

# Redirected Walking

231114 19101188 고은수

# Camera Latency 줄이기

```
11 cam1 = cv2.VideoCapture(cam_list[0])
12 cam2 = cv2.VideoCapture(cam_list[1])
13
14 while True:
15     beforeRead = time.time()
16
17     ret1, frame1 = cam1.read()
18     ret2, frame2 = cam2.read()
19
20
21     afterRead = time.time()
22
23     print('read time : ' + str(afterRead - beforeRead))
24
25     if not ret1:
26         print("cam1 is not connected")
27         break
28
29     if not ret2:
30         print("cam2 is not connected")
31         break
32
33     beforeShow = time.time()
34
35     cv2.imshow("1", frame1)
36     cv2.imshow("2", frame2)
37
38     afterShow = time.time()
39
40     print('show time : ' + str(afterShow - beforeShow))
41
42     if cv2.waitKey(1) == 27:
43         break
```

기존 방식 -> 약 1.9초정도 딜레이 발생

```
t2.py > ...
1 import cv2
2 import threading
3
4 cam_list = ['rtsp://admin:duSrntlf123@192.168.0.133:554/Streaming/Channels/1', 'rts
5
6 cam_index = {
7     'rtsp://admin:duSrntlf123@192.168.0.133:554/Streaming/Channels/1': 0,
8     'rtsp://admin:duSrntlf123@192.168.0.138:554/Streaming/Channels/1': 1
9 }
10
11 cam1 = cv2.VideoCapture(cam_list[0])
12 cam2 = cv2.VideoCapture(cam_list[1])
13
14 def show_camera(cam, cam_name):
15     while True:
16         ret, frame = cam.read()
17         if not ret:
18             print(f"{cam_name} is not connected")
19             break
20         cv2.imshow(cam_name, frame)
21         if cv2.waitKey(1) == 27:
22             break
23
24 thread1 = threading.Thread(target=show_camera, args=(cam1, "Camera 1"))
25 thread2 = threading.Thread(target=show_camera, args=(cam2, "Camera 2"))
26 thread1.start()
27 thread2.start()
28
29 thread1.join()
30 thread2.join()
31
32 cv2.destroyAllWindows()
33
```

멀티 쓰레드 방식 -> 약 0.7초정도 딜레이 발생

# Camera Latency 줄이기

```
11 cam1 = cv2.VideoCapture(cam_list[0])
12 cam2 = cv2.VideoCapture(cam_list[1])
13
14 while True:
15     beforeRead = time.time()
16
17     ret1, frame1 = cam1.read()
18     ret2, frame2 = cam2.read()
19
20
21     afterRead = time.time()
22
23     print('read time : ' + str(afterRead - beforeRead))
24
25     if not ret1:
26         print("cam1 is not connected")
27         break
28
29     if not ret2:
30         print("cam2 is not connected")
31         break
32
33     beforeShow = time.time()
34
35     cv2.imshow("1", frame1)
36     cv2.imshow("2", frame2)
37
38     afterShow = time.time()
39
40     print('show time : ' + str(afterShow - beforeShow))
41
42     if cv2.waitKey(1) == 27:
43         break
```

버퍼 사이즈 줄이기

```
cam1.set(cv2.CAP_PROP_BUFFERSIZE, 1)
cam2.set(cv2.CAP_PROP_BUFFERSIZE, 1)
```

1번 방식에서 약 1.9초 -> 약1.7초로 0.2초정도 딜레이 감소



# Camera Latency 줄이기

## -Bot-sort에 적용

```
def detection_thread(cam_num):  
    detection(cam_num)
```

```
def detect(cam_num):  
    global coordinates  
    global x_average, y_average  
    weights, view_img, save_txt, imsz, trace = opt.weights, opt.view_img, opt.save_txt, opt.img_size, opt.trace  
    if cam_num==1:  
        source = cam_list[0]  
    else:  
        source = cam_list[1]
```

```
# parser.add_argument('--source', type=str, default='inference/images', help='source') # file/folder, 0 for webcam
```

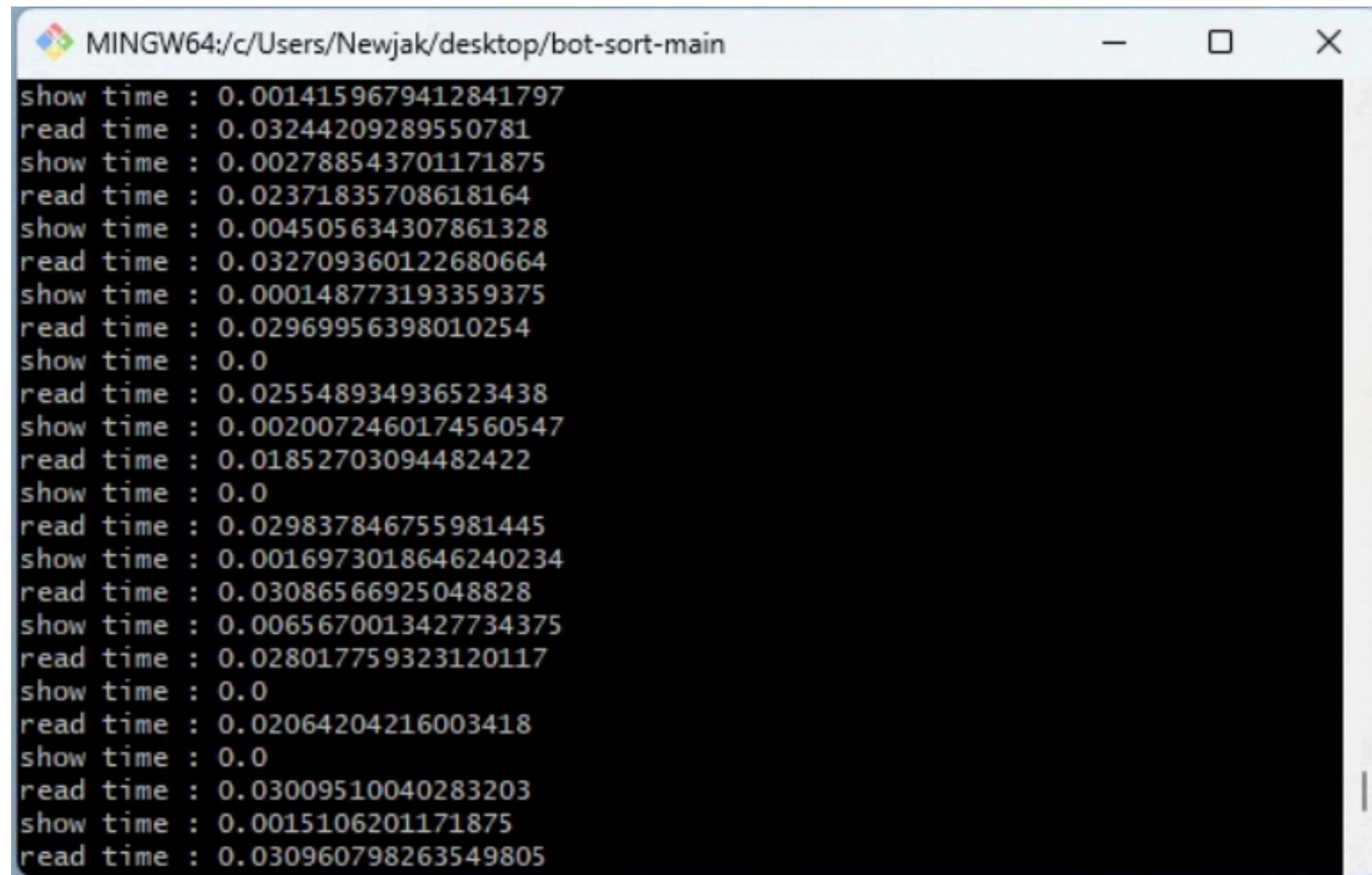
```
cam1 = threading.Thread(target=detection_thread, args=(1,))  
cam2 = threading.Thread(target=detection_thread, args=(2,))  
  
cam1.start()  
cam2.start()
```

```
cap = cv2.VideoCapture(url)  
cap.set(cv2.CAP_PROP_BUFFERSIZE, 1)
```

```
self.cap = cv2.VideoCapture(pipe) # video capture from pipe  
self.cap.set(cv2.CAP_PROP_BUFFERSIZE, 1)
```

```
self.cap = cv2.VideoCapture(path)  
self.cap.set(cv2.CAP_PROP_BUFFERSIZE, 1)
```

# Camera Latency 줄이기



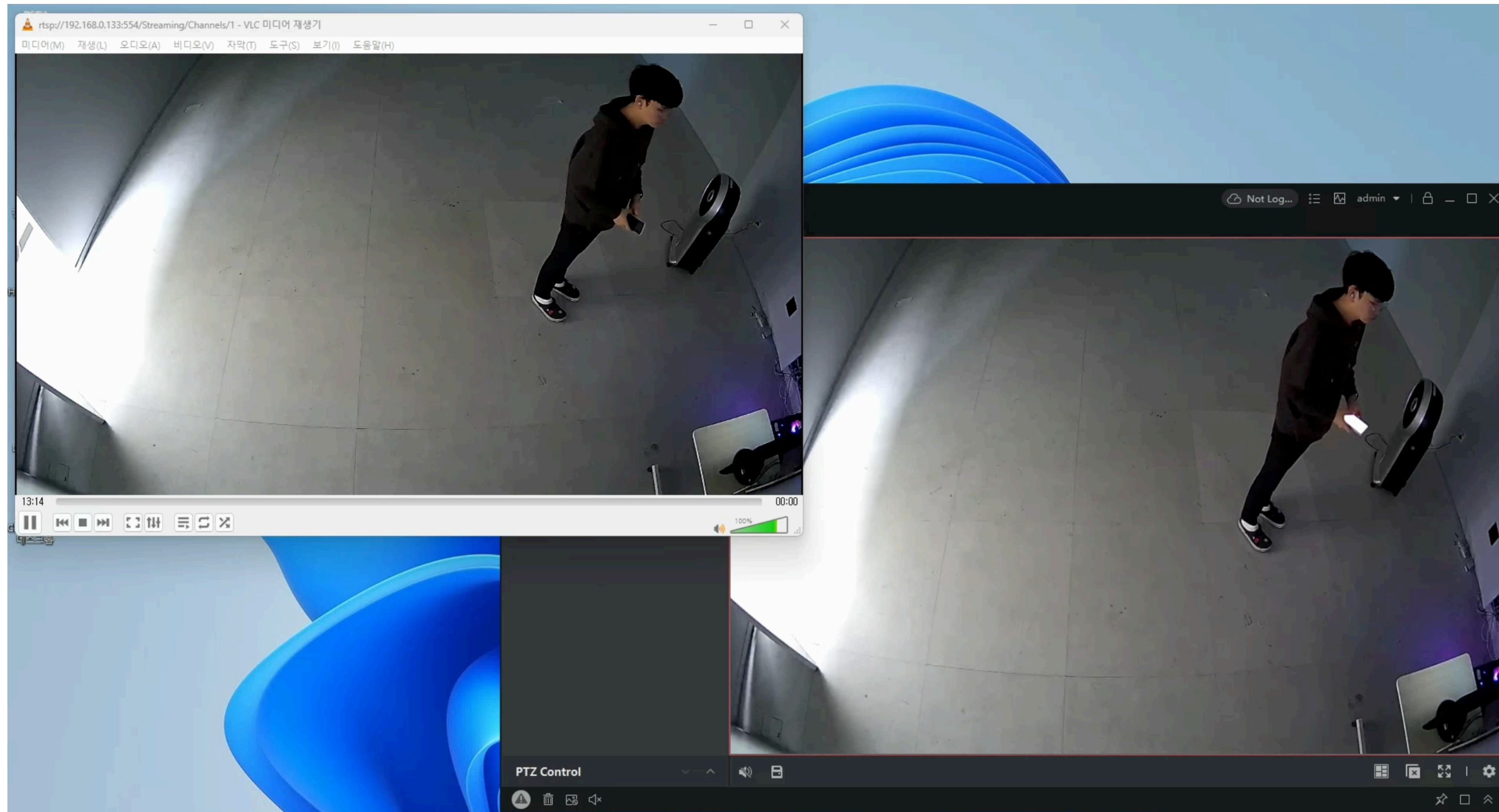
```
MINGW64:/c/Users/Newjak/desktop/bot-sort-main
show time : 0.0014159679412841797
read time : 0.03244209289550781
show time : 0.002788543701171875
read time : 0.02371835708618164
show time : 0.004505634307861328
read time : 0.032709360122680664
show time : 0.000148773193359375
read time : 0.02969956398010254
show time : 0.0
read time : 0.025548934936523438
show time : 0.0020072460174560547
read time : 0.01852703094482422
show time : 0.0
read time : 0.029837846755981445
show time : 0.0016973018646240234
read time : 0.03086566925048828
show time : 0.0065670013427734375
read time : 0.028017759323120117
show time : 0.0
read time : 0.02064204216003418
show time : 0.0
read time : 0.03009510040283203
show time : 0.0015106201171875
read time : 0.030960798263549805
```

1번(기존) 방식에서 두 카메라의 프레임을 읽어오는 데 걸리는 시간 : 약 0.03초 내외  
매 프레임을 화면에 출력하는 데 걸리는 시간 : 약 0.002초 미만  
=> 네트워크 속도, 카메라 자체 레이턴시의 비중이 크다고 판단



# Camera Latency 줄이기

-순수하게 카메라로부터 프레임 받아오는 지연이 얼마나 되는지?



VLC 플레이어 RTSP 스트리밍 - 약 1.2초

IVMS - 약 0.2초

감사합니다