**종합설계 프로젝트 수행 보고서**

|  |  |
| --- | --- |
| **프로젝트명** | **실시간 총기 위협 감지를 위한 CCTV 감시 시스템** |
| **팀번호** | **S1-7** |
| **문서제목** | **수행계획서( )**  **2차발표 중간보고서( O )**  **3차발표 중간보고서( )**  **최종결과보고서( )** |

**2022.03.07**

**팀원 : 정수민 (팀장)**

**배성은 (팀원)**

**지도교수 : 나보균**

**문서 수정 내역**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **작성일** | **대표작성자** | **버전** | **수정내용** |  |
| 2022.01.03 | 정수민(팀장) | 1.0 | 수행계획서 | 최초작성 |
| 2022.02.27 | 정수민(팀장) | 2.0 | 2차발표자료 | 설계서추가 |

**문서 구성**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **진행단계** | **프로젝트**  **계획서 발표** | **중간발표1**  **(3월)** | **중간발표2**  **(5월)** | **학기말발표**  **(6월)** | **최종발표**  **(10월)** |
| **기본양식** | 계획서 양식 | 계획서 양식 | 계획서 양식 | 계획서 양식 | 계획서 양식 |
| **포함되는**  **내용** | Ⅰ. 서론  (1~6)  Ⅱ. 본론  (1~3)  참고자료 | Ⅰ. 서론  (1~6)  Ⅱ. 본론  (1~4)  참고자료 | Ⅰ. 서론  (1~6)  Ⅱ. 본론  (1~5)  참고자료 | Ⅰ. 서론  (1~6)  Ⅱ. 본론  (1~7)  참고자료 | I  II  III |

**이 문서는 한국산업기술대학교 컴퓨터공학부의 “종합설계”교과목에서 프로젝트 “실시간 총기 위협 감지를 위한 CCTV 감시 시스템”을 수행하는**

**S1-7팀(정수민, 배성은)이 작성한 것으로 사용하기 위해서는 팀원들의 허락이 필요합니다.**

**목 차**

**Ⅰ. 서론**

1. 작품선정 배경 및 필요성

2. 기존 연구/기술동향 분석

3. 개발 목표

4. 팀 역할 분담

5. 개발 일정

6. 개발 환경

**Ⅱ. 본론**

1. 개발 내용

2. 문제 및 해결방안

3. 시험 시나리오

4. 상세 설계

5. Prototype 구현

6. 시험 / 테스트 결과

7. Coding & DEMO

**Ⅲ. 결론**

1. 연구 결과

2. 작품제작 소요재료 목록

참고자료

**I 서론**

**1 작품선정 배경 및 필요성**

**① 작품 선정 배경**

현재 미국은 총기 소유가 합법인 국가로 Covid-19 사태 이전인 2019년

기준 총기 사고 피살자 수 15,208명, 미성년(0-17세) 총기 부상사〮망자수 37760명으로 총기 관련 범죄에 굉장히 취약한 상황임. 공공시설 및 주택에서의 CCTV 설치율이 계속 증가하고 있지만 땅이 워낙 넓은 탓에 인적이 드문 길거리나 야간에는 여전히 총기 관련 범죄에 취약함.

최근 미국의 CCTV 수요가 계속해서 증가하고 있으나, 미국 내 CCTV 수입 제품 점유율에서 중국이 44%를 차지할 정도로 중국에 대한 의존도가 큼.

중국은 국제 정세 이슈로 미국과의 외교적 관계가 현재 매우 악화된 상태이며, 중국발 랜섬웨어, 샤오미 스마트폰 보안 이슈, 레노버 노트북 백도어 사건 등으로 중국 제품들에 대한 보안성 신뢰도가 굉장히 떨어진 상황임.

**② 작품의 필요성**

* 유동인구가 많은 곳에서 총기 사고가 일어날 경우, 목격자가 신고하면

경찰 이 출동하여 상황을 정리할 수 있지만, 목격자가 없거나 신고하기 어려운 상황에 처하면 큰 인명 피해가 발생하거나 사건 사후 처리가 늦어질 수 있음.

* 기존에도 situation detection 기술로 특정 상황을 예방/신고 하는 시스템

은 있었으나 대부분 서버에서 모든 처리를 하는 서버 의존적인 시스템임.

* Gun threat detection은 최근에 활발히 연구되고 있지만 아직 application

이 많지 않으며 군 등에서 활용할 수 있음에도 불구하고 총기 소유가 불법인 우리나라의 특성상 국내 연구가 거의 없음.

**2 기존 연구/기술동향 분석**

**Real-time gun detection in CCTV: An open problem**

Gun detection model 학습을 위한 dataset이다.

Mock Attack datset과 Unity Synthetic dataset으로 나누어져 있다.

Mock Attack dataset은 미국에서 촬영한 현실세계의 영상 데이터를 기반으로 레이블링한 것이다. 세가지 분류로 나누어져 있으며 각각 607개, 3511개, 1031개의 이미지를 포함한 데이터셋이다.

Unity Synthetic datset은 가상의 상황을 Unity 엔진을 사용하여 모델링한 결과를 렌더링한 영상을 레이블링한 것이다. 똑같이 세가지로 분류되어있으며 각각 500개, 1000개, 2500개의 이미지를 포함한다.

**Omnilert Gun Detect**

2003년에 창업한 미국의 카메라 보안 솔루션 회사 Omnilert에서 2020년부터 제공하는 솔루션이다. Gun Detection, Monitoring, Immediate response, Priority Alert 등을 제공한다.

Detection을 포함한 모든 기능은 Omnilert에서 제공하는 EMNS(Emergency

Mass Notification and Management System)에서 처리한다.

**3 개발 목표**

* Object detection model을 학습시켜 gun object detection을 수행
* 미리 녹음된 음성을 사용하여 위험 상황 발생시 경고 방송
* RTSP 프로토콜을 사용하여 low-delay video streaming 구현
* DVR 서버에 detection result 전송
* PTZ Controllable Camera 등의 하드웨어를 원격지에서 제어
* 관리 서버와 양방향 Audio Streaming 구현

**4 팀 역할 분담**

**정수민**

시스템 모듈 설계

하드웨어(카메라, 마이크, 조명 등) 컨트롤 기능 구현

Yolo, fast-RCNN 등의 모델을 사용하여 object detection 성능 측정 및 학습

Gstreamer 등을 이용한 네트워킹 기능 구현

시스템 통합

**배성은**

Object detection 관련 사전조사

Opencv를 이용한 Video Capture 구현

Video Encoding 기능 구현

시스템 문서화

**5 개발 일정**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 추진계획 | 12월 | 1월 | 2월 | 3월 | 4월 | 5월 | 6월 | 7-9월 |
| 조사 및 학습 |  |  |  |  |  |  |  |  |
| 시스템 설계 |  |  |  |  |  |  |  |  |
| 구현 |  |  |  |  |  |  |  |  |
| 테스트 및 데모 |  |  |  |  |  |  |  |  |
| 문서화 및 발표 |  |  |  |  |  |  |  |  |
| 최종보고서 작성 |  |  |  |  |  |  |  |  |

**6 개발 환경**

* **Hardware**: Jetson Nano Development Kit
* **OS**: Linux (Ubuntu, Arch)
* **Langauage**: Python 3
* **Frameworks**: Opencv, numpy, matplotlib, Glib,

GStreamer, Darknet, OpenGL, pyglet

* **Development Methodology**: Agile

**II 본론**

1. **개발 내용**

**1.1 Object detection**

Detection model은 darknet based yolov4-tiny를 사용한다. 작은 물체 감지를 위해 darknet github repository의 How to improve object detection 항

목을 참고하여 모델을 최적화하고, How to train tiny-yolo (to detect your custom objects) 항목을 참고하여 모델을 학습시킨다.

이와는 별개로 python에서 opencv로 신경망을 로드해 Frame Buffer에

쌓인 이미지 데이터에서 gun detection을 수행하는 모듈을 개발한다.

**1.2** **Video Streaming**

개발의 편의성을 위해 GStreamer Pipeline을 적극 이용한다. Gst-parse-launch로 파이프라인을 쉽게 구성할 수 있다. CPU 부하를 줄이기 위해

OpenMax 인코더(omxh264, omxh265 element)로 인코딩 시에 NVENC

하드웨어 인코더를 사용하도록 강제한다.

높은 품질보다는 low-delay가 우선이며 RTSP 프로토콜로 감싸 영상을

전송한다. 음성 데이터가 필요할 시 gstreamer pipeline element 중 mux

계열의 element를 사용하여 영상 데이터와 묶어 전송한다.

**1.3 Networking**

Camera identification 및 Detection Result transport는 TCP 프로토콜을

이용한다. Data serialize는 json을 이용하여 Detection Result 및 Detection

Box 클래스에 미리 as\_dict 함수를 정의하여 dictionary 형태로 변환 후

json.dumps() 함수를 통해 json data로 serialize 한 후에 전송한다.

연결이 끊길 경우 주기적으로 재접속을 시도한다. Timeout은 연결 시도

횟수에 따라 유동적으로 변한다.

**1.4 Hardware control**

Camera의 PTZ Control과 Microphone 및 Speaker를 관리 서버에서 원격으로 제어 할 수 있어야 한다. Camera PTZ Control에 대해서는 일단 API

가 존재하는 TP-link의 IP Camera에 대하여 모듈을 제작한 후, 여건이 되면

Serial Communication을 통한 카메라 제어로 전환한다. 관리 서버에서 전송

한 음성 데이터를 스피커로 재생할 수 있어야 하며, 반대로 마이크에 입력

된 음성 데이터를 서버로 보낼 수 있어야 한다.

* 1. **Audio Device Control**

Gun detect 이벤트 발생시에 미리 녹음된 TTS 음성을 재생한다. 또 시스템에 내장된 스피커를 통해 관리 서버에서 스트리밍한 음성 데이터를 재생할 수 있어야 하며, 반대로 마이크를 통해 받은 음성 데이터를 관리 서버로 스트리밍할 수 있어야 한다. Audio device controller 개발에 linux의 alsa library를 사용한다.

* 1. **Renderer**

테스트 및 디버깅을 위한 렌더러를 개발한다. Frame Buffer에서 이미지 데이터를, Detection Buffer에서 Detection Result를 가져와 opencv를 이용해

하나의 이미지로 만든다.

이후 matplotlib, opencv, opengl 등의 렌더링 엔진을 사용해 Detection

결과를 포함한 영상을 화면에 표시하도록 한다.

1. **문제 및 해결방안**

**본 항목은 문제 발생/해결 시마다 지속적으로 업데이트 한다.**

**2.1** 기존에 yolov3-tiny detection model을 사용하였으나 detection performance 및 fps performance가 요구 수준에 미치지 못하였음. 추가적인 조사 결과 yolov4-tiny 모델이 있다는 것을 발견하고 성능 측정 결과yolov4-tiny가 두 측면에서 yolov3-tiny 모델보다 우세하다 판단하여 모델을 변경함.

**2.2** video streaming에서 need-data event 발생 시 framebuffer가 비어있는 등의 문제로 데이터를 전송하지 않으면 파이프라인이 블로킹되므로, 만약 framebuffer가 비어있다면 더미 프레임을 전송함.

**2.3** Laptop environment 에서 테스트시 OpenCV 및 matplotlib 기반의 렌더러가 정상적으로 작동하지 않으므로 OpenGL 혹은 qt 등의 렌더링 엔진으로의 변경이 필요함.

**2.4** 프로그램 장시간 가동 시 발열로 인해 방열판이 있음에도 시스템 온도가 80도까지 증가함. 시스템 안정성 확보를 위해 시스템 보드에 쿨러 장착이 필요함.

**2.5** 기존에 detection result 전송을 위해 python의 dill 라이브러리로 serialize하여 전송했으나, python-python 통신밖에 불가능한 dill의 단점으로 인해 DetectionResult 클래스 인스턴스를 json 형식으로 변환하여 serialize 후 전송하는 방법으로 변경함.

**2.6** Video Capture Thread에서 frame을 저장할 때 timestamp를 찍는데, 이를 그대로 사용하여 streaming 할 경우 전송 프로토콜 상에서 timestamp sync 문제가 발생하여, Video Streamer가 독자적인 timestamp를 다시 버퍼에 담아 전송하는 방식으로 변경함.

1. **시험 시나리오**

테스트 시나리오는 Local Dummy Server Test, Unified System Test 두가지로 나뉜다.

1. **Local Dummy Server Test**

메인 프로그램과 통신이 가능한 테스트용 Dummy Server 프로그램들을 제작한다. (구현의 용이성을 위해 기능별로 각각 다른 프로그램으로써 실행되도록 제작한다)

각각의 Dummy Server는 메인 프로그램의 관점에서 예상되는 실제 관리 시스템과 완전히 동일하게 동작해야 한다.

모든 필요한 프로그램들을 실행한 후, Debug Output으로 로직이 제대로

처리가 되는지, Networking에 문제가 없는지 확인한다.

**Unified System Test**

구현된 실제 관리 시스템과 통합하여 테스트하는 시나리오이다. 본 프로그램과 원격지에 있는 관리 시스템의 모든 기능이 온전히 작동하여야 테스트 시나리오를 통과할 수 있다. Configs를 수정하여 해당 관리 서버를 타겟

팅 한 후, 관리 서버가 온전히 작동 중인지 확인하여 접속을 시도한다.

이때 체크해야 하는 사항은 다음 4가지이다.

A. 관리 서버와 정상적으로 연결되었고 올바른 데이터를 전송하고 있는가?

B. 영상 스트리밍의 딜레이가 과도하게 발생하지 않는가?

C. 네트워크가 불안정해도 프로그램의 작동이 멈추지 않는가?

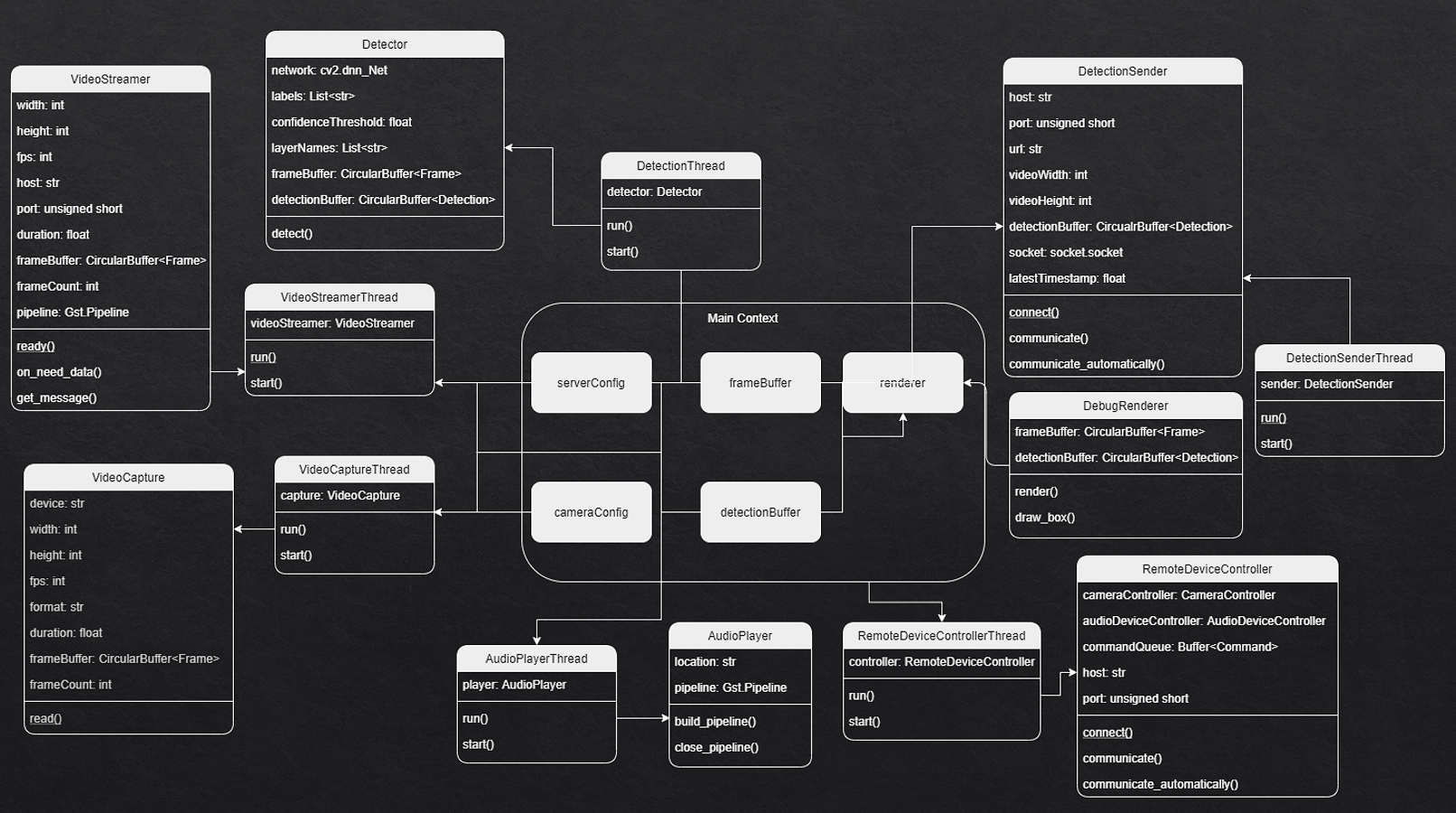
D. Audio 관련 기능이 정상적으로 작동하는가?

Local Dummy Server Test 및 A, B, C, D 기준을 모두 충족하면 모든 테스트

시나리오를 통과했다고 할 수 있다.

1. **상세 설계**

**4.1 시스템 구성도**



프로그램을 실행하면, main() 함수에서 server와 camera에 대한 설정값을 불러오고, frame buffer와 detection buffer를 생성한 뒤, Video Capture Thread, Video Streaming Thread, Detection Thread, Detection Sender Thread, Audio Player Thread, Remote Device Controller Thread를 생성한다.

모든 스레드를 시작시킨 뒤, Render Context를 생성하여 메인 스레드에서 결과값을 Rendering한다.

**4.2 시스템 모듈 상세 설계**

**기반 클래스**

|  |  |
| --- | --- |
| **클래스명** | Buffer |
| **설명** | 가장 기본적인 List 기반의 Buffer이다. |
| **속성명** | rawbuffer |
| **타입** | List<Any> |
| **설명** | 실제 데이터가 들어있는 리스트 객체이다. |
| **속성명** | size |
| **타입** | int |
| **설명** | buffer의 현재 사이즈이다. |
| **메소드명** | \_\_init\_\_ |
| **파라미터** | x |
| **반환타입** | NoneType |
| **설명** | Buffer 클래스의 Initializer |
| **메소드명** | add |
| **파라미터** | data: Any |
| **반환타입** | NoneType |
| **설명** | 데이터를 버퍼에 저장한다. |
| **메소드명** | clear |
| **파라미터** | x |
| **반환타입** | NoneType |
| **설명** | 데이터를 버퍼에 저장한다. |
| **메소드명** | get |
| **파라미터** | index: int |
| **반환타입** | Any |
| **설명** | 버퍼에서 특정 index의 데이터를 가져온다. |
| **메소드명** | pop |
| **파라미터** | index: int |
| **반환타입** | Any |
| **설명** | 버퍼에서 특정 index의 element를 버퍼에서 지우고 값을 반환한다. |
| **메소드명** | insert |
| **파라미터** | index: int |
| data: Any |
| **반환타입** | NoneType |
| **설명** | 버퍼의 특정 index에 element를 삽입한다. |
| **메소드명** | first |
| **파라미터** | x |
| **반환타입** | Any |
| **설명** | 버퍼의 가장 앞에 있는 데이터를 반환한다. |
| **메소드명** | last |
| **파라미터** | x |
| **반환타입** | Any |
| **설명** | 버퍼의 가장 마지막에 있는 데이터를 반환한다. |
| **메소드명** | head |
| **파라미터** | x |
| **반환타입** | Any |
| **설명** | 버퍼의 가장 앞에 있는 데이터를 반환한다. |
| **메소드명** | tail |
| **파라미터** | x |
| **반환타입** | Any |
| **설명** | 버퍼의 가장 마지막에 있는 데이터를 반환한다. |

|  |  |
| --- | --- |
| **클래스명** | StaticTypeBuffer<T> |
| **설명** | 한가지 타입의 element만을 저장하는 버퍼이다. Buffer의 자식 클래스이다.  부모 클래스와 겹치는 속성 및 메소드는 생략한다. |
| **속성명** | elementType |
| **타입** | Any |
| **설명** | 버퍼가 저장하는 element의 Type이다. |
| **메소드명** | \_\_init\_\_ |
| **파라미터** | elementType: Any |
| **반환타입** | NoneType |
| **설명** | StaticTypeBuffer 클래스의 Initializer |

|  |  |
| --- | --- |
| **클래스명** | CircularBuffer |
| **설명** | 원형 큐 인터페이스를 제공하는 버퍼이다. Buffer의 자식 클래스이다.  부모 클래스와 겹치는 속성 및 메소드는 생략한다. |
| **속성명** | maxlen |
| **타입** | int |
| **설명** | 버퍼의 최대 길이를 지정한다. 이 길이를 넘어서면 가장 앞에 있는  element를 삭제한다. |
| **메소드명** | \_\_init\_\_ |
| **파라미터** | maxlen: int |
| **반환타입** | NoneType |
| **설명** | CircularBuffer 클래스의 Initializer |

|  |  |
| --- | --- |
| **클래스명** | StaticTypeCircularBuffer |
| **설명** | 원형 큐 인터페이스를 제공하는 Static Type Buffer이다. StaticTypeBuffer와 CircularBuffer의 자식 클래스이다. Buffer 클래스와 겹치는 속성 및 메소드는 생략한다. |
| **속성명** | maxlen |
| **타입** | int |
| **설명** | 버퍼의 최대 길이를 지정한다. 이 길이를 넘어서면 가장 앞에 있는  element를 삭제한다. |
| **속성명** | elementType |
| **타입** | Any |
| **설명** | 버퍼가 저장하는 element의 Type이다. |
| **메소드명** | \_\_init\_\_ |
| **파라미터** | elementType: Any |
| maxlen: int |
| **반환타입** | NoneType |
| **설명** | StaticTypeCircularBuffer 클래스의 Initializer |

|  |  |
| --- | --- |
| **클래스명** | Config |
| **설명** | ConfigParser로 설정값을 불러와 내부에 저장하는 클래스이다. |
| **메소드명** | \_\_init\_\_ |
| **파라미터** | x |
| **반환타입** | NoneType |
| **설명** | Config 클래스의 Initializer |

|  |  |
| --- | --- |
| **클래스명** | ServerConfigLoader |
| **설명** | ConfigParser로 서버 설정값을 불러와 HTTP 서버, TCP 서버, Streaming 서버의 정보를 Config 클래스로 저장하는 클래스이다. |
| **메소드명** | \_\_init\_\_ |
| **파라미터** | filePath |
| **반환타입** | NoneType |
| **설명** | ServerConfigLoader 클래스의 Initializer |
| **메소드명** | load |
| **파라미터** | filePath |
| **반환타입** | NoneType |
| **설명** | Server Config을 불러와 각각 detection, streaming, http property로 저장한다. |

**핵심 클래스**

|  |  |
| --- | --- |
| **클래스명** | VideoCapture |
| **설명** | 카메라 디바이스로부터 이미지 데이터를 읽어와 frame buffer에 저장하는 클래스이다. 내부적으로 opencv의 VideoCapture 클래스를 사용한다. |
| **속성명** | device |
| **타입** | string |
| **설명** | 카메라 device file의 파일 시스템 상의 경로를 지정한다. |
| **속성명** | width |
| **타입** | int |
| **설명** | 카메라에서 읽어올 이미지의 width |
| **속성명** | height |
| **타입** | int |
| **설명** | 카메라에서 읽어올 이미지의 height |
| **속성명** | fps |
| **타입** | int |
| **설명** | 카메라에서 읽어올 image sequence의 초당 framerate |
| **속성명** | format |
| **타입** | string |
| **설명** | 카메라에서 읽어올 이미지의 color format을 지정한다. |
| **속성명** | duration |
| **타입** | float |
| **설명** | 한 프레임이 몇 초동안 재생되는지를 나타낸다. |
| **속성명** | frameCount |
| **타입** | int |
| **설명** | 현재까지 카메라에서 받아온 이미지의 개수를 나타낸다. |
| **속성명** | framebuffer |
| **타입** | Buffer<Frame> |
| **설명** | 이미지를 저장할 프레임 버퍼를 지정한다. |
| **메소드명** | \_\_init\_\_ |
| **파라미터** | device: str |
| width: int |
| height: int |
| fps: int |
| location: str |
| framebuffer: Buffer<Frame> |
| **반환타입** | NoneType |
| **설명** | VideoCapture 클래스의 Initializer |
| **메소드명** | read |
| **파라미터** | x |
| **반환타입** | NoneType |
| **설명** | 카메라에서 한 프레임을 읽어와 frame buffer에 저장한다. |

|  |  |
| --- | --- |
| **클래스명** | VideoStreamer |
| **설명** | frame buffer로부터 이미지 데이터를 가져와 H264 혹은 H265로 인코딩하여 음성 데이터와 합쳐 관리 서버로 전송한다. Gstreamer의 Pipeline 클래스를 사용하여 RTSP 프로토콜로 영상 데이터를 전송한다. |
| **속성명** | device |
| **타입** | string |
| **설명** | 카메라 device file의 파일 시스템 상의 경로를 지정한다. |
| **속성명** | width |
| **타입** | int |
| **설명** | framebuffer에서 가져올 이미지의 width |
| **속성명** | height |
| **타입** | int |
| **설명** | framebuffer에서 가져올 이미지의 height |
| **속성명** | fps |
| **타입** | int |
| **설명** | framebuffer에서 가져올 image sequence의 초당 framerate |
| **속성명** | location |
| **타입** | string |
| **설명** | 데이터를 전송할 관리 서버의 uri를 지정한다. |
| **속성명** | duration |
| **타입** | float |
| **설명** | 한 프레임이 몇 초동안 재생되는지를 나타낸다. |
| **속성명** | frameCount |
| **타입** | int |
| **설명** | 현재까지 전송한 이미지의 개수를 나타낸다. |
| **속성명** | framebuffer |
| **타입** | Buffer<Frame> |
| **설명** | 이미지를 받아올 frame buffer를 지정한다. |
| **속성명** | pipeline |
| **타입** | Gst.Pipeline |
| **설명** | 영상을 스트리밍하는 gstreamer pipeline이다. |
| **메소드명** | \_\_init\_\_ |
| **파라미터** | width: int |
| height: int |
| fps: int |
| format: str |
| framebuffer: Buffer<Frame> |
| **반환타입** | NoneType |
| **설명** | VideoStreamer 클래스의 Initializer |
| **메소드명** | build\_pipeline |
| **파라미터** | x |
| **반환타입** | NoneType |
| **설명** | streaming에 사용될 gstreamer pipeline을 만든다. |
| **메소드명** | close\_pipeline |
| **파라미터** | x |
| **반환타입** | NoneType |
| **설명** | 현재 사용중인 gstreamer pipeline을 닫는다. |
| **메소드명** | ready |
| **파라미터** | x |
| **반환타입** | NoneType |
| **설명** | video streaming을 시작하기 위해 pipeline을 playing 상태로 전환한다. |
| **메소드명** | on\_need\_data |
| **파라미터** | src: Gst.Element  length: int |
| **반환타입** | NoneType |
| **설명** | 파이프라인을 통해 영상 데이터를 전송하기 위해, 데이터 요청시의 콜백으로써 파이프라인에 연결한다. |
| **메소드명** | get\_message |
| **파라미터** | x |
| **반환타입** | bool |
| **설명** | 파이프라인의 상태를 확인하여 에러 시 재시작을 위해 timeout callback으로써 주기적으로 호출하는 함수이다. |

|  |  |
| --- | --- |
| **클래스명** | ObjectDetector |
| **설명** | framebuffer로부터 이미지 데이터를 가져와 Neural Network를 사용하여 object detection을 수행한다. 수행 결과는 frame의 time stamp와 함께 detection buffer에 저장한다. |
| **속성명** | network |
| **타입** | cv2.dnn\_Net |
| **설명** | 실제 object detection을 수행하는 network 객체이다. |
| **속성명** | labels |
| **타입** | List<str> |
| **설명** | object class labeling을 위한 label list이다. index는 classID를 의미한다. |
| **속성명** | confidenceThreshold |
| **타입** | float |
| **설명** | bounding box 후보를 선택하기 위한 confidence의 threshold이다. |
| **속성명** | layerNames |
| **타입** | List<str> |
| **설명** | output layer의 layer name list이다. |
| **속성명** | frameBuffer |
| **타입** | Buffer<Frame> |
| **설명** | detection을 수행하기 위한 image data를 가져올 buffer이다. |
| **속성명** | detectionBuffer |
| **타입** | Buffer<Detection> |
| **설명** | detection result를 저장하기 위한 buffer이다. |
| **메소드명** | \_\_init\_\_ |
| **파라미터** | labelFileName: str |
| configFileName: str |
| weightsFileName: str |
| confidenceThreshold: float |
| framebuffer: Buffer<Frame> |
| detectionbuffer: Buffer<Detection> |
| **반환타입** | NoneType |
| **설명** | ObjectDetector 클래스의 Initializer |
| **메소드명** | detect |
| **파라미터** | x |
| **반환타입** | NoneType |
| **설명** | frame buffer에서 한 프레임을 가져와 detection을 수행하여 detection buffer에 저장한다. |

|  |  |
| --- | --- |
| **클래스명** | DetectionSender |
| **설명** | Detection data를 관리 서버에 전송하는 역할을 한다. 서버 상태에 상관없이 항시 작동해야 하므로 연결이 끊어지면 일정 시간마다 연결을 재시도한다. |
| **속성명** | host |
| **타입** | str |
| **설명** | 관리 서버의 host ip 주소를 지정한다. |
| **속성명** | port |
| **타입** | unsigned short |
| **설명** | TCP 서버의 port number를 지정한다. |
| **속성명** | url |
| **타입** | str |
| **설명** | 관리 서버의 HTTP url을 지정한다. |
| **속성명** | videoWidth |
| **타입** | int |
| **설명** | VideoStreamer가 전송하는 video의 width이다. |
| **속성명** | videoHeight |
| **타입** | int |
| **설명** | VideoStreamer가 전송하는 video의 height이다. |
| **속성명** | detectionBuffer |
| **타입** | Buffer<Detection> |
| **설명** | detection data를 가져올 buffer를 지정한다. |
| **속성명** | reconnectInterval |
| **타입** | float |
| **설명** | 몇 초마다 재접속을 시도할 지를 지정한다. |
| **메소드명** | \_\_init\_\_ |
| **파라미터** | host: str |
| port: unsigned int |
| url: str |
| videoWidth: int |
| videoHeight: itn |
| detectionbuffer: Buffer<Detection> |
| reconnectInterval: float |
| **반환타입** | NoneType |
| **설명** | DetectionSender 클래스의 Initializer |
| **메소드명** | connect |
| **파라미터** | x |
| **반환타입** | NoneType |
| **설명** | 관리 서버에 TCP 프로토콜 접속을 시도한다. 접속 실패시 일정 시간마다 재접속을 시도한다. |
| **메소드명** | communicate |
| **파라미터** | x |
| **반환타입** | NoneType |
| **설명** | 관리 서버에 접속 한 후에 실행하는 메소드이다. 관리 서버와 TCP 통신을 수행하며, 처음에 Camera Identification을 수행한 뒤 계속해서 detection buffer에서 데이터를 가져와 전송한다. last timestamp를 기록해 현재 buffer의 가장 앞에 있는 데이터의 timestamp와 비교해 작거나 같으면 전송하지 않고, 크다면 전송하고 last timestamp를 업데이트한다. |
| **메소드명** | communicate\_automatically |
| **파라미터** | x |
| **반환타입** | NoneType |
| **설명** | 서버에 접속을 시도하고 communicate 하는 과정을 자동으로 수행한다. 만약 연결이 끊어지면 다시 connect 함수를 호출하여 계속해서 재접속을 시도한다. |

|  |  |
| --- | --- |
| **클래스명** | DebugRenderer |
| **설명** | Debuging을 위한 클래스이다. OpenGL을 이용하여 image data와 detection result를 조합하여 화면에 출력한다. |
| **속성명** | frameBuffer |
| **타입** | Buffer<Frame> |
| **설명** | Frame data를 가져올 frame buffer이다. |
| **속성명** | detectionBuffer |
| **타입** | Buffer<Detection> |
| **설명** | detection data를 가져올 buffer이다. |
| **메소드명** | \_\_init\_\_ |
| **파라미터** | frameBuffer: Buffer<Frame> |
| detectionBuffer: Buffer<Detection> |
| **반환타입** | NoneType |
| **설명** | DebugRenderer 클래스의 Initializer |
| **메소드명** | draw\_box |
| **파라미터** | image: numpy.ndarray  box: DetectionBox |
| **반환타입** | numpy.ndarray |
| **설명** | numpy ndarray로 표현되어 있는 image data에 detection result인 bounding box, label, confidence를 입힌다. |
| **메소드명** | render |
| **파라미터** | x |
| **반환타입** | NoneType |
| **설명** | 실제로 OpenGL을 사용하여 렌더링을 수행하는 함수이다. 일정 시간마다 호출되며 OpenGL frontend library인 pyglet을 사용하여 image에 detection result를 입힌 결과를 렌더링한다. |

|  |  |
| --- | --- |
| **클래스명** | AudioPlayer |
| **설명** | 관리 서버로부터 받은 Audio Input을 재생하는 클래스이다. gstreamer pipeline을 사용해 audio data를 받아온다. |
| **속성명** | location |
| **타입** | str |
| **설명** | audio input을 받아올 서버의 uri 주소를 지정한다. |
| **속성명** | pipeline |
| **타입** | Buffer<Frame> |
| **설명** | Frame data를 가져올 frame buffer이다. |
| **메소드명** | \_\_init\_\_ |
| **파라미터** | location |
| **반환타입** | NoneType |
| **설명** | AudioPlayer 클래스의 Initializer |
| **메소드명** | build\_pipeline |
| **파라미터** | x |
| **반환타입** | NoneType |
| **설명** | streaming에 사용될 gstreamer pipeline을 만든다. |
| **메소드명** | close\_pipeline |
| **파라미터** | x |
| **반환타입** | NoneType |
| **설명** | 현재 사용중인 gstreamer pipeline을 닫는다. |

|  |  |
| --- | --- |
| **클래스명** | RemoteDeviceController |
| **설명** | 관리 서버와 TCP 통신을 수행해 제어 데이터를 받아와 command queue에 넣고, queue에 들어있는 명령을 하나씩 꺼내 디바이스를 제어하는 클래스이다. |
| **속성명** | cameraController |
| **타입** | CameraController |
| **설명** | 카메라를 제어하는 Device Controller이다. |
| **속성명** | AudioDeviceController |
| **타입** | Buffer<Frame> |
| **설명** | Speaker, Microphone 등을 제어하는 Device Controller이다. |
| **속성명** | commandQueue |
| **타입** | Buffer<str> |
| **설명** | command가 저장되어있는 queue이다. |
| **속성명** | host |
| **타입** | str |
| **설명** | 관리 서버의 host ip를 지정한다.ㄴ |
| **속성명** | port |
| **타입** | unsigned short |
| **설명** | 관리 서버와 통신을 수행할 포트 번호를 지정한다. |
| **메소드명** | connect |
| **파라미터** | x |
| **반환타입** | NoneType |
| **설명** | 관리 서버에 TCP 프로토콜 접속을 시도한다. 접속 실패시 일정 시간마다 재접속을 시도한다. |
| **메소드명** | communicate |
| **파라미터** | x |
| **반환타입** | NoneType |
| **설명** | 관리 서버에 접속 한 후에 실행하는 메소드이다. 관리 서버와 TCP 통신을 수행하며, 제어 명령어 스트림을 받아와 controller로 전달한다. |
| **메소드명** | communicate\_automatically |
| **파라미터** | x |
| **반환타입** | NoneType |
| **설명** | 서버에 접속을 시도하고 communicate 하는 과정을 자동으로 수행한다. 만약 연결이 끊어지면 다시 connect 함수를 호출하여 계속해서 재접속을 시도한다. |