

1. 개발 과제의 요약

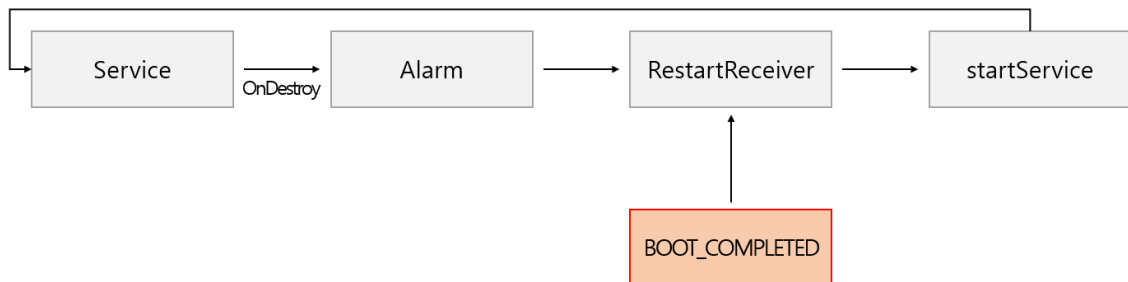
어르신들의 건강 및 안전 케어에 대한 관심이 높아지고 있으며, 이를 위한 다양한 서비스가 검토되고 있다. 특히 어르신들의 거주 공간에서 생활 정보를 바탕으로 어르신들의 응급 안전 대응 및 건강 관리를 할 수 있다면 이는 매우 효과적인 서비스가 될 것이다. 본 개발의 목표는 디스플레이형 AI 단말기를 활용하여 65세 이상의 시니어를 대상으로 일상생활 데이터를 모으고, 앞에서 말한 서비스 중 실시간 행동 분석을 통한 어르신들의 안전 케어 및 위험 상황 판단에 대한 스마트 서비스를 제공하는 시스템을 개발하는 것이다.

2. 개념설계안

가. 죽지 않는 서비스

서비스는 앱이 UI 없이 백그라운드에서 특정 시간 동안 실행되는 것을 의미한다. 많은 앱은 서비스를 데이터 처리, 연산 등에서 많이 사용한다. 일반적인 안드로이드 앱들은 죽지 않는 서비스(Immortal Service)를 주로 이용한다. 왜냐하면, 백그라운드 서비스는 OS가 Task Kill을 해서 서비스를 종료시켜버릴 수 있지만 죽지 않는 서비스는 여기에서 서비스가 종료되었을 때, 다시 서비스를 재실행시키는 방식으로 구현하여 서비스가 완전히 종료되지 않도록 하기 때문이다.

죽지 않는 서비스의 구현은 일반적인 서비스를 구현하는 것과 비슷하지만 차이점이 존재한다. 먼저 서비스를 실행시키고 서비스가 어떠한 조건으로 죽으면 알람을 통해 다시 startService 메서드를 사용, 서비스를 재실행시킨다. 제일 중요하게 처리해야 하는 조건은 위의 문단에서 말했던 OS가 서비스를 죽이는 경우이다. 앱을 종료하거나 내리거나 화면을 끄는 경우는 서비스의 특성상 괜찮지만 OS가 서비스를 죽이는 경우에는 서비스가 시작돼야 하는 단이 백그라운드이기 때문에 조금 더 복잡하다. 지금 안드로이드 버전이 Snow Cone(12.1)까지 나왔는데 안드로이드 Oreo(8.0) 이후 버전에서는 서비스를 백그라운드에서 실행하는 것을 금지하기 때문에 포그라운드에서 실행해야 한다. 그래서 죽지 않는 서비스를 구현할 때 startForegroundService 메서드라는 한 가지 과정이 더 필요하지만, 우리가 사용하는 단말기의 버전은 Oreo 이전 버전(7.1)이기 때문에 startService 메서드와 리시버만 가지고 죽지 않는 서비스를 구현할 수 있다. 이외의 죽지 않는 서비스를 위하여 구현이 필요한 부분은 재부팅 시 서비스 자동 실행이다. 이 부분은 브로드캐스트리시버를 사용하여 액션을 받아 현재 부팅이 되었다는 것을 인식하면 startService를 실행시키는 것으로 해결할 수 있다.

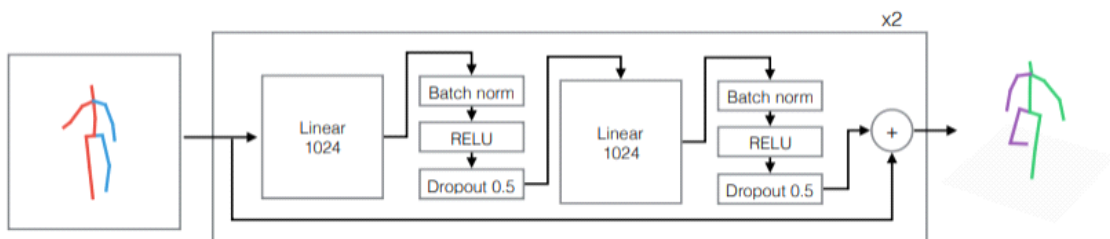


나. 영상에서의 행동 인식

영상을 입력받으면, 영상을 실시간 행동 분석이 가능한 프레임 수로 나누어 인간 객체를 Skeleton으로 벡터화시키는 과정을 진행한다. 이때, 구글에서 만들어서 배포한 딥러닝 API인 Pose Detection API를 사용한다.

Pose Detection API를 거친 영상은 프레임별로, 인간 객체의 골격 위치 정보를 저장할 수 있다.

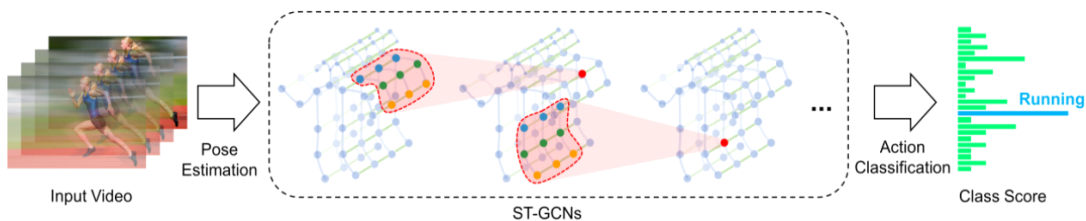
이러한 정보를 곧바로 모델의 입력에 사용할 수 있지만, 2차원의 관절 정보로는 분류 정확도가 낮음을 확인했다. 실제로, 실시간 행동 인식 모형의 State of Art는 Kinetic 카메라로부터 얻어진 RGB+D 영상을 기준으로 처리하는 모형이 많다. 하지만, 실제 AI 단말기 내에서 사용되는 카메라는 RGB를 출력하는 일반 카메라로, 분류 성능을 위한 정보의 양이 한정되어있다.



이러한 문제를 해결하기 위해, deep-learning 기반 모델로 주어진 2D 관절 좌표를 3차원으로 매핑해주는 오픈소스인 3D-Pose-Baseline을 이용한다. 이는 RGB 카메라에서 얻은 정보를 3차원 공간정보로 바꾸어주는 과정을 추가하여, 카메라의 성능 개선 부담을 줄이면서, 정보의 양은 늘리는 효과를 가진다.

즉, 앞선 두 과정을 파이프라인으로 설정하여, 입력 데이터가 들어오면, 프레임별로 2D-Skeleton 정보를 추출하고 이를 3D-Skeleton 정보로 변환하는 과정을 거치게 된다.

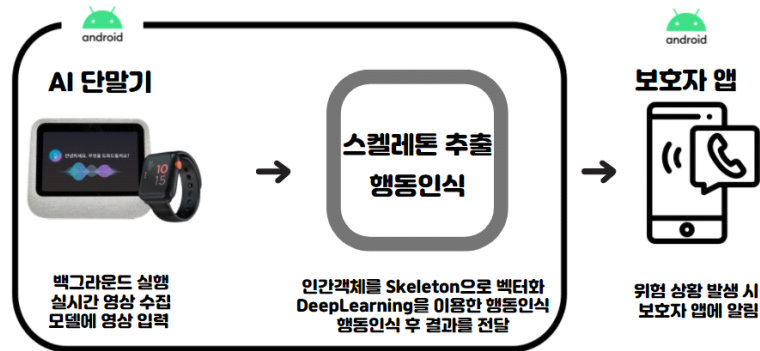
입력 파이프라인을 거친 3D 공간정보는 행동 분류 모델의 입력이 된다. 행동 분류 모델은 ST-GCN(Spatial Temporal Graph Convolutional Networks)을 사용한다.



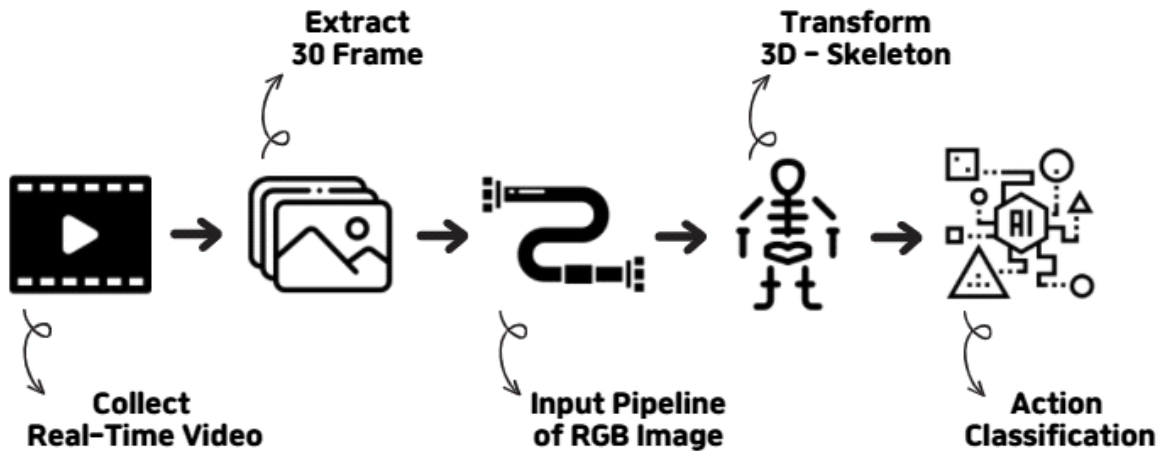
어르신들의 70세 이상의 고령자 53명의 자택을 방문하여 기상부터 취침까지의 하루 행동을 직접 관찰한 데이터에서 245개의 일상 활동 유형 중 빈번하게 나타나는 행동 55가지 중에 쓰러지기 행동은 위험상황으로 정의하고 나머지 TV 시청, 식사 관련 활동, 화장실 사용, 식사 준비, 전화 통화, 약 복용, 요리, 청소 등은 쓰러지기가 아닌 일상생활로 정의한다. 해당 라벨들은 분류 모형의 예측 클래스가 되며, 각 행동들에 대한 로짓값을 출력하도록 한다. 그렇게 쓰러지기와 일상생활을 이진분류하는 모델을 만든다.

다. AI 단말기 내에서 실행되는 실시간 행동분석

앞의 두 가지 개념을 이용해서 결과적으로 다음과 같은 시스템을 만들고자 한다.



실시간 행동 분석을 위해, AI 단말기의 전원을 켜면 단말기 내의 카메라를 제어하여 영상(또는 프레임)을 자동으로 수집하여, 윈도우 환경에서 학습한 입력 파이프라인과 행동 분류 모델의 입력으로 넣는다.



실시간 영상(일정 프레임)에서의 행동을 분류하기 위해, 입력 이미지의 개수(프레임의 개수)를 30개로 제한하여 매 30 프레임마다 어떤 행동인지 판단하게 된다. 윈도우 환경에서 작성한 모델은 텐서플로 라이트로 변환 후 이식을 진행한다. 모델 과정의 출력 중 이상 행동이 검출되면, 사전에 연결된 보호자에게 해당 상황을 설명하는 알림을 전송한다.

라. 위험 상황에 대한 정의 및 범위

저희 조가 정의한 위험 상황은 '시니어 분들이 집 안에서 혼자 계실 때, 어떠한 이유로 보호자 측에 연락을 취할 수 없는 상황이 되어 빠른 대처가 되지 않았을 때, 시니어 분들의 건강 악화가 우려되는 경우나 목숨의 위험이 있는 경우 또는 시니어분들이 돌아가셨지만 보호자 측에서 인지하지 못하고 있는 경우'이다.

그리고 위험상황을 분류할 행동들은 낙상이다. 낙상을 감지하는 기준은 모델을 학습시킬 때 들어간 ETRI-Activity3D 데이터셋의 영상들이 기준이 될 것이다.



3. 이론적 계산 및 시뮬레이션

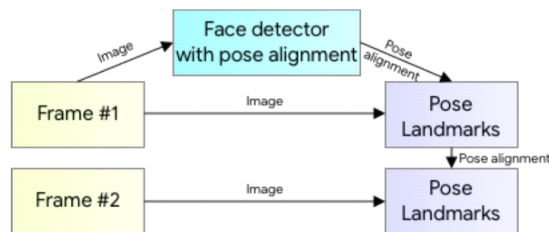
가. Pose Detection API의 이론적 과정

Pose Detection API는 BlazePose를 기반으로 작동하는 API이다.

BlazePose는 모바일에서도 human pose estimation에 대한 real-time inference가 가능한 모델이다. BlazePose는 heatmap과 regression을 모두 이용해 lightweight pose estimation을 한다는 장점이 있다.

기존의 자세 추정 모델은 heatmap을 이용해 각 관절에 대한 heatmap을 생성해 각 관절 좌표에 대한 offset을 수정하는 방식으로 진행되었지만, single person에 대한 실시간 자세 추론을 하기에는 모델이 너무 크다는 단점이 존재했다.

Regression 기반 모델은 컴퓨팅 리소스가 저렴하고 확장성이 크나, 관절이 맞물리는 경우에 예측이 취약하다는 단점이 있지만, BlazePose는 인코더 디코더 신경망을 이용해 모든 관절의 heatmap을 예측한 후, 다른 encoder로 regression을 이용해 각 관절의 좌표를 예측하는 구조를 가지고있으며, 추론당시에 heatmap부분을 제외해 모바일에서도 돌릴 수 있을 정도로 가볍게 만들어진다.

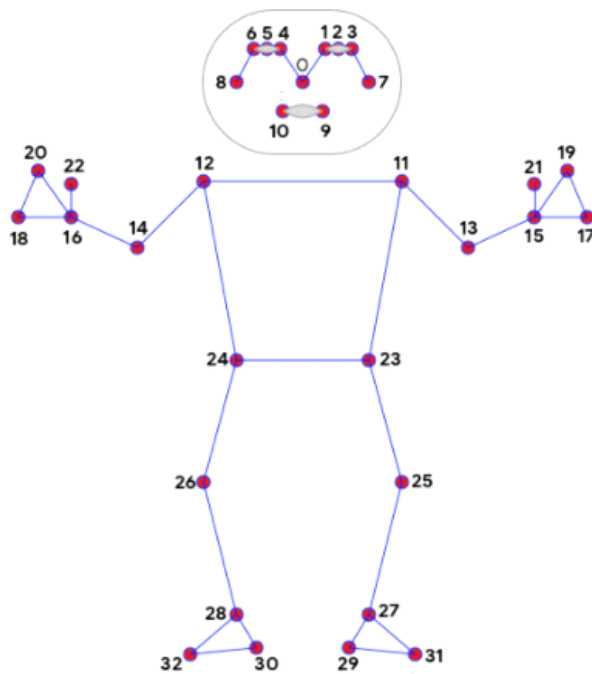
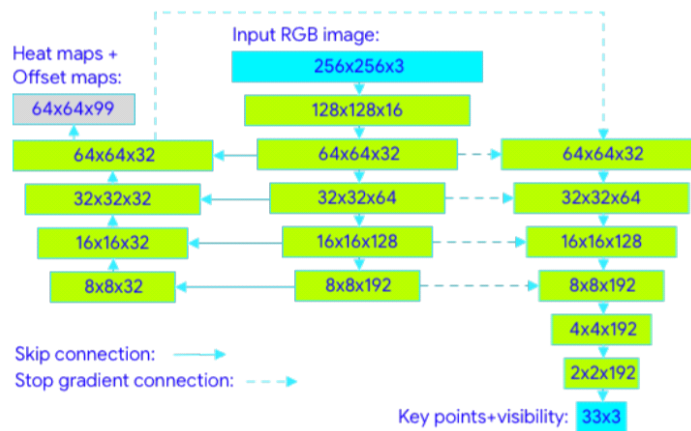


추론 파이프라인은 lightweight body pose detector와 pose tracker 순서로 구성된다. tracker는 keypoint 좌표, 사람 존재 유무, 현재 프레임의 ROI(Region Of Interesting)를 예측한다. Tracker에서 사람이 없다고 판단할 경우 다음 프레임에서 detector를 재실행한다.

최근의 객체 탐지 방법은 대부분 NMS(Non-Maximum Suppression)으로 후처리를 한다. 하지만 NMS는 keypoint가 겹치는 동작에서 오류가 발생한다는 단점이 있다.

BlazePose에서는 detector의 기준을 명확한 특징이있고, 편차가 적은 얼굴로 정하여, 사람의 골반 중앙값, 사람을 포함하는 원의 크기, 사람의 기울기 등의 alignment parameter를 예측한다.

BlazePose는 heatmap, offset, regression을 결합한 방식을 이용했다. Heatmap과 offset Loss는 학습과정에서만 이용하고, 추론을 할 때에는, output Layer를 제거해 경량화하여 regression encoder에서 활용할 수 있도록 했다. 해당 방식은 인코더-디코더 히트맵 기반 신경망 뒤에 regression 신경망이 따라오는 구조를 쌓는 방식으로 적용했다.



- | | |
|--------------------|----------------------|
| 0. nose | 17. left_pinky |
| 1. left_eye_inner | 18. right_pinky |
| 2. left_eye | 19. left_index |
| 3. left_eye_outer | 20. right_index |
| 4. right_eye_inner | 21. left_thumb |
| 5. right_eye | 22. right_thumb |
| 6. right_eye_outer | 23. left_hip |
| 7. left_ear | 24. right_hip |
| 8. right_ear | 25. left_knee |
| 9. mouth_left | 26. right_knee |
| 10. mouth_right | 27. left_ankle |
| 11. left_shoulder | 28. right_ankle |
| 12. right_shoulder | 29. left_heel |
| 13. left_elbow | 30. right_heel |
| 14. right_elbow | 31. left_foot_index |
| 15. left_wrist | 32. right_foot_index |
| 16. right_wrist | |

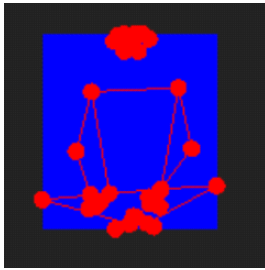
< Pose Detection API의 출력으로 나오는 33개 관절 정보 >

나. 1차 판단 알고리즘 이론적 과정 및 시뮬레이션

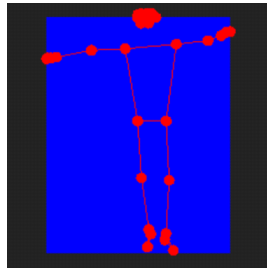
기존에 낙상에 따른 신체 관절의 동적 특성을 분석한 논문[1]에 따르면, 낙상 행동에서 땅에 닿기까지 상체는 약 0.8초, 하체는 약 0.5초의 시간이 걸린다고 분석했다. 하지만 해당 1차 판단 알고리즘에서는 어르신들의 움직임을 고려해서 시간의 폭을 좀 더 늘려 사람이 서 있던 자세에서 1초만에 바닥에 쓰러진 자세로 인식된다면 이를 낙상으로 의심한다. 이 특성과 Pose Detection API의 관절 정보로 그린 바운딩 박스의 비율로 1차적으로 낙상 의심 상황을 검출한다.

◇ 바운딩 박스

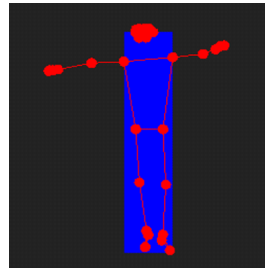
스켈레톤으로 낙상을 감지했던 기존 논문[4]을 참고하여 바운딩 박스를 이용하여 1차 판단을 사용하기로 하였다. 하지만 기존 알고리즘과 해당 알고리즘의 차이점은 Pose Detection API 추출한 33개의 정보 중에서 팔과 관련된 관절 정보(팔꿈치, 손목, 손가락)를 제외한 나머지 관절들을 사용하여 바운딩 박스를 그린다는 점이다. 팔을 제외한 이유는 서 있을 때의 자세와 앉아있을 때의 자세를 구별하기 위함이다. 일반적으로 사람이 바닥에 앉을 때는 바운딩 박스가 정사각형 모양이고, 서 있는 경우에는 세로로 긴 직사각형 모양이다. 이런 상황이라면 서 있을 때나 앉아있을 때를 구별할 수 있다. 하지만, 만약 사람이 서 있는 상태로 양팔을 벌린다면, 아래의 설명 그림 1, 2처럼 두 자세의 바운딩 박스는 둘 다 정사각형에 가깝게 그려지기 때문에 바운딩 박스만 보고 서 있거나 앉아있는 것은 구분하는 것은 어렵다. 이외에도 어떤 자세를 취하던 팔은 움직임이 많기 때문에, 팔을 제외하지 않고는 바운딩 박스의 수식 조건을 일반화하는 것은 어려웠기 때문에 바운딩 박스를 그릴 때 양팔은 제외하기로 하였다. 그 결과, 그림 3, 4처럼 사람이 서 있는 상태로 팔로 다른 행동을 하고 있더라도 바운딩 박스가 똑같이 그려지기 때문에 서 있다는 상태를 판단할 수 있게 되었다.



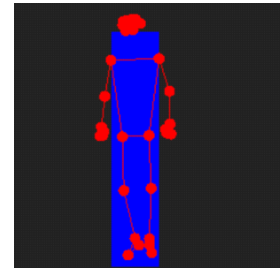
< 설명 그림1
앉을 때 (양팔 포함) >



< 설명 그림2
팔을 벌리고 서 있을 때
(양팔 포함) >



< 설명 그림3
팔을 벌리고 서 있을 때
(양팔 제외) >



< 설명 그림4
서 있을 때 (양팔 제외) >

이렇게 구한 바운딩 박스의 폭과 높이를 가지고, 바운딩 박스의 비율 R 을 계산한다. 일반적으로 서 있거나 앉아서 무언가를 한다면 $R < 1$, 바닥에 길게 누워있는 상태에서는 $R > 1$ 이다.

$$R = \text{Width} / \text{Height}$$

◇ 1차 판단 조건문

앞선 바운딩 박스 비율로 현재의 자세를 판단하고, 그 자세가 1.0초만에 서 있는 자세에서 누워있는 자세로 변경된다면 이를 낙상 의심 상황으로 판단한다. 그러기 위해서는 일단 일정 시간마다 바운딩 박스의 비율을 계산해야 한다. $w(t)$, $h(t)$ 는 각각 현재 시간 t 의 바운딩 박스의 폭과 높이이고, $R(t)$ 는 현재 시간의 바운딩 박스의 비율이다.

$$R(t) = \frac{w(t)}{h(t)}$$

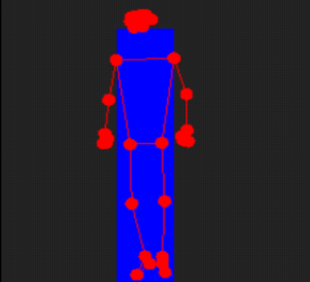
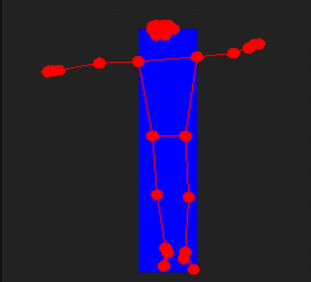
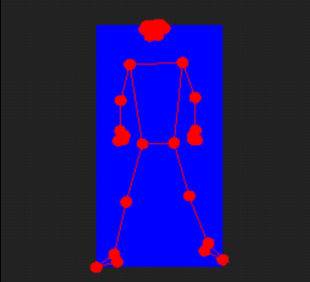
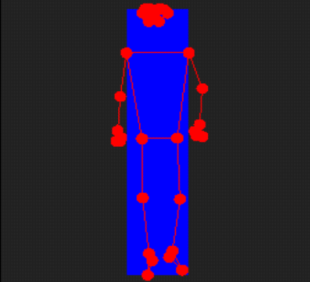
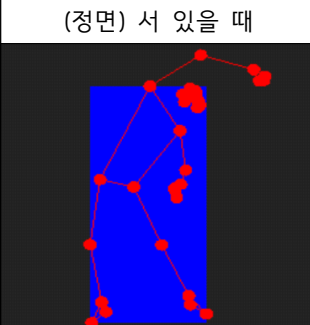
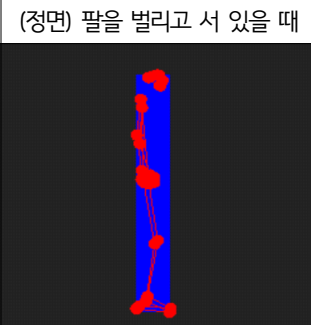
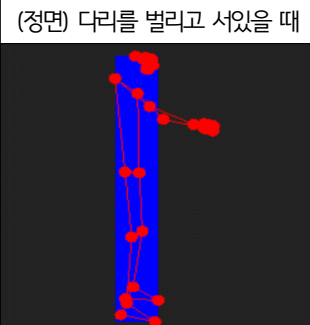
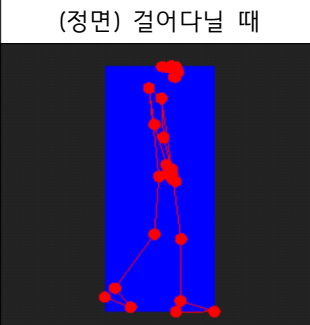
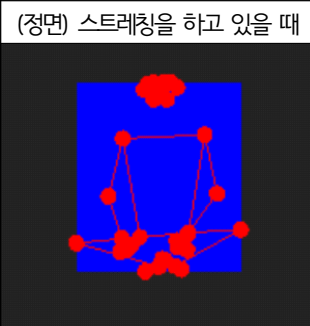
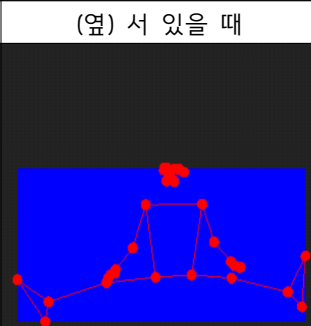
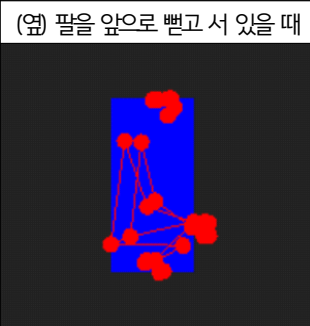
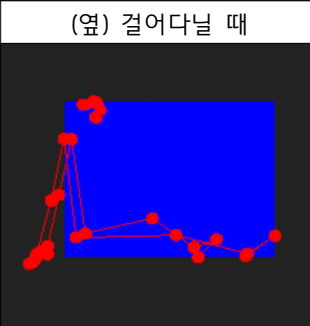
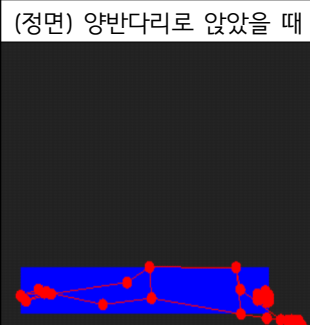
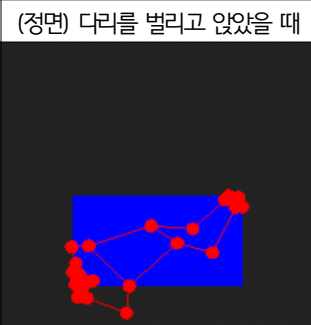
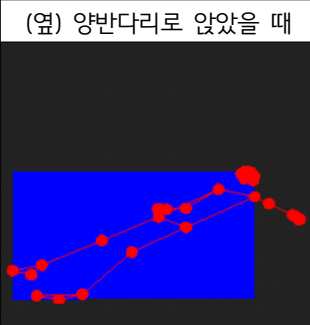
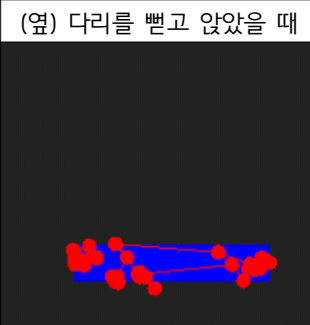
현재 시간의 $R(t)$ 값과 1.0초전의 바운딩 박스의 비율 $R(t-2)$ 값을 사용하여 조건식을 만들면 1.0초만에 서 있던 자세가 누운 자세로 변경되었는지를 판단할 수 있게 된다. 아래의 조건식에서 α , β 값은 각각 서 있는 자세를 구분하기 위한 임계값, 쓰러진 자세를 구분하기 위한 임계값이다.

$$\text{if } R(t-2) < \alpha \text{ and } R(t) > \beta :$$

낙상 의심

◇ 임계값을 찾기 위한 시뮬레이션

바운딩 박스의 비율로 자세를 구분할 수 있는 적절한 임계값을 구하기 위해서 일상생활에서 일어날 수 있는 자세들을 직접 취하고, 해당 자세의 비율을 분석하였다. 정면이란 카메라를 정면으로 바라봤을 때를 의미하고, 옆은 카메라가 찍는 방향에 수직한 방향을 바라봤을 때를 의미한다.

			
(정면) 서 있을 때	(정면) 팔을 벌리고 서 있을 때	(정면) 다리를 벌리고 서있을 때	(정면) 걸어다닐 때
			
(정면) 스트레칭을 하고 있을 때	(옆) 서 있을 때	(옆) 팔을 앞으로 뻗고 서 있을 때	(옆) 걸어다닐 때
			
(정면) 양반다리로 앉았을 때	(정면) 다리를 벌리고 앉았을 때	(옆) 양반다리로 앉았을 때	(옆) 다리를 뻗고 앉았을 때
			
쓰러졌을 때, 누울 때	카메라에 머리를 두고 쓰러졌을 때	카메라에 발을 두고 쓰러졌을 때	대각선으로 쓰러졌을 때,

< 여러 자세들의 바운딩 박스 >

실험한 자세 바운딩 박스의 R(t)값의 범위를 측정한 결과는 다음과 같다.

번호	자세	R(t)
1	(정면) 서 있을 때	0.18~0.21
2	(정면) 팔을 벌리고 서 있을 때	0.18~0.21
3	(정면) 다리를 벌리고 서있을 때	0.49~1.05
4	(정면) 걸어다닐 때	0.20~0.24
5	(정면) 스트레칭을 하고 있을 때	0.34~0.43
6	(옆) 서 있을 때	0.11~0.15
7	(옆) 팔을 앞으로 뻗고 서 있을 때	0.11~0.21
8	(옆) 걸어다닐 때	0.37~0.47
9	(정면) 양반다리로 앉았을 때	0.78~0.89
10	(정면) 다리를 벌리고 앉았을 때	1.80~2.04
11	(옆) 양반다리로 앉았을 때	0.49~0.68
12	(옆) 다리를 뻗고 앉았을 때	1.31~1.80
13	쓰러졌을 때, 누울 때	5.0 이상
14	카메라에 머리를 두고 쓰러졌을 때	1.46 이상
15	카메라에 발을 두고 쓰러졌을 때	1.2 이상
16	대각선으로 쓰러졌을 때,	3.0 이상

팔을 제외하였기 때문에 일반적으로 다리를 벌리고 서있는 경우를 제외하면 서 있거나 걸어 다닐 때 (1,2,4,5,6,7,8) 0.45이하의 R(t) 값을 가진다. 또한, 쓰러진 자세(13,14,15,16)는 찍히는 각도에 따라 값의 변동이 심했지만 누워있다는 특징으로 인해 $w(t)$ 의 값이 $h(t)$ 의 값보다 일반적으로 비율이 컸다. 쓰러진 자세 중에서도 $w(t)$ 가 가장 짧게 찍히는 각도인 14,15의 경우에도 R(t)값이 대략 1.2보다는 크다는 것을 알 수 있었다. 이 결과를 통해서 서 있는 자세를 구분하는 임계값 α 값은 0.45, 쓰러진 자세를 구분하는 임계값 β 값은 1.2로 정하였다. β 인 1.2를 넘는 자세에는 쓰러지기 자세 외에도 10,12번 등의 자세들이 있지만, 해당 항목들은 서 있다가 다른 중간 동작없이 1.0초만에 취할 수 없는 자세들이다. 임계값을 이렇게 정함으로써, 서 있다가 앉는 경우나 앉아있다가 눕는 경우는 임계값으로 세워진 조건식을 만족하지 못하기 때문에 서 있다가 쓰러지는 경우만 잡을 수 있게 된다.

◇ 프레임 속도

해당 알고리즘에서는 1.0초를 기준으로 자세를 판단한다. 하지만 1.0초 간격으로 촬영한다면 $t-1$ 초와 t 초 사이(1.0초)인 중간 지점부터 넘어지는 자세가 시작한다면 정확도가 떨어지게 된다. 그렇기 때문에 이 간격을 줄이고자 프레임 속도를 0.5초로 하여, 0.5초마다 자세를 추출하고 대신 1.0초 전의 값을 비교하는 알고리즘에는 영향을 받지 않도록 2 프레임 전의 R(t)값을 사용하는 방법을 사용한다.

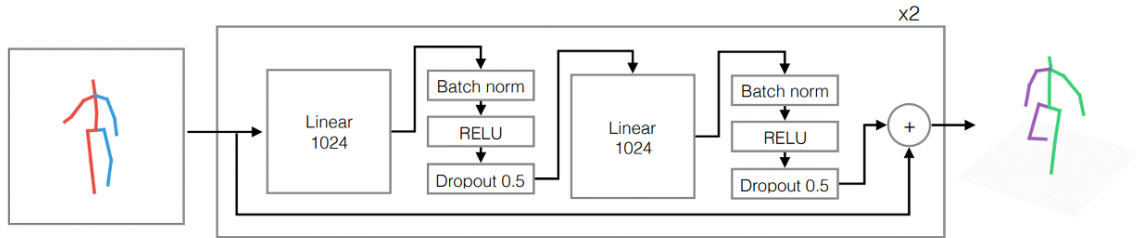
◇ 조건식

앞선 내용을 모두 종합하여 1차적으로 낙상 의심 상황을 판단하는 조건식을 다음과 같다.

if $R(t-2) < 0.45$ and $R(t) > 1.2$:
낙상 의심

$R(t) = \frac{w(t)}{h(t)} = \frac{\text{바운딩 박스의 가로 길이}}{\text{바운딩 박스의 세로 길이}}$
 $R(t-1) = 1$ 프레임 (0.5초전)전의 R(t) 값
 $R(t-2) = 2$ 프레임 (1.0초전)전의 R(t) 값

다. 3D-Pose-Baseline의 이론적 과정



3D-Pose-Baseline은 딥러닝 기반 모델로 주어진 2D 관절 좌표를 3차원으로 변환해주는 모델이다.

$$f^* = \min_f \frac{1}{N} \sum_{i=1}^N L(f(x_i) - y_i)$$

(N : Number of Action, i : Number of Articulation)

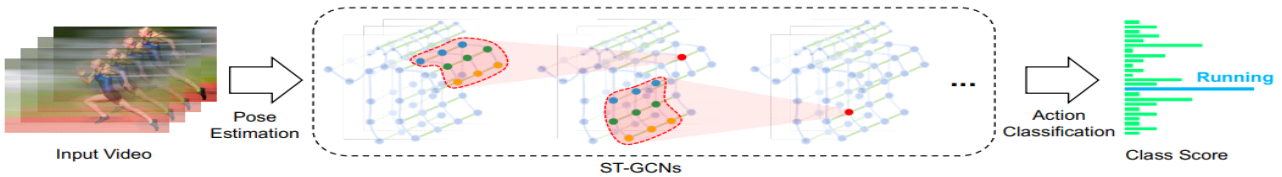
기본적인 수식은 위의 수식과 같이 2차원의 점 $x \in R^{2n}$ 을 딥러닝 모형 $f^* : R^{2n} \rightarrow R^{3n}$ 을 통해 예측된 값 $f(x)$ 와 실제 타깃값(실측값)인 점 $y \in R^{3n}$ 과의 오차가 최소화되도록 하는 함수를 학습하도록 한다.

즉, x_i 는 카메라 parameter들을 알고 있는 상태에서 얻은 2D Ground Truth이거나 2D 자세추정 알고리즘에서 얻은 좌표값이다. 최종적으로 얻은 3차원의 좌표들은 global position이 아닌 hip 관절을 원점 중심으로 해서 이루어져있다.

네트워크의 Building Block은 배치정규화와 드롭아웃, ReLU 활성화함수를 포함하는 선형 레이어고, Residual Connection으로 둘러싸인 2개의 블록으로 2번 반복된다.

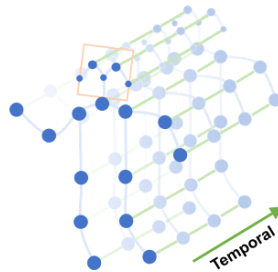
라. ST-GCN의 이론적 과정

행동 인식을 하기 위해 사용하는 데이터셋의 modality는 다양하다. RGB 영상, depth 영상, 움직임의 방향이 표현된 optical flow, 그리고 skeleton data가 있다. ST-GCN은 skeleton을 이용한 행동인식 분류 모델이다. skeleton data의 경우 자연스럽게 사람의 관절의 위치를 시간축에 따라 표현되어진다. 이러한 구조의 data로 학습을 진행하기 위해 이전의 방식에서는 한 frame에서의 관절의 위치 값들을 하나의 feature vector로 사용해서 학습을 시켰었다. 이 방식의 경우, 행동인식에서 관절 간의 관계성 또한 매우 중요함에도 불구하고 해당 정보가 학습에 포함되지않는다. 해당 모델은 공간적 관계성을 찾아내기 위해 GCN(Graph Convolutional Neural Network)를 이용한다.



개략적인 모델의 개요는 위의 그림과 같다. 동영상으로부터 skeleton을 추출하고, skeleton data를 그래프 형태로 만든다. 각각의 joint들을 노드로 만들고, 각각의 노드가 이어지는 부분(공간, 시간)을 edge로 연결한다. 총 9개의 ST-GCN 모듈을 통해 feature를 추출한 후, Softmax 함수를 이용하여 행동을 분류한다.

◇ Skeleton Graph Construction



위의 그림과 같이 Skeleton data를 그래프로 만든다. 모든 frame에서의 skeleton의 joint들이 하나하나의 vertex로 구성되고, 한 frame 내에서의 주변 joint들과 edge로 연결된다. 또한 frame 사이에서 같은 joint끼리도 edge로 연결된다. 이러한 그래프 구조는 dataset의 종류와 관계없이 모두 동일하게 적용 가능하다.

그래프에서 vertex의 집합은 다음과 같이 표기된다 T는 frame의 개수, N은 한 Skeleton에서 joint의 개수를 의미한다.

$$V = \{v_{ti} | t = 1, \dots, T, i = 1, \dots, N\}$$

edge의 경우 두 종류로 나뉘는데 한 frame안에서 joint의 edge는 다음과 같다.

$$E_S = \{v_{ti}v_{tj} | (i, j) \in H\}, \quad H \text{ is the set of naturally connected human body joints}$$

$$E_F = \{v_{ti}v_{(t+1)i}\}$$

◇ Graph Convolutional Neural Network

ST-GCN을 설명하기 전, GCN에 대해 먼저 설명하겠다. 한 장의 프레임을 가지는 GCN은 다음과 같은 수식을 가진다. 시점 τ 에서의 한 프레임에서 N 개의 joint node V_t 와 edge $E_S(\tau) = \{v_{ti}v_{tj} | t = \tau, (i, j) \in H\}$ 를 가질 것이다. 이때, kernel size가 $K * K$ 이며, 채널의 수 c 를 가지는 input 특성맵 f_{in} 이 주어졌다고 하자. 공간위치 x 에서의 단일 채널의 output 값은 다음과 같이 쓸 수 있다.

$$f_{out}(x) = \sum_{h=1}^K \sum_{w=1}^K f_{in}(p(x, h, w)) \cdot w(h, w)$$

여기서, sampling function $p : Z^2 * Z^2 \rightarrow Z^2$ 한 위치 x 로부터 주변 픽셀들을 가져오는 함수이다. weight function $w : Z^2 \rightarrow R^c$ 는 위의 sampling function에서 얻어와진 픽셀들에 weight를 더하는 함수이다.

◇ Sampling function

이미지에서는 sampling function $p(h, w)$ 는 중심 위치 x 에 대하여 이웃하는 픽셀들이라고 정의한다. 그래프에서는 이미지에서와 유사하게 정의하며 다음과 같은 이웃 집합 B 를 정의한다.

$$B(v_{ti}) = \{v_{tj} | d(v_{tj}, v_{ti}) \leq D\} \text{ of a node } v_{ti}$$

여기서 $d(v_{tj}, v_{ti})$ 는 v_{tj} 로부터 v_{ti} 로 도달하는 어떠한 길들 중에 가장 짧은 길이를 의미한다. 여기서 D 는 이웃을 정의하는 거리의 threshold라고 생각할 수 있다.

정리하면, 그래프에서의 sampling function $p : B(v_{ti}) \rightarrow V$ 는 다음과 같이 쓸 수 있다.

$$p(v_{ti}, v_{tj}) = v_{tj}$$

◇ Weight function

이미지의 경우 중심 위치 주위로 이미 정해진 grid가 존재하기 때문에, 거리적으로 weight가 정해져있다. 반면 그래프의 경우 어떤 것이 더 가까운지 정의하기 어렵기 때문에, 해당 모델에서는 모든 이웃 노드들에게 unique label을 주는 것 대신에, 특정 joint node v_{ti} 의 이웃 집합 $B(v_{ti})$ 을 고정된 K 개의 부분 집합으로 분할하는 방법을 선택한다. 각 부분 집합은 숫자 label을 가지고 있다.

그러므로, 이웃 노드들을 부분 라벨 집합으로 넣는 매핑함수 $l_{ti} : B(v_{ti}) \rightarrow \{0, \dots, K-1\}$ 가 존재한다.

결국, weight function $w(v_{ti}, v_{tj}) : B(v_{ti}) \rightarrow R^c$ 는 (c, K) 차원 tensor를 인덱싱하여 실행되거나, 다음과 같이 정의된다.

$$w(v_{ti}, v_{tj}) = w(l_{ti}(v_{tj}))$$

◇ Spatial Graph Convolution

앞서 설명한 이미지에서의 GCN을 그래프 형태로 다시 쓰면 다음과 같다.

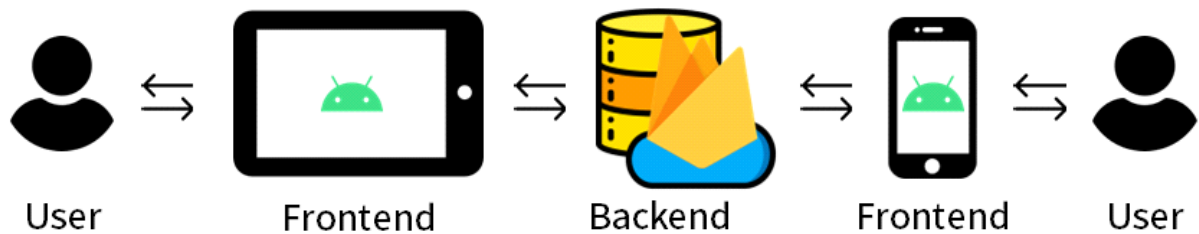
$$f_{out}(v_{ti}) = \sum_{v_{tj} \in B(v_{ti})} \frac{1}{Z_{ti}(v_{tj})} f_{in}(p(v_{ti}, v_{tj})) \cdot w(v_{ti}, v_{tj})$$

위의 식에서 $Z_{ti}(v_{tj}) = |\{v_{tk} | l_{ti}(v_{tk}) = l_{ti}(v_{tj})\}|$ 는 대응하는 부분집합의 기수와 동일하다. 해당 항은 출력에 다른 부분 집합들의 기여도를 균형 잡기 위해 넣었다. 앞서 정의한 sampling function과 weight function을 위의 식에 대입하면 최종적으로 다음과 같은 식을 만족한다.

$$f_{out}(v_{ti}) = \sum_{v_{tj} \in B(v_{ti})} \frac{1}{Z_{ti}(v_{tj})} f_{\in}(v_{tj}) \cdot w(l_{ti}(v_{tj}))$$

4. 소프트웨어 설계

가. 시스템 설계



◇ Frontend

- Android Studio를 이용하여 구현한다.
- 회원가입, 로그인, 주요 기능을 위한 화면을 생성하고, User가 발생시키는 이벤트를 받아 Backend단으로 넘긴다.

◇ Backend

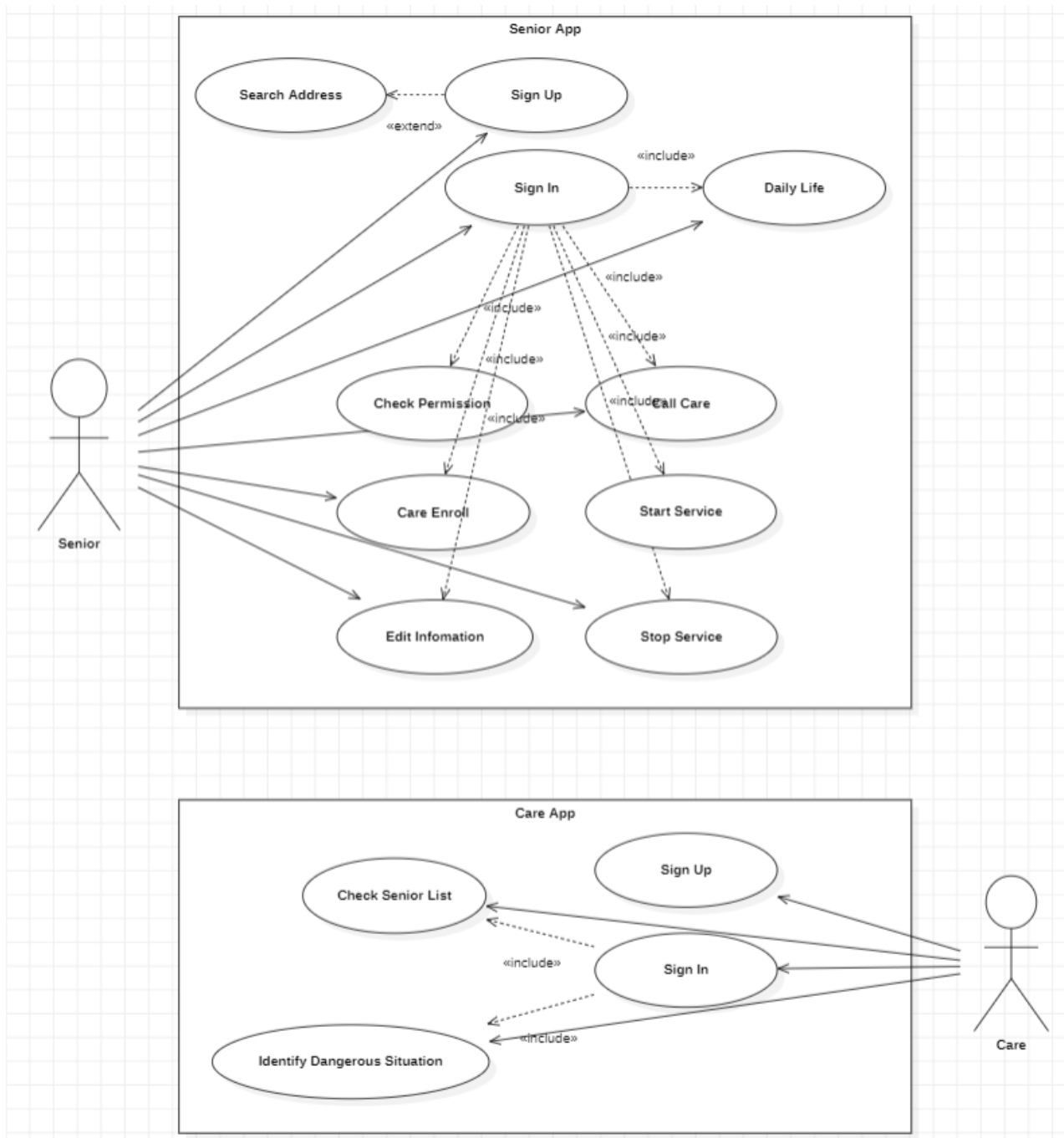
- Firebase가 제공하는 기능들을 이용하여 구현한다.
- 실시간 데이터베이스 기능을 이용하여 User들의 정보를 저장/수정한다.
- 인증 기능을 이용하여 User 인증을 진행한다.
- 스토리지 기능을 이용하여 이미지를 저장/불러오기한다.
- 호스팅 기능을 이용하여 주소API를 가져와 주소검색 액티비티를 제공한다.
- Frontend단에서 온 이벤트를 처리하고 결과값을 형태에 맞춰 반환한다.
- 답러닝 모델은 서버에서 처리하지 않고 앱 내부에서 처리하여 결과만 Backend로 넘긴다.

◇ 장점

- 모든 데이터는 Backend에 모이기 때문에 데이터의 구성과 관리 측면에서 유용하다.
- 서버를 따로 만들어서 쓰는 것이 아니라 제공하는 서비스를 통해 쓰기 때문에 유지보수에 좋고 편리성이 뛰어나다.

나. 유스케이스

◇ 유스케이스 다이어그램



◇ 액터 목록

액터명	구분	설명
Senior	사용자	AI 스피커로 Senior 앱을 사용하는 사용자
Care	사용자	스마트폰으로 Care 앱을 사용하는 사용자

◇ 유즈케이스 설명

이름	CheckPermission	관련 액터	Senior
설명	사용자가 권한 설정을 하는 유즈케이스		
사건흐름	기본 흐름	1. 권한 설정을 요구하는 박스가 생성된다. 2. 권한 설정 허용 버튼을 누른다.	
	대안 흐름	1.a. 권한 설정을 이미 모두 했다면 박스는 생성되지 않고 바로 유즈케이스를 종료한다.	
	예외 흐름	2.a. 허용을 누르지 않고 취소 버튼이나 뒤로 가기를 한다면 앱은 종료된다.	
조건	사전 조건	SignIn	
	사후 조건	없음	
참고사항			

이름	SignUp	관련 액터	Senior / Care
설명	사용자가 회원가입을 하는 유즈케이스		
사건 흐름	기본 흐름	1. 사용자 정보를 입력한다. 2. 하단의 “가입하기” 버튼을 누른다.	
	대안 흐름	1.a. Senior인 경우, 주소를 포함한 모든 정보를 필수로 입력하여 가입한다. 1.b. Care인 경우, 주소를 제외한 모든 정보를 필수로 입력하여 가입한다.	
	예외 흐름	2.a. 이미 존재하는 ID, 비밀번호일 경우 오류 메시지를 출력한다. 2.b. 사용자 정보 형식이 올바르지 않은 경우 오류 메시지를 출력한다.	
조건	사전 조건	없음	
	사후 조건	로그인 액티비티로 이동한다.	
참고사항			

이름	SearchAddress	관련 액터	Senior
설명	Senior 사용자가 주소를 검색하는 유즈케이스		
사건흐름	기본 흐름	1. 주소를 검색한다. 2. 주소를 선택한다.	
	대안 흐름	없음	
	예외 흐름	없음	
조건	사전 조건	없음	
	사후 조건	없음	
참고사항			

이름	EditInfomation	관련 액터	Senior
설명	Senior 사용자가 자신의 정보를 수정하는 유즈케이스		
사건 흐름	기본 흐름	1. 수정할 정보를 입력한다. 2. 하단의 수정 버튼을 눌러 정보를 수정한다.	
	대안 흐름	없음	
	예외 흐름	2.a. 수정한 사용자 정보의 형식이 올바르지 않은 경우, 오류 메시지를 출력한다.	
조건	사전 조건	SignIn	
	사후 조건	없음	
참고사항			

이름	SignIn	관련 액터	Senior / Care
설명	사용자가 로그인을 하는 유즈케이스		
사건흐름	기본 흐름	1. 사용자가 아이디와 비밀번호를 입력한다. 2. 로그인 버튼을 눌러 로그인한다.	
	대안 흐름	2.a. Senior 앱에서 보호자 정보를 등록했다면 메인 액티비티로 이동한다. 2.b. Senior 앱에서 보호자 정보를 등록하지 않았다면 보호자 등록 창으로 이동한다. 2.c. Care앱에서 로그인했을 경우 메인 액티비티로 이동한다.	
	예외 흐름	2.a. 아이디나 비밀번호가 올바르지 않은 경우 오류 메시지를 출력한다.	
조건	사전 조건	없음	
	사후 조건	없음	
참고사항			

이름	CareEnroll	관련 액터	Senior
설명	Senior 사용자가 보호자를 등록하는 유즈케이스		
사건흐름	기본 흐름	1. 사용자가 보호자 아이디를 입력한다. 2. 등록 버튼을 눌러 정보를 등록한다.	
	대안 흐름	없음	
	예외 흐름	2.a. 존재하지 않는 보호자 아이디를 입력했을 경우, 오류 메시지를 출력한다.	
조건	사전 조건	SignIn	
	사후 조건	없음	
참고사항			

이름	StopService	관련 액터	Senior
설명	Senior 사용자가 서비스를 중지하는 유즈케이스		
사건흐름	기본 흐름	1. 사용자가 “서비스 중지” 버튼을 누른다. 2. 서비스가 중지된다.	
	대안 흐름	없음	
	예외 흐름	없음	
조건	사전 조건	SignIn, 서비스가 동작 중인 경우	
	사후 조건	없음	
참고사항			

이름	StartService	관련 액터	Senior
설명	Senior 사용자가 서비스를 동작시키는 유즈케이스		
사건 흐름	기본 흐름	1. 사용자가 “서비스 동작” 버튼을 누른다. 2. 서비스가 동작한다.	
	대안 흐름	없음	
	예외 흐름	없음	
조건	사전 조건	SignIn, 서비스가 중단된 경우	
	사후 조건	없음	
참고사항			

이름	CallCare	관련 액터	Senior
설명	Senior 사용자가 보호자를 호출하는 유즈케이스		
사건 흐름	기본 흐름	1. 사용자가 보호자 호출 버튼을 누른다. 2. 보호자 앱으로 알림을 보낸다.	
	대안 흐름	없음	
	예외 흐름	2.a 인터넷에 연결되지 않은 경우에는 메시지를 보낼 수 없기 때문에 오류 메시지를 출력한다.	
조건	사전 조건	SignIn, 인터넷이 연결되어있는 경우	
	사후 조건	없음	
참고사항			

이름	DailyLife	관련 액터	Senior
설명	Senior 사용자가 일상생활을 하는 유즈케이스		
사건흐름	기본 흐름	1. 사용자가 단말기 앞에서 행동을 취한다. 2. 카메라 프리뷰 화면에 현재 행동이 출력된다.	
	대안 흐름	1.a. 낙상이 의심되면 알림 메시지를 보호자 측에 보낸다.	
	예외 흐름	없음	
조건	사전 조건	SignIn, 서비스가 동작중인 경우	
	사후 조건	없음	
참고사항			

이름	CheckSeniorList	관련 액터	Care
설명	Care 사용자가 관리하는 Senior를 보여주는 유즈케이스		
사건흐름	기본 흐름	1. 관리 중인 Senior의 이름, 전화번호와 주소를 띄운다.	
	대안 흐름	없음	
	예외 흐름	1.a. 관리하는 Senior가 없는 경우 디스플레이에 “없음”이라고 출력한다.	
조건	사전 조건	SignIn	
	사후 조건	없음	
참고사항	Care 앱의 메인 액티비티		

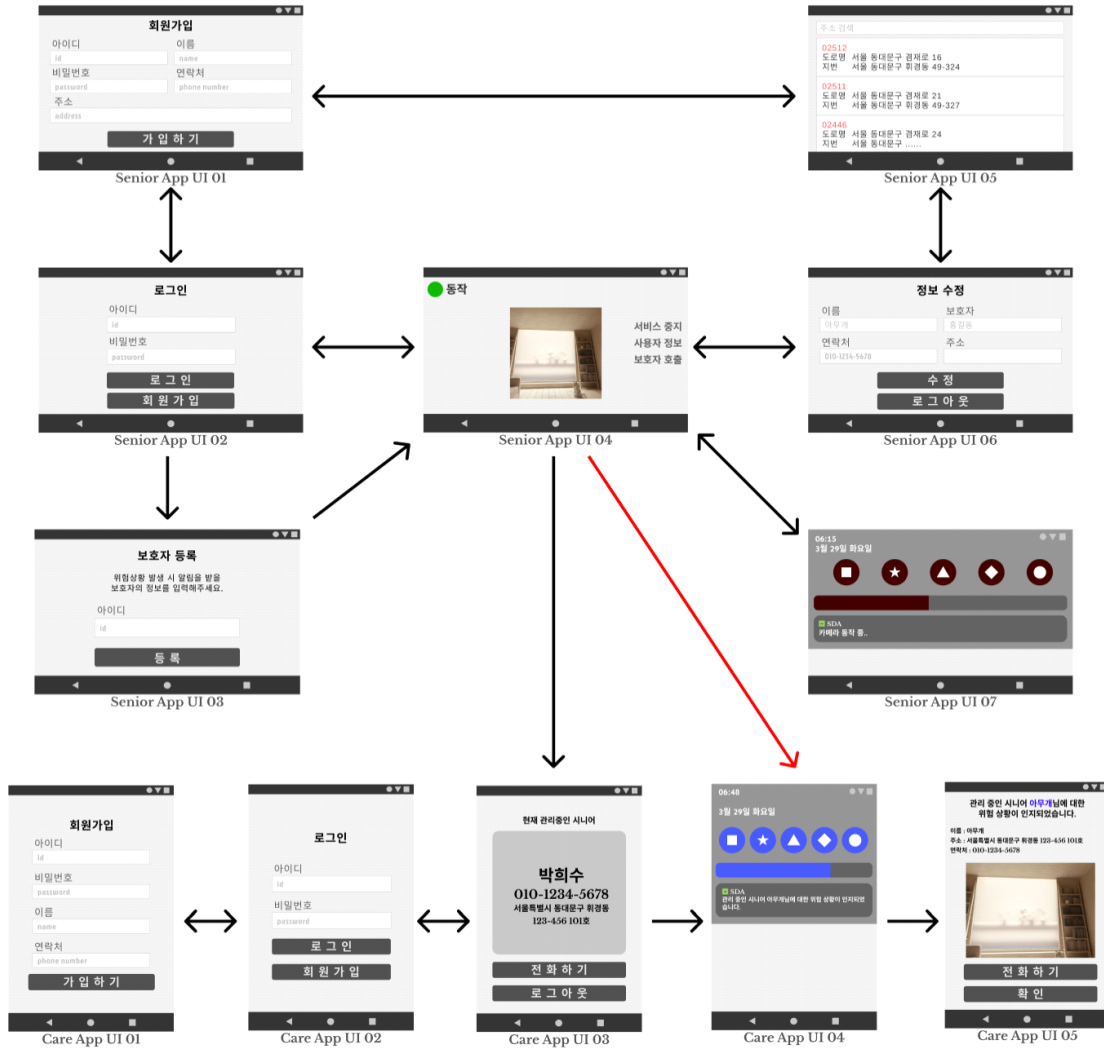
이름	IdentifyDangerousSituation	관련 액터	Care
설명	Care 사용자가 Senior의 위험상황을 인지하게 되는 유즈케이스		
사건흐름	기본 흐름	1. Care 사용자가 알림 메시지를 누른다. 2. Senior에 대한 정보와 위험 상황 당시의 이미지를 확인한다.	
	대안 흐름	없음	
	예외 흐름	없음	
조건	사전 조건	SignIn, Senior가 위험상황 알림을 보낸 경우	
	사후 조건	없음	
참고사항			

다. UI

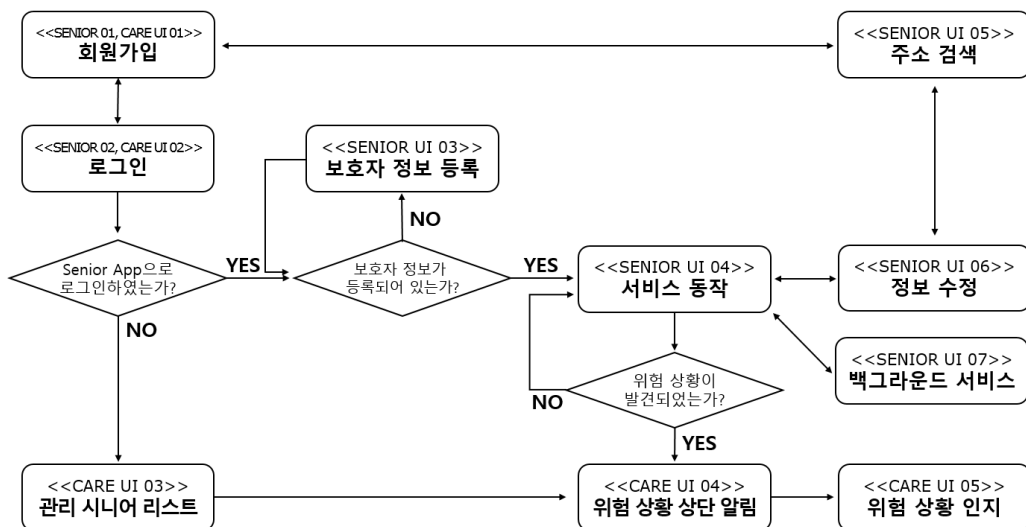
◇ 화면 목록

UI	화면	설명	구성 요소
SENIOR UI 01 / CARE UI 01	회원가입	회원가입 화면	입력창 : ID, Password, 이름, 연락처, 주소(시니어앱) 버튼 : 가입하기
SENIOR UI 02 / CARE UI 02	로그인	로그인 화면	입력창 : ID, Password 버튼 : 로그인, 회원가입
SENIOR UI 03	보호자 정보 등록	위험 상황 발생 시 알림을 받을 보호자 정보 등록	입력창 : 보호자 아이디 버튼 : 등록
SENIOR UI 04	서비스 동작	카메라가 작동 중이고 AI가 분석을 하고 있는 상태 카메라 프리뷰 제공	버튼 : 정보 수정, 서비스 중지, 보호자 호출 텍스트 : 서비스 동작 여부 뷰 : 카메라 프리뷰
SENIOR UI 05	주소 입력	주소 검색 API를 사용하여 도로명, 지번을 검색하여 입력	Javascript Web View 로드
SENIOR UI 06	정보 수정	이름, 보호자, 연락처, 주소 등의 사용자 정보를 수정	입력창 : 이름, 보호자, 연락처, 주소 버튼 : 수정, 로그아웃
SENIOR UI 07	백그라운드 서비스	앱이 꺼지거나 내려가도 백그라운드로 동작하는 상태 (상단 알림 제공)	상단 알림 메시지
CARE UI 03	관리 시니어 리스트	보호자가 현재 관리 중인 시니어 대상자 리스트를 확인할 수 있는 화면	리스트 : 관리 중인 시니어 버튼 : 전화하기, 로그아웃
CARE UI 04	위험 상황 상단 알림	관리 중인 시니어에게 위험 상황이 인지되었을 때 상단 알림으로 발생	상단 알림 메시지
CARE UI 05	위험 상황 인지	낙상이 발생한 시니어에 대한 정보와 당시 이미지 제공	텍스트 : 시니어의 정보 이미지뷰 : 당시 이미지 버튼 : 전화하기, 확인

◇ UI FLOW



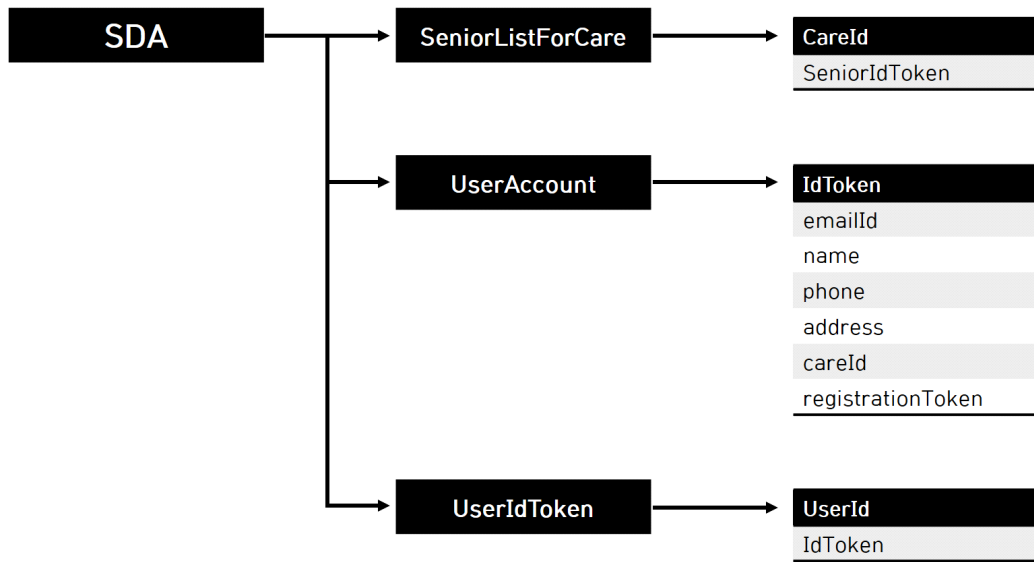
< FIGMA >



< UI FLOW >

라. Firebase Realtime Database

◇ DB 구성도



< DB 구성도 >

SDA	<pre> "SeniorListForCare":{ "CareId":"SeniorIdToken", ... } "UserAccount":{ "IdToken":{ "emailId":"email Id", "name":"user name", "phone":"phone number", "address":"address info", "careId":"care id for senior account", "registrationToken":"registration token value" }, ... } "UserIdToken":{ "userId":"idToken" ... } </pre>
-----	--

마. 모델 설계

Task 1.



Task 2.

