프로젝트 제안서	
주요목표	낮은 클러스터링 성능으로 인한 이상치 극복을 위하여 기존의 비지도 클러스터링 기법을 비지도 클러스터링 기법으로 변환
필요자원	추가사항 없음

순차적 달성 목표

1. 기존 클러스터링의 한계점 및 원인 분석 (달성)

- 동일 인물에 대해 클러스터 분리되는 현상
- 여러 인물이 하나로 합쳐지는 현상
- 노이즈 문제 : 위 3가지 문제는 이미지가 아닌 레이더 결과값을 클러스터링하면서 생기는 문제점으로 파악, 이를 극복 가능한 지도 클러스터링 학습을 제안함
- 판별에 사용하는 frame sequence 길이가 너무 짧은 문제 : 30 sequence 이상, 즉 1초 정도의 길이를 사용하는 것으로 해결 기대

2. Visualizer 새로 구현하여 기존 3D 모델의 plotting 버그 해결

- 일정 시간 이후에 3D 관련 모듈만 중지, AI 구현물과 충돌 가능성 존재
- 3D 모델에 대한 메모리 내에서의 할당 해제가 정상적으로 이루어지지 않는 경우도 의심
- Visualizer 재구현으로 이러한 버그 잡는 것 기대

3. 기존 학습 데이터에 대해 global clustering 결과를 csv table에 추가

- global clustering : 한 프레임에 대한 클러스터링의 결과물에 여러 프레임에 대한 헝가리안을 반영시킨 것
- 더 좋은 클러스터링 기법 또한 탐색이 필요

4. 학습 데이터의 리플레이 기능을 구현 (신규 프로그램)

- Visualizer와 유사한 환경에 입력된 csv를 replay 하는 시스템을 구현
- 3번 과정의 결과물인 csv table을 다시 레이더 영상으로 확인하며, 정지와 시간 이동 기능을 구현할 예정

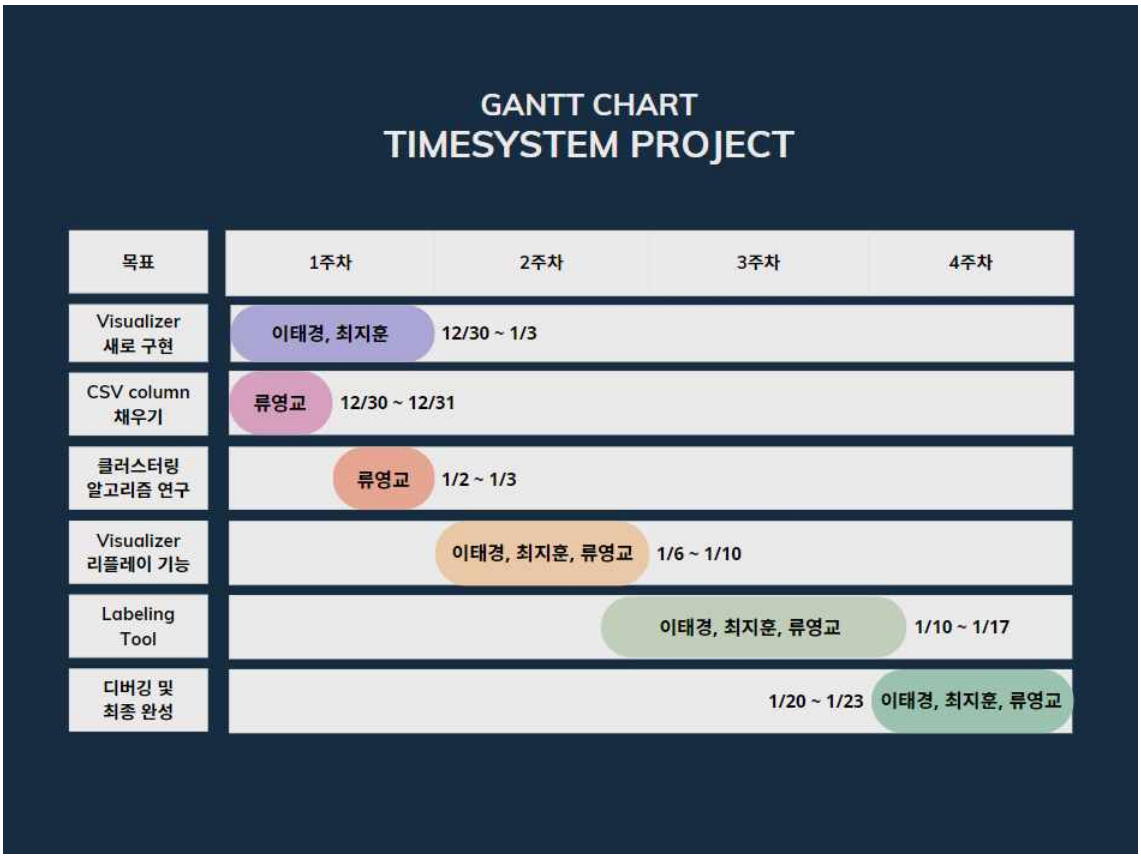
5. 지도 클러스터링에 필요한 labeling tool 구현 (4번 프로그램의 확장)

- 3번의 결과물에 명시된 클러스터 값에 따라 레이더 영상에 보이는 각 클러스터에 색을 할당
- 화면 우측에 클러스터에 해당하는 색을 띄우고, 이를 선택하며 합치는 기능을 제공한다
- 클러스터를 합치게 되면, 기존 비지도 클러스터링 기법이 여러 사람으로 잘못 판별한 대상을 조작자가 판별하여 하나의 사람으로 표시되도록 도와주는 프로그램
- 헝가리안에 의해 연속 sequence에 같은 대상으로 판별된 대상도 동일한 결과를 연쇄적으로 반영하는 등의 기능을 제공
- 그 외의 labeling에 필요한 추가 기능 제공(GUI로 클러스터 번호를 쉽게 편집하는 기능)

향후 목표

- 위 labeling tool 통해 지도 클러스터링에 필요한 학습 데이터를 생성 가능
- 다양한 환경에 대한 시나리오 캡쳐 후, 지도학습 기반 클러스터링 학습
- 학습된 모델 결합한 GRU 모델(긴 sequence 길이 사용) 사용하여 전체 평가 모델 학습 (주목표)
- 실사용 환경에 맞는 형태로 AI 모델과 Visualizer 변경
- 신뢰도 높은 낙상감지 모델로의 개선 (상업적 배포를 고려한 모델)
- 모델 경량화

- 프로젝트 일정 계획



- 현재 구현 상황과 제안 사항의 비교

현재 구현물	제안서 구현물
- AI 학습에 필요한 센서의 출력 값을 .csv 형태로 출력하는 프로그램 - 센서 출력 값에 대해 파일 단위로 label 부여하여 GRU 기반 AI 모델에 학습시키는 코드 - 위 학습을 위해 시나리오 단위로 센서 촬영 진행하고 csv 표를 구분하여 저장함 - GRU 학습 이전에 사람 하나하나를 구분하기 위한 비지도 클러스터링인 DBSCAN을 적용하였으며, 이러한 클러스터링이 time sequence를 넘어 유지되도록 Hungarian Algorithm을 적용함 - 위 학습 완료된 AI 모델을 포함하여 실시간으로 낙상 및 여러 자세를 포착 및 3D 모델 - 로 확인할 수 있는 Visualizer 프로그램	- 레이더 센서의 값 활용하기에 이미지와 달리 사람 하나를 한 개의 클러스터로 구분하지 못하며, 대상이 멈추어 신호가 들어오지 않을 때 Hungarian이 객체를 지속적으로 추적하지 못하는 한계 존재함 - 이를 극복하기 위한 AI 기반 지도 클러스터링 알고리즘이 필요함을 확인하였으며, 이 과정에 필수적인 labeling tool을 개발하는 것을 제안함 - 이러한 툴 개발을 순차적으로 개발함과 동시에 기존 구현물의 한계점을 개선할 수 있는 계획을 따름 -> 1페이지 프로젝트 제안서에 세부 내용 설명됨

- 각 프로세스 별 세부 구현 사항 및 다른 시스템과의 비교 사항

1. Visualizer

- Ti에서 예제로 제공하는 Visualizer에 회사 측 요구사항인 multi-class, multi-label classification과 사람이 이를 쉽게 받아들일 수 있는 그래픽 기반 UI / UX를 적용함

- 사람의 낙상이 감지되는 경우 해당하는 인물에 대해 컬러 레이블의 형태로 표시되어 경고하는 형식으로 구성되어 있다. 다른 자세와 다르게 3D 모델이 아닌 컬러 레이블로 표시되는 이유는 실시간 낙상 모니터를 통해 확인하지 못하더라도 이를 지속적으로 표현하여 보고하는 시스템을 구현할 수 있으며, 기존 프로젝트에서 구현한 구현물을 계승하는 의미도 가지기에 이와 같이 구현하였다.
- AI에 의해 판별된 사람의 자세는 3D 모델을 통해 직접 표현되며, 이는 기존의 cloud point와 겹쳐 표현된다. 기존 cloud point를 그대로 사용하지 않는 이유는 cloud point만 표현되는 경우 사람이 화면을 보았을 때 사람의 자세를 판별하기 어려우며, 무엇보다 사람이 하나인지 둘인지, 사람인지 노이즈인지 조차 판별이 어려운 상황 많았기에 이를 개선하기 위한 UX로 이를 택하였다.
- 해당 Visualizer의 경우 20 ~ 30초 작동 이후에 멈추는 현상이 발견되어 개선이 필요하다.

2. 학습 데이터 생성 프로그램

- 기존 Visualizer에서 내부적으로 처리되는 데이터를 추출한 뒤, AI 학습에 용이한 csv 파일 형태로 변환시키는 프로그램이다. 기존 프로그램 재사용이 아닌 별도의 프로그램으로 재구현한 이유는, 기존 Visualizer의 많은 자원 사용량을 단축하면서도 오류로 인한 중단 가능성을 배제하며 시스템 구조 자체를 학습하여 새로운 Visualizer 개발에도 도움이 될 방법 찾기 위함이다.
- 해당 프로그램은 1학기에 진행된 프로젝트와 비교하였을 때, 자신이 사용한 펌웨어나 출력 형식과 맞지 않더라도 펌웨어 무관하게 모든 출력을 받아서 저장하기에 AI 학습에 용이한 모든 feature를 다룰 수 있다.
- 향후 프로젝트 이어가면서 다른 펌웨어 사용 가능성을 염두에 두어 펌웨어를 선택하는 기능과 확장에 용이한 구조로 클래스 설계하였으며, 현재 2가지 펌웨어를 지원하는 상태이다.
- Visualizer와 다르게 센서가 리셋되지 않은 상태에서의 전송에 프로그램 멈추는 현상을 해결하고, ini 파일로 주요 정보를 디스크에 따로 저장하여 연속적인 학습 데이터 수집 과정에 편리성 및 효율성을 증가시킨 설계를 적용하였다.

3. AI 모델

- 학습 때 csv 파일의 형식이 기존 Velocity, Position, Height만 사용하던 것에서 pointNum 단위의 세부 데이터가 담긴 csv 파일로 변경하였다. 이전에 사용한 csv 데이터가 pointNum 단위의 데이터를 센서 자체에서 가공해서 만든 데이터였기 때문에 AI 학습에서 저차원의 편향된 데이터를 받아 높은 성능을 가진 복잡한 모델을 구현하는데 한계가 있다고 판단하였기 때문이다. 그래서 변경된 입력 형식이 딥러닝 학습 과정에서 더 효과적일 것이라 생각하였으나, 실제로 좋은 결과를 내었는지는 잘 알 수 없었다. 이전 구현에는 pointNum 자체를 frame 단위로 평균해서 사용하였기 때문에 클러스터링 과정에서 이로 인한 성능 저하가 생겨날 가능성을 염두해 두고 있으며, 모델 개선을 통한 검증이 필요한 부분이다.
- csv 파일 형식이 바뀌면서 데이터를 받아올 때 frameNum이 1부터 시작하지 않거나 순서가 섞이고 값이 1씩 증가하지 않고 갑자기 많이 늘어나는 문제들이 발생하여 이를 정렬하는 코드를 제작할 필요성이 있다.
- 기존의 모델의 경우 시계열 데이터 처리를 위하여 먼저 각 파일을 읽어서 frameNum 단위로 window size = 5, stride = 2의 5 * 5 시계열 데이터를 추출하게 된다. 해당 과정을 거쳐 데이터가 1000개 정도 나오게 되었는데 시계열의 크기를 더 늘리면 데이터가 너무 적어지기에 학습 성능의 하락을 우려하여, sequence 크기를 5로 지정하여 구현하게 되었다.
- 하지만 시계열의 크기를 5로 하면 0.275초 정도의 데이터가 모델의 입력으로 들어가는 것이기 때문에 문제의 여지가 존재한다 (기존 시스템의 한계, 실제 사람의 상태를 판별하기에는 너무 짧은 시간). 이것에 우리가 만든 결과물에서 너무 자세 전환이 빨랐던 이유라고 생각하며 최소한 1초 정도의 정보를 담을 수 있게 크기를 최소한 30으로 변경해야 할 것으로 보인다.
- 데이터의 레이블링은 one-hot encoding 방식을 사용하였다. 레이블이 서로 관련 없는 값들이기 때문에 서로 정보를 공유하지 않게 독립시켜야 하기 때문이다.

- 제안한 프로그램의 구현 세부 사항 및 제안 사유

1. Visualzier 재구현

- 현재 사용하는 visualizer는 구동 후 일정 시간(20~30초) 후에 낙상 감지는 정상적으로 작동하지만, 3D people_tracking이 중지되는 현상이 일어난다. 해당 현상의 원인을 알아내기 위한 실험에서 AI를 접목시키지 않은 visualizer에서는 정상적으로 작동하지만, AI를 접목시킨 visualizer는 위 현상이 발생함을 확인하였다. 이로부터 구현된 AI 모듈과 visualizer가 충돌하는 것으로 추정할 수 있었다. 또는, 3D 모델의 메모리 할당 해제가 정상적으로 이루어지지 않는 경우도 의심이 된다. 이러한 문제를 해결하기 위해서는 구조 자체를 고치는 것이 필요하다고 판단하였다.
- 이러한 프로그램 수정에 있어 가장 큰 문제인 것은 현재 visualizer가 TI사의 visualizer에서 필요한 시스템을 확장하여 구현한 프로그램이라는 점이다. 기본적으로 내부 시스템에 대한 이해도가 낮은 상황이기에 디버깅이 어려워서 버그의 원인을 파악하는것 조차 힘들기에 visualizer 프로그램을 새로 구현하면서 디버깅을 통해 버그의 원인을 확실히 파악하고 이를 제거하고자 한다. 단, 원인 파악과는 별개로 현재의 개발 환경에서 해결이 가능한 문제인지는 확인이 불가능하다.
- 또한 이 과정을 통해 사용하지 않는 데모와 클래스, 함수들을 제거하여 무거워진 프로그램을 경량화시켜서 더 원활하게 작동되도록 하고자 한다.

2. csv 파일에 클러스터링 attribute 추가하는 프로그램

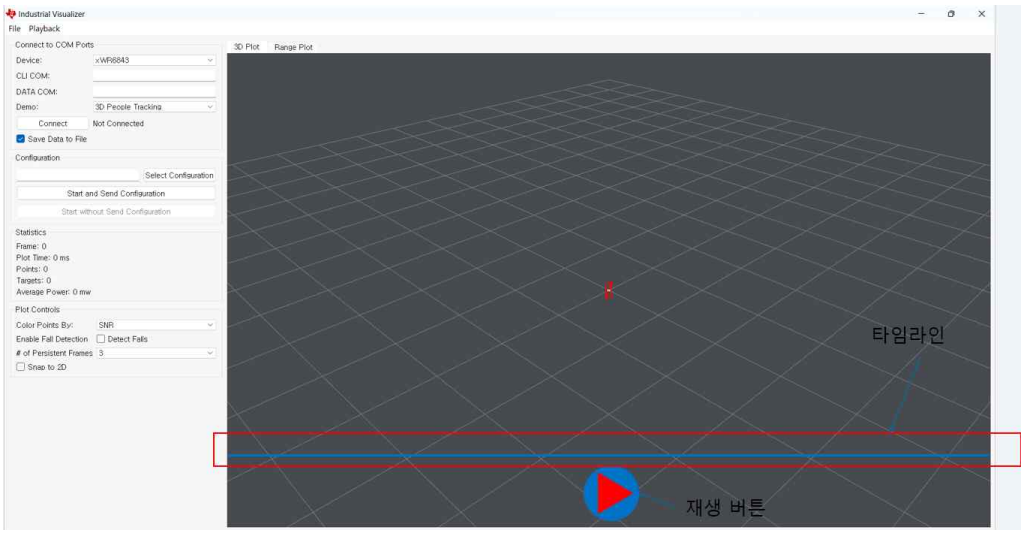
- 지도 학습을 진행하기 위해서는 labeling이 필수적이다. 단, 이 과정을 테이블을 보고 수작업으로 진행하기에는 시간과 노력이 비현실적으로 많이 들기에 이를 빠르고 효율적으로 진행하는 프로그램의 필요성 느꼈으며, 이를 위한 시초로 attribute 추가하는 프로그램을 작성하게 되었다. 또한 이러한 전처리 과정을 거친다면 향후 labeling tool에서 빠른 작업을 가능하게 해 주는 장점도 더해준다. 세부적으로는, 이미 클러스터링 된 결과물에 대해 사람이 인식한 동일 객체를 묶는 형식으로 하나 하나의 포인트에 대해 클러스터 지정하는 방식보다 훨씬 빠른 결과물의 생성이 가능하게 된다.

3. 클러스터링 알고리즘의 연구

- 영상 데이터가 아닌 레이더 센서 기반한 데이터이기에 이상치가 필연적으로 발생, 이를 극복 가능한 알고리즘을 탐색 시도한다.
- 기본적으로 core는 visualizer의 것을 사용하면서 현재 낙상 감지 시스템에 필요한 것(클래스 or 함수)인지, 필요 없는 것인지 사용 가능성을 따져서 core부터 전체적인 layout, 3D_tracking, 낙상 감지 순으로 build up할 예정이다. 구현체는 PySide2 라이브러리를 중심으로 한 python ui 프로그램이다.
- Layout은 현재 사용 중인 프로그램에서 3D와 관련된 부분만을 가져와서 사용할 예정이며, 3D_tracking은 현재 사용 중인 프로그램의 graph_utilities.py 파일과 gui_threads.py 파일에서 필요한 함수와 클래스만 가져와서 한 파일에 합칠 예정이다. 가능하다면 people_tracking.py 데모 파일까지 합치고 싶으나, 실제 구현을 통한 효율성 검증이 필요하다.
- 낙상 감지 관련은 현재 시스템을 그대로 쓰되, 색을 칠하거나 사람 머리를 구로 만드는 등의 부분은 여유 시간에 구현 시도할 예정이다.

4. Visualzier Replay 프로그램

- Visualizer와 유사한 환경에서 csv 파일의 값을 제어하고 확인하는 기능을 제공하는 프로그램이다. 현재의 구현물은 csv 파일의 값이 한 번 정해진 후, 이를 확인하기 위해 csv 문서를 일일히 확인하여야 하며 검증을 위해 동영상 촬영을 해도 비교하기가 어렵다. 그래서 csv 파일의 값을 visualizer와 매칭시켜 csv table과 레이더 영상을 함께 보며 확인함으로서 작업의 효율성 및 모델 정확도의 향상을 기대한다.
- 향후에 구현할 labeling tool과의 연계를 통해 csv table의 global clustering값을 쉽게 바꿀 수 있다는 점에서도 해당 프로그램이 필요하다. 즉, 검증 시에는 동영상 촬영 등을 사용 가능하지만, 실사용에서는 레이더 신호만으로 판별해야 하므로 레이더만으로 csv 검증 또는 확인을 하기 위함이다.
- 아래는 해당 프로그램의 초안이다.



5. Labeling Tool

- 지도 클러스터링 학습에 필요한 학습 데이터를 만드는 tool. 4번의 replay 프로그램에 기반하여 3번의 출력물인 csv 파일에 클러스터링 번호에 따라 구분되는 색을 출력하는 것이 주요 특징이다. 3번 과정에서 처리된 클러스터링의 결과는 이전 진행한 프로젝트와 동일하게 밀도에 기반한 클러스터링으로 레이더 cloud point를 통한 인물 추적에 있어 사람을 적절하게 구분하지 못하는 한계점이 여전히 존재한다. 이를 AI가 적절하게 처리하기 위해서는 사람이 이를 직접 정정하여 알려주는 학습 데이터를 생성하고, 이를 바탕으로 AI 모델이 학습을 하면서 극복하는 것을 기대한다.
- labeling tool은 기존의 replay 프로그램에서 클러스터에 맞는 색을 띄는 point 출력하는 기능 외에도 오른쪽에 추가적인 레이아웃을 추가한다. 해당 레이아웃에서는 클러스터에 해당하는 색들이 표시된 label 들이 나열되어 있으며, 버튼과 같이 선택이나 선택 해제가 가능하다. 선택된 색(클러스터)에 대해서 합치기 버튼을 누르면 두 클러스터는 하나의 클러스터로 합쳐지게 된다. 가령, 레이더를 통해 사람의 팔이 하나의 클러스터로, 사람의 하체가 하나의 클러스터로 인식되게 된다면, AI는 이를 서로 다른 사람으로 인지하게 될 것이다. 이러한 경우를 막기 위해서 두 클러스터를 하나로 인식하도록 학습 데이터를 변경하는 것이며, 지도 클러스터링은 이를 학습하여 단순 밀도 뿐만 아닌 레이더를 통한 사람 인식에 최적화된 형태로 사람을 클러스터링하게 된다.
- 이러한 클러스터링은 시계열 데이터에 대해 연속적으로 적용될 필요가 있기에 위의 합치기 기능은 연속된 클러스터에 대해 동일하게 적용되며, 더 긴 sequence를 가지는 클러스터로 합쳐지게 구현할 것이다. 또한 이러한 연속적 적용의 용이성을 위해, 3번 단계에서 클러스터링은 하나의 파일(10초 이하, 300 프레임 이하)에 대해 각 클러스터가 고유한 클러스터 번호를 가져야 한다. 즉 1~3 프레임에서 감지된 클러스터 1번이 있는 경우, 다른 프레임이나 동일 프레임에 다른 클러스터에 대해 같은 1번 클러스터가 존재하면 안된다는 의미이다. 이러한 구조를 채택시, 합치기 기능에 대한 연속적 할당의 알고리즘을 간단하게 구현가능하다.
- 3번의 학습 데이터 클러스터링에서 헝가리안을 적용을 함에도, 레이더 특성상 행동이 멈춘 사람에 대해서 cloud point가 인식되지 않는 경우에 프레임에 대한 클러스터의 연속성이 끊기는 현상이 생겨 연속적인 클러스터 인식이 불가능한 한계가 생긴다. 예를 들자면, 1~5 프레임동안 관측된 클러스터가 잠시 멈추어 6번 프레임에서 관측이 되지 않고, 다시 7~10 프레임동안 움직임이 관측될 시, AI는 시간의 관점에서 이 둘을 같은 사람으로 인지하는 것이 불가능하다. 예상되는 AI 모델은 약 30 프레임 간격동안 사람의 상태를 판별하기에, 이러한 클러스터 연속성의 파괴는 AI 성능 및 객체 추적에 있어 치명적이다. 이러한 문제점을 해결하기 위해, 지도 클러스터링에 있어서도, 포착된 객체의 위치를 잠시동안 기억하는 기능을 부여하면 좋다고 생각한다. 이를 위해서는 RNN과 같이 time step을 넘어 정보를 확인하는 구조를 채택할 필요가 있으며, 이러한 기능을 발휘하기 위한 지도 학습 데이터 또한 필요하다. 이러한 학습

데이터를 마련하기 위해, labeling tool에도 끊기는 클러스터를 같은 클러스터로 연결하는 기능이 필요하다. 이러한 기능을 구현하는 UI는 테스트를 통한 적절한 UI 확인이 선행되어야 한다.