스마트폰과 카메라를 활용한 학생 인식 및 위치 파악 시스템

캡 스 톤 디 자 인 I I 중 간 보 고 서

20172608 이동진
20172598 김규진
20172615 최우석
20172616 한수호

CONTENTS

01

개요

프로젝트 소개 및 구현 계획

01 개요

프로젝트 소개 및 구현 계획

스마트폰의 어플리케이션과 카메라를 활용하여
학생의 위치를 파악하고 출석을 확인할 수 있는 시스템.

교수자와 학생 간의 적절한 피드백과 소통을 이끌어내어
수업의 질적 향상을 기대.

1. 카메라의 영상을 분석하여 객체를 식별, 추적하는 프로그램을 개발하고,
   이 정보를 실시간으로 DB와 사용자에게 제공할 서버를 웹서버로 구현.

2. 학생들을 개인별로 식별하기 위해 QR코드 인식 후 영상 속 객체에 ID를
   부여하고, 이를 추적하여 학생들의 위치를 파악할 수 있는
   시스템 개발을 목표.

3. 또한 사용자가 추가적인 행동을 하지 않아도 환경에 맞게 실행될 수
   있도록 사용자 맞춤 시스템을 개발할 예정. (책상 좌표)

02

구현

항목별 구현한 방법

1. 시스템 구현 방법
2. QR코드 인식을 통한 ID 부여
3. 사용자 환경 맞춤

02 구현

(1) 시스템 구현 방법

객체 추적 프로그램은 Yolov5와 StrongSORT (DeepSORT에서 변경) 알고리즘을 활용하여 Python으로 개발.

웹서버는 Node.js와 Vue.js를 활용하여 백엔드/프론트엔드를 구현.

DB는 Google Firebase Realtime DB를 활용.

02 구현

(2) QR코드 인식을 통한 ID 부여

QR코드는 스마트폰 어플리케이션에서 학번으로 QR코드 생성.

QR코드 스캔은 별도의 리더기를 설치하여 스캔하고,
스캔한 QR코드 정보가 서버로 전송되면 영상에 새롭게 인식된 객체에 학번을 ID로 부여.

객체의 위치 정보는 객체 추적 프로그램에 의해 구해진 후 서버로 전송됨.
서버는 사용자에게 정보를 제공하고 DB에 정보 갱신.

02 구현

(3) 사용자 환경 맞춤

책상 좌표를 획득하기 위한 방법으로 실사용 전에 영상에서
1프레임을 추출하고 이를 활용하여 좌석마다 좌표를 획득.

파이썬 GUI 프로그래밍으로 ROI 클릭 이벤트를 구현하여
박스 영역의 꼭지점 좌표를 획득하고 이 정보를 json으로 저장.

저장된 json 좌표 파일을 좌석 관리 클래스가 읽어 활용함.
해당 박스 내부에 존재하는 좌표의 경우 그 자리에 있는 객체로 인식.

03

문제점

항목별 문제점 소개 및
해결 방안 제시

03 문제점

문제점

1. ID 스위칭, 차폐 문제

- 카메라의 위치와 각도를 최적화하여 설치.
- 학생의 위치가 자주 변동되지 않는다는 점을 고려하여 좌석의 정보를 반고정 형태로 취급하는 방법.
- Naver CLOVA Face Recognition, StrongSORT 알고리즘 등을 활용하여 외형에 대한 정보를 추가적으로 입력.

2. 서버 처리 지연

- 프레임 스킵, DeepSORT(StrongSORT)와 SORT 알고리즘 혼합 사용 등의 방법.

계획 당시 예상했던 문제와 해결법에 대한 항목별 피드백

03 문제점

(1) ID 스위칭

객체 간의 겹침으로 인해 서로의 ID가 뒤바뀌는 현상. 탐지 결과가 유사하여 겹침이 없는 상황에 뒤바뀌는 것도 포함하여 다룸.

ID 스위칭을 예방하는 것을 힘들다고 판단하여 ID 스위칭이 발생했을 때 이것을 감지해 다시 바꾸는 방법을 선택함. DB와 프로그램 내부에 저장된 객체들의 위치 정보를 바탕으로 ID 스위칭이 발생한 이후에 저장된 위치에 있는 ID와 현재 ID가 다를 경우 두 객체의 ID를 다시 바꿔주는 코드를 작성.

```
i = 0
while (True):
    if (i == len(matches)):
        break
    track_idx, detection_idx  = matches[i][0], matches[i][1]
    cx, cy, _, _ = detections[detection_idx].to_xyah()
    loc = Location(cx, cy)
    location= loc.get_location()

    if ((location != 0) and (location != self.tracks[track_idx].location)):
        id = int(self.db.get_id(location))

        for j in range(len(matches)):
            if (self.tracks[matches[j][0]].track_id == id):
                temp_detect_idx = matches[j][1]
                matches[j] = (j, detection_idx)
                matches[i] = (track_idx, temp_detect_idx)
                break
    i += 1
```

03 문제점

(2) 차폐

객체가 사물에 가려지거나 화면 밖으로 나가서 추적이 해제되는 문제.
또한 탐지기의 탐지 실패로 추적이 해제되는 문제도 포함하여 다룸.

기존에 추적하던 객체가 해제되고 다시 등장했을 때 새로운 ID를
부여해버리는 상황이 발생하는 것을 완화하기 위해서 ID의 최대값을
설정하고, ID를 부여하기 전에 추적이 해제된 객체 중 가장 외형 정보가 유사한 객체
의 ID를 부여하도록 설정함.

```
if ((len(self.ids) >= self.max_id) and (len(self.deleted_tracks) > 0)):
    max_feature_distance = 9999
    for i in self.deleted_tracks:
        i_feature = np.array(i.features)
        i_feature_distance = np.mean(np.abs(detection.feature-i_feature))
        if (max_feature_distance > i_feature_distance):
            max_feature_distance = i_feature_distance
            self.id = i.track_id
            i.state = 2
            self.deleted_tracks.remove(i)
            print(i.track_id)
            break
else:
    self.id = self.get_id()
    self.ids.append(self.id)
```

03 문제점

(3) 처리 시간

20개 이내의 객체를 대상으로 객체 추적 프로그램은
0.2초 이내로, DB와 웹서버 모두 실시간 동작.

CPU 환경에서는 객체 추적 프로그램이 프레임 당
8초 이상 걸리는 것을 확인함.

따라서 다수의 좌석과 사람을 관리하는 상황에,
현재 개발된 수준으로는, GPU 활용이 불가피하다고 판단함.

가벼운 모델 사용은 정확도가 떨어져 ID 스위칭과 탐지 불가 문제 발생.
프레임 스킵 시 ID 스위칭과 차폐를 해결하기 위한 위 항목들의 솔루션 적용 불가.
SORT와 DeepSORT를 섞는 시도는 시간 부족으로 인하여 적용하지 않음.

GPU

```
Speed: 0.7ms pre-process, 58.1ms inference, 1.8ms NMS, 140.2ms strong sort update per image at shape (1, 3, 640, 640)
```

CPU

```
Speed: 3.4ms pre-process, 2897.4ms inference, 2.2ms NMS, 5523.1ms strong sort update per image at shape (1, 3, 640, 640)
```

04

참고 문헌

04 참고 문헌

참고 문헌

SIMPLE ONLINE AND REALTIME TRACKING WITH A DEEP ASSOCIATION METRIC
- Nicolai Wojke, Alex Bewley, Dietrich Paulus. (2017).

객체 추적 알고리즘을 활용한 딥러닝 기반 실시간 화재 탐지 시스템
- 박종혁, 박도현, 현동환, 나유민, 이수홍. (2022).

딥러닝 기반 실시간 다중 객체 추적 시스템
- 김경훈, 강석주. (2019).

SORT와 DeepSORT의 혼합을 이용한 실시간 다중객체 추적
- 양수진, 정인화, 강동화, 백형부. (2021).

감사합니다