Assignment#1

25.02.06~07

0. RTSP Connection Info

IP Address: 192.168.200.132

ID: admin

Password: 1234567s

Suffix: /Streaming/Channels/101

rstp://{ID}:{Password}@{IP Address}:554/{Suffix}

rstp://admin:1234567s@192.168.200.132:554/Streaming/Channels/101

1. Find the library for RTSP in Python

Library	Ease of Use	Reliability	Low Latency	Hardware Acceleration	Best Use Case
OpenCV	✅ Easy	⚠️ Sometimes unstable	❌ High latency	❌ No	General video processing
FFmpeg	⚠️ Medium	✅ Very reliable	❌ High latency	❌ No	Handling unstable streams
GStreamer	❌ Complex	✅ Reliable	✅ Low latency	✅ Yes	Low-latency real-time streaming
Livestreamer	⚠️ Medium	✅ Good network handling	❌ High latency	❌ No	Network-heavy streaming

OpenCV는 영상처리 라이브러리 이고, **FFmpeg**나 **GStreamer**를 **백엔드** 로 사용함

High Latency Issues

Latency?? OpenCV has an internal **buffer**, so it might **lag** behind live video.

오래된 이미지를 버퍼형태로 저장해 놓기 때문에 -> 버퍼사이즈를 줄여 놓거나 적절하게 조절

Codec Incompatibility

OpenCV relies on FFmpeg for decoding, but some cameras use unsupported codecs.

Some IP cameras use H.265, which OpenCV might not decode properly.

비디오 코덱 문제 발생할 수도

Memory Leaks

OpenCV might not properly release memory if the stream is opened/closed frequently.

=> 전반적으로 보았을 때 GStreamer 사용이 적절해 보임

2. Connect IP camera using OpenCV + ffmpeg (codec)

Python 코드는 완성

https://answers.opencv.org/question/230143/opencv-connect-and-process-an-ip-camera-stream-rtsp-protocol/

But, 1. 에서 언급한 몇가지 에러사항을 테스트 해볼려고 함

cv2.getBuildInformation() 시 opencv 빌드 정보를 알수 있고, ffmpeg 내 코덱을 사용하였음

*** Error 1 : RSTP 서버 접근 실패의 경우 cv2.VideoCapture에서 대기 시간이 길 (30초)**

```python
def repeat_access(rtsp_url, try_times, try_interval):
    # print("connecting..")
    for i in range(try_times):
        cap = cv2.VideoCapture(rtsp_url)
        # print("connecting2..")
        if cap is None:
            print(f"RTSP 서버 접근 실패... Try{i+1}!")
            time.sleep(try_interval)
        else:
            ret, frame = cap.read()
            if not ret or frame is None:
                print(f"프레임 읽기 실패. 스트림을 재시도합니다... Try{i+1}!")
                cap.release()
                time.sleep(try_interval)
            else:
                return cap
    return None
```

rtsp 서버접근 실패의 경우 cv2.VideoCapture에서

cv2.VideoCapture 에 30초동안 막혀 있음(Blocking 현상)

opencv 이전 버전 및 최신버전 모두에서 공통적인 문제

https://github.com/opencv/opencv/issues/24393

Sol 1.

export OPENCV_FFMPEG_CAPTURE_OPTIONS="timeout;5000000|rtsp_transport;tcp"

5초로 제한 가능하나, cap 이 None이 아닌상태로 강제 종료시켜 RTSP 서버 접근에서 실패하여야 하는데 cap.read() 에서 error

Sol 2. Thread를 만들어 해결가능하나 복잡해짐

=> Gstreamer로 해결해 보기로함

*** Other Errors**

High Latency Issues, Codec Incompatibility, Memory Leaks 문제도 실제 테스트 후 고려해봐야함

항목	GStreamer	FFmpeg
설계 철학	파이프라인(Element) 기반으로 모듈화. 동적인 파이프라인 변경과 확장에 최적화	단일 툴/라이브러리(FFmpeg 명령어·libav API). 필터그래프 사용 가능하나 구조가 단순화됨
파이프라인 유연성	런타임 중 Element 교체/추가/제거 용이. 이벤트 기반 흐름 제어 등에 강점	파이프라인 구성은 가능하지만, GStreamer만큼 직관적이지 않을 수 있음 (필터그래프 설정 필요)
하드웨어 가속	NVDEC/NVENC, VAAPI, V4L2, OpenMAX, DeepStream 등 다양한 GPU/ASIC 플러그인 공식 지원. 임베디드 호환성↑	NVENC, VAAPI, QSV 등 하드웨어 가속 지원. 다만 설정이 옵션 기반이고, 파이프라인 자동 연결은 제한적

3. Connect IP camera using Gstreamer

https://gitlab.freedesktop.org/gstreamer/gstreamer/-/tree/main/subprojects/gst-python

Opencv + gstreamer 설치 필요

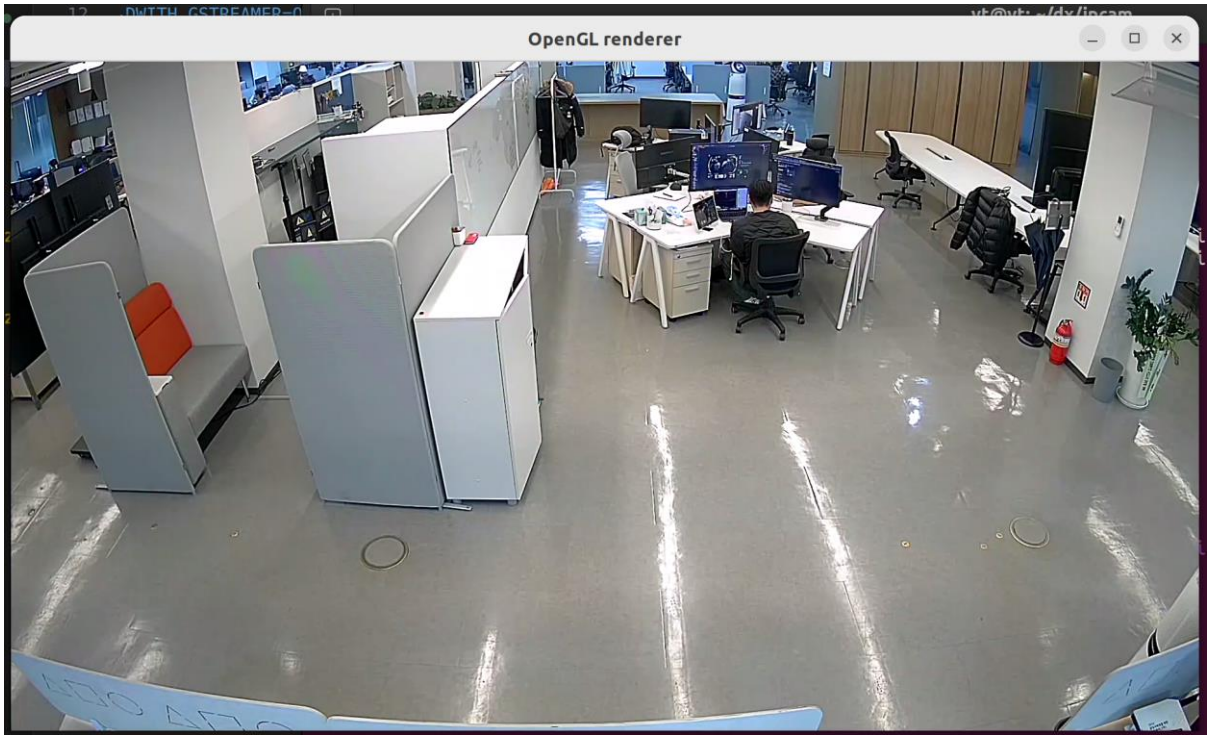
Conda 환경에서 gstreamer 는 아래와 같이 설치

conda install -c conda-forge gstreamer gst-plugins-base gst-plugins-good gst-plugins-bad gst-plugins-ugly

conda install -c conda-forge gst-libav pygobject

gst-launch-1.0 playbin uri=rtsp://admin:1234567s@192.168.200.132:554/Streaming/Channels/101

(실행 관련 명령어 별도 첨부)


OpenGL renderer