**종합설계 프로젝트 수행 보고서**

|  |  |
| --- | --- |
| **프로젝트명** | **지능형 카메라를 이용한 요양원 상태 분석 시스템** |
| **팀번호** | **S3-7** |
| **문서제목** | **수행계획서( O )**  **2차발표 중간보고서( O )**  **3차발표 중간보고서( O )**  **최종결과보고서( O )** |

**2021.11.08**

**팀원 : 김주영 (팀장)**

**김민규**

**박세경**

**지도교수 : 전광일 교수**

**문서 수정 내역**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **작성일** | **대표작성자** | **버전(Revision)** | **수정내용** |  |
| 2020.12.18 | 김주영(팀장) | 1.0 | 수행계획서 | 최초작성 |
| 2020.03.08 | 김주영(팀장) | 2.0 | 2차발표자료 | 설계서추가 |
| 2020.05.06 | 김주영(팀장) | 3.0 | 3차발표자료 | 프로토타입 추가 |
| 2020.06.16 | 김주영(팀장) | 4.0 | 최종발표자료 | 코드와 데모 추가 |

**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 “지능형 카메라를 이용한 요양원 상태 분석 시스템” 을 수행하는**

**(S3-7, 김주영, 김민규, 박세경)들이 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. 시험시나리오

4. 상세 설계

5. Prototype 구현

6. 시험/ 테스트 결과

7. Coding & DEMO

**Ⅲ. 결론**

1. 연구 결과

2. 작품제작 소요재료 목록

참고자료

**Ⅰ. 서 론**

**1. 작품선정 배경 및 필요성**

대한민국은 세계에서 유례가 없을 정도로 고령화가 급속하게 진행되고 있는 국가라고 할 수 있다. 1999년에는 노인 비율이 7%를 넘어서 고령화 사회가 되었고 2017년에는 노인 비율이 14%를 넘어서 고령사회가 되었다. 1970년부터 2010년까지 대한민국의 노인 인구 증가율은 OECD 국가 중에서 가장 높다. 또한 최근에는 코로나로 인한 팬데믹 상황에 들어서며 돌봄이 필요한 노인의 수가 지속적으로 증가하고 있음을 알 수 있다.

2020년 들어 베이비붐 세대가 고령층으로 진입하면서 고령 인구 비율이 급등하고 있다. 2019년 8월 15.2% 에서 2020년 6월 16.0%로 거의 한 달에 0.1%p씩 증가하고 있다.

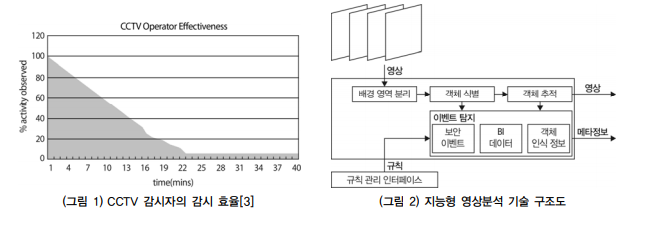
이에 따라 요양원의 입소 희망자는 늘어가고 있다. 최근 7년간 요양원의 수는 4배 이상 증가하였다. 하지만, 현재 노인 요양 시설 간호 인력 현황을 보면 현재 간호 인력의 합계는 9,278명으로 늘어나는 노인 비율에 비해 턱없이 부족하고 간호 인력 1인당 80명의 노인을 감당해야 하는 실상이다. 이에 따라 노인 요양 시설의 노동을 줄이기 위한 시스템이 필요하다.

요양원에서 가장 많이 발생하는 유형의 사건 사고는 낙상, 추락 등의 미끄러짐이다. 이러한 문제를 1차로 방지하기만 해도 간호 인력의 일을 줄일 수 있다. 이에 따라 요양원에서의 지능형 CCTV의 필요성이 제고되고 지능형 CCTV로 인해 빠른 사고 관리와 간호 인력의 일의 효율을 증대 시킬 수 있다.

**2. 기존 연구/기술동향 분석**

현재 요양원은 CCTV의 설치가 의무화가 되어있지 않고 CCTV가 설치되더라도 일반 카메라형 CCTV를 사용하여 녹화기능만을 제공하고 있다. 따라서 요양원을 위한 지능형 CCTV는 현재 상용화되어 있지 않다. 현재 보건복지부에서 요양원 CCTV 모니터링 제도와 CCTV 설치 가이드 라인을 마련하도록 권고하고 있다.

일반적인 지능형 CCTV의 연구는 활성화되어 있다. 특히 교통이나 길거리 범죄를 예방하기 위한 지능형 CCTV연구가 활성화되어 있다.



2011년 경찰청 자료를 보면 서울 지역 경찰서 CCTV 관제센터에 근무하는 감시 인력은 1인당 평균 45대를 모니터링하는 것으로 조사된다. (그림 1)에서 보듯 2대 이상의 CCTV를 감시하는 사람의 경우에는 12분이 지나면 위험 상황의 45%, 22분이 지나면 위험 상황의 95%를 놓치는 것을 확인할 수 있다. 지능형 영상분석 기술은 영상분석을 통하여 사전 정의된 이벤트가 발생될 때 감시자에게 경보를 생성하여 알려주기 때문에, 모든 영상을 24시간 365일 쉼 없이 감시하지 않고, 경보가 발생할 때 해당 화면을 보고 실시간으로 상황을 판단하고 대처하면 된다.

지능형 영상분석 기술이란 영상의 정보를 분석하여 자동으로 이상 행위를 탐지하는 기술로써, 일반적인 구조는 (그림 2)와 같이 배경 영역 분리 단계, 객체 식별단계, 객체 추적 단계 및 사전 정의된 규칙을 기반으로 이벤트를 탐지하는 이벤트 탐지 단계로 구성된다.

이벤트 탐지 단계에서 이벤트 유형으로 보안 이벤트, BI이벤트, 객체 인식이 있다. 보안 이벤트는 영상에서 객체의 움직임, 상태 정보를 분석하여 사용자가 정의한 규칙에 위반되는 행위를 탐지하는 유형이고 BI이벤트는 영상에서 분석을 통하여 비즈니스에 유용한 통계 정보를 생성하는 것이다. 객체 인식은 영상에서 객체를 검출하고 인식하여 객체의 식별 정보를 생성하는 유형이다.

사람의 행동인식에서 인식하는 동작은 신체 일부의 움직임을 의미하는 제스처, 한 사람에 대한 다양한 제스처의 조합으로 정의되는 액션, 두 객체 사이에서 발생되는 행동인 인터랙션, 다수의 객체로 구성된 그룹에서 발생되는 그룹 행동으로 구분될 수 있다. 현재 사람의 행동인식에 대한 접근 방법은 싱글 레이어드 기반의 방법과 계층적 기반의 방법으로 구분할 수 있다. 싱글 레이어드 기반의 방법은 영상에서 사람의 행동을 직접 구분하는 방법으로써 상대적으로 간단하고 짧은 사람의 동작을 인식하는 것이 목적이기 때문에 걷기, 뛰기 등의 비교적 단순한 제스처, 액션 등의 행동인식에 주로 사용된다. 계층적 기반의 방법은 상대적으로 인식하기 쉬운 단순한 행동 정보를 인식한 후, 행동 정보들을 조합함으로써 고수준의 행동을 판단하는 방법이기 때문에 복잡한 사람의 행동을 인식하는 데 주로 사용된다. 예를 들어 싸움과 같은 행동의 경우 영상에서 연속적인 펀칭과 키킹의 행위를 탐지함으로써 인식할 수 있다.

또한 관리자가 설정한 규칙을 회피하는 경로가 존재하는 경우에는 탐지할 수 없는 단점이 있는데 이 부분은 영상에서 일어나는 객체들의 행동 패턴을 학습하여 행위 기반으로 이상 현상을 탐지하는 방법으로 행위 기반의 이상 현상 탐지로 분류한다.

앞으로 지능형 CCTV는 기존의 규칙 기반 이벤트 탐지 방법에서 벗어나 사람의 개입 없이 행위 기반의 이벤트 탐지 방법이 주류로 등장할 것이다. 그리고 군중들이 움직이는 환경에서 영상의 전역 정보를 분석하여 이상 행위를 인식하는 이벤트 탐지 기술이 시장에 보급될 것이다. 또한 기존의 서버/에지 기반의 지능형 영상인식 구조에서 벗어난 새로운 구조를 가지는 기술들이 속속 개발될 것이다.

**3. 개발 목표**

위에 언급한 연구 개발 배경에 언급했듯 노인의 수와 요양원의 수는 지속적으로 증가하는 반면, 요양원에서는 간호인력이 환자의 수에 비해 턱없이 부족하고, 이에 따라 돌봄 서비스를 효율적으로 제공하지 못한다는 문제점이 있음을 파악하였다. 따라서 요양원에서 자주 일어나는 낙상과 실신을 자동으로 탐지하는 지능형 CCTV를 개발하고자 한다.

인간의 행동을 자동으로 탐지하는 지능형 CCTV 개발이 이뤄지면 1명당 80명의 환자를 돌봐야 하는 간병인의 부담이 줄어들어 요양원 서비스를 효율적으로 제공할 수 있다. 간병인은 1차적 상황 판단을 쉽게 할 수 있는데, 1차적 상황 판단이란 CCTV가 환자가 실신하거나 침대에서 낙상, 사람들 간의 폭력이 일어나는 것을 감지하여 클라이언트 앱으로 알림을 제공함으로써 구체적인 상황 판단은 사용자에게 맡긴다는 의미에서 1차적 상황 판단이라 한 것이다.

지능형 CCTV의 효과적인 감시 시스템은 다양한 방식으로도 또한 이용될 수 있다. 객체의 정확한 탐지를 통해 미아 식별, 위험인물 색출 등을 사람이 24시간 수동적으로 모니터링할 필요 없이 할 수 있다. 또한 더 나아가 행동과 상황 판단까지 이뤄진다면 요양시설보호에 사용될 수 있고, 위험 행동에 대해 효과적으로 대응할 수 있다.

**4. 팀 역할 분담**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 김민규 | 김주영 | 박세경 |
| 자료수집 | - HLS  - 라이브 인코더 | - YOLO v5  - 행동 분석 알고리즘 | - CDN  - 영상 분석 서버 |
| 설계 | 하드웨어 설계 | 행동 알고리즘 설계 | 서버 설계 |
| 구현 | - 라즈베리파이 카메라 모듈 제어 구현 | - 스트리밍 영상 🡪 분석 시스템 구현 | - 클라이언트 앱 구현  - 인터페이스 매커니즘 |
| 테스트 | - 장비 작동/제어 테스트  - 소프트웨어 작동/제어 테스트  - 통합테스트/ 유지보수 | | |

**5. 개발 일정**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 항목 | 12 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 요구사항 정의 및 분석 |  |  |  |  |  |  |  |  |  |  |  |
| 요구사항 명세 |  |  |  |  |  |  |  |  |  |  |  |
| 시스템 설계 |  |  |  |  |  |  |  |  |  |  |  |
| 상세 설계 |  |  |  |  |  |  |  |  |  |  |  |
| 코딩 |  |  |  |  |  |  |  |  |  |  |  |
| 시스템 통합 시험 |  |  |  |  |  |  |  |  |  |  |  |
| 졸업작품 완전성 보강 |  |  |  |  |  |  |  |  |  |  |  |
| 졸업작품 중간 보고서 작성 |  |  |  |  |  |  |  |  |  |  |  |
| 발표 |  |  |  |  |  |  |  |  |  |  |  |
| 산업 기술 대전 참가 |  |  |  |  |  |  |  |  |  |  |  |
| 졸업작품 최종 보고서 작성 |  |  |  |  |  |  |  |  |  |  |  |
| CD 패키징 |  |  |  |  |  |  |  |  |  |  |  |

**6. 개발 환경**

|  |  |
| --- | --- |
| **구분** | **개발 환경** |
| **H/W** | ⦁ 라즈베리 파이 4 B |
| ⦁ 라즈베리파이 카메라모듈 V2, 8MP |
| **S/W** | ⦁ Python |
| ⦁ Android Studio (Kotlin) |
| ⦁ Openpose |
| **Server** | ⦁ Flask |
| ⦁ MJPG-Streamer |
| **Database** | ⦁ Firebase |

**Ⅱ. 본 론**

**1. 개발 내용**

|  |  |
| --- | --- |
| 구성 | 내용 |
| S/W | ● 영상 스트리밍  → RTSP서버를 이용한 데이터 스트리밍  → DROID CAM을 이용하여 핸드폰의 카메라화 및 데스크 탑으로 영상 전송  ● 영상 분석  → 스트리밍 받은 영상을 OPENPOSE를 사용하여 행동인식  → 낙상 관련 자세 학습  ● 서버  → FLASK를 사용하여 웹 서버 구현  → 웹 서버에서 실질적인 프로그램 동작  ● 클라이언트  → ANDROID APP을 통해 FLASK 웹 서버와 통신  ● 데이터베이스  → 영상의 데이터를 저장 |
| H/W | ● 카메라  → 라즈베리파이 V4, PI CAM, ANDROID 기반 핸드폰 |

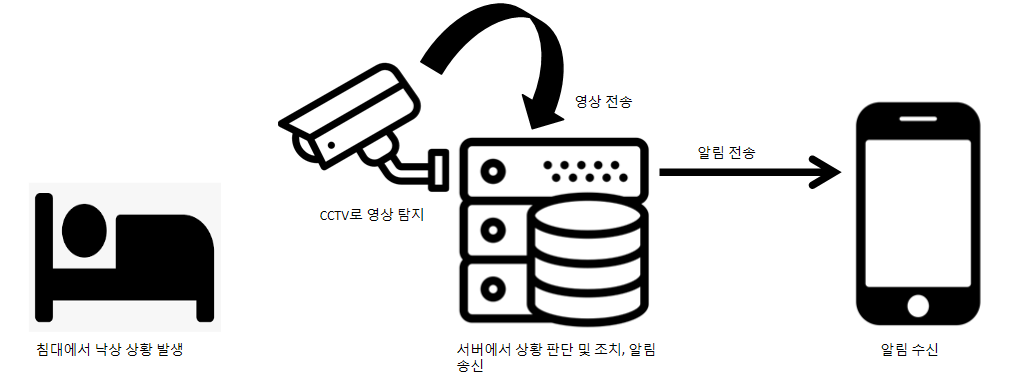
**2. 문제 및 해결방안**

- 실시간 스트리밍을 통해 CCTV에서 인간의 행동을 인식하는 딥 러닝 서버가 필요하다. 이상 행동을 파악하기 위해 인간의 관절점을 학습시켜 스켈레톤 구조를 뽑아내고, 이 스켈레톤 구조를 활용하여 인간의 행동을 분류할 수 있다.

- 스켈레톤을 추출할 때 한 명의 스켈레톤 정보만 추출됨을 파악하여 이를 해결하기 위해 최소 관절 점 거리를 계산하는 함수를 추가하여 다수의 스켈레톤 정보도 추출할 수 있도록 하였다.

- 인간의 자세가 파악되면 낙상이 발생할 것 같은 상황이 연출된 경우 동영상 정보와 동영상 클립을 실시간 데이터베이스에 저장하고 클라이언트 앱으로 알림을 보내 사용자가 백그라운드에서 항상 확인할 수 있도록 한다.

**3. 시험 시나리오**

****

- 환자의 침대에서의 낙상에 대한 상황이 발생하였을 시 CCTV로 실시간 영상을 분석해 상황을 판단하고 낙상 방지를 위한 시스템이 작동한다. 또한 방지 시스템이 작동하면 CCTV로 스트리밍 중인 화면을 캡쳐 하여 확인할 수 있도록 한다. 이 때, 낙상에 대한 상황을 방지하지 못하거나 보호자에게 알려야 한다고 판단되면 알림을 전송한다.

**4. 상세 설계**

**1) 영상 스트리밍**

① 기능

◦ 영상 촬영 하드웨어를 사용하여 영상을 촬영한 후 실시간으로 스트리밍

② 다루는 정보

◦ 실시간 CCTV환경을 구현하기 위한 스트리밍 중인영상 정보를 다룸

◦ IP카메라 및 라즈베리 파이,휴대전화 등을 카메라로 사용하기 위한 IP정보 및 기술

◦동영상의 FPS, 화면 크기 등의 설정 사항

③ 하드웨어 정보

◦라즈베리 파이 v4, 파이 카메라, 안드로이드 기반 휴대전화

④ 고려사항

◦동영상에 대한 분석이 필요하기 때문에 높은 해상도의 영상 정보 필요

◦동영상에 대한 실시간 행동분석이 목적이므로 빠른 스트리밍 영상 필요

⑤ 카메라 하드웨어 선정 대상

◦ 라즈베리 파이 카메라

◦ MJPG-Streamer를 사용하여 영상 스트리밍

◦ MJPG-Streamer를 통해 Flask서버에서의 코드 작성이 용이함

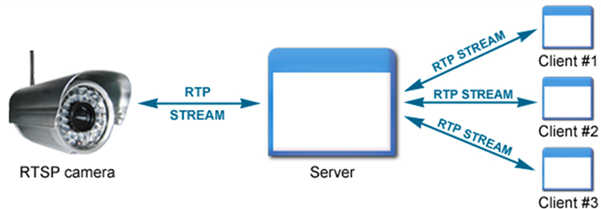
◦ 속도가 현저하게 떨어지는 문제점이 있음

◦ IP카메라

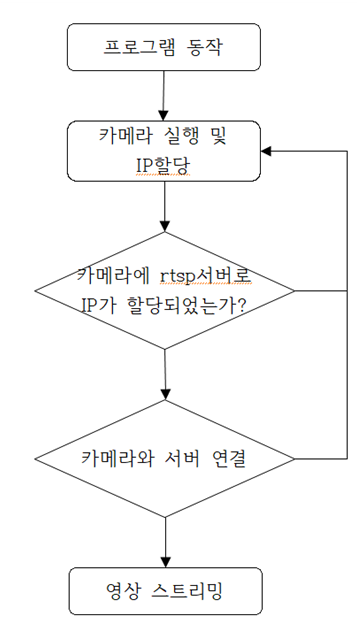
◦RTSP서버를 사용하여 핸드폰을 데스크탑의 영상입력 장치로 사용하게 해주는Droid cam App이용

◦ 고화질의 영상과 빠른 속도를 제공함

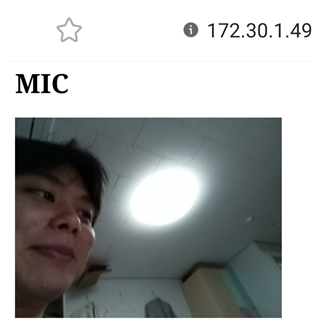
⑥ 비디오 스트리밍 구조, 비디오 스트리밍 서버



⑦ 순서도



⑧ 실행 화면



**2) 영상 분석**

① 기능

◦ 스트리밍으로 받은 영상 정보에서 사람의 관절을 추출

◦ 추출된 관절 정보를 토대로 Openpose모델에 따라 사람의 뼈대 형성

◦ 행동 분석을 하기 전 사람의 행동을 인식하는 과정

② 다루는 정보

◦ 사람의 행동에 따른 뼈대 정보

◦ 다수의 사람을 구분하는 정보

③ 고려 사항

◦ 사람을 detection하고 포즈를 추정하는 top-down 방식에서 키포인트 간의 상관관계를 분석하여 포즈를 추정하는 bottom-up 방식을 선택

◦ Top-down방식은 사람의 수에 따라 시간이 기하급수적으로 늘어나기 때문

④ 사용 알고리즘

◦ Openpose 모델 중 coco 모델을 사용

⑤ 함수

* getKeyPoints()

◦스트리밍으로 받은 영상 정보에서 사람의 관절을 추출

◦추출된 관절 정보를 토대로 Openpose모델에 따라 사람의 뼈대 형성

◦행동 분석을 하기 전 사람의 행동을 인식하는 과정

* getValidPairs()

◦스트리밍으로 받은 영상 정보에서 사람의 관절을 추출

◦추출된 관절 정보를 토대로 Openpose모델에 따라 사람의 뼈대 형성

◦행동 분석을 하기 전 사람의 행동을 인식하는 과정

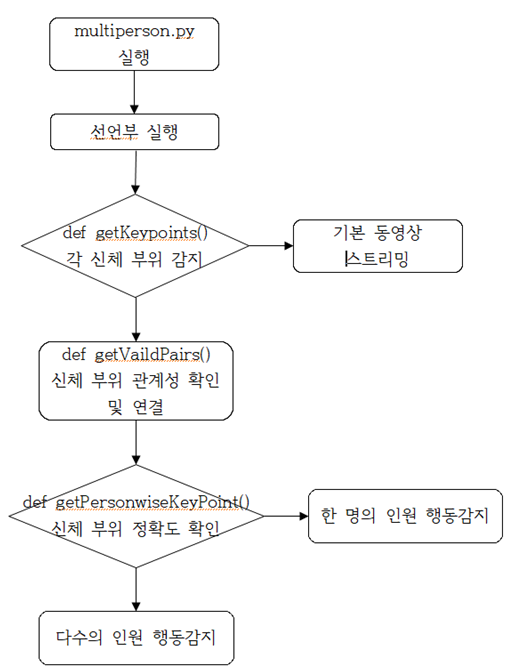
* getPersonwiseKeyPoints()

◦스트리밍으로 받은 영상 정보에서 사람의 관절을 추출

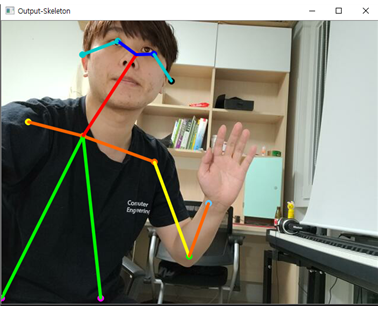
◦추출된 관절 정보를 토대로 Openpose모델에 따라 사람의 뼈대 형성

◦행동 분석을 하기 전 사람의 행동을 인식하는 과정

⑥ 순서도



⑦ 실행 화면



**3) 상황 판단**

① 기능

◦ 영상 분석 모듈에서 추출한 정보를 바탕으로 환자의 위험성을 판단함

◦ 낙상에 대한 자세를 판단

② 다루는 정보

◦ 영상 분석 단계에서 분석한 객체의 뼈대 정보

◦ 낙상 클래스

|  |  |
| --- | --- |
| **행동 클래스** | **설명** |
| Fall (쓰러짐) | 서 있는 상태에서 바닥에 눕는 상태가 되는 상황 |

③ 고려 사항

◦ 낙상에 대한 자세

* openpose의 스켈레톤과 math함수를 사용하여 계산

**4) 알림 전송**

① 기능

◦ 상황 판단 모듈에서 판단한 결과를 바탕으로 낙상 가능성이 있는 상황이 연출될 경우 User Application 으로 알림을 전송

② 다루는 정보

◦ 상황 판단 모듈에서의 판단 결과 및 현재 상황의 정보

③ 고려 사항

◦ 알림을 수신하는 사용자의 앱이 사용 중이 아닌 상태에서도 전송이 가능해야 함

**5) 알림 수신**

① 기능

◦ 딥 러닝 서버로부터 이상 행동이 감지되면 Flask 웹 서버는 데이터베이스에 상황 판단에 대한 결과를 기록 : Firebase의 Realtime Database

◦ 안드로이드 클라이언트에게 낙상 가능성을 감지했다는 알림을 보냄 : Firebase의 Cloud Messaging

② 고려 사항

◦ 애플리케이션이 종료되거나 슬립 상태에서도 사용자가 언제든지 알림을 받을 수 있어야 함

**6) 영상 시청**

① 기능

◦ 사용자가 실시간 스트리밍 중인 영상을 확인할 수 있다.

- 클라이언트로부터 영상 시청 요청이 오면 데이터베이스로부터 동영상을 전달 받음

- 사용자 정보를 확인한 후 Firebase의 Storage에 저장된 동영상 리스트를 클라이언트 앱으로 가져옴

② 고려 사항

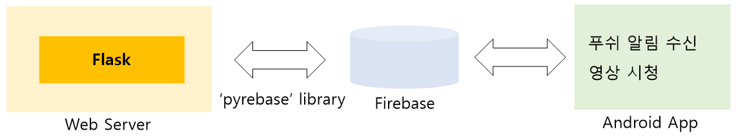
◦ 환자에 대한 영상이므로 사용자에 대한 인증 절차가 필요

◦ 사용자로 하여금 시청이 발생하는 경우는 2가지가 존재

- 실시간으로 낙상 가능성을 판단하여 사용자에게 알림이 온 시점의 동영상 클립을 시청하는 경우

- 사용자가 임의적으로 Storage에 저장된 전체 동영상 리스트를 시청하는 경우

③ 구조도

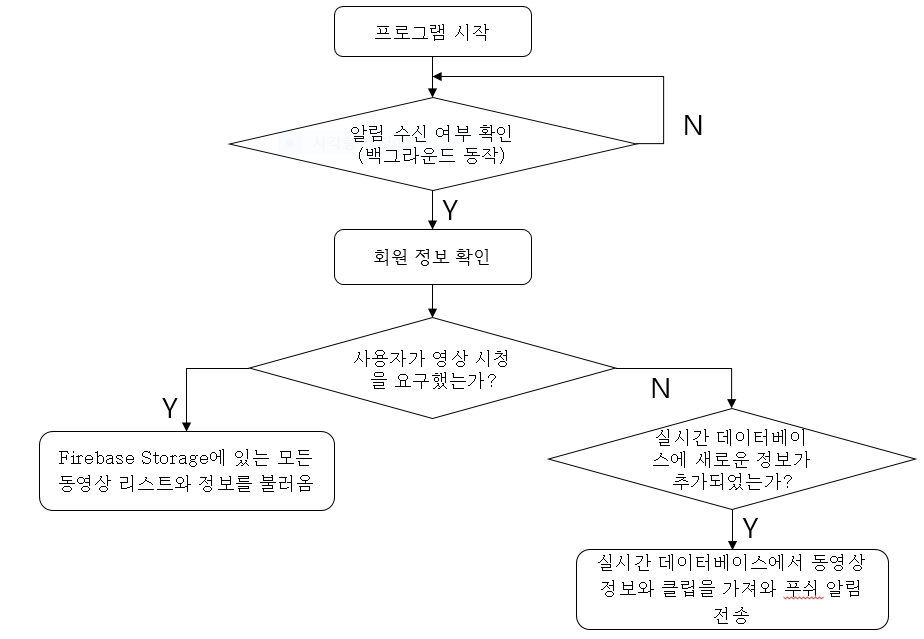


◦ Flask 웹 서버와 Firebase 간의 통신을 위해 REST API를 사용

◦ 안드로이드 앱과 Flask 웹 서버간 통신은 Retrofit 사용

◦ Firebase와 Flask 웹 서버간 통신은 pyrebase 라이브러리를 활용

④ 순서도



**7) 데이터베이스**

① 기능

◦ 상황 판단에 대한 결과와 동영상 정보를 데이터베이스에 저장한다.

◦ 영상 시청에 이용하는 사용자 인증에 대한 정보를 저장한다.

② 다루는 정보

◦ 동영상 정보

- 동영상이 저장된 경로

- 사용자 아이디

- 촬영 날짜 및 촬영 장소

◦ 사용자 정보

- 사용자 이름

- 사용자 아이디, 비밀번호

- 사용자 이메일

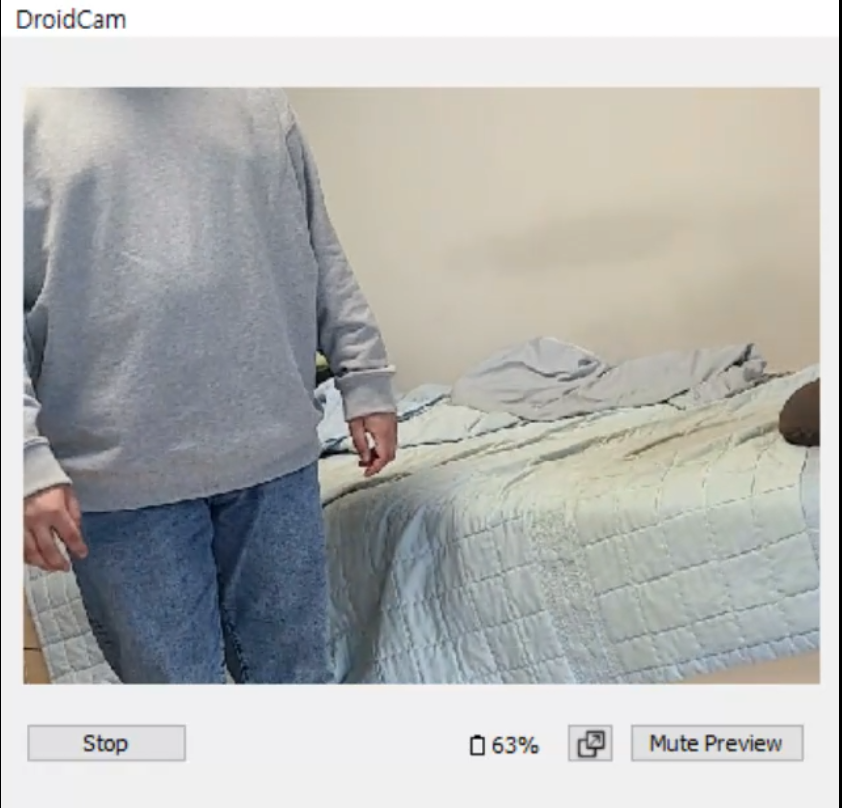
**5. Prototype 구현**

* 데이터 스트리밍

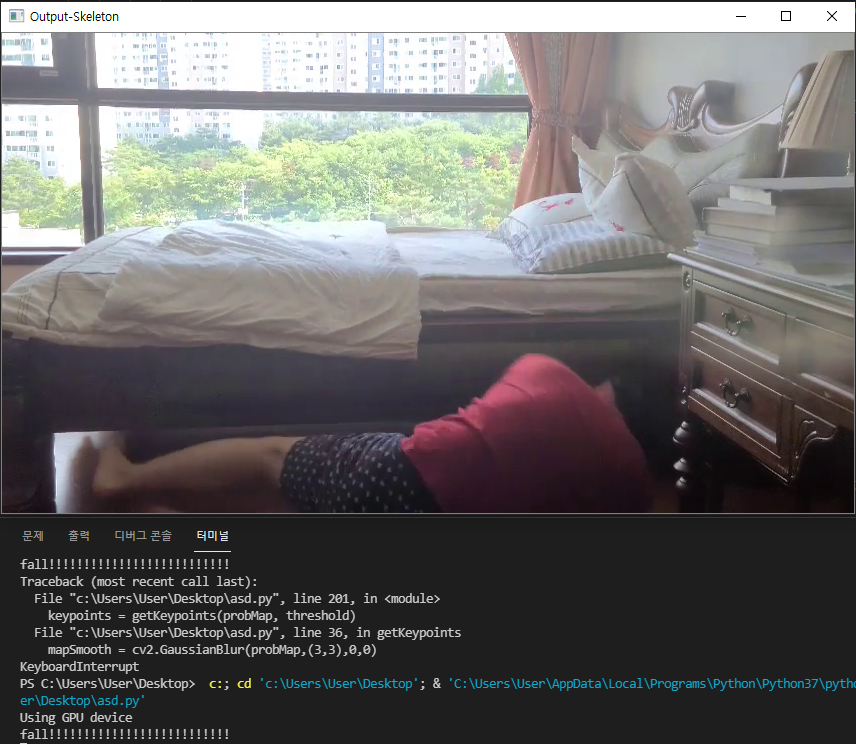
(카메라 촬영, 송신부)



(데스크탑, 수신부)

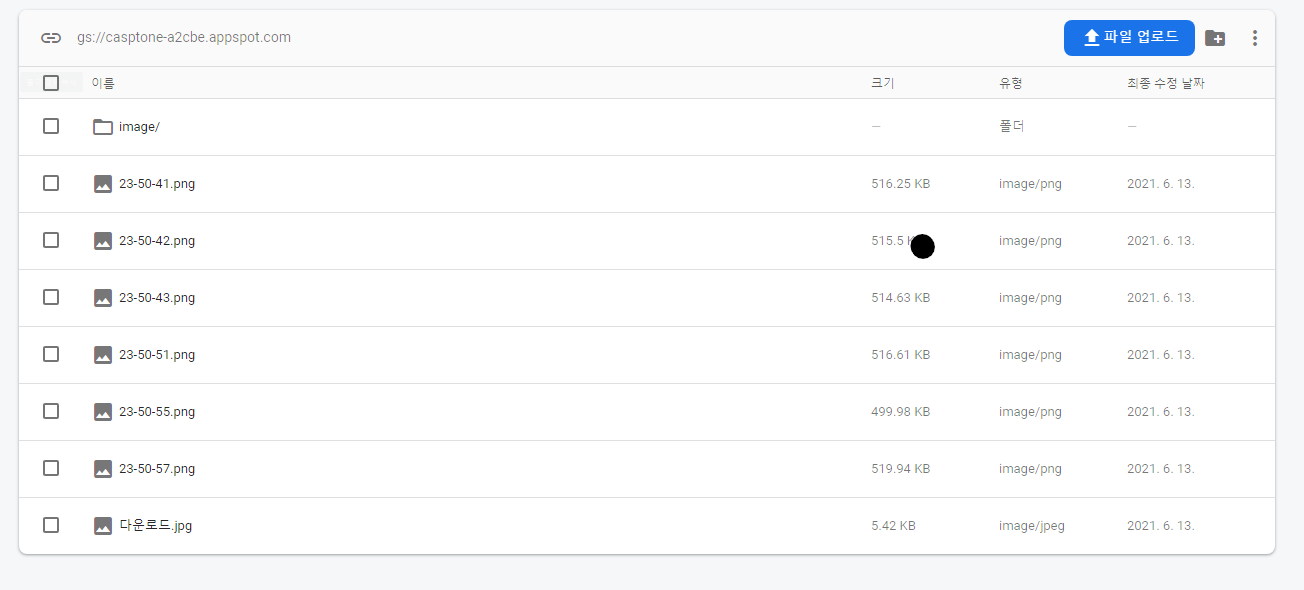


* 모바일 어플 중 핸드폰을 IP카메라처럼 사용할 수 있는 droidcam을 사용하여 촬영 후 RTSP서버 주소를 할당하는 코드를 결합하여 서버가 운영되는 데스크 탑으로 전송
* 낙상 가능성 감지 웹 서버 구현



- 낙상 예방 감지 시스템은 파이썬 기반의 플라스크(Flask) 웹 서버에서 구현

- 기능: 사용자의 스켈레톤을 추출하여 낙상 가능성을 감지한다. 낙상 가능성이 감지된 경우 웹서버의 지정된 경로로 CCTV 캡쳐본이 저장된다.



- pyrebase 라이브러리를 사용하여 Firebase와의 연동을 통해 데이터베이스에 CCTV의 영상/사진을 저장하여 관리

* 클라이언트 앱



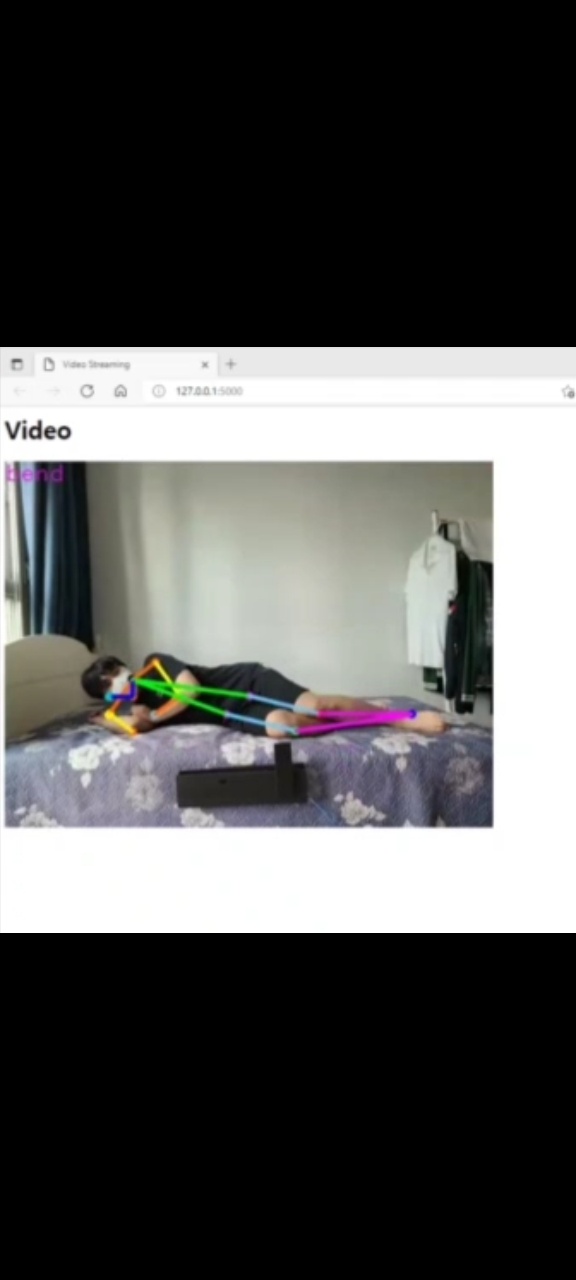
- 기능 : 낙상 시 저장된 화면을 확인할 수 있다.

- 기능: Retrofit 라이브러리를 활용하여 인터넷을 통해 플라스크 서버에 접근 후 저장된 사진을 사용자 어플리케이션으로 가져온다.

**6.시험테스트 결과**

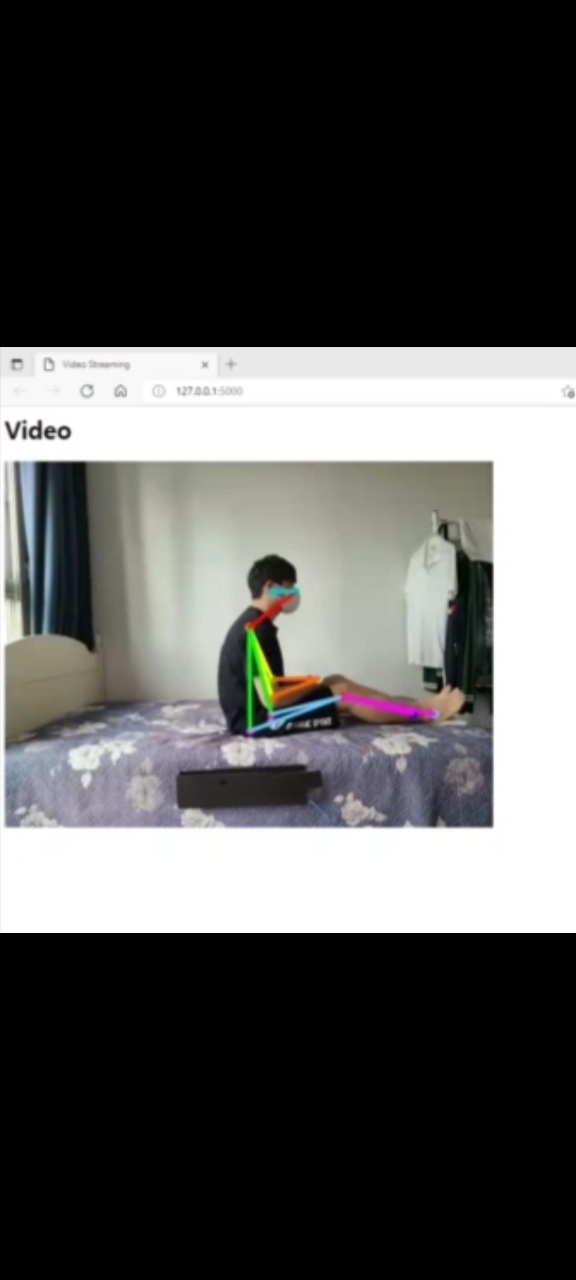
* 난간 작동

(낙상 위험 시)



* 낙상에 대한 위험이 판단되면 난간이 작동
* 시리얼 통신을 이용하여 아두이노와 위험에 대한 통신
* 위험상태일 경우 1을 전송

(낙상 위험 아닐 시)



* 낙상에 대한 위험이 판단되지 않으면 난간이 작동하지 않음
* 시리얼 통신을 이용하여 아두이노와 위험에 대한 통신
* 위험상태일 경우 0을 전송

**6.coding & demo**

* **Flask 서버 코드 (openpose 행동 계산 & 행동 분석 & 데이터 베이스 연결)**
* import cv2
* import time
* import math
* import numpy as np
* import argparse
* import datetime
* import serial
* import threading
* import pyrebase
* from flask import \*
* import os
* from pyfcm import FCMNotification
* #--------------------------------데베 부분
* import firebase\_admin
* from firebase\_admin import credentials, db
* #Firebase 콘솔에서 비밀키를 생성하고 파일 다운로드 -> key json 파일로 인증
* cred = credentials.Certificate("C:\\Users\\User\\test\\templates\\serviceAccountKey.json")
* firebase\_admin.initialize\_app(cred, {
* "databaseURL" : "https://casptone-a2cbe-default-rtdb.firebaseio.com/"
* })
* #---------------------------데베 부분
* #---------------------------알림 부분
* #알림을 위한 설정
* #registration\_id = "eBAOp5UUQxiuA6F9NsKc5T:APA91bGomx0X63bpVmgoeLlVSfN-YJ4U6EeW\_xlnVow0QrtuZQGuaOScLzXcmrRDo8n8ItoRMgn-T48ntTINPRIc8le20IIV\_9GQY\_-jlibIGHwblXiQ4Vx5Htu2YaehJLJt4e3CHTcQ"
* push\_service = FCMNotification(
* api\_key="AAAAOYtUnXk:APA91bGSzC-3SOeVbEgiu3uV\_udbcGYk2vzKW9UpCdEghFsMrQBzoePYinKJBrU9HHWeL0EvMRhdarnXcE3rHxtQWqJt5z4EAKJecSCtVL1rHMNTDKQQq1eM3XbZZrEgLHDkx36p3wT2")
* mk = db.reference('users/testid99/token') # 데베에서 토큰값 불러오기
* sk = db.reference('users/fcmcheck/token') # 데베에서 토큰값 불러오기
* registration\_ids = [mk.get(),sk.get()] # 토큰값 추가
* #---------------------------알림 부분
* # print('serial ' + serial.\_\_version\_\_)
* #파이어베이스 정보
* config = {
* "apiKey": "AIzaSyAOksU2ogxblR2vUVczwffzmv-x15v2OJI",
* "authDomain": "casptone-a2cbe.firebaseapp.com",
* "databaseURL": "https://casptone-a2cbe-default-rtdb.firebaseio.com",
* "projectId": "casptone-a2cbe",
* "storageBucket": "casptone-a2cbe.appspot.com",
* "messagingSenderId": "247150714233",
* "appId": "1:247150714233:web:31b6a3fdcc9d5138c5e3a8",
* "serviceAccount": "C:\\Users\\User\\test\\serviceAccountKey.json"
* }
* #파이어베이스 연결
* firebase = pyrebase.initialize\_app(config)
* storage = firebase.storage()
* app = Flask(\_\_name\_\_)
* pushtest = 1
* @app.route("/")
* def index():
* return render\_template('video.html') #html파일 이름 multiperson 이나 video
* def gen():
* #PORT = 'COM4'
* #BaudRate = 9600
* #ARD = serial.Serial(PORT, BaudRate)
* parser = argparse.ArgumentParser(description='Run keypoint detection')
* parser.add\_argument("--device", default="gpu", help="Device to inference on")
* args = parser.parse\_args()
* protoFile = "C:\\project\\models\\coco\\pose\_deploy\_linevec.prototxt"
* weightsFile = "C:\\project\\models\\coco\\pose\_iter\_440000.caffemodel"
* nPoints = 18
* # COCO Output Format
* keypointsMapping = ['Nose', 'Neck', 'R-Sho', 'R-Elb', 'R-Wr', 'L-Sho', 'L-Elb', 'L-Wr', 'R-Hip', 'R-Knee', 'R-Ank',
* 'L-Hip', 'L-Knee', 'L-Ank', 'R-Eye', 'L-Eye', 'R-Ear', 'L-Ear']
* POSE\_PAIRS = [[1, 2], [1, 5], [2, 3], [3, 4], [5, 6], [6, 7],
* [1, 8], [8, 9], [9, 10], [1, 11], [11, 12], [12, 13],
* [1, 0], [0, 14], [14, 16], [0, 15], [15, 17],
* [2, 17], [5, 16]]
* mapIdx = [[31, 32], [39, 40], [33, 34], [35, 36], [41, 42], [43, 44],
* [19, 20], [21, 22], [23, 24], [25, 26], [27, 28], [29, 30],
* [47, 48], [49, 50], [53, 54], [51, 52], [55, 56],
* [37, 38], [45, 46]]
* colors = [[0, 100, 255], [0, 100, 255], [0, 255, 255], [0, 100, 255], [0, 255, 255], [0, 100, 255],
* [0, 255, 0], [255, 200, 100], [255, 0, 255], [0, 255, 0], [255, 200, 100], [255, 0, 255],
* [0, 0, 255], [255, 0, 0], [200, 200, 0], [255, 0, 0], [200, 200, 0], [0, 0, 0]]
* inWidth = 168
* inHeight = 168
* pre\_Y = 0
* present\_Y = 0
* threshold = 0.1
* input\_source = 0
* cap = cv2.VideoCapture("C:\\openpose\\examples\\media\\fall.mp4")
* hasFrame, frame = cap.read()
* vid\_writer = cv2.VideoWriter('output.mp4', cv2.VideoWriter\_fourcc('M', 'J', 'P', 'G'), 10,
* (frame.shape[1], frame.shape[0]))
* net = cv2.dnn.readNetFromCaffe(protoFile, weightsFile)
* if args.device == "cpu":
* net.setPreferableBackend(cv2.dnn.DNN\_TARGET\_CPU)
* print("Using CPU device")
* elif args.device == "gpu":
* net.setPreferableBackend(cv2.dnn.DNN\_BACKEND\_CUDA)
* net.setPreferableTarget(cv2.dnn.DNN\_TARGET\_CUDA)
* print("Using GPU device")
* while cv2.waitKey(1) < 0:
* count = 0
* t = time.time()
* swtich\_degree = False
* hasFrame, frame = cap.read()
* frameCopy = np.copy(frame)
* if not hasFrame:
* cv2.waitKey()
* break
* frameWidth = frame.shape[1]
* frameHeight = frame.shape[0]
* inpBlob = cv2.dnn.blobFromImage(frame, 1.0 / 255, (inWidth, inHeight),
* (0, 0, 0), swapRB=False, crop=False)
* net.setInput(inpBlob)
* output = net.forward() # make a prediction
* H = output.shape[2]
* W = output.shape[3]
* detected\_keypoints = []
* keypoints\_list = np.zeros((0, 3))
* keypoint\_id = 0
* threshold = 0.1
* post\_y = 0
* rHipY = 0
* lHipY = 0
* neck = 0
* flag\_for = True
* for part in range(nPoints):
* probMap = output[0, part, :, :]
* probMap = cv2.resize(probMap, (frame.shape[1], frame.shape[0]))
* # getKeypoints
* mapSmooth = cv2.GaussianBlur(probMap, (3, 3), 0, 0)
* mapMask = np.uint8(mapSmooth > threshold)
* keypoints = []
* contours, \_ = cv2.findContours(mapMask, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)
* for cnt in contours:
* blobMask = np.zeros(mapMask.shape)
* blobMask = cv2.fillConvexPoly(blobMask, cnt, 1)
* maskedProbMap = mapSmooth \* blobMask
* \_, maxVal, \_, maxLoc = cv2.minMaxLoc(maskedProbMap)
* keypoints.append(maxLoc + (probMap[maxLoc[1], maxLoc[0]],))
* # getKeyPoints
* # print("Keypoints - {} : {}".format(keypointsMapping[part], keypoints))
* if keypointsMapping[part] == "Neck":
* mapSmooth = cv2.GaussianBlur(probMap, (3, 3), 0, 0)
* mapMask = np.uint8(mapSmooth > threshold)
* keypoints = []
* contours, \_ = cv2.findContours(mapMask, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)
* for cnt in contours:
* blobMask = np.zeros(mapMask.shape)
* blobMask = cv2.fillConvexPoly(blobMask, cnt, 1)
* maskedProbMap = mapSmooth \* blobMask
* \_, maxVal, \_, maxLoc = cv2.minMaxLoc(maskedProbMap)
* keypoints.append(maxLoc + (probMap[maxLoc[1], maxLoc[0]],))
* neck = maxLoc[1]
* elif keypointsMapping[part] == "R-Hip":
* mapSmooth = cv2.GaussianBlur(probMap, (3, 3), 0, 0)
* mapMask = np.uint8(mapSmooth > threshold)
* keypoints = []
* contours, \_ = cv2.findContours(mapMask, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)
* for cnt in contours:
* blobMask = np.zeros(mapMask.shape)
* blobMask = cv2.fillConvexPoly(blobMask, cnt, 1)
* maskedProbMap = mapSmooth \* blobMask
* \_, maxVal, \_, maxLoc = cv2.minMaxLoc(maskedProbMap)
* keypoints.append(maxLoc + (probMap[maxLoc[1], maxLoc[0]],))
* rHipY = maxLoc[1]
* elif keypointsMapping[part] == "L-Hip":
* mapSmooth = cv2.GaussianBlur(probMap, (3, 3), 0, 0)
* mapMask = np.uint8(mapSmooth > threshold)
* keypoints = []
* contours, \_ = cv2.findContours(mapMask, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)
* for cnt in contours:
* blobMask = np.zeros(mapMask.shape)
* blobMask = cv2.fillConvexPoly(blobMask, cnt, 1)
* maskedProbMap = mapSmooth \* blobMask
* \_, maxVal, \_, maxLoc = cv2.minMaxLoc(maskedProbMap)
* keypoints.append(maxLoc + (probMap[maxLoc[1], maxLoc[0]],))
* lHipY = maxLoc[1]
* # print(neck, rHipY, lHipY)
* if neck != 0 and rHipY != 0 and lHipY != 0:
* post\_y = neck + rHipY + lHipY
* keypoints\_with\_id = []
* minVal, prob, minLoc, point = cv2.minMaxLoc(probMap)
* for i in range(len(keypoints)):
* keypoints\_with\_id.append(keypoints[i] + (keypoint\_id,))
* keypoints\_list = np.vstack([keypoints\_list, keypoints[i]])
* keypoint\_id += 1
* # Add the point to the list if the probability is greater than the threshold
* detected\_keypoints.append(keypoints\_with\_id)
* present\_Y = post\_y
* frameClone = frame.copy()
* for i in range(nPoints):
* for j in range(len(detected\_keypoints[i])):
* cv2.circle(frameClone, detected\_keypoints[i][j][0:2], 5, colors[i], -1, cv2.LINE\_AA)
* # getValidPairs
* valid\_pairs = []
* invalid\_pairs = []
* n\_interp\_samples = 10
* paf\_score\_th = 0.1
* conf\_th = 0.7
* for k in range(len(mapIdx)):
* # A->B constitute a limb
* pafA = output[0, mapIdx[k][0], :, :]
* pafB = output[0, mapIdx[k][1], :, :]
* pafA = cv2.resize(pafA, (frameWidth, frameHeight))
* pafB = cv2.resize(pafB, (frameWidth, frameHeight))
* candA = detected\_keypoints[POSE\_PAIRS[k][0]]
* candB = detected\_keypoints[POSE\_PAIRS[k][1]]
* nA = len(candA)
* nB = len(candB)
* if (nA != 0 and nB != 0):
* valid\_pair = np.zeros((0, 3))
* for i in range(nA):
* max\_j = -1
* maxScore = -1
* found = 0
* for j in range(nB):
* # Find d\_ij
* d\_ij = np.subtract(candB[j][:2], candA[i][:2])
* norm = np.linalg.norm(d\_ij)
* if norm:
* d\_ij = d\_ij / norm
* else:
* continue
* # Find p(u)
* interp\_coord = list(zip(np.linspace(candA[i][0], candB[j][0], num=n\_interp\_samples),
* np.linspace(candA[i][1], candB[j][1], num=n\_interp\_samples)))
* # Find L(p(u))
* paf\_interp = []
* for k in range(len(interp\_coord)):
* paf\_interp.append([pafA[int(round(interp\_coord[k][1])), int(round(interp\_coord[k][0]))],
* pafB[int(round(interp\_coord[k][1])), int(round(interp\_coord[k][0]))]])
* # Find E
* paf\_scores = np.dot(paf\_interp, d\_ij)
* avg\_paf\_score = sum(paf\_scores) / len(paf\_scores)
* # Check if the connection is valid
* # If the fraction of interpolated vectors aligned with PAF is higher then threshold -> Valid Pair
* if (len(np.where(paf\_scores > paf\_score\_th)[0]) / n\_interp\_samples) > conf\_th:
* if avg\_paf\_score > maxScore:
* max\_j = j
* maxScore = avg\_paf\_score
* found = 1
* # Append the connection to the list
* if found:
* valid\_pair = np.append(valid\_pair, [[candA[i][3], candB[max\_j][3], maxScore]], axis=0)
* # Append the detected connections to the global list
* valid\_pairs.append(valid\_pair)
* else: # If no keypoints are detected
* # print("No Connection : k = {}".format(k))
* invalid\_pairs.append(k)
* valid\_pairs.append([])
* # getValidPairs
* # getPersonwiseKeypoints
* personwiseKeypoints = -1 \* np.ones((0, 19))
* for k in range(len(mapIdx)):
* if k not in invalid\_pairs:
* partAs = valid\_pairs[k][:, 0]
* partBs = valid\_pairs[k][:, 1]
* indexA, indexB = np.array(POSE\_PAIRS[k])
* for i in range(len(valid\_pairs[k])):
* found = 0
* person\_idx = -1
* for j in range(len(personwiseKeypoints)):
* if personwiseKeypoints[j][indexA] == partAs[i]:
* person\_idx = j
* found = 1
* break
* if found:
* personwiseKeypoints[person\_idx][indexB] = partBs[i]
* personwiseKeypoints[person\_idx][-1] += keypoints\_list[partBs[i].astype(int), 2] + \
* valid\_pairs[k][i][2]
* # if find no partA in the subset, create a new subset
* elif not found and k < 17:
* row = -1 \* np.ones(19)
* row[indexA] = partAs[i]
* row[indexB] = partBs[i]
* # add the keypoint\_scores for the two keypoints and the paf\_score
* row[-1] = sum(keypoints\_list[valid\_pairs[k][i, :2].astype(int), 2]) + valid\_pairs[k][i][2]
* personwiseKeypoints = np.vstack([personwiseKeypoints, row])
* # getPersonwiseKeypoints
* for i in range(17):
* for n in range(len(personwiseKeypoints)):
* index = personwiseKeypoints[n][np.array(POSE\_PAIRS[i])]
* if -1 in index:
* continue
* B = np.int32(keypoints\_list[index.astype(int), 0])
* A = np.int32(keypoints\_list[index.astype(int), 1])
* cv2.line(frameClone, (B[0], A[0]), (B[1], A[1]), colors[i], 3, cv2.LINE\_AA)
* # 기울기 계산
* if colors[i] == [0, 255, 0]:
* # calculate\_degree
* dx = B[1] - B[0]
* dy = A[1] - A[0]
* rad = math.atan2(abs(dy), abs(dx))
* deg = rad \* 180 / math.pi
* if deg < 45:
* string = "bend"
* cv2.putText(frameClone, string, (0, 25), cv2.FONT\_HERSHEY\_DUPLEX, 1, (255, 0, 255))
* print(f"[degree] {deg} ({string})")
* op = 'a'
* #ARD.write(op.encode())
* elif deg > 45:
* op = 'b'
* #ARD.write(op.encode())
* if pre\_Y - present\_Y > 900: # 숫자가 커질수록 민감도 높아짐
* print("fall!!!!!!!!!!!!!!!!!!!!!!!!!!")
* filename = datetime.datetime.now().strftime("%H-%M-%S")
* foldername = datetime.datetime.now().strftime("%F")
* cv2.imwrite("C:\\capture\\ " + str(filename) + ".png", frame) # 데스크탑에 저장 >> 파이어 베이스 업로드로 변경하면됨
* storage.child(str(foldername) + "/" + str(filename) + ".png" + ".png").put("C:\\capture\\ " + str(filename) + ".png")
* #timeMsg = datetime.datetime.now().strftime("%H시-%M분-%S초 발생")
* message\_title = "낙상 발생"
* message\_body = "낙상 발생 " + filename
* # + timeMsg
* #result = push\_service.notify\_multiple\_devices(registration\_ids=registration\_ids,
* # message\_title=message\_title, message\_body=message\_body)
* result = push\_service.notify\_topic\_subscribers(topic\_name="news", message\_title=message\_title,
* message\_body=message\_body)
* pre\_Y = present\_Y
* present\_Y = 0
* cv2.putText(frame, "time taken = {:.2f} sec".format(time.time() - t), (50, 50), cv2.FONT\_HERSHEY\_COMPLEX, .8,
* (255, 50, 0), 2, lineType=cv2.LINE\_AA)
* imgencode = cv2.imencode('.jpg', frameClone)[1]
* stringData = imgencode.tostring()
* yield (b'--frame\r\n'
* b'Content-Type: text/plain\r\n\r\n' + stringData + b'\r\n')
* vid\_writer.write(frameClone)
* cv2.waitKey(0)
* del (cap)
* @app.route('/calculation')
* def calculation():
* return Response(gen(),
* mimetype='multipart/x-mixed-replace; boundary=frame')
* if \_\_name\_\_ == "\_\_main\_\_":
* app.run(debug=True)
* **아두이노 동작 코드**
* #include <Servo.h>
* Servo servo1;
* Servo servo2;
* #define OFFMODE -1
* #define ONMODE 1
* char ch;
* int state = OFFMODE;
* void setup() {
* // put your setup code here, to run once:
* Serial.begin(9600);
* servo1.attach(7);
* servo2.attach(8);
* servo1.write(0);
* servo2.write(0);
* }
* void loop() {
* // put your main code here, to run repeatedly:
* if(Serial.available()){
* ch = Serial.read();
* }
* if(ch=='a'){
* servo1.write(90);
* servo2.write(90);
* }else if(ch=='b'){
* servo1.write(0);
* servo2.write(0);
* }
* }
* **APP 동작 코드**
* import android.content.Intent
* import android.content.pm.PackageInfo
* import android.content.pm.PackageManager
* import androidx.appcompat.app.AppCompatActivity
* import android.os.Bundle
* import android.util.Base64
* import android.util.Log
* import android.widget.Toast
* import com.facebook.\*
* import com.google.android.gms.auth.api.Auth
* import com.google.android.gms.auth.api.signin.GoogleSignIn
* import com.google.android.gms.auth.api.signin.GoogleSignInAccount
* import com.google.android.gms.auth.api.signin.GoogleSignInClient
* import com.google.android.gms.auth.api.signin.GoogleSignInOptions
* import com.google.firebase.auth.FirebaseAuth
* import com.google.firebase.auth.FirebaseUser
* import com.google.firebase.auth.GoogleAuthProvider
* import kotlinx.android.synthetic.main.activity\_login.\*
* import com.facebook.appevents.AppEventsLogger;
* import com.facebook.login.LoginManager
* import com.facebook.login.LoginResult
* import com.google.firebase.auth.FacebookAuthProvider
* import java.security.MessageDigest
* import java.security.NoSuchAlgorithmException
* import java.util.\*
* class LoginActivity : AppCompatActivity() {
* var auth: FirebaseAuth? = null // 로그인을 관리해주는 클래스 FirebaseAuth
* var googleSignInClient: GoogleSignInClient?=null
* var GOOGLE\_LOGIN\_CODE=9001 // 임의 숫자 넣으면 됨
* var callbackManager:CallbackManager?=null
* override fun onCreate(savedInstanceState: Bundle?) {
* super.onCreate(savedInstanceState)
* setContentView(R.layout.activity\_login)
* auth = FirebaseAuth.getInstance() //auth 초기화
* email\_login\_button.setOnClickListener {
* createAndLoginEmail()
* }
* google\_sign\_in\_button.setOnClickListener {
* googleLogin()
* }
* facebook\_login\_button.setOnClickListener{
* facebookLogin()
* }
* // 구글 시작하기 전에 세팅해주는 것
* var gso = GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT\_SIGN\_IN)
* .requestIdToken(getString(R.string.default\_web\_client\_id))//Client id= 구글로그인에 접근할 수 있도록 허가해주는 아이디 키값. 얘가 있어야 구글 로그인에 접근 가능 (일종의 로그인 인증키)
* .requestEmail()
* .build() // 조립 완성됐다는 뜻
* googleSignInClient= GoogleSignIn.getClient(this,gso) // 구글 로그인하는 클래스 완성됨!
* callbackManager= CallbackManager.Factory.create() // 초기화
* }
* fun createAndLoginEmail() {
* auth?.createUserWithEmailAndPassword(
* email\_edittext.text.toString(),
* password\_edittext.text.toString()
* )?.addOnCompleteListener { task -> //파라미터를 받아옴
* if (task.isSuccessful) {
* //Toast.makeText(this, "아이디 생성이 완료되었습니다.", Toast.LENGTH\_LONG).show()
* moveMainPage(auth?.currentUser)
* } else if (task.exception?.message.isNullOrEmpty()) {
* // 익셉션에 메시지가 있을 경우 메세지 출력
* //isnullorempty 대신 !=null 해도 됨
* Toast.makeText(this, task.exception?.message, Toast.LENGTH\_LONG).show()
* } else {
* // 로그인임. 아이디&비번도 있고, 익셉션도 없다면 로그인!
* signinEmail()
* }
* }
* }
* fun signinEmail() {
* auth?.signInWithEmailAndPassword(
* email\_edittext.text.toString(),
* password\_edittext.text.toString()
* )
* ?.addOnCompleteListener { task ->
* if (task.isSuccessful) {
* //Toast.makeText(this, "로그인이 성공했습니다.", Toast.LENGTH\_LONG).show()
* moveMainPage(auth?.currentUser)//auth는 로그인이 성공하면 유저에 대한 정보를 가지게 된다.
* } else {
* Toast.makeText(this, task.exception?.message, Toast.LENGTH\_LONG).show()
* }
* }
* }
* fun moveMainPage(user: FirebaseUser?) {
* if (user != null) {
* //유저가 있을 경우 다음 페이지로 넘어감
* startActivity(Intent(this, MainActivity::class.java))
* finish()
* }
* }
* fun googleLogin() {
* var signInIntent=googleSignInClient?.signInIntent
* startActivityForResult(signInIntent,GOOGLE\_LOGIN\_CODE)
* }
* fun facebookLogin(){
* LoginManager.getInstance()
* .logInWithReadPermissions(this, Arrays.asList("public\_profile","email"))
* LoginManager
* .getInstance()
* .registerCallback(callbackManager,object:FacebookCallback<LoginResult>{
* override fun onSuccess(result: LoginResult?) {
* println("onSuccess")
* handleFacebookAccessToken(result?.accessToken)
* }
* override fun onCancel() {
* println("onCancel")
* }
* override fun onError(error: FacebookException?) {
* println("onError")
* }
* })
* }
* fun handleFacebookAccessToken(token:AccessToken?){
* var credential=FacebookAuthProvider.getCredential(token?.token!!)
* auth?.signInWithCredential(credential)!!.addOnCompleteListener {
* task->
* println("task"+task.isSuccessful)
* Log.d("페이스북",task.isSuccessful.toString())
* }.addOnFailureListener{
* exception ->
* println("exception"+exception.message)
* }
* }
* fun firebaseAuthWithGoogle(account:GoogleSignInAccount){
* // 인증서를 먼저 만들어줘야 함
* var credential=GoogleAuthProvider.getCredential(account.idToken,null)
* auth?.signInWithCredential(credential)
* }
* // 결과값을 받는 함수, 구글 버튼 누르고 사용자의 계정을 선택하면 그 결과값이 넘어옴
* // 사용자의 계정을 선택하는 창이 꺼졌을 때 이전 액티비티 (로그인 액티비티) 에게 값이 넘어가고
* // 그 값이 firebase에 갱신되도록 한다.
* override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
* super.onActivityResult(requestCode, resultCode, data)
* callbackManager?.onActivityResult(requestCode,resultCode,data)
* // 넘어온 사용자의 정보를 가공
* if(resultCode==GOOGLE\_LOGIN\_CODE ){
* var result= Auth.GoogleSignInApi.getSignInResultFromIntent(data)
* if(result.isSuccess){
* // 구글 로그인에 성공했을 경우 파이어베이스에게 값 넘겨줘야 함 - firebaseAuthWithGoogle
* var account=result.signInAccount
* firebaseAuthWithGoogle(account!!)
* }
* }
* }
* }

**Ⅲ. 결론**

**1. 연구 결과**

- 코로나19로 인해 노인들이 고위험군으로 지정되어 대면으로 관리하는 것이 어려운 현재 상황에서 1차적으로 CCTV를 통해 상황 판단과 낙상 방지를 하는 것은 요양사의 수고를 덜어줄 수 있다.

- 프로그램을 통해 사람의 움직임을 감지하고 낙상을 감지하며 자세를 판단하여 난간을 동작 시킬 수 있었고 상황을 app에서 확일 할 수 있었다.

- 낙상 알고리즘의 경우 openpose의 스켈레톤에서 추출한 (x,y) 값을 이전 루프와 비교하여 변화에 따른 (x,y)값을 기준으로 하기 때문에 카메라의 위치에 따른 사람의 비율과 크기에 따라 민감도가 달라지는 것을 확인 할 수 있었다.

- openpose의 모델 중 coco model을 사용하여 프로그램을 돌리는 것이 연산속도와 처리능력 면에서 가장 훌륭한 결과를 보여주었다.

- 개선사항으로 카메라에 인식되는 사람이 여러 명일 때 각각의 사람을 openpose를 동작시키는 것 까지는 수행하였으나 각 사람의 낙상과 자세를 분리하여 판단하는 것에 보충이 필요하다.

**2. 작품제작 소요재료 목록**

**-** Raspberry Pi 4B 4G

사양  
프로세서 : Broadcom BCM2711, quad-core Cortex-A72(ARM v8) 64-bit SoC @1.5GHz 메모리 : 2GB, 4GB, 8GB LPDDR4

- 라즈베리파이 카메라모듈 V2(RPI 8MP CAMERA BOARD)

사양  
이미지 센서 : Sony IMX 219 PQ CMOS image seonsor in a fixed-focus module  
방식 : RTSP 서버를 사용하여 IP주소 할당 후 IP카메라 처럼 사용