소프트웨어 종합설계 최종 보고서

다중 CCTV 이용,
객체 동기화 및 객체 인식 성능 향상

연세대학교 컴퓨터과학과 데이터베이스 연구실

지도교수: 이원석

지도조교: 박기종

TEAM: CSI-15

김도희 이경우 정유정

목차

1. 연구 주제

다중 카메라를 이용하여 객체 인식의 성능을 향상시키고, 서로 다른 카메라에서 찍힌 객체를 동기화하여 객체 추적 및 의미 있는 그리드 정보를 데이터베이스로부터 도출하기.

2. 연구의 필요성

기존의 객체 인식 프로그램은 촬영 환경에 따른 날씨 및 빛에 영향을 크게 받거나, 객체가 겹치는 현상으로 인해 오류가 잦게 발생한다. 많은 연구에서 이 문제를 해결하기 위해 AI 및 비전 기술을 이용해 객체 인식 성능을 높이려고 노력하고 있지만, 이것은 카메라 한 대만을 이용하여 객체를 인식하고 할 때 필연적으로 발생하는 문제이고, 복잡하고 어려운 알고리즘 및 계산이 요구되어 아직까지 큰 성과를 얻지 못하고 있다. 따라서 우리는 AI 와 비전의 시각이 아닌, 다중 카메라와 데이터베이스를 이용하여 기존의 객체 인식 프로그램의 결과를 후작업으로 보정해 오류율을 줄여 보다 쉽고 효과적으로 객체 인식 성능을 높이는 방향으로 연구를 진행해보기로 했다.

기존에 비슷한 연구가 있는지 찾아본 결과, 아래와 같이 다중 카메라 환경에서 다중 객체 추적을 한 연구가 있었지만, 이 연구는 객체 정보 추출 및 추적을 위해 영상의 RGB 값과 LPB 를 구해 다중 카메라의 객체를 동기화 했다. 이것은 비전의 기술로 객체를 동기화 한 것으로 많은 계산을 필요로 하고, 색과 질감을 이용하는 것이기 때문에 여전히 촬영 환경에 영향을 크게 받을 수밖에 없다.

> *다중 카메라 환경에서 다중 객체추적 기술 연구. 한국정보과학회 학술발표논문집, 36(2C), 414-417*
>
> *객체 정보 추출 밀 추적을 위해 RGB 컬러 정보를 추출하고, LBP 를 이용하여 질감 정보를 추출하여 컬러와 질감을 분리했을 경우와 결합했을 경우의 객체 인식률을 비교, 결합했을 때 다중 카메라에서 동일 객체로 인식 및 추적 성능을 높임.*

우리의 연구는 데이터베이스 기반으로 학습이 이루어지기 때문에 객체 인식 성능을 향상시킬 수 있을 뿐만 아니라, 객체 추적 및 추후 쌓인 데이터를 통해 어떤 곳에서 사람이 주로 나가고 들어오는지, 어떤 장소에 오래 머물러 있는지 등 시공간에 대한 정보도 파악할 수 있다.

3. 연구 내용

3.1. 알고리즘 개요

다중 객체를 동기화하기 위해 먼저 다중 영상 속 공간을 동기화할 필요가 있다. 공간 동기화 방법은 영상 속 공간을 그리드별로 나누어 학습을 통해 어떤 그리드끼리 연결되는지 맵핑 테이블을 만드는 것이다. 맵핑 테이블 상에서 하나의 레이블은 각각의 그리드가 같은 공간을 나타낸다는 의미를 갖는다. 그리드 맵핑 테이블이 완성되면 영상 1 에서 찍힌 특정 객체가 영상 2 의 어떤 위치에 존재할 것인지 예측할 수 있게 된다. 이 방법을 통해 객체를 동기화할 수 있다. 그리고 객체 인식에서 카메라를 여러 대 이용한다면 한 카메라 영상에서 오류가 발생하더라도 오류를 일으키지 않은 카메라가 한 대라도 존재한다면 맵핑 테이블과 현재까지 찍힌 로그 데이터를 이용하여 오류를 보정할 수 있다. Entry Point 와 Exit Point 를 학습을 통해

도출하여 어떤 지점에서 객체가 갑자기 생기거나 사라졌을 때 그 현상이 오류인지, 아니면 사람이 정말 들어오거나 빠져나간 것인지 판단할 수 있다.

3.2. 세부 알고리즘

3.2.1. 카메라 별 그리드 사이즈 결정 알고리즘

다중 카메라의 그리드 사이즈를 고정된 같은 사이즈로 지정하게 되면 각 카메라 별로 한 그리드에 맵핑 되는 실제 공간의 크기가 다를 수 있다. 예를 들어 멀리서 찍고 있는 카메라와 가까이서 찍고 있는 카메라를 비교 시, 멀리서 찍고 있는 카메라는 가까이서 찍고 있는 카메라보다 훨씬 더 넓은 실제 공간이 한 그리드 맵핑 된다. 카메라 별로 한 그리드에 맵핑 되는 실제 공간 크기가 가능한 같아야 각 카메라별로 실제 같은 공간에 대한 그리드 맵핑이 1 : 1 대응이 될 수 있고, 1 : 1 대응이면 오류 탐지 및 보정 시 다양한 경우를 배제하고, 맵핑 된 그리드 set 만 보고 판단하는 것이 가능하게 된다.

따라서 카메라 별로 한 그리드가 실제 같은 공간을 맵핑 하게 하기 위해 우리는 각 카메라의 화면 가운데에서의 평균 객체 크기를 구해 1/N 하여 그리드 사이즈를 카메라 별로 다르게 설정해주기로 했다.

3.2.2. 그리드 좌표 동기화 알고리즘

초기 학습 시에는 세 대의 카메라에서 하나의 객체만 등장할 때 각 카메라에서 나타난 객체의 그리드 좌표를 그리드 맵핑 테이블에 저장한다. 카메라에 여러 개의 객체가 나타난 경우에는 학습하지 않는다. 초기학습 시 한 명일 때에만 학습을 시키는 이유는 맵핑 테이블이 충분히 쌓이지 않으면 객체 동기화를 할 수 없기 때문이다. 이후 모니터링과 학습을 동시에 하는 단계에서는 객체 동기화가 한 번 이루어진 후로는 여러 객체가 출현하더라도 객체 동기화 정보를 보고 맵핑 테이블 학습을 시킬 수 있다.

3.2.3. 그리드 별 객체 사이즈 동기화 알고리즘

이상 크기 오류를 보정할 때, 이상 크기를 '적절한' 사이즈로 보정하려면 3.2.2 에서 맵핑 된 각 카메라 그리드 별로 객체 평균 사이즈를 학습시켜 해당 그리드 맵핑에 대한 객체 사이즈 비율을 얻을 필요가 있다. 실제 같은 공간에 대한 세 대의 카메라의 그리드 맵핑에 맞춰 세 대의 카메라에서의 같은 공간에 대한 사이즈 비율을 알고 있다면, 이상 크기 오류를 탐지할 수 있을 뿐만 아니라 적절한 크기로 보정을 할 수 있다.

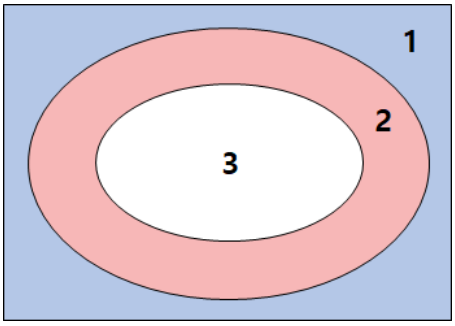
그리드 별로 객체의 사이즈를 저장, 평균 객체 사이즈를 구한다. 그리고 3.2.2 에서 구해지는 그리드 맵핑에 대한 객체 사이즈 비율을 계산한다. 이때, 특정 객체 사이즈 비율은 가로와 세로의 비율을 따로 구한다.

카메라 1 에서의 가로 : 카메라 2 에서의 가로 : 카메라 3 에서의 가로

카메라 1 에서의 세로 : 카메라 2 에서의 세로 : 카메라 3 에서의 세로

3.2.4. Entry, Exit Point 알고리즘

오류의 발생을 고려하여 객체의 출현이나 소멸이 몇 초간 유지된 경우, 그리고 임계 값 이상의 횟수로 발생한 경우에 대해서만 Entry Point, Exit Point 로 지정한다.

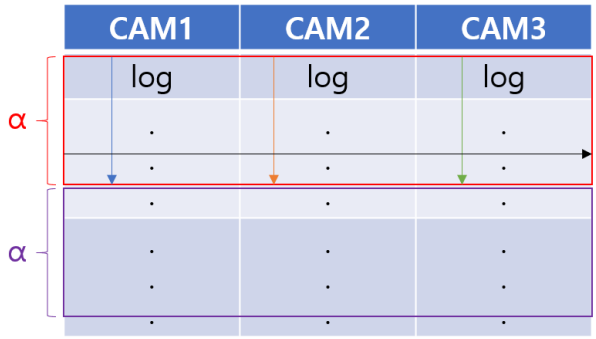


<그림 2>

<그림 2>은 카메라 1 에서 바라본 공간이다. 1 번 공간은 카메라 1 에서는 보이지만 카메라 2 또는 카메라 3 에서는 보이지 않는 공간이다. 2 번 공간과 3 번 공간은 세 대의 카메라에서 모두 보이는 공간이다. 2 번 공간은 1 번 공간에서 3 번 공간으로 들어오기 위해 반드시 통과해야만 하는 지점이다. 2 번 공간을 Entry, Exit Point 로 지정한다. Entry Point 에서 객체가 생성되고 Exit Point 에서 객체가 사라져야 한다. 공간의 특성에 따라 Entry, Exit Point 가 일치하지 않을 수 있다. Entry Point 와 Exit Point 를 단순히 영상의 테두리 공간으로 설정하지 않은 이유는 세 대의 카메라가 모두 바라보지 않는 영역에서는 오류가 발생하더라도 보정할 수 없기 때문이다.

3.2.5. 오류 보정 알고리즘

세 대의 카메라의 세 프레임을 동기화하여 동시에 비교하면서 오류를 탐지하는 방식이 아닌, 하나의 카메라의 프레임만을 보면서 이상 상황이 감지되었을 때만 나머지 두 카메라의 프레임을 확인해서 오류를 보정하는 방식을 택했다. 전자의 방식을 택하지 않는 이유는 시간 동기화를 매 프레임마다 하기보다는 오류가 발생했을 때만 하는 편이 더 효율적이기 때문이다. 이 때 카메라 데이터를 보는 순서는 오류율이 높은 카메라부터 선정하여 먼저 보정이 이루어지도록 한다. 여기서 말하는 오류는 기존에 존재하던 객체의 id 가 갑작스럽게 사라지거나 생기는 현상을 말한다. 다만 오류 보정 시 오류가 보정되지 않은 다른 두 카메라 데이터로 보정하게 되면 보정의 신뢰도가 떨어지기 때문에 시간 간격을 두고 그 시간 간격 내에서는 모든 카메라의 오류 보정이 완료될 수 있도록 한다.



<그림 3>

<그림 3>는 카메라의 오류율이 CAM1>CAM2>CAM3 순으로 높고 시간간격은 α 라 가정했을 때, α 시간 내에서 CAM1 의 프레임만을 먼저 순서대로 보고, CAM1 의 보정이 다 끝난 후에 CAM2, CAM3 에 같은 작업을 시행한다. 그 이후에 다음 α 시간만큼의 프레임에 대해서 같은 작업을 반복하게 된다.

객체인식에서 가장 빈번하게 발생되는 오류로는 객체가 없는데 객체가 인식되는 현상, 객체 인식은 되지만 비정상적인 크기로 인식되는 이상크기, 객체 인식이 끊어지는 단절, 하나의 객체가 여러 객체로 분리되는 현상이 있다. 오류 보정 시 중요한 점은 이렇게 가장 빈번하게 발생되는 오류를 최대한 보정하는 것이다. 객체의 출현이나 소멸이 정상적인지 여부를 판단하기 위해 Entry point 와 Exit point 개념을 도입하고 한 객체임에도 불구하고 오류로 인해 수시로 변하는 id 를 추적하기 위한 테이블을 만든다.

id_prev	id_next
a	d
b	null
c	null
d	f
.	.
.	.
.	.
f	x

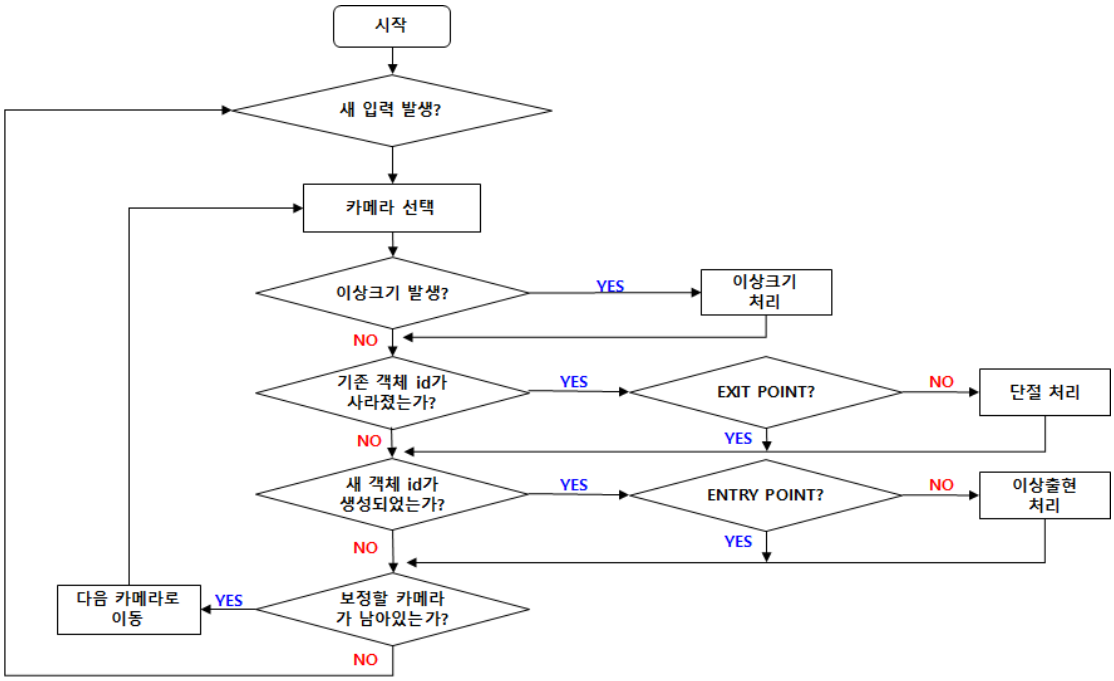
a → d → f → x
b →
c →

<그림 4>

<그림 4>는 CCTV1 의 객체 id 추적 테이블이다. 한 카메라 영상 내의 하나의 객체도 오류로 인해 id 가 수시로 변하기 때문에 이를 추적하기 위한 테이블이다. 예를 들어 객체 a 는 이후 id 가 d 로 변했고 그 후로 f, x 로 변했음을 알 수 있다

3.2.6. 오류 상황 별 동작

- **순서도**



- **이상크기**

  모든 프레임에 대해 이상 크기가 발생했는지 확인한다. 이상 크기가 발생했다면 그리드 별 비율이 정상인지 학습 데이터와 비교해보고, 정상으로 분류되면 아무런 액션을 취하지 않는다.

  비교 결과 비정상으로 분류될 시 보정 과정은 아래와 같다.

  - 맵핑 테이블과 다른 카메라 영상에서 찍힌 그리드 좌표를 이용해 객체의 중심점을 알맞은 그리드 좌표로 수정한다.
  - 그리드 별 크기 비율 데이터와 다른 카메라 영상에서 찍힌 동일 객체의 가로 세로 길이 데이터를 이용하여 적절한 크기로 수정한다.

- **이전 프레임과 비교했을 때 객체 id 에 변화가 생긴 경우**

  - 세 카메라의 프레임 중 오류가 가장 많이 발생한 프레임부터 보정한다. 프레임 선정 시 우선순위는 아래와 같다.

    A. 사라진 객체의 수가 많을수록

    B. 새로 생긴 객체의 수가 많을수록

    C. 카메라의 오류율이 높을수록

A, B, C 순으로 오류수가 많은 카메라의 프레임부터 보정을 한다.

- 새로운 객체가 생긴 경우
  - 이전 프레임에 존재하는 기존 객체를 후보군으로 선정한다.
  - 후보군 중에서 새 객체와 크기가 비슷하고 위치가 가까운 객체를 선정한다.
  - 선정된 객체와 새 객체를 연결하고 객체 id 추적 테이블에 데이터를 추가한다.
- 기존에 있던 객체가 사라진 경우 (단절)
  - 바로 이전 프레임을 확인하고 맵핑 테이블을 이용해 맵핑이 되는 객체끼리 연결시킨다.
  - 위에서 맵핑 된 정보를 이용해서 현재 프레임에서 사라진 객체와 맵핑되는 객체가 다른 카메라 프레임에 있는지 확인한다.
  - 있을 경우 맵핑 테이블을 보고 사라진 객체의 위치를 도출해낸다.
  - 오류 보정을 마친 후 이상크기 보정을 한다.

- **분리**

  분리가 발생할 경우 두 가지 케이스로 나타나게 된다.

  - 기존 객체의 id 는 유지, 새로운 id 생성
    - 새로운 id 가 오류로 탐지되면 제거하고 보정 후 테이블에 넣지 않는다.
    - 기존 id 의 객체는 이상크기 보정을 통해 보정한다.
  - 기존 객체의 id 소멸, 새로운 id 생성
    - 새로운 id 가 오류로 탐지, 제거한다.
    - 없어진 기존 id 는 단절로 연결하여 보정한다.

4. 연구 결과

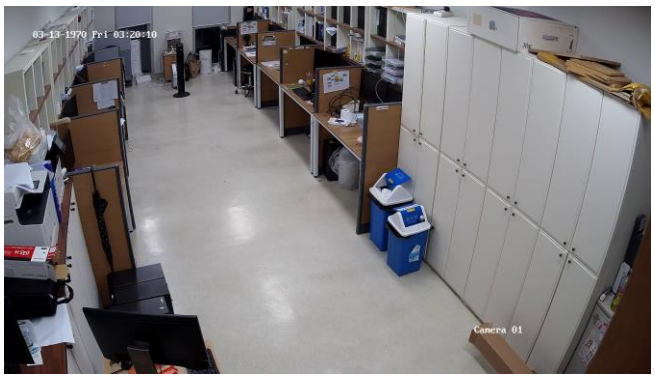
4.1. 실험 환경

4.1.1. 촬영 환경

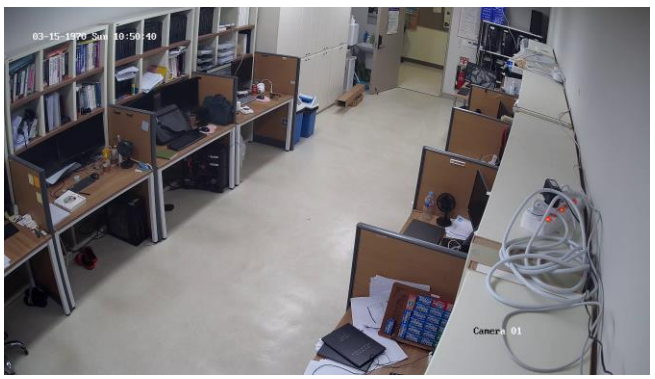
한 공간의 서로 다른 각도에서 세 대의 카메라로 찍었다.

- cam1

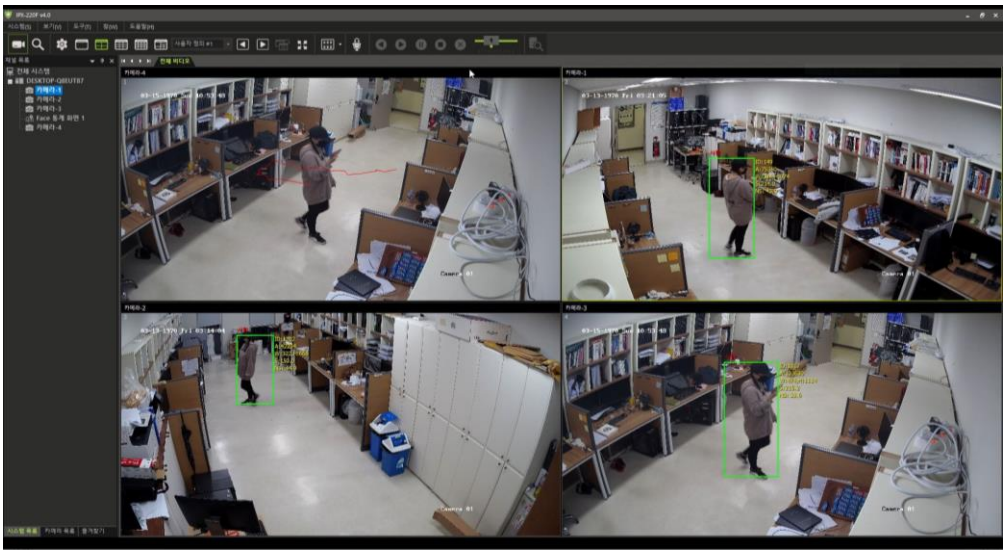
- cam2



- cam3

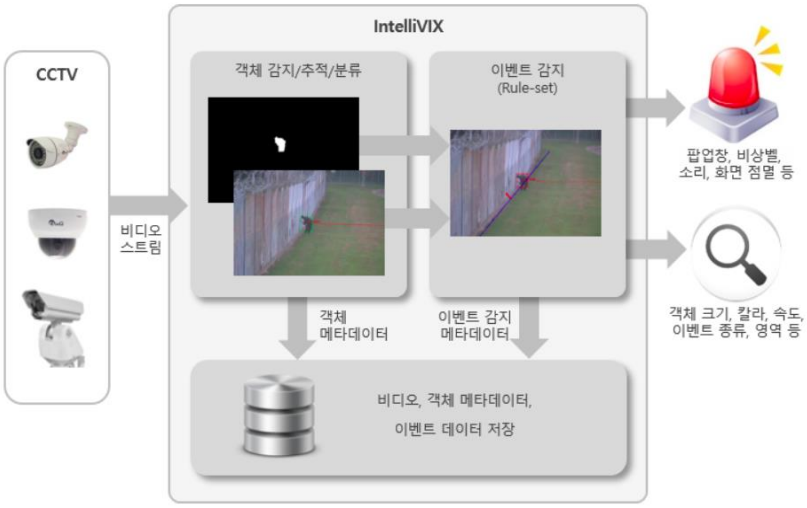


4.1.2. 객체 인식 프로그램



컴퓨터 비전 관련 소프트웨어를 연구 개발하는 전문 기업, 인텔리빅스의 지능형 영상 감시 소프트웨어 패키지인 **'IntelliVIX'**를 사용.

IntelliVIX 는 CCTV 카메라 혹은 녹화된 비디오에서 입력되는 실시간 영상을 분석하여 움직임이 있는 물체를 감지/추적/분류하여 사전 정의된 이벤트(룰셋)를 감지하고 녹화/재생/검색할 수 있는 올인원 지능형 영상 감시 시스템이다. 감지/추적/분류된 물체(보행자, 차량 등)가 사전 설정된 이벤트에 감지될 경우 운영자에게 감지 사실을 실시간으로 알려줄 수 있으며 또한, 관련 비디오 및 객체, 이벤트 메타데이터를 저장하고 사고 후 메타데이터를 이용하여 빠르게 영상 검색을 수행할 수 있다.



4.2. 학습 결과

4.2.1. 그리드 맵핑 테이블 정확도 비교

- **맵핑 테이블 정확도 정의**

  반복성 (Repeatability)을 통해 맵핑 테이블의 정화도를 측정

  반복성이란, 같은 측정 조건에서 연속적으로 측정하여 얻은 결과들 사이의 일치하는 정도를 의미하는 것으로, 새로운 영상의 맵핑 그룹 정보가 학습된 맵핑 테이블의 그룹에 얼마나 다시 맵핑되는지를 통해 즉정한다.

  맵핑 테이블 정확도 = (테스트 영상의 맵핑 그룹 중 학습 맵핑 테이블에도 있는 그룹의 수)%(테스트 영상의 맵핑 그룹 수)

- **그리드 사이즈와 모양에 따른 정확도 변화**

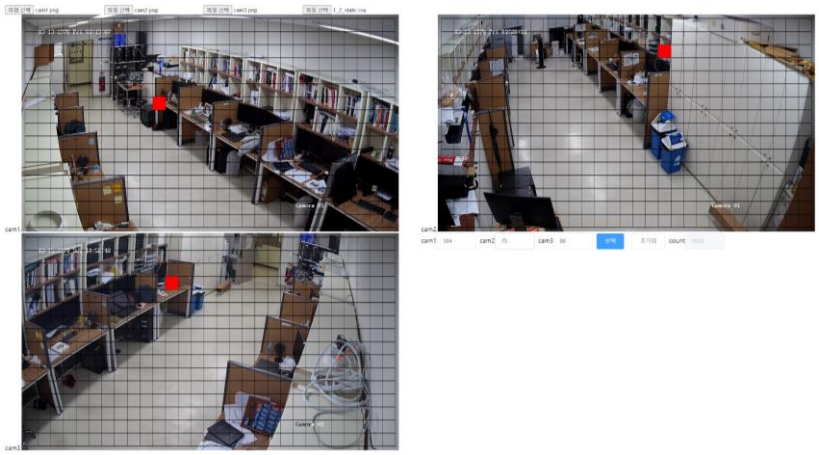
  그리드 사이즈와 모양을 바꿔가면서 같은 영상으로 80 분동안 학습을 진행하고, 만들어진 맵핑 테이블의 정확도를 측정하기 위해, 20 분 동안 테스트를 진행

  - **기준**

    각 카메라의 화면 가운데에서의 평균 객체 크기를 구해 다음 세 기준으로 그리드 모양을 결정 후 1/n 로 그리드 사이즈를 결정. n={1,2,3,4,5,6}

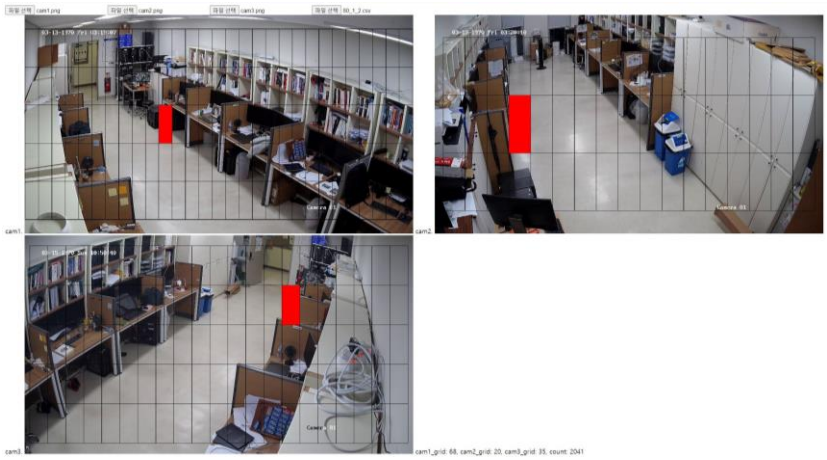
    - **고정 정사각형**

      1 번 카메라의 평균 객체 사이즈의 가로를 한변으로 하는 정사각형을 그리드 모양으로 결정. 세 대의 카메라 모두 같은 모양, 사이즈.



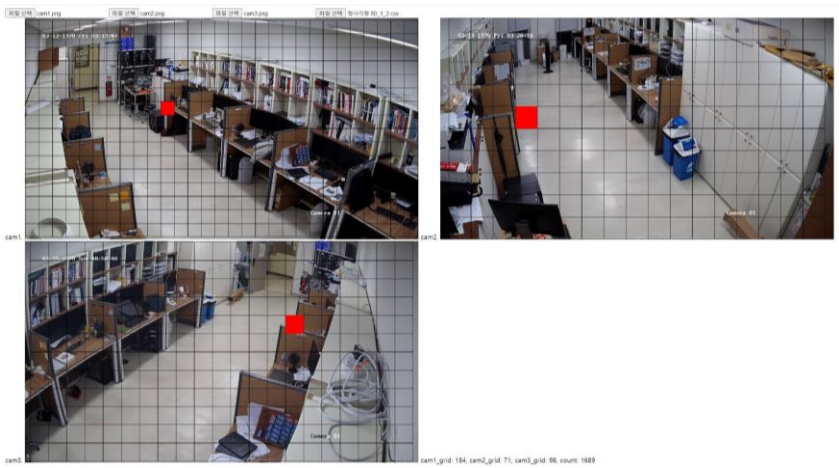
- **가변 직사각형**

  각 카메라에서의 평균 객체 사이즈를 그리드 모양 결정. 카메라마다 다른 모양, 사이즈.

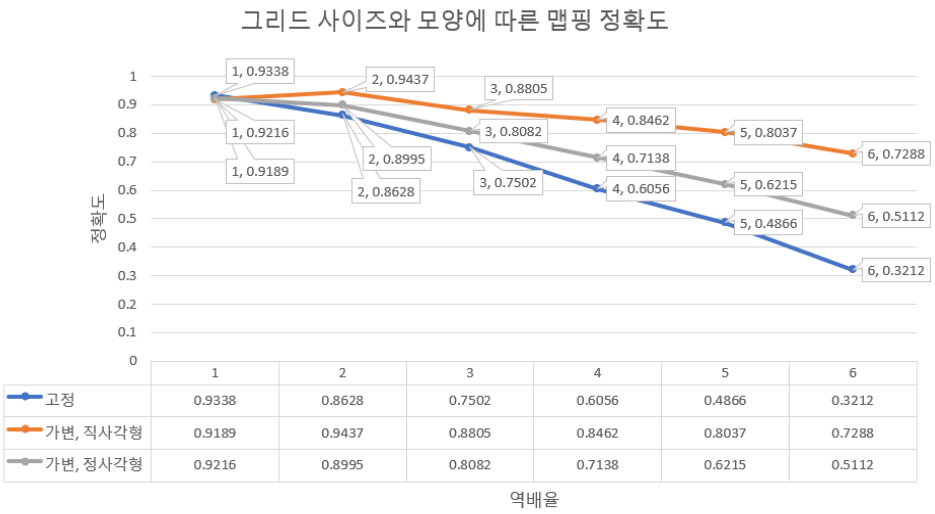


- **가변 정사각형**

  각 카메라에서의 평균 객체 사이즈의 가로를 한 변으로 하는 정사각형을 그리드 모양으로 결정. 카메라마다 다른 모양, 사이즈.

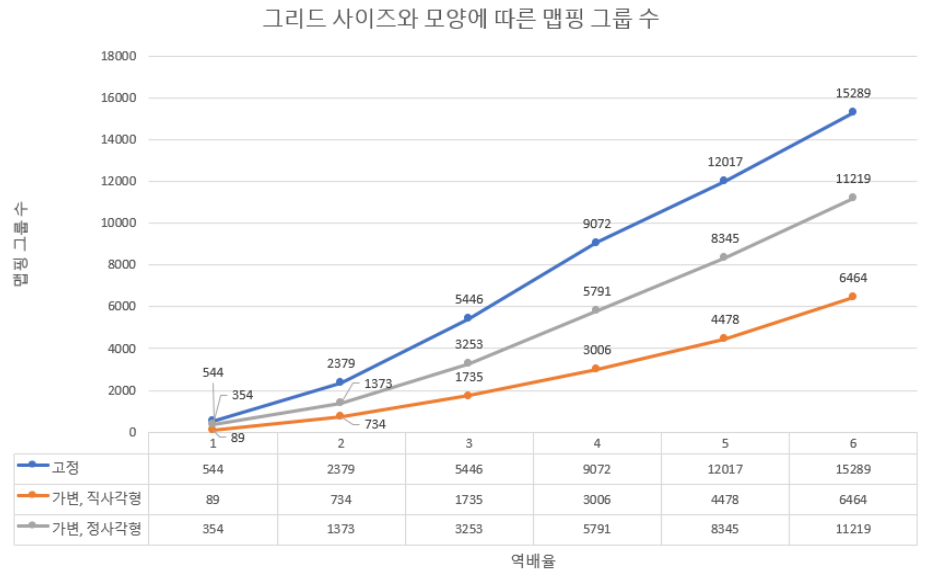


- **결과**



	1	2	3	4	5	6
고정	0.9338	0.8628	0.7502	0.6056	0.4866	0.3212
가변, 직사각형	0.9189	0.9437	0.8805	0.8462	0.8037	0.7288
가변, 정사각형	0.9216	0.8995	0.8082	0.7138	0.6215	0.5112

- n 이 커질수록 정확도가 감소하는 경향이 보인다.
- 고정 > 가변 정사각형 > 가변 직사각형 순으로 감소하는 폭이 크다.
- 가변 직사각형 > 가변 정사각형 > 고정 순으로 n 에 따른 값이 크다.
- 가변 직사각형 ½ 의 정확도가 가장 높다.



	1	2	3	4	5	6
고정	544	2379	5446	9072	12017	15289
가변, 직사각형	89	734	1735	3006	4478	6464
가변, 정사각형	354	1373	3253	5791	8345	11219

- n 이 커질수록 점점 증가한다.
- 고정 > 가변 정사각형 > 가변 직사각형 순으로 n 에 따른 값과, 증가하는 기울기가 크다.

- n 이 커질수록 작아진다. 모든 그리드 모양에서, 1 에서 2 로 넘어갈 때 크게 감소한다. 특히 가변 직사각형에서 그 폭이 다른 모양에 비해 크다.

- 가변 직사각형 > 가변 정사각형 > 고정 순으로 n 에 따른 값이 크다.

▪ **분석**

- n 이 커질수록 그리드 사이즈는 작아진다. 그리드 사이즈가 작으면 그리드의 해상도(가까운 영역을 구분할 수 있는 정도)는 커지지만, 그만큼 반복적으로 같은 맵핑이 이루어질 확률이 낮아져 정확도가 낮아지는 것으로 예상된다. 실제로 가변 직사각형에 대해, 학습된 맵핑 그룹의 평균 카운트 수가 1 에서는 210.75 였지만, 1/6 에서는 3.6023 로 차이가 컸다.

- 고정 > 가변 정사각형 > 가변 직사각형 순으로 감소하는 키울기가 큰 것은 각 카메라 별로 한 그리드가 실제 같은 공간을 맵핑 하는 정도에 따른 것으로 예상된다. 카메라에 상관없이 고정 크기를 사용한 경우 그리드 사이즈가 작아질수록 정확도가 크게 떨어졌고, 각 카메라에서의 객체 사이즈 비율을 지킨 가변 정사각형은 비교적 덜 떨어졌다. 그리고 각 카메라에서의 객체 사이즈를 그대로 사용한 가변 직사각형이 가장 기울기가 작았다.

- 고정, 가변 정사각형과 달리 가변 직사각형에서 ½ 에서 정확도가 가장 높은 것은 ½ 의 해상도가 객체 사이즈를 그대로 사용한 것보다 높기 때문인 것으로 예상 된다. 객체 사이즈를 그대로 사용한 경우 학습된 맵핑 그룹 수가 89 밖에 되지 않았지만, ½ 은 734 로 약 9 배가까이

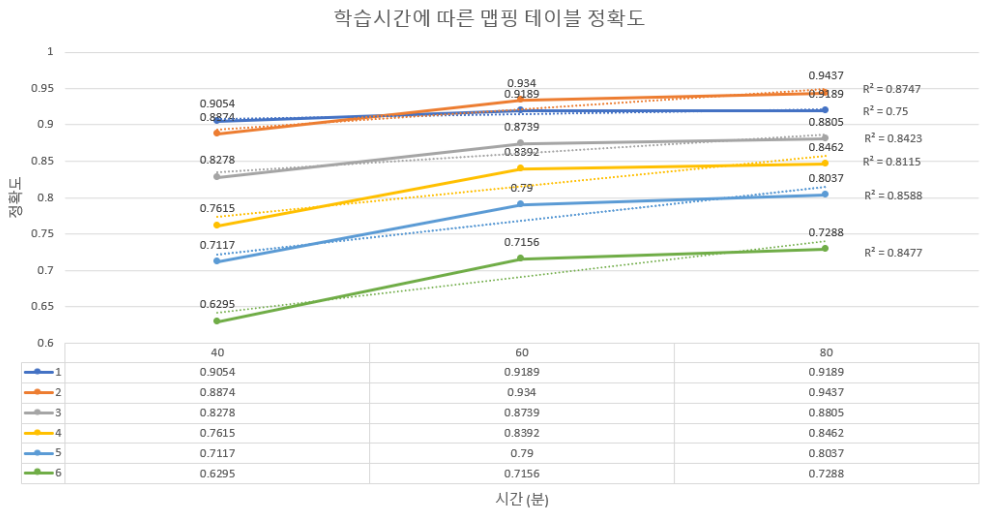
증가했다. 평균 카운트 수는 ½ 이 더 작지만, ½ 기준 평균 카운트 수보다 해상도가 정확도에 더 영향을 끼친 것으로 보인다.

- **학습 시간에 따른 정확도 변화**

  각 시간만큼 학습된 맵핑 테이블을 20 분의 새로운 영상 데이터로 비교.

  그리드 모양 : 가변 직사각형 (가장 정확도가 높았던 데이터)

  - **결과**



  - **분석**

    - 맵핑 정확도가 90 퍼센트 이상이기 위해서는 적어도 80 분 이상 학습할 필요가 있다.

    - 1/2 일 때 학습 시간 대비 정확도의 추세선 R^2 값도 가장 크다. 이것은 실험환경에서 그리드 모양 및 사이즈가 가변 직사각형 1/2 일 때 학습시간 대비 정확도의 효율이 좋음을 의미한다.

4.2.2. ENTRY & EXIT POINT

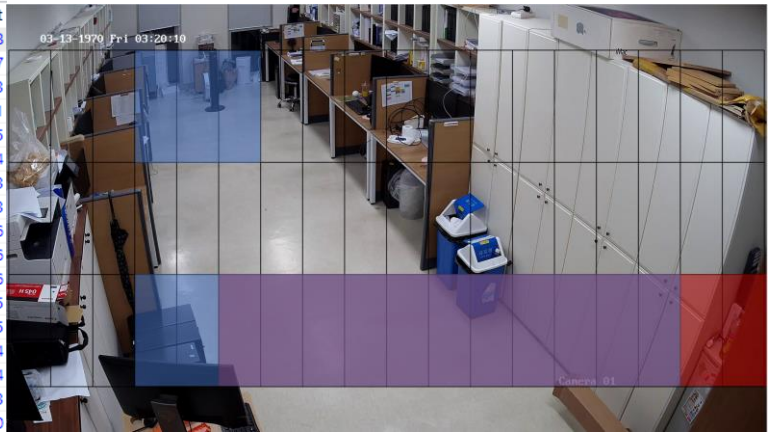
- **빨강 - 처음 Entry/Exit Point 로 예상했던 부분**
- **파랑 - 실제 Entry/Exit Point 로 찍힌 부분**
- **보라 - Entry/Exit Point 로 예상했고 실제로 Entry/Exit Point 로 찍힌 부분 (빨강&파랑)**

- **cam1 entry/exit table 과 이를 시각화한 모습**

TYPE	grid	count
exit	33	161
entry	33	140
entry	32	68
exit	32	53
exit	137	31
entry	134	29
exit	35	26
entry	35	23
entry	36	23
exit	138	22
exit	132	22
entry	135	22
entry	130	22
exit	37	22
exit	135	22
entry	138	22
entry	133	20

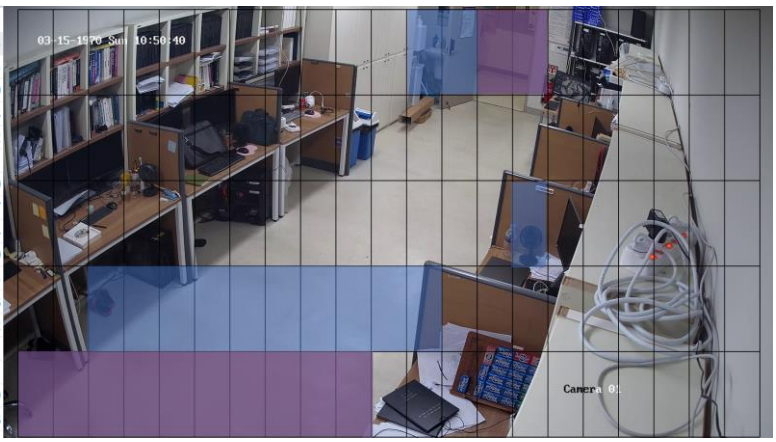
- **cam2 entry/exit table 과 이를 시각화한 모습**

TYPE	grid	count
entry	4	118
exit	4	117
exit	3	73
entry	3	71
exit	44	35
entry	5	34
exit	5	33
entry	45	33
entry	48	26
exit	49	26
exit	45	26
exit	48	25
entry	50	25
entry	47	24
entry	43	24
entry	44	23
exit	47	20



- **cam3 entry/exit table 과 이를 시각화한 모습**

TYPE	grid	count
exit	56	445
entry	56	445
exit	13	70
entry	13	64
exit	64	62
exit	35	51
entry	64	50
entry	14	47
exit	14	44
entry	35	40
entry	65	35
entry	34	28
exit	70	22
entry	89	21
exit	65	21
entry	67	19
entry	12	18



- **예측과 맞아 떨어진 그리드 / 예측과 다르게 나타난 그리드**

  - **예측과 맞아 떨어진 부분**

    - 카메라의 시야에서 들어오고 벗어나는 지역은 Entry/Exit Point 로 학습이 이루어졌다.

  - **예측과 달랐던 부분**

    - 객체가 화면 안에 들어온 뒤 객체인식이 이루어지는 데에 지연시간이 생기고 그 지연시간 동안 객체의 움직임으로 인해 입구에서 다소 떨어진 지역에서도 Entry/Exit Point 로 인식되는 문제

      - 해결 방법: 객체인식이 다소 늦게 이루어지더라도 이 지점까진 Entry/Exit Ponit 로 채용

    - 오류가 많이 발생하는 특정 포인트에서 오류로 인해 객체가 자주 나타나고 사라지기 때문에 Entry/Exit Point 로 인식되는 문제

      - 해결 방법: 단순히 나타나고 사라지는 정도에 따라서 Entry/Exit Point 를 정하는 것이 아닌, 객체가 나타난 뒤 최소 유효시간 동안 유지되었을 때에만 해당 객체가 나타난 지점을 Entry Point 로, 사라진 지점을 Exit Point 로 지정

4.3. 오류 보정 결과

4.3.1. 단절 보정

- **보정 전**

- **보정 후**

- **보정 전 로그(왼쪽) 과 보정 후 로그(오른쪽)**

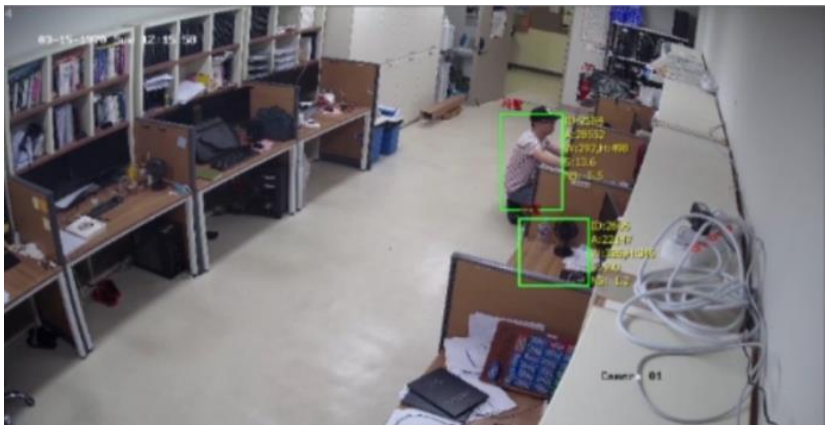
2020-05-29 23:24:48.000	785	693	470	237	337
2020-05-29 23:24:48.100	785	689	463	226	348
2020-05-29 23:24:48.200	785	690	458	224	350
2020-05-29 23:24:48.300	785	690	458	224	350
2020-05-29 23:24:48.400	785	704	448	170	319
2020-05-29 23:24:48.500	785	700	452	199	337
2020-05-29 23:24:48.600	785	692	456	214	347
2020-05-29 23:24:48.700	785	692	456	214	347
2020-05-29 23:24:48.800	785	702	450	162	315
2020-05-29 23:24:49.200	786	658	462	225	356
2020-05-29 23:24:49.300	786	651	462	237	355
2020-05-29 23:24:49.400	786	651	462	237	355
2020-05-29 23:24:49.500	786	651	463	234	356
2020-05-29 23:24:49.600	786	651	463	240	357
2020-05-29 23:24:49.700	786	651	461	237	360
2020-05-29 23:24:49.800	786	651	461	237	360
2020-05-29 23:24:49.900	786	651	457	234	362
2020-05-29 23:24:50.000	786	670	453	231	365
2020-05-29 23:24:50.100	786	686	453	230	366

2020-05-29 23:24:48.000	785	693	470	237	337
2020-05-29 23:24:48.100	785	689	463	226	348
2020-05-29 23:24:48.200	785	690	458	224	350
2020-05-29 23:24:48.300	785	690	458	224	350
2020-05-29 23:24:48.400	785	704	448	170	319
2020-05-29 23:24:48.500	785	700	452	199	337
2020-05-29 23:24:48.600	785	692	456	214	347
2020-05-29 23:24:48.700	785	692	456	214	347
2020-05-29 23:24:48.800	785	702	450	162	315
2020-05-29 23:24:48.900	785	659	465	225	356
2020-05-29 23:24:49.000	785	657	464	237	355
2020-05-29 23:24:49.100	785	657	464	237	355
2020-05-29 23:24:49.200	785	658	462	225	356
2020-05-29 23:24:49.300	785	651	462	237	355
2020-05-29 23:24:49.400	785	651	462	237	355
2020-05-29 23:24:49.500	785	651	463	234	356
2020-05-29 23:24:49.600	785	651	463	240	357
2020-05-29 23:24:49.700	785	651	461	237	360
2020-05-29 23:24:49.800	785	651	461	237	360
2020-05-29 23:24:49.900	785	651	457	234	362
2020-05-29 23:24:50.000	785	670	453	231	365
2020-05-29 23:24:50.100	785	686	453	230	366

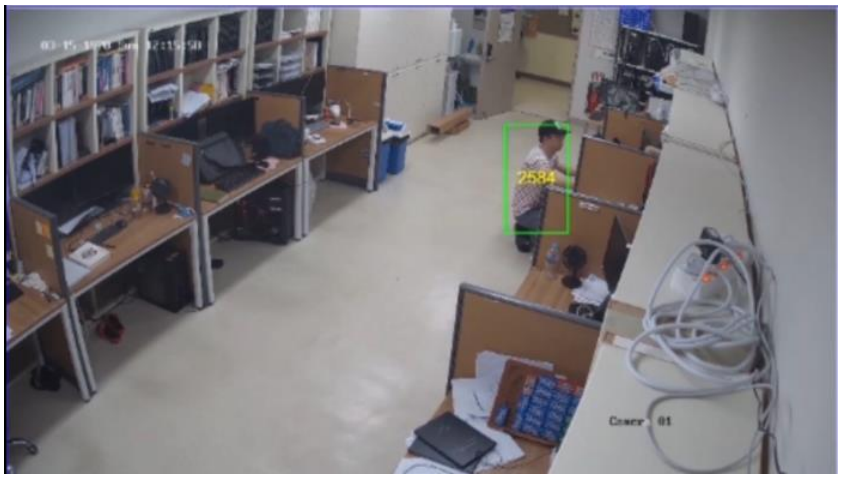
  - 보정 전: 객체(id = 785)의 단절이 발생한 뒤 같은 객체를 재인식하였으나 새로운 객체(id=786)로써 인식하였다.
  - 보정 후: 단절이 발생하여 빠진 로그들을 생성하고 새로 인식된 객체(id=786)을 기존 객체(id=785)로 변경시켜 객체인식이 연결되도록 하였다.

4.3.2. 이상 객체 보정

- **보정 전**



- **보정 후**



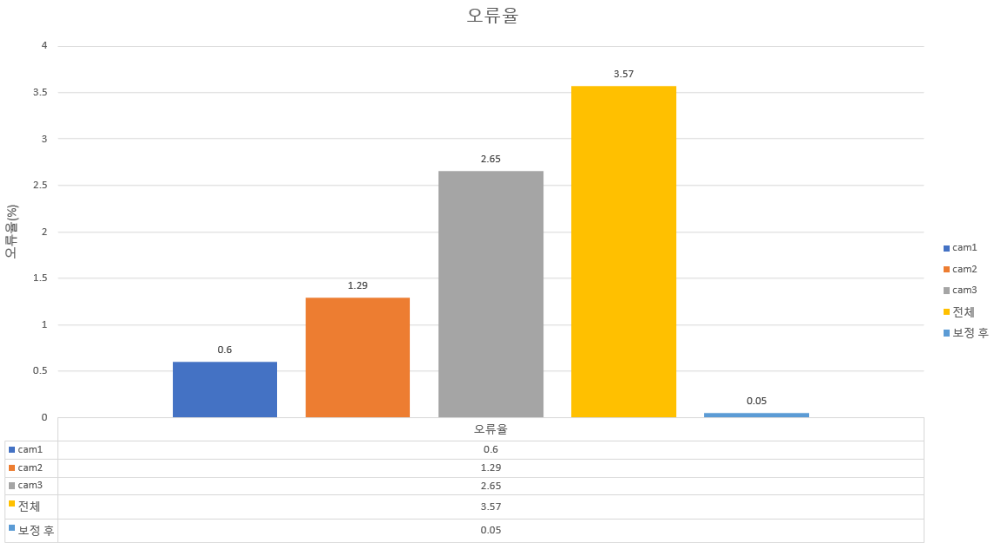
- **보정 전 로그(왼쪽) 과 보정 후 로그(오른쪽)**

2020-05-29 23:24:41.600	2,584	1,225	400	133	252
2020-05-29 23:24:41.600	2,584	1,225	400	133	252
2020-05-29 23:24:41.700	2,584	1,225	402	139	251
2020-05-29 23:24:41.800	2,584	1,225	402	139	251
2020-05-29 23:24:41.900	2,584	1,225	402	142	250
2020-05-29 23:24:42.000	2,584	1,225	397	139	251
2020-05-29 23:24:42.000	2,606	1,274	629	167	179
2020-05-29 23:24:42.100	2,584	1,222	395	142	250
2020-05-29 23:24:42.100	2,606	1,285	627	167	171
2020-05-29 23:24:42.200	2,584	1,234	396	149	258
2020-05-29 23:24:42.200	2,606	1,275	637	168	178
2020-05-29 23:24:42.300	2,584	1,254	405	156	265
2020-05-29 23:24:42.300	2,606	1,276	625	166	179
2020-05-29 23:24:42.400	2,584	1,254	405	156	266
2020-05-29 23:24:42.400	2,606	1,287	624	168	177
2020-05-29 23:24:42.500	2,584	1,252	403	156	265
2020-05-29 23:24:42.500	2,606	1,282	630	167	172
2020-05-29 23:24:42.600	2,584	1,253	405	147	254
2020-05-29 23:24:42.600	2,606	1,280	637	163	176
2020-05-29 23:24:42.700	2,584	1,220	395	145	248
2020-05-29 23:24:42.700	2,606	1,281	632	167	173
2020-05-29 23:24:42.800	2,584	1,227	399	145	246
2020-05-29 23:24:42.800	2,606	1,282	634	160	174
2020-05-29 23:24:42.900	2,584	1,220	395	145	246
2020-05-29 23:24:42.900	2,606	1,274	629	168	172
2020-05-29 23:24:43.000	2,584	1,220	395	145	246

2020-05-29 23:24:41.600	2,584	1,225	400	133	252
2020-05-29 23:24:41.600	2,584	1,225	400	133	252
2020-05-29 23:24:41.700	2,584	1,225	402	139	251
2020-05-29 23:24:41.800	2,584	1,225	402	139	251
2020-05-29 23:24:41.900	2,584	1,225	402	142	250
2020-05-29 23:24:42.000	2,584	1,225	397	139	251
2020-05-29 23:24:42.100	2,584	1,222	395	142	250
2020-05-29 23:24:42.200	2,584	1,234	396	149	258
2020-05-29 23:24:42.300	2,584	1,254	405	156	265
2020-05-29 23:24:42.400	2,584	1,254	405	156	266
2020-05-29 23:24:42.500	2,584	1,252	403	156	265
2020-05-29 23:24:42.600	2,584	1,253	405	147	254
2020-05-29 23:24:42.700	2,584	1,220	395	145	248
2020-05-29 23:24:42.800	2,584	1,227	399	145	246
2020-05-29 23:24:42.900	2,584	1,220	395	145	246
2020-05-29 23:24:43.000	2,584	1,220	395	145	246
2020-05-29 23:24:43.100	2,584	1,220	395	145	246
2020-05-29 23:24:43.200	2,584	1,223	397	145	246

- 보정 전 로그를 보면 기존 객체(id = 2584)와 함께 이상객체인 선풍기(id = 2606)가 인식되었으나 이상객체를 제거한 보정 후 로그에서는 기존 객체(id = 2584)만이 남아있다.

4.3.3. 케이스별 보정 + 오류율



세 대의 카메라 중 cam1 의 오류율이 0.6%, cam2 의 오류율이 1.29%, cam3 의 오류율이 2.65%정도로 카메라별로 차이를 보였다. 모두 같은 종류의 카메라를 사용하였으나 카메 라의 위치와 각도에 따라 오류율에 차이가 나는 것으로 보인다. 전체 오류율은 3.57%로 보정 후 오류율은 0.5%로 줄어들었다.

4.4. 정리

4.4.1. 장점과 단점

- **장점**
  - 주변 환경에 영향을 받지 않고 객체 인식 프로그램의 오류율을 낮출 수 있다.
  - 데이터를 분석하여 공간적 정보를 도출할 수 있다.
    - 사람이 오래 머무르는 자리
    - 오류가 많이 발생하는 그리드
- **단점**
  - 그리드 맵핑 테이블이 신뢰할 수 있을 정도로 학습시키는 데 시간이 소요된다.
  - 카메라의 위치가 변경되거나 새로운 카메라를 추가할 시 학습을 처음부터 다시 시행해야 한다.

- 학습을 위해 사람이 학습 공간을 직접 돌아다닐 필요가 있다.
- 한 공간을 다양한 각도에서 바라보는 다중 카메라가 필요하다. 카메라가 가까이 위치해 있는 것보다는 멀리 위치해서 객체가 약간 작게 보이는 편이 좋으며, 카메라끼리는 직교하는 위치에 설치될수록 효과가 좋다.
- 어떤 객체 인식 프로그램을 사용하느냐에 따라 보내주는 로그의 형태가 다르기 때문에 인텔리빅스가 아닌 다른 객체인식 프로그램을 사용할 경우 초기 로그를 받아서 DB 로 저장하는 부분의 코드를 수정해야 한다.

5. 일정 및 역할 분배

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
개발환경 구축 및 DB 설계														
객체 메타 데이터 추출														
타임스탬프 동기화														
중간 발표 준비														
그리드 좌표 나누기														
단일 객체 그리드 좌표 학습														
다중 객체 동기화														
객체 오류 분석 / 보정														
포스터 및 최종보고서														

: 모두 : 이경우 : 김도희 : 정유정