**응용SW개발 부(모델 - 서버 통신 부) 전체 목차**

**1. 모델부**

(1)이벤트 알고리즘

(2)이미지, 로그, 영상 저장

(3)객체 위치 파악과 경보

(4)AWS 서버로 자료 전송

(5)안전한 AWS 연결을 위한 환경변수 설정

(6)다수 작업의 동시처리를 위한 multi threading

(7)모델 속도 개선

**2. 서버부**

(1)DB 구성

(2)S3 구성

**3. vision 연구부 - 위치를 어디로 해야하지?**

(1)단안 카메라를 통한 거리 탐지

**4. 리뷰/스토리**

(1)모델 - 서버 통신 부 담당으로의 리뷰

(2)vision 연구부로의 리뷰

(3)팀원, 팀장으로의 리뷰, 팀원들에게

+정말 열심히 한 프로젝트. 향후 각자 길이 있지만..

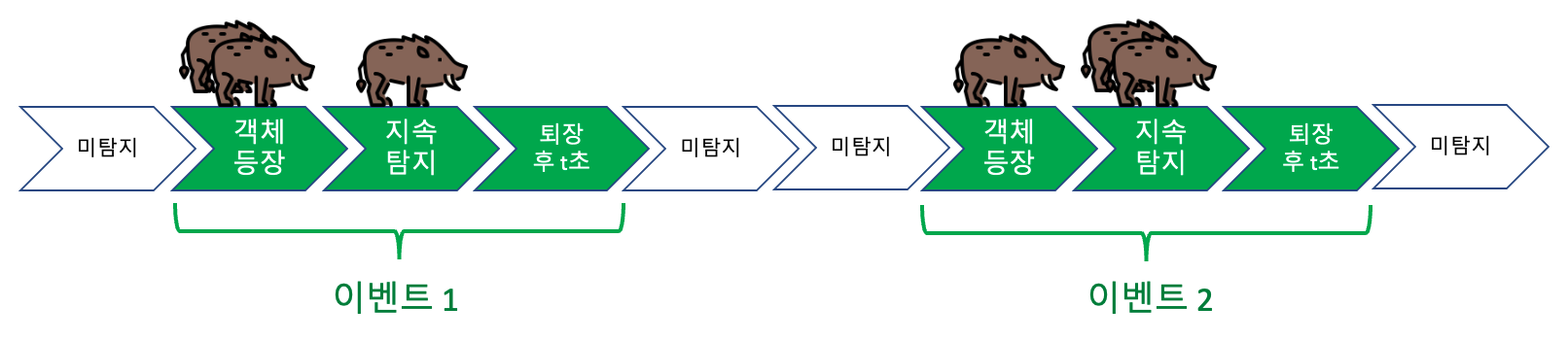
**1. 모델부**

**(1)이벤트 알고리즘**

객체들이 화면 내에 등장하여 탐지되고, 모든 객체가 퇴장할 때까지를 하나의 이벤트로 분류합니다.

비슷한 예로 자동차의 블랙박스를 떠올리시면 적절할 것입니다. 자동차의 블랙박스는 충돌이 발생하기 전 수 초 전 부터, 충돌이 발생한 후 수 초 동안을 이벤트로 기록합니다.

이와 비슷하게 우리의 서비스에서는 객체들이 등장해 탐지되고 모든 객체가 퇴장한 후 수 초까지를 하나의 이벤트로 묶습니다. 이렇게 이벤트를 묶음으로써 객체 탐지의 결과값이 무질서하게 나열되는 것이 아닌, 탐지 시점에 따라 체계적으로 정리될 수 있는 효과를 볼 수 있습니다. 또한 이벤트를 묶음으로써 자료를 저장하는 AWS 데이터베이스와의 통신 횟수를 줄여, 부하를 줄일 수 있습니다.



*<fig. 이벤트 묶임 예시>*

*\* 멧돼지 아이콘 : flaticon.com*

하나의 이벤트는 (1) 객체의 최초 등장(=이벤트 시작), (2) 객체의 지속 탐지, (3) 객체의 퇴장, (4) 이벤트 종료 총 네 단계의 상태로 나뉘게 됩니다. 이를 보조하기 위해 코드 상으로는 event\_type이라는 변수의 선언을 통해 현재의 탐지 상황을 구분함으로써 각각의 상황마다 그에 맞는 작업이 이뤄질 수 있도록 구현하였습니다.

다음은 이벤트를 묶는 알고리즘을 나타낸 의사코드입니다.

```이벤트 알고리즘 의사코드

# event\_type 0 : 탐지 객체가 없는 상시 상태. 기본 상태.

# event\_type 1 : 탐지 객체가 있는 상태

# event\_type 2 : 탐지 객체가 사라지고 event\_time\_limit 초 이내의 상태

# event\_time\_limit : 탐지 객체가 사라지고나서 이벤트 종료까지의 유예 시간

# 최초 탐지

if 탐지된 객체가 1개 이상이고 & event\_type = 0 :

event\_type = 1 로 변경

event\_name = 최초 탐지시각으로 선언

event\_name 폴더 생성

event\_name 폴더에 최초 탐지 이미지, 로그, 동영상 저장 시작

위험지역 진입시 알람

# 지속 탐지

elif 탐지된 객체가 1개 이상이고 & event\_type >= 1 :

event\_name 폴더에 이미지, 로그, 동영상 저장

위험지역 진입시 알람

# 객체가 사라짐

elif 탐지된 객체가 없고 & event\_type = 1 & 객체 퇴장 후 t 초 이하:

event\_type = 2 로 변경

elif event\_type = 2 & 객체 퇴장 후 t초 초과:

이벤트 종료

최종 이미지 저장

동영상 저장 종료

AWS RDS로 로그 전송

AWS S3로 영상, 동영상 전송

```

**(2)이미지, 로그, 영상 저장**

탐지 모델을 통해 탐지된 이벤트의 결과물은 로그, 영상, 그리고 동영상으로 저장됩니다.

이러한 자료를 저장하는 목적은 두 가지가 있습니다.

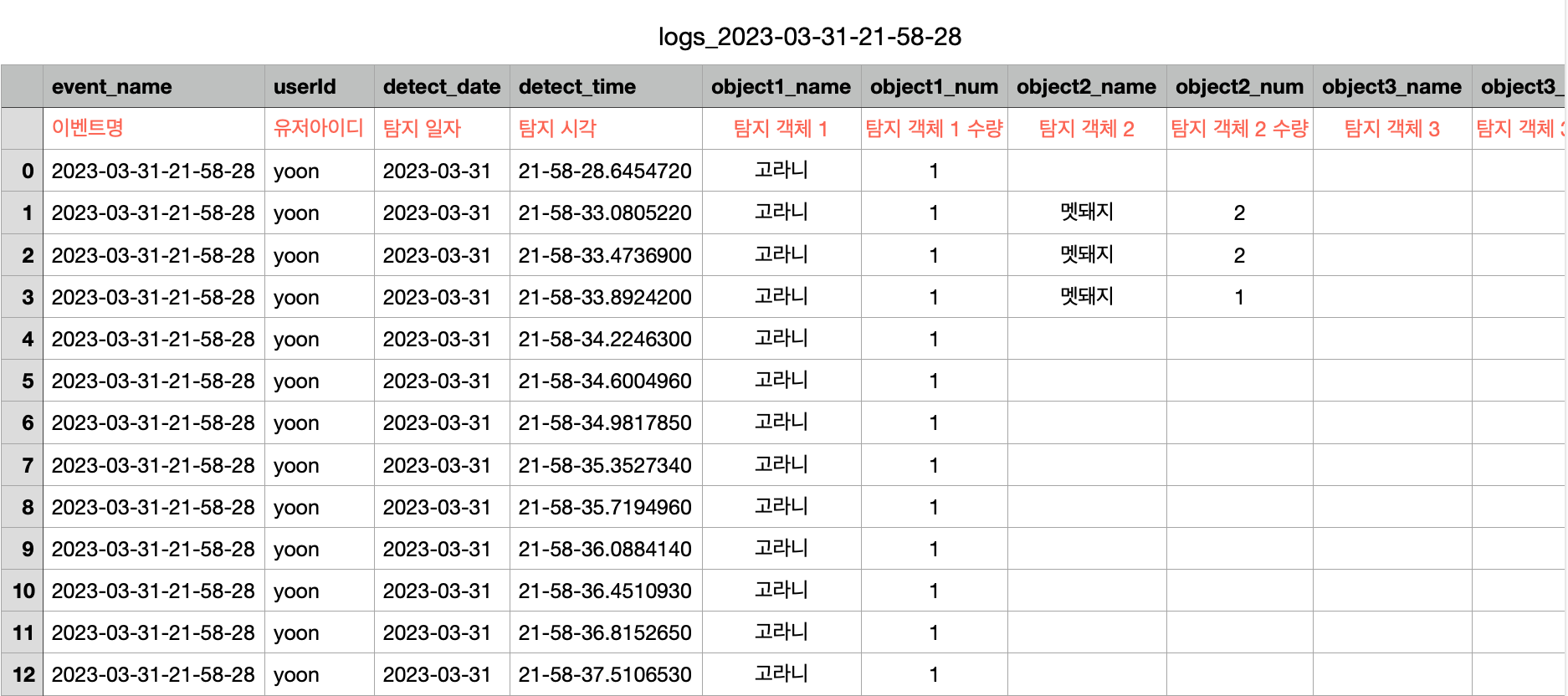
첫째, 서비스 사용자에게 감시 지역에서 발생한 이벤트를 보여주기 위해서입니다. 단순히 객체를 탐지하고 퇴치하는 기능 외에도, 서비스 사용자가 이벤트의 상세 내용을 볼 수 있도록 함으로써 서비스에 대한 신뢰도를 높일 수 있습니다.

둘째, 서비스 제공자가 서비스를 더욱 발전시키기 위해서입니다. 이벤트 상세 내용을 살펴보면서 탐지율과 퇴치율에 대한 정량적, 정성적 평가를 함으로써 서비스에서 부족한 점을 보완하고 새로운 기능 추가에 대한 자가 점검을 할 수 있습니다.

로그 정보에는 탐지 날짜와 시각, 탐지된 객체 종류와 수가 기록됩니다.

영상(이미지)은 최초 등장시, 그리고 최종 퇴장시 총 두 번 저장됩니다. 최초 등장시에는 객체 등장 전의 감시 지역의 모습을 알 수 있으며, 최종 퇴장시 영상에는 탐지된 객체의 이동 경로 및 객체가 지나간 뒤의 감시 지역의 모습을 볼 수 있습니다.

동영상은 객체가 탐지되는 모든 순간에 기록되며, 객체의 이동경로가 화살표로 표시됩니다.



*<fig. 로그 데이터>*



*<fig. 영상 데이터. 왼쪽은 최초 탐지 영상, 오른쪽은 퇴장 후 경로가 표시된 영상.>*

**(3)객체 위치 파악과 경보**

객체의 퇴치 기능은 객체의 탐지 기능과 함께 본 서비스의 핵심적인 기능입니다.

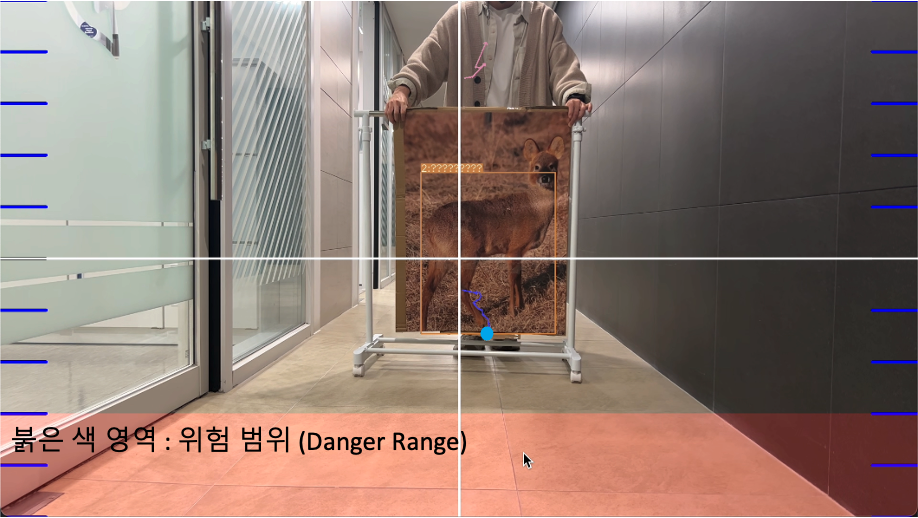
유해한 동물 객체가 사용자가 설정한 위험 범위(Danger Range) 내에 진입하면 경고 알람을 재생하여 이를 내쫓습니다. 퇴치 방법은 각 객체마다 다르게 구현할 수 있지만, 현재는 가장 직관적인 경보 알람 재생 방식을 사용하고 있습니다. 위험 범위는 사용자 및 설치 환경에 따라 맞춤 설정할 수 있습니다.

이 서비스를 위해 두 가지 기술을 접목하였습니다.

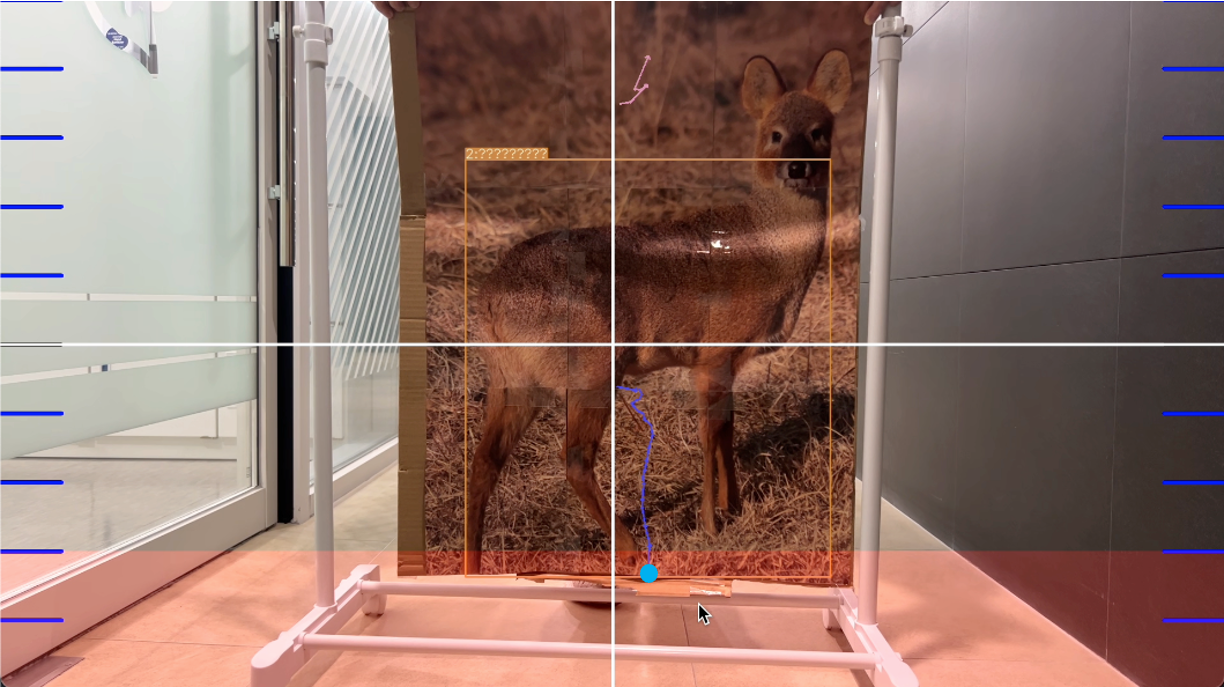
첫 번째는 탐지 객체의 위치값 판단입니다. 탐지 모델을 통해 탐지된 객체는 영상에서 바운딩 박스로 표시됩니다. 우리는 바운딩 박스의 중앙 하단, 즉 객체가 땅을 딛고 서 있는 부분을 객체의 위치로 설정했습니다.

두 번째는 위험 구역의 설정입니다. 영상은 최소 단위인 픽셀로 이루어져 있으며, 각 픽셀에는 좌표값이 부여됩니다. 이 중 높이(y값)을 통해 위험 구역을 설정했습니다.

객체의 위치값이 위험구역 y값보다 높아질 경우(=더 가까이 올 경우) 경보 알람이 울리도록 구현하였습니다.



*<fig. 객체가 위험범위 내 진입하기 전>*



*<fig. 객체가 위험범위 내 진입 함 : 경보 재생>*

**(4)AWS 서버로 자료 전송**

모델부에서 저장된 로그, 영상, 동영상 자료는 이벤트 종료 후 AWS로 자동 전송됩니다. 데이터 전송을 위해 pymysql, boto3 라이브러리를 사용했습니다. 이렇게 전송된 자료는 서비스 사용자에게 이벤트 내용을 제공하여 더욱 현장감 있는 이벤트 결과를 볼 수 있도록, 나아가 서비스에 대한 신뢰도를 높이는 데 사용됩니다. 더불어 서비스 제공자 즉 관리자는 저장된 자료를 장소에 구애받지 않고 열람할 수 있어 업무 대응의 유연성을 높일 수 있습니다.

정형데이터인 로그 데이터는 pymysql로 작성된 코드를 통해 AWS의 RDS로 업로드 되며, 비정형데이터인 영상과 동영상은 boto3를 사용하여 AWS의 S3에 저장됩니다.

로컬에 있는 모델부와 AWS간의 통신에 있어 Access Key 나 RDS 계정정보 등은 AWS CLI와 환경변수 파일(env) 설정을 통해 외부에 노출되지 않고 안전하게 전송됩니다.

**(5)안전한 AWS 연결을 위한 환경변수 설정**

로컬에서 AWS로 자료를 업로드하려면 AWS에 접근할 수 있는 Access Key나 RDS 관리자 계정 정보가 필요합니다. 이러한 Access Key와 계정 정보는 외부로 노출되면 안되는 특성이 있기 때문에 더욱 보안에 신경을 써야 합니다.

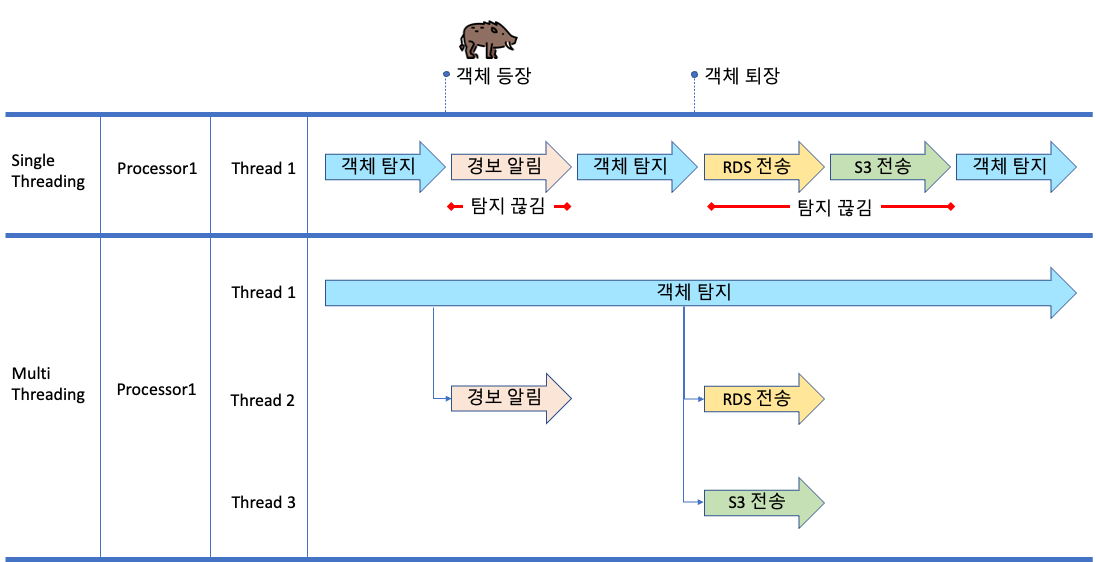
이를 위해 Access Key는 AWS CLI를 통해 로컬 환경 변수에 보관하면서 안전하게 외부에 노출되지 않도록 구현하였으며, RDS 관리자 정보는 환경변수 파일(env) 에 담아 파이썬 코드에서 envdot 모듈을 통해 불러오도록 하여 또한 외부에 노출되지 않도록 구현했습니다.

**(6)다수 작업의 동시처리를 위한 multi threading**

본 서비스의 로컬 모델부에서는 앞서 언급한 Yolo 객체탐지, 경보, AWS 서버로의 자료 전송이 이루어집니다. 이 기능들은 모두 실시간으로, 그리고 끊김 없이 처리되어야 합니다. 예를 들어, 객체가 퇴장하여 이벤트가 종료된 뒤 로그와 영상을 AWS로 전송하는 동안에도 Yolo를 통한 객체 탐지가 끊기면 안됩니다. 이를 위해 기능들 중 하나의 기능이 작동될 동안 다른 기능이 중지되지 않도록 Multi Threading을 통해 설계를 하였습니다.

Thread 는 하나의 프로세스 내에서 실행되는 흐름의 단위를 말하며, Multi Threading은 하나의 프로세스 내에 둘 이상의 흐름을 실행시키는 것을 의미합니다.

본 서비스에서는 Yolo 객체 탐지, 경보 재생, AWS 서버로의 자료 전송을 Multi Threading을 통해 모두 각각의 스레드에 올려 실시간으로, 그리고 동시에도 실행될 수 있게 구현하였습니다. 이를 통해, 서로 다른 작업들이 서로에게 영향을 미치지 않도록 하였으며, 특히 객체 탐지 코드의 실행이 끊기지 않도록 할 수 있었습니다.



*<fig. Single Threading 과 Multi Threading 비교표>*

*\* 멧돼지 아이콘 : flaticon.com*

**(7)모델 속도 개선**



*<fig. 모델 속도 측정 표>*

**2. 서버부**

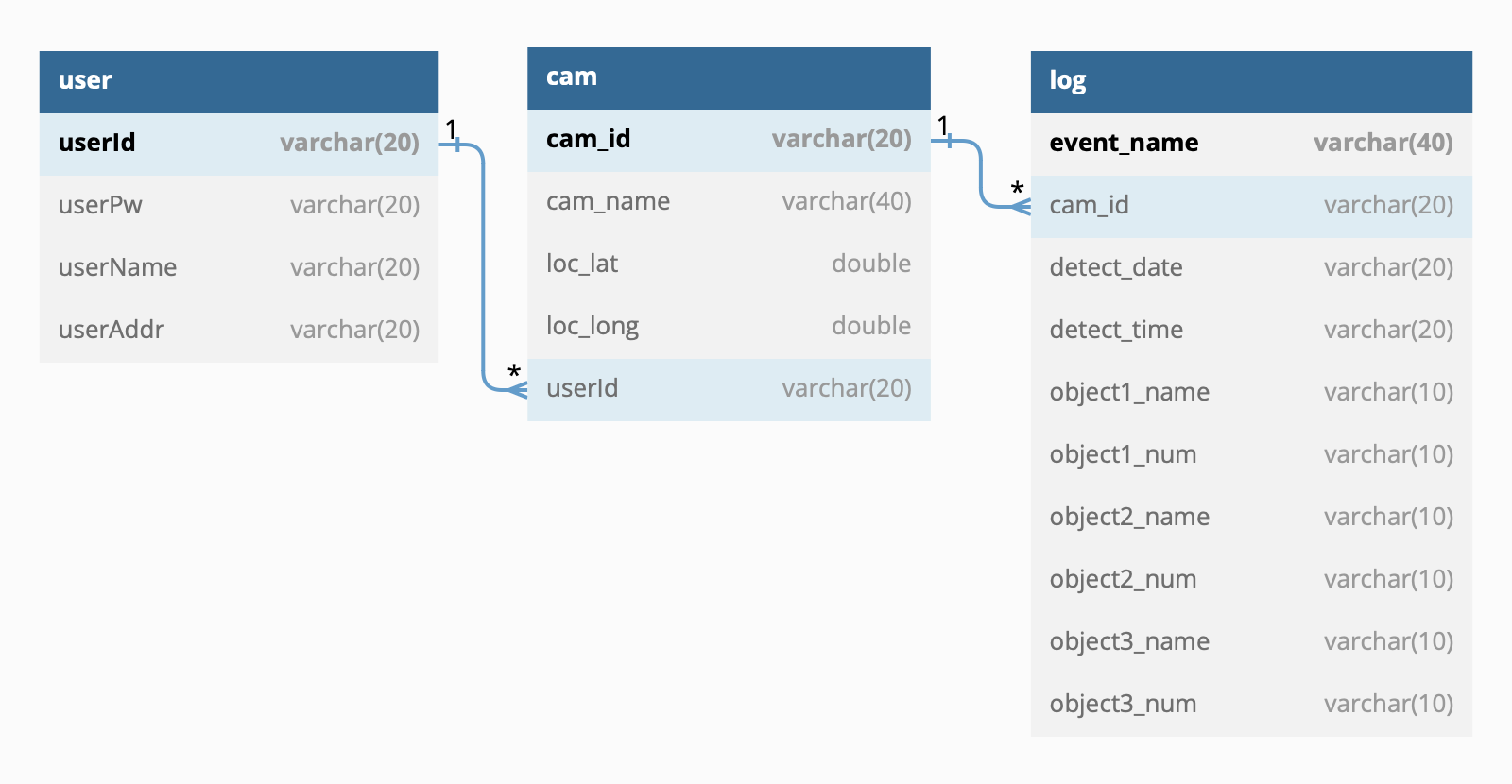
**(1)DB 구성**

DB는 로그를 저장하는 로그 테이블, 서비스 대상인 사용자의 정보를 저장하는 user 테이블, 카메라 정보를 담고 있는 cam 테이블로 구성되어 있습니다.

우리는 서비스 사용자에게 웹을 통해 로그데이터 제공을 염두에 두고 있기 때문에 AWS의 정형 데이터베이스 관리 시스템인 RDS에 DB를 위치시켰습니다.

현재는 객체가 등장했을 때의 로그만을 기록하고 있기 때문에 db.t3.micro 의 최소 사양의 RDS를 구동 중이며, 향후 서비스 발전에 따라 객체 이벤트 뿐 아니라 객체 탐지 IOT 기기의 상태까지 기록될 것을 염두에 뒀을 때엔 더 높은 사양의 RDS 이용이 필요합니다.

아래는 DB의 ERD 입니다.



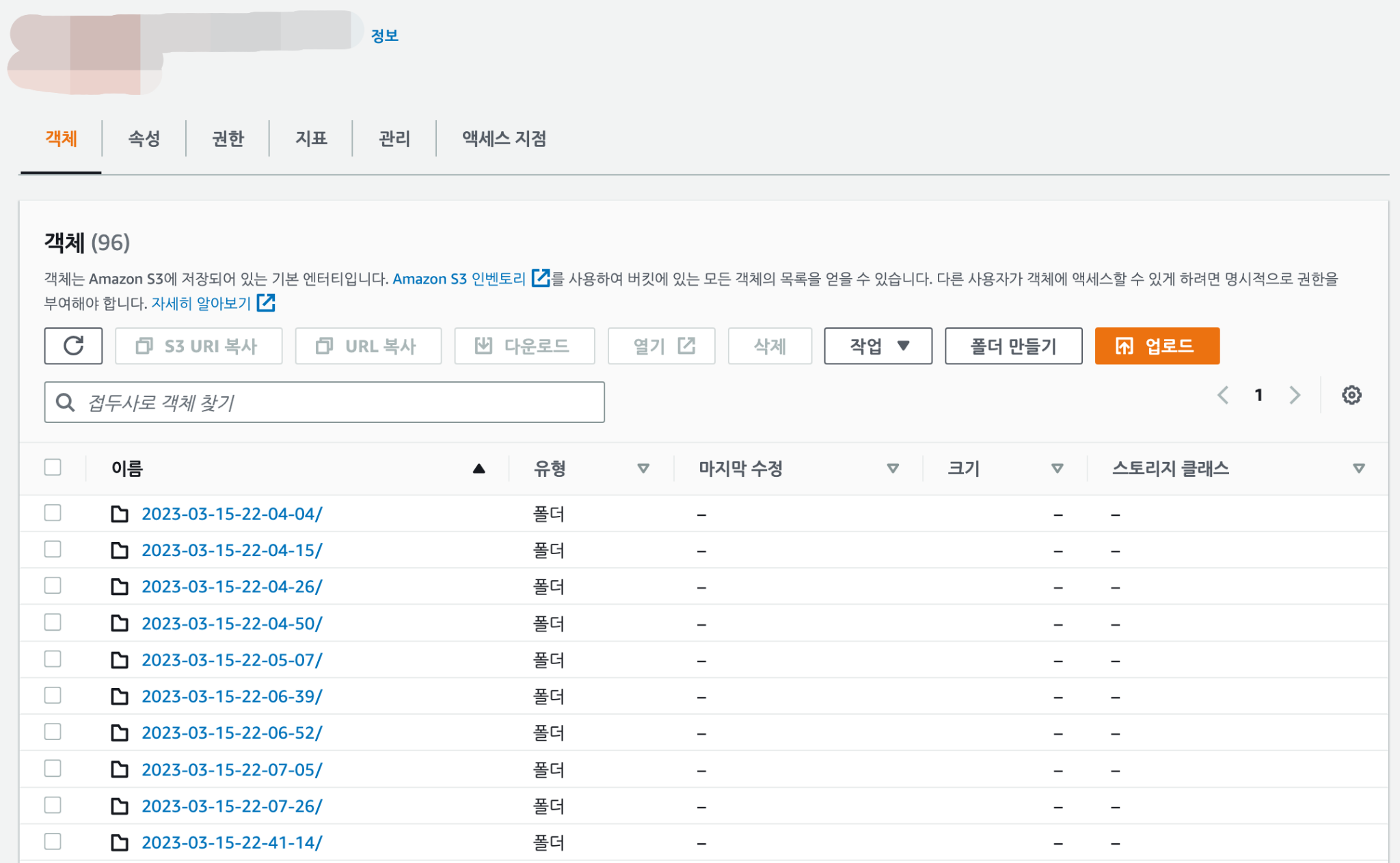
*<fig. ERD>*

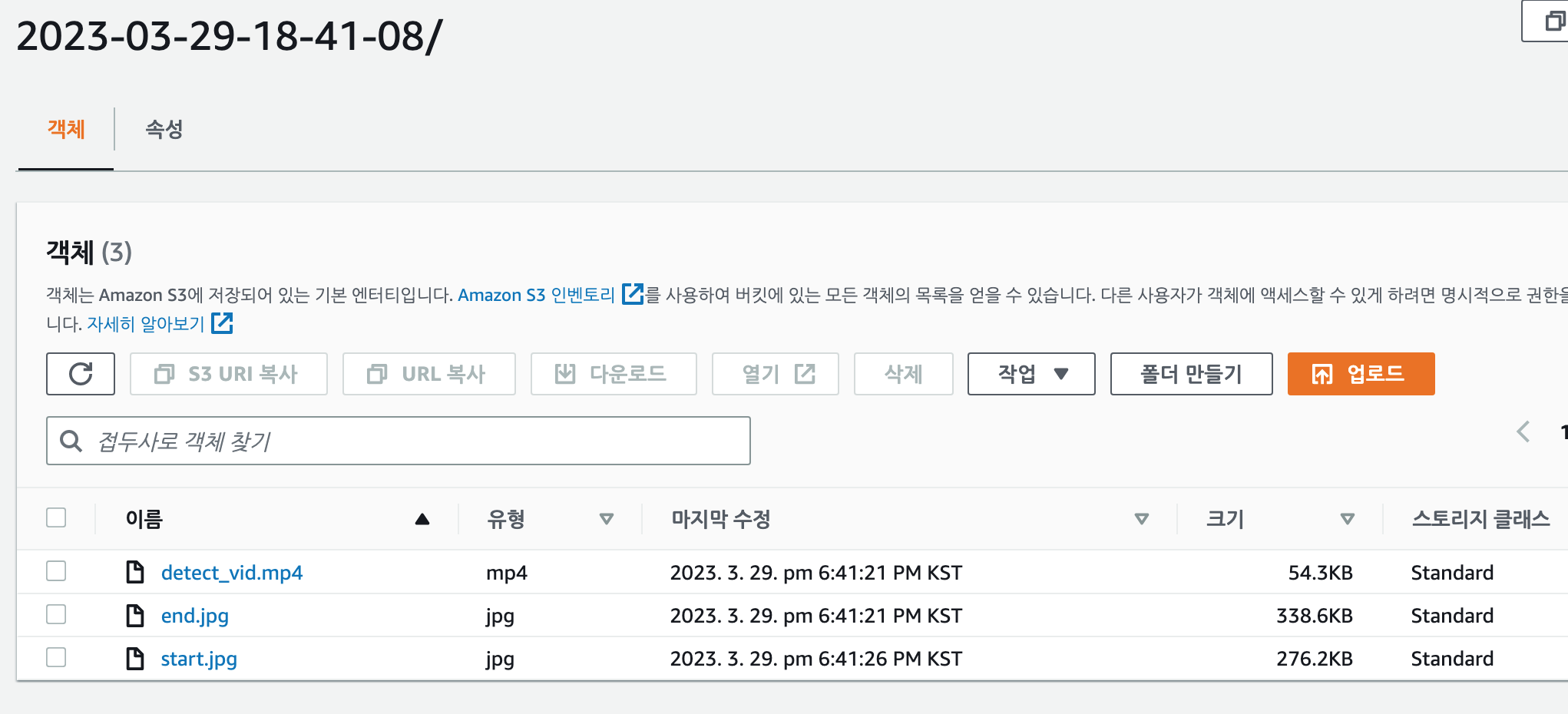
*\* 작성 : dbdiagram.io*

**(2)S3 구성**

이미지와 동영상과 같은 비정형 데이터는 S3에 저장되게 됩니다. S3에 저장되는 자료는 그 키값(주소값)을 일정한 규칙으로 만들어, 사용자가 요청한 자료를 쉽게 가져올 수 있도록 설계했습니다.

S3에 저장된 이미지와 동영상은 웹을 통해 사용자에게 제공됩니다.





*<fig. S3 저장 자료 화면>*

**3. 비전 연구부**

**(1)단안 카메라를 통한 거리 탐지**

본 프로젝트의 “서비스화”를 염두에 뒀을 때, 주요한 점을 아래 두 가지로 잡았습니다.

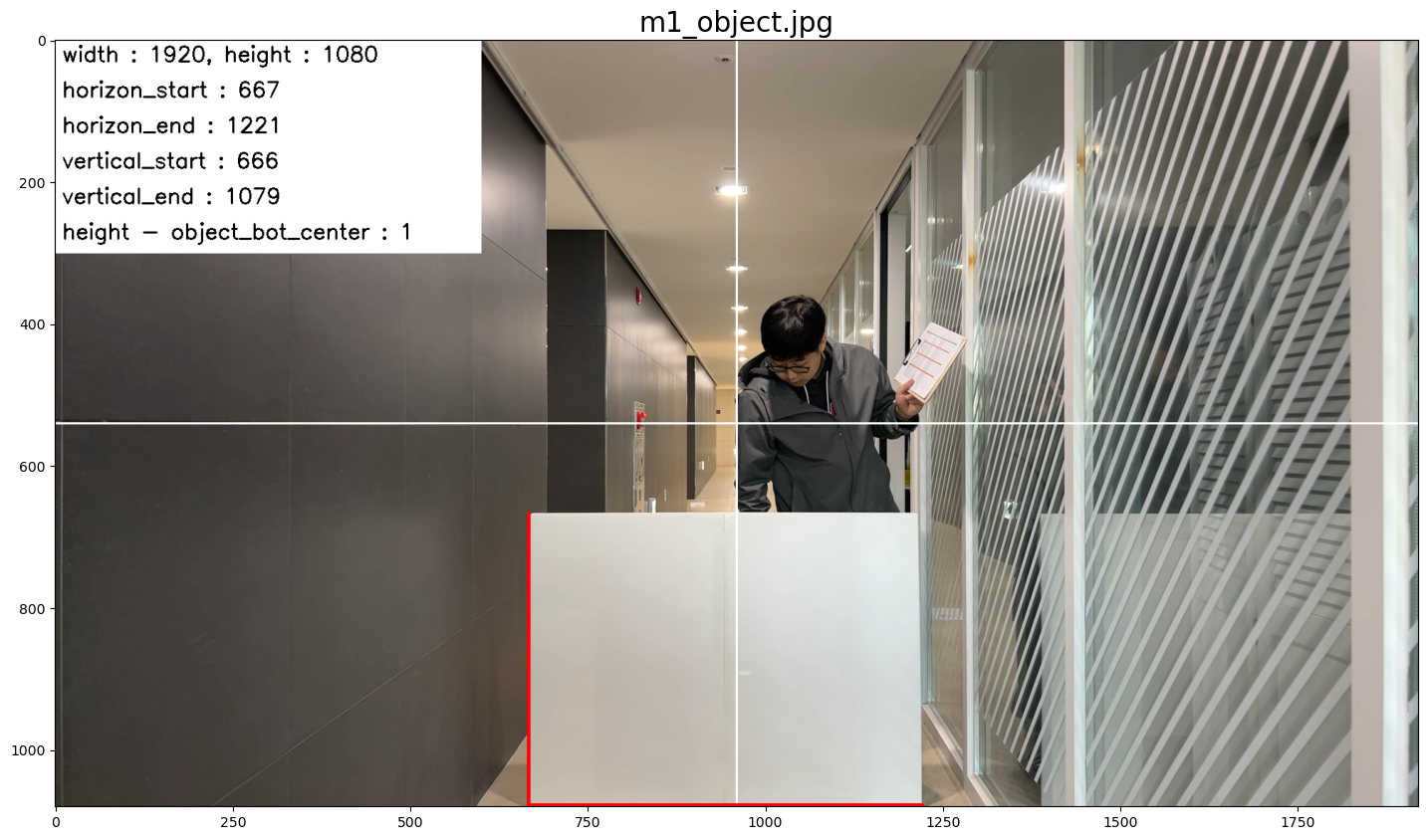
* 고객의 환경에 맞춘 최적의 커스터마이징
* 서비스의 효과를 객관적으로 파악하고 발전시킬 수 있는 방안

이를 위해 탐지되는 객체의 거리 측정을 구현하려 시도했습니다. 거리 측정이 가능하다면, 현실 거리를 이용한 최적 커스터마이징이 가능할 뿐더러, 거리에 따른 퇴치율 분석을 통해 서비스 자체를 발전시킬 수 있기 때문입니다.

이를 위해 관련 선행 연구인 <영상기반의 차량검출 및 차간거리 추정방법 (2012)> 논문의 연구방향을 바탕으로 실제 단안카메라를 통한 거리측정 실험을 진행하였습니다.

관련 내용은 함께 첨부되는 연구노트를 참고해주시기 바랍니다.

실제 거리와 영상 내 거리의 환산식을 구현하지 못해 비록 본 연구를 실제 프로젝트에 적용해보지는 못하였으나, 본 실험을 통해 실제 환경과 영상자료간 왜곡현상이 큰 영향을 미침을 깨달을 수 있었고, “현장” 자체의 중요성에 대해서도 인식할 수 있었습니다.





*<fig. 거리탐지 자료 영상>*

**4. 리뷰/스토리**

**(1)응용SW부(모델 - 서버 통신 부) 담당으로의 리뷰**

맡은 역할인 모델 - 서버 통신 부는 한 마디로 “응용 SW 개발"의 범위라고 할 수 있습니다. 인공지능 모델을 튜닝하거나, 학습을 통해 가중치 파일을 만들어내는 등의 인공지능 모델 자체를 건드리는 작업보다는, 모델부에서 구현한 모델을 이용해 특정 기능들을 구현하고 기획에 맞춰 기능을 추가하며, 개발과 유지보수를 통해 서비스가 잘 동작할 수 있도록 하는 것이게 핵심 태스크이기 때문입니다.

역할에서 중요한 점은 아래의 몇 가지로 꼽을 수 있습니다.

첫째, 기획자의 니즈를 파악하고, 이를 위해 긴밀히 소통하는 것입니다.

둘째, 가지고 있는 데이터와 하드웨어적 한계를 파악하고 기능 구현 방법과 구현 가능 범위를 정하는 것입니다.

셋째, 기획 의도를 최대한 달성할 수 있도록 기술적인 탐색을 하고, 기능을 잘 구현하는 것입니다.

넷째, 모델부와 서버부 간 원활한 데이터 통신을 위한 표준을 정하고 효율적인 통신 방법을 선택하는 것입니다.

SeSAC 교육기간 동안 배웠던 인공지능 기술과 프로그래밍 언어의 기초 지식을 기반으로, 특히 비전처리에서 필요한 OpenCV에 대한 추가 공부와, 기능 구현에 필요한 Multi Threading 과 같은 기술을 익히면서 기능들을 구현해냈습니다. 이러한 공부와 기술 구현을 통해 많은 것을 배우고 성장할 수 있었습니다. 이 중 OpenCV의 공부를 통해 이미지 처리의 기본 지식과 응용 방법에 대해 학습할 수 있었고, 이를 기반으로 객체의 위치 탐색, 객체 이동 경로의 표시, 위험구역 진입시 알람 등의 기능을 우리 프로젝트에 녹여낼 수 있었던 것이 역할을 맡으면서 가장 잘 했던 일이라고 생각합니다.

하지만 처음 다뤄봐서 더욱 조심해야했던 AWS의 Access Key 보안에 둔감했던 탓에 필요치 않은 EC2가 생성되며 불필요한 비용이 발생하기도 했습니다. 이에 대해서는 교육 과정을 준비해주신 모든 분들과, 과정을 함께한 학생들, 그리고 함께한 팀원들에게 미안합니다. 이후부터는 보안 관련 문제가 다시는 재발하지 않도록 Access Key를 CLI에 담아 외부로 노출되지 않도록 하며, 기타 보안 사항들도 환경변수로 저장하여 유출되지 않도록 하였습니다. 더불어 Access Key는 사용 후 즉시 삭제하고, 최소 하루 세 번 AWS에 불필요한 EC2가 생성되지는 않았는지 체크를 하고 있습니다.

앞서 말한 기능 구현과 더불어, 모델부와 서버부 간의 데이터 표준을 맞추고 그 통신에 문제가 없도록 하였습니다. 기능 추가와 유지보수에 있어서는 모듈화의 중요성을 많이 느꼈는데, 이는 기능의 모듈화를 통해 "쉬운 기능 추가와 원활한 유지보수"를 달성할 수 있기 때문입니다.

잘한 점도 있고, 잘못한 점도 있고, 많은 것을 느낄 수 있었고 학습적으로도 많은 공부가 되어 정말 좋은 기회였습니다. 이는 잘 협력할 수 있었던 팀원들이 있었고, 강사님과 멘토님을 비롯해 여러 도움을 준 다른 팀 팀원들이 있었던 덕분으로 생각합니다. 이후 다른 프로젝트나 업무에서는 더 나은 소통을 할 수 있도록 노력하며, 같은 실수를 반복하지 않고, 비슷한 실수는 살펴 예방하며, 더 많은 공부를 통해 더욱 효율적인 프로그래밍을 할 수 있도록 노력할 것입니다.

**(2)vision 연구부로의 리뷰**

이번 프로젝트는 ‘컴퓨터 비전'이 밑바탕에 깔려 있습니다. 그리고 이 비전 기술에는 자연어 처리나 음성 처리에 비해 특별한 특징이 있다는 것을 느꼈는데, 바로 ‘현장'이 중요하고 ‘실제와 컴퓨터 간의 차이'가 많이 신경써야 할 주요 변수라는 것입니다.

비전 처리는 대상 환경에 대해 카메라를 통해 데이터를 수집합니다. 이렇게 수집 데이터는 영상 파일이 됩니다. 하지만 영상 파일은 실제 현장을 100% 반영하지 못하며 왜곡이 일어납니다. 이는 실제 현장은 3차원인 데 비해 영상 파일은 2차원의 자료이기 때문입니다.

이번 프로젝트에서 vision 연구부로, 기능 구현을 위해 이러한 제한적이고 현장을 왜곡해 보여주는 영상 자료에서 객체의 위치, 크기 등의 실제 현장의 정보를 추출하기 위해 가설을 세우고, 실험하고, 검토하고, 재실험을 하는 과정을 경험 하였습니다.

아직 객체와 카메라 간의 정확한 거리 측정 또는 다중 카메라를 통한 미니맵의 구현 등은 부족한 공부로 인해 만족스러운 결과물을 도출하지는 못했습니다. 그러나 실제 현장의 정보를 영상에서 뽑아낼 수 있도록 가설을 세우고, 거리 측정을 위해 현장에서 실험 현장을 통제하고, 거리와 각도를 측정했던 경험들은 매우 가치있었습니다.

프로그래머는 책상에 앉아있는 만큼이나 현장을 느끼는 것 또한 중요하다는 것을 깨달았습니다. 영상 처리를 위해선 현장을 알아야 함은 물론이고, 현장에는 프로그래밍 성과를 높이고 일을 더 효율적으로 처리할 수 있는 힌트가 많이 있기 때문입니다. 더불어 현장의 실험의 통제와 측정 자체의 난이도가 상당히 높고, 이에 따라 현실적인 제한사항과 어느정도의 타협이 필요하다는 것도 느꼈습니다.

우리 팀에는 다행히도 훌륭한 기획과 함께 현장 실험 계획 수립과 수행에 능숙한 윤율 팀원이 있어, 예상보다 많은 실험을 할 수 있었고, 많은 것을 배울 수 있었습니다.

**(3)팀장으로의 리뷰**

이번 프로젝트에서는 정말 많은 일들이 있었습니다.

**(1) 결정과 설득에 대해**

이 프로젝트를 하기 전 과거의 나는, 어떤 결정을 내리고, 설득하고, 그에 대한 책임을 지는 일에 익숙하지 않았습니다.

팀장의 역할을 하면서 결정을 한 경험이 크게 두 번이 기억에 남습니다.

첫 번째는, 프로젝트 초반, 프로젝트를 어떻게 만들어갈까 하는 기획회의에 집중해 코딩 착수가 늦어지던 시점이었습니다. 구현하고자 하는 기능과 프로젝트의 세부 사항에 대한 의견들이 달랐습니다. 그때 심재훈 팀원이 “말만 하지 말고 일단 시작해보는 게 어떨까?”는 의견을 제시했고, 그대로 기획회의를 끝내고 일단은 최소한의 목표를 정하고 코딩에 착수기로 결정했습니다.

두 번째는, 프로젝트 중후반에 프로젝트 중간 발표를 할지 말지에 대한 의견이 갈렸던 시점이었습니다. 모두들 체력적으로 부담이 있던 시점이었고, 발표 반대 의견으로는 중간발표 준비로 시간을 날리는 것 보다는 기능구현에 집중하자는 의견이 있었습니다. 하지만 중간발표를 통해 받는 피드백을 통해 남은 시간을 훨씬 고민없이 효율적으로 쓸 수 있을 것이라는 판단에 중간발표가 필요하다는 결론을 내리고 설득했습니다. 발표 준비에 들이는 시간을 최소화하기 위해 중간 발표를 준비하는 우리의 자세에 대한 협의했습니다. 발표를 통해 우리가 얻어야 할 것은 "서비스 시연에 대한 피드백"임을 명확히 하였으며, 이를 위해 필요한 최소한의 준비만을 할 것임을 정하였습니다.

위의 두 결정이 어떤 결과를 가져왔는지는 팀 구성원마다 평가가 다를 것이라고 생각합니다. 또한, 이러한 결정에 반대되는 결정을하거나 보류했을 때 더 나은 결과가 있을 수도 있습니다.

하지만 한 가지 좋았던 점은, 의견이 분분하고 고민이 깊어지는 시점에 일단 결정을 내림으로써 팀의 모든 구성원이 같은 목표를 가지고 움직일 수 있었다는 점입니다. 이를 통해 우리 팀은 수많은 고민을 단 한 가지의 고민, 즉 “결정된 목표를 어떻게 달성할 것인가"로 줄일 수 있었습니다.

결정과 설득은 말로도 했지만, 행동으로도 하려고 노력했습니다. 결정에 대해 좋은 결과를 가져오기 위해 정말 열심히, 그리고 잘 하려고 노력할 수밖에 없었습니다. 그 결정은 곧 우리 팀의 결과물로 이어지기 때문이고, 또한 결정에 대한 책임을 지기 위해서입니다. 이러한 노력은 당연한 것이며, ‘결정’이 ‘보류’보다는 조금이라도 나은 결과를 가져오기 위한 최소한의 조건임도 함께 느꼈습니다.

프로젝트에 결과물이 각 팀원들에게 만족스러울지, 어떠한 의미를 가질지는 모두 다를 것입니다. 만약 결과물이 만족스럽고 프로젝트가 좋은 경험 이었다면 그것은, 못미더울 수 있음에도 불구하고 일단 합심해서 열심히 진행한 팀 구성원들 덕분에, 거기에 약간의 행운이 더해졌기 때문일 것입니다.

드라마 스토브리그에서 나오는 한 장면이 떠오릅니다. 야구단 단장인 백승수는, 구단을 해체의 위기에서 구할 수 있는 구단 인수 PT에서 이렇게 말합니다. “나는 모든 직원들을 책임지는 사람이라고 생각했다. 하지만 정신 차리고 보니 팀원들도 나를, 그리고 서로를 책임져주고 있더라. 잘 생각해보시라, 대표님의 결정에서 상대방을 설득한 것은 대표님이었나? 대표님 본인이 설득당하지 않은 것은 아니었나?”. 앞으로 항상 책임과 설득에 대한 생각을 할 때에, 이번 프로젝트를 떠올릴 것입니다.

[[#족집게\_스토브리그] 말빨 zzㅓ는 단장님과 지지 않는 이제훈 대표님🔥 (ft. 단장직 해임)ㅣ스토브리그(Stove League)ㅣSBS DRAMA](https://youtu.be/KEqhjw4zX4c?t=703)

*11분 43초부터*

**(2) 결과물이 중요할까 학습이 중요할까**

프로젝트를 진행하면서, 가장 큰 고민 중 하나는 팀원 모두가 어떻게 성취감을 느끼게 할 것인가였습니다. 다른 말로 하면, "프로젝트의 초점을 결과물에 맞출 것인가, 학습에 맞출 것인가" 였습니다. 이 두 가지가 황금 밸런스를 맞추기 위해서는 많은 고민과 경험, 그리고 소통이 필요함을 깨달았습니다.

지금 와서 돌아보면 우리 팀은, 그리고 저는 “조금 욕심이 많은 최소한의 결과물"을 달성하는 것에 우선 순위를 두고 움직였습니다. 처음부터 차근차근 공부하면서 하나씩 해나가기 보다는, 조금 공부가 덜 되더라도 빠르게 구현을 할 수 있도록 진행되었습니다.

반대급부로 아쉬웠던 점은, 학습이라는 목표를 위해 각자 차분히 배웠던 것에 대한 정리나 추가로 하고 싶던 공부, 혹은 교육 과정 동안 있었던 학습에 대한 블로그 등 기록의 정리에 시간을 많이 쏟지 못했다는 점입니다. 결과물을 만들기 위해 학습 부분의 무게를 조금 덜었던 것입니다.

결과물(완성도)과 학습 간의 밸런스를 맞추는 일은 앞으로도 계속해서 고민할 일입니다. 업무를 위한 프로젝트인가, 학습을 위한 프로젝트인가, 현재 상황은 어떤가에 따라 그 밸런스의 기준은 매번 달라질 것입니다. 이 때마다 조금 더 나은, 이전보다 더 만족할 수 있도록, 밸런스를 맞추는 데 대한 고민은 정말 중요한 일임을 인식하고, 진지하게 고민을 할 것입니다.

큰 그림을 보고 훌륭한 기획을하고 이를 통해 팀이 가야할 목표를 정조준 할 수 있게 한 율이 형님,

빠른 기술 탐색과 구현을 통해 팀이 지지부진한 고민을 할 때마다 팀에 속도감을 불어넣어준 재훈이,

사려 깊은 탐색을 통해 팀의 방향성과 모델 학습에 대한 논리적 근거를 차근차근 쌓아준 원준이.

많은 노력을 하고, 서로를 책임져준 팀 구성원 모두에게 너무도 감사합니다. 모두 원하는 만큼의 성취를 얻기를 바랍니다.