.check_ip(self.btn_input.get()):
        self.btn_server['state'] = tkinter.DISABLED # 연결버튼 비활성화
        self.btn_client['state'] = tkinter.DISABLED # 연결버튼 비활성화
        self.btn_kill['state'] = tkinter.NORMAL # 연결시도 중단 버튼 활성화
        self.waiting_or_connected_event.set()
        threading.Thread(target=self.main_client).start() # main_client 함수로
self.btn_client=tkinter.Button(btn_frame, text="client연결", width=10, command=btn_client_callback, bg=self.from_rgb((52, 61, 70)), fg="white")
self.btn_client.pack(side="left", padx=10, pady=10)
```

서버, 클라이언트 버튼 콜백함수

각각 버튼의 기능을 정의하는 용도로 사용하고 있는 함수입니다. IP 주소를 잘못 입력하면 버튼은 비활성화됩니다.

8

App 클래스

```
# 연결 시도할 중단하는 버튼
def btn_kill_callback():
    if DEBUG==True: print('btn_kill_callback')
    self.btn_kill['state'] = tkinter.DISABLED # 연결시도 중단 버튼 비활성화
    self.connect_stopped_event.clear()
    self.connect_stopped_event = threading.Event()
    self.connect_stopped_event.set()
    self.btn_kill=tkinter.Button(btn_frame, text="연결 시도 중단", width=10, command=btn_kill_callback, bg=self.from_rgb((52, 61, 70)), fg="white")
    self.btn_kill.pack(side="left", padx=10, pady=10)
    self.btn_kill['state'] = tkinter.DISABLED

# 화상통화를 끊는 버튼
def btn_exit_callback():
    if DEBUG==True: print('btn_exit_callback')
    self.btn_exit['state'] = tkinter.DISABLED # 통화종료 버튼 비활성화
    self.disconnected_event.clear()
    self.disconnected_event = threading.Event()
    self.disconnected_event.set()
    self.btn_exit=tkinter.Button(btn_frame, text="통화 종료", width=10, command=btn_exit_callback, bg=self.from_rgb((52, 61, 70)), fg="white")
    self.btn_exit.pack(side="right", padx=10, pady=10)
    self.btn_exit['state'] = tkinter.DISABLED
```

연결시도 중단 버튼, 통화 종료 버튼 콜백함수

연결 중인 상태를 중단하거나 이미 연결되어 통화중인 상태에서 종료하는 버튼입니다.

9

App 클래스

```
# 닫기 버튼
def window_close_callback():
    if DEBUG==True: print('window_close_callback')
    if self.waiting_or_connected_event.is_set():
        self.window_close_event.clear()
        self.window_close_event.wait()
        self.window.destroy()
        exit()
    self.waiting_or_connected_event = threading.Event()
    self.window_close_event = threading.Event()
    self.window_close_event.set()
    self.window.protocol("WM_DELETE_WINDOW", window_close_callback)

    self.show_other_video = False

    self.delay=15
    self.update()

    self.window.mainloop()
```

닫기 버튼 콜백함수

윈도우 창에 있는 닫기 버튼을 누르면 창이 사라지게 되며, 기능하고 있던 화상통화 역시 종료됩니다.

```
def update(self):
    if self.show_other_video == False:
        ret, frame=self.vid.get_frame()
        if ret:
            self.photo = PIL.ImageTk.PhotoImage(image=PIL.Image.fromarray(frame))
            self.canvas.create_image(0,0, image=self.photo, anchor=tkinter.NW)
            self.window.after(self.delay,self.update)

    def from_rgb(self,rgb):
        return "#%02x%02x%02x" % rgb

    def confirm():
        in_text = "인력 내용 : " + input_text.get()
        label.configure(text=in_text)

    def check_ip(self,ip):
        # 자기 자신에게 연결하려는 경우 확인
        if ip == socket.gethostbyname(socket.gethostname()):
            answer = askyesno('','자기 자신에게 연결할 수 없습니다. 그래도 강제로 연결합니까?')
            return answer
        # 아니면 True
        else:
            return True

    # sock.recv()로 size 크기의 바이트열을 받아오는 함수
    def full_recv(self,sock,size):
        a = sock.recv(size)
        while len(a) != size:
            a += sock.recv(size-len(a))
        return a
```

기타 함수

10


```

def main_server(self):
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.settimeout(2)
    server_socket.bind((self.btn_input.get(), 9999))
    server_socket.listen(1)
    while True:
        if DEBUG==True: print('클라이언트 연결 시도중...')
        try:
            client_socket, address = server_socket.accept()
            break # 연결 성공시 while문 바깥으로
        except TimeoutError:
            # 연결시도 중단 버튼 또는 닫기버튼 눌렀으면 중단
            if (not self.connect_stopped_event.is_set()) or (not self.window_close_event.is_set()):
                if DEBUG==True: print('클라이언트 연결 시도 중단됨')
                server_socket.close()
                self.connect_stopped_event.set()
                self.window_close_event.set()
                self.waiting_or_connected_event.clear()
                self.btn_server['state'] = tkinter.NORMAL # 연결버튼 활성화
                self.btn_client['state'] = tkinter.NORMAL # 연결버튼 활성화
                return
        except OSError: # 올바른지 않은 IP 입력 시
            if DEBUG==True: print('올바르지 않은 IP:', self.btn_input.get())
            sock.close()
            self.waiting_or_connected_event.clear()
            self.btn_server['state'] = tkinter.NORMAL # 연결버튼 활성화
            self.btn_client['state'] = tkinter.NORMAL # 연결버튼 활성화
            self.btn_kill['state'] = tkinter.DISABLED # 연결시도 중단 버튼 비활성화
            return
        if DEBUG==True: print('클라이언트 연결됨')
        self.btn_kill['state'] = tkinter.DISABLED # 연결시도 중단 버튼 비활성화
        self.btn_exit['state'] = tkinter.NORMAL # 종료종료 버튼 활성화
        # main_sender 함수와 main_receiver 함수 호출
        send = threading.Thread(target=self.main_sender, args=(client_socket,))
        rcv = threading.Thread(target=self.main_receiver, args=(client_socket,))
        send.start()
        rcv.start()
        send.join()
        rcv.join()
        # main_sender 함수와 main_receiver 함수가 종료된 후
        if DEBUG==True: print('클라이언트 연결 끊김')
        client_socket.close()
        server_socket.close()
        self.disconnected_event.set()
        self.window_close_event.set()
        self.waiting_or_connected_event.clear()
        self.btn_server['state'] = tkinter.NORMAL # 연결버튼 활성화
        self.btn_client['state'] = tkinter.NORMAL # 연결버튼 활성화
        self.btn_kill['state'] = tkinter.DISABLED # 종료종료 버튼 비활성화

```

11

```

def main_server(self):
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.settimeout(2)
    server_socket.bind((self.btn_input.get(), 9999))
    server_socket.listen(1)
    while True:
        if DEBUG==True: print('클라이언트 연결 시도중...')
        try:
            client_socket, address = server_socket.accept()
            break # 연결 성공시 while문 바깥으로
        except TimeoutError:
            # 연결시도 중단 버튼 또는 닫기버튼 눌렀으면 중단
            if (not self.connect_stopped_event.is_set()) or (not self.window_close_event.is_set()):
                if DEBUG==True: print('클라이언트 연결 시도 중단됨')
                server_socket.close()
                self.connect_stopped_event.set()
                self.window_close_event.set()
                self.waiting_or_connected_event.clear()
                self.btn_server['state'] = tkinter.NORMAL # 연결버튼 활성화
                self.btn_client['state'] = tkinter.NORMAL # 연결버튼 활성화
                return
        except OSError: # 올바른지 않은 IP 입력 시
            if DEBUG==True: print('올바르지 않은 IP:', self.btn_input.get())
            sock.close()
            self.waiting_or_connected_event.clear()
            self.btn_server['state'] = tkinter.NORMAL # 연결버튼 활성화
            self.btn_client['state'] = tkinter.NORMAL # 연결버튼 활성화
            self.btn_kill['state'] = tkinter.DISABLED # 연결시도 중단 버튼 비활성화
            return
        if DEBUG==True: print('클라이언트 연결됨')
        self.btn_kill['state'] = tkinter.DISABLED # 연결시도 중단 버튼 비활성화
        self.btn_exit['state'] = tkinter.NORMAL # 종료종료 버튼 활성화
        # main_sender 함수와 main_receiver 함수 호출
        send = threading.Thread(target=self.main_sender, args=(client_socket,))
        rcv = threading.Thread(target=self.main_receiver, args=(client_socket,))
        send.start()
        rcv.start()
        send.join()
        rcv.join()
        # main_sender 함수와 main_receiver 함수가 종료된 후
        if DEBUG==True: print('클라이언트 연결 끊김')
        client_socket.close()
        server_socket.close()
        self.disconnected_event.set()
        self.window_close_event.set()
        self.waiting_or_connected_event.clear()
        self.btn_server['state'] = tkinter.NORMAL # 연결버튼 활성화
        self.btn_client['state'] = tkinter.NORMAL # 연결버튼 활성화
        self.btn_kill['state'] = tkinter.DISABLED # 종료종료 버튼 비활성화

# 소켓 클라이언트
def main_client(self):
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    while True:
        if DEBUG==True: print('서버 연결 시도중...')
        try:
            sock.connect((self.btn_input.get(), 9999))
            break # 연결 성공시 while문 바깥으로
        except ConnectionRefusedError:
            # 연결 시도 중단 버튼 또는 닫기버튼 눌렀으면 중단
            if (not self.connect_stopped_event.is_set()) or (not self.window_close_event.is_set()):
                if DEBUG==True: print('서버 연결 시도 중단됨')
                sock.close()
                self.connect_stopped_event.set()
                self.window_close_event.set()
                self.waiting_or_connected_event.clear()
                self.btn_server['state'] = tkinter.NORMAL # 연결버튼 활성화
                self.btn_client['state'] = tkinter.NORMAL # 연결버튼 활성화
                return
        # 올바른지 않은 IP 입력 시
        except OSError:
            if DEBUG==True: print('올바르지 않은 IP:', self.btn_input.get())
            sock.close()
            self.waiting_or_connected_event.clear()
            self.btn_server['state'] = tkinter.NORMAL # 연결버튼 활성화
            self.btn_client['state'] = tkinter.NORMAL # 연결버튼 활성화
            self.btn_kill['state'] = tkinter.DISABLED # 연결시도 중단 버튼 비활성화
            return
        if DEBUG==True: print('서버 연결됨')
        self.btn_kill['state'] = tkinter.DISABLED # 연결시도 중단 버튼 비활성화
        self.btn_exit['state'] = tkinter.NORMAL # 종료종료 버튼 활성화
        # main_sender 함수와 main_receiver 함수 호출
        send = threading.Thread(target=self.main_sender, args=(sock,))
        rcv = threading.Thread(target=self.main_receiver, args=(sock,))
        send.start()
        rcv.start()
        send.join()
        rcv.join()
        # main_sender 함수와 main_receiver 함수가 종료된 후
        if DEBUG==True: print('서버 연결 끊김')
        sock.close()
        self.disconnected_event.set()
        self.window_close_event.set()
        self.waiting_or_connected_event.clear()
        self.btn_server['state'] = tkinter.NORMAL # 연결버튼 활성화
        self.btn_client['state'] = tkinter.NORMAL # 연결버튼 활성화
        self.btn_kill['state'] = tkinter.DISABLED # 종료종료 버튼 비활성화

```

12

App 클래스

```
# 소켓으로 데이터를 보내는 함수
def main_sender(self, sock):

    # 오디오 데이터 전송
    def audio_callback(in_data, frame_count, time_info, status):
        try:
            data_size = struct.pack('>I', len(in_data))
            sock.send(b'\x01'+data_size+in_data)
            # 상대방이 통화를 종료했을 경우 중단
            except (ConnectionResetError, ConnectionAbortedError):
                pass
            return (None, pyaudio.paContinue)
        stream = pa.open(rate=44100, channels=1, format=pyaudio.paInt16, input=True, frames_per_buffer=8192, stream_callback=audio_callback)

    # 비디오 데이터 전송
    widthheight = struct.pack('>II', int(self.vid.width), int(self.vid.height))
    # 통화를 종료 또는 대기대문 열 수 있으면 중단
    while self.disconnected_event.is_set() and self.window_close_event.is_set():
        try:
            ret, frame = self.vid.get_frame()
            if ret:
                sock.send(b'\x02'+widthheight+frame.tobytes())
            # 상대방이 통화를 종료했을 경우 중단
            except (ConnectionResetError, ConnectionAbortedError):
                break
        stream.close()
        self.disconnected_event.clear()
```

데이터를 전송하는 스레드 함수

데이터를 전송할 때 사용하는 스레드를 실행할 때 사용하는 함수입니다.

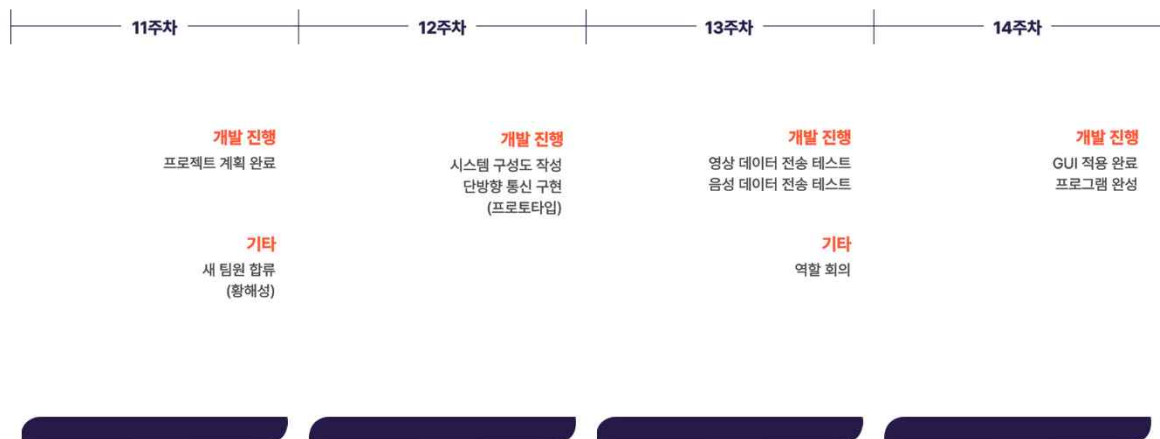
```
# 소켓으로부터 데이터를 받는 함수
def main_receiver(self, sock):
    stream = pa.open(rate=44100, channels=1, format=pyaudio.paInt16, output=True, frames_per_buffer=8192)
    # 통화를 종료 또는 대기대문 열 수 있으면 중단
    while self.disconnected_event.is_set() and self.window_close_event.is_set():
        try:
            # 연결 오류
            except:
                pass
            # 오디오 데이터가 온 경우: b'\x01'
            # 비디오 데이터가 온 경우: b'\x02'
            rcvd_data = sock.recv(1)
            if rcvd_data == b'\x01':
                # 오디오 데이터를 처리하여 오디오로 출력
                elif rcvd_data == b'\x01':
                    size = struct.unpack('>I', self.full_rcv(sock, 4))[0]
                    volume = 4
                    rcvd_data = np.frombuffer(self.full_rcv(sock, size), dtype=np.int16) * volume
                    stream.write(rcvd_data.tobytes())
                # 비디오 데이터가 처리된 이미지로 출력
                elif rcvd_data == b'\x02':
                    width, height = struct.unpack('>II', self.full_rcv(sock, 8))
                    rcvd_data = self.full_rcv(sock, width*height*3)
                    # 이미지 비디오 코덱은 코덱을 지정
                    self.show_other_video = True
                    frame = np.frombuffer(rcvd_data[:width*height*3], dtype='uint8').reshape((height, width, 3))
                    self.photo = PIL.ImageTk.PhotoImage(image=PIL.Image.fromarray(frame))
                    self.canvas.create_image(0, 0, image=self.photo, anchor=tkinter.NW)
            # 상대방이 통화를 종료했을 경우 중단
            except (ConnectionResetError, ConnectionAbortedError):
                break
        stream.close()
        cv2.destroyAllWindows()
        self.disconnected_event.clear()
```

데이터를 받는 스레드 함수

데이터를 받을 때 사용하는 스레드의 실행에 필요한 함수입니다.

13

프로젝트 일정



14

프로젝트 결과

USER1

[illegible]

USER2

```

C:\Users\hik\Desktop\KIDnapAttack\KIDnapAttack.py
File Edit Format Run Options Window Help
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
103
```

15

역할



메인 프로그래머, 김민석

네트워크 통신과 영상 처리 및 음성 데이터 통신 부분을 맡았습니다.



메인 프로그래머, 황해성

그래픽 인터페이스 위주로 맡아서 개발하였습니다.



발표 및 문서 자료 제작, 최신희

계획 및 시스템 구성도, 회의 내용 취합, 프리젠테이션 발표를 맡았습니다.

16

출처

https://tayrelgrin.blogspot.com/2018/09/python-opencv_9.html