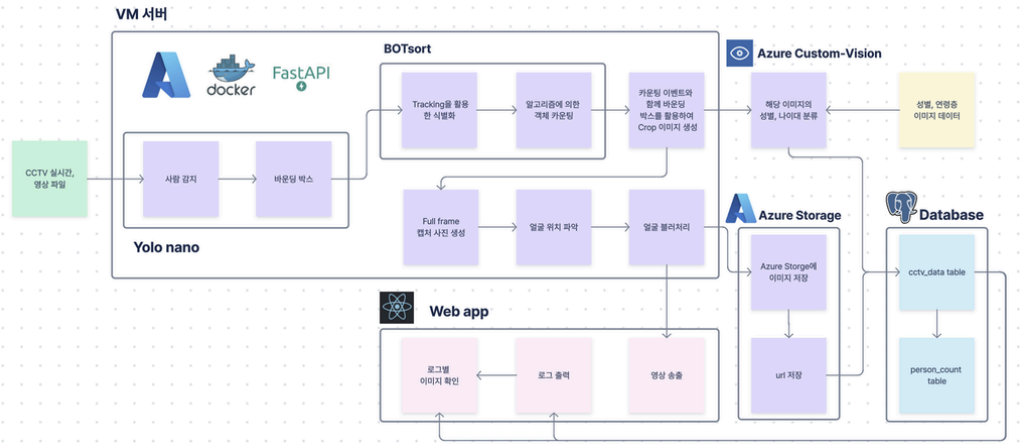
CV 알고리즘 문서



1. 전체 개요

이 코드는 CCTV 영상 스트림 내에서 사람을 탐지하고 추적하는 애플리케이션입니다. 주요 기능은 다음과 같습니다.

- **객체 탐지 및 추적:** Ultralytics의 YOLO 모델을 사용하여 영상 내 사람(객체 클래스 0)을 탐지 및 추적합니다.
- **얼굴 및 객체 처리:** 탐지된 사람의 얼굴 영역을 추출하여 블러 처리하고, 크롭된 이미지와 전체 프레임 이미지를 저장합니다.
- **Azure API 연동:** 저장된 사람 이미지에 대해 Azure API를 호출하여 성별 및 연령대 예측을 수행하고, 예측 결과를 정규화합니다.
- **백엔드 전송:** 분석 결과와 이미지를 HTTP 요청으로 백엔드 서버에 전송합니다.
- **비동기 처리:** `asyncio`와 `aiohttp`를 활용하여 비동기 방식으로 API 호출 및 데이터 전송을 수행합니다.

---

2. FastAPI 및 데이터 모델

- **FastAPI 애플리케이션:**
  `app = FastAPI()`를 통해 FastAPI 애플리케이션을 초기화합니다.
- **Pydantic 모델 (`DetectionRequest`):**
  프론트엔드에서 전달받는 요청의 구조를 정의하며, `cctv_url`과 `cctv_id`를 포함합니다.

---

3. Azure API 클래스 (AzureAPI)

이 클래스는 Azure의 이미지 분석 API와의 통신을 담당합니다.

- **환경 변수 로드:**
  `dotenv`를 사용하여 `.env` 파일에서 API URL 및 키를 불러옵니다.
- **세션 관리:**
  비동기 `aiohttp.ClientSession`을 생성하고, 종료할 수 있도록 메서드(`start`, `close`)를 제공합니다.
- **이미지 분석:**
  `analyze_image` 메서드를 통해 파일을 읽어 Azure API에 POST 요청을 보내고, 결과 JSON을 받아옵니다.
- **결과 정규화:**
  `normalize_predictions` 메서드에서는 예측 결과에서 성별 및 연령대에 해당하는 태그의 확률을 백분율로 변환하고, 그룹별로 비율을 재계산합니다.

---

4. PersonTracker 클래스

이 클래스는 YOLO 모델을 이용하여 사람을 탐지하고, 추적한 후 추가 처리를 수행합니다.

4.1. 초기화 및 속성

- **모델 로딩:**
  생성자에서 YOLO 모델을 로드하며, 디바이스 선택은 GPU(cuda)가 가능하면 사용합니다.
- **설정 값:**
  결과 저장 디렉터리, 추적기 설정 파일(`botsort.yaml`), 신뢰도(confidence), IoU, 이미지 크기, 비디오 결과 저장 경로 등을 초기화합니다.
- **상태 변수:**
  프레임, 박스, 색상 매핑, 이미 처리된 객체 ID 등을 저장하여 추적 상태를 관리합니다.
- **AzureAPI 객체 생성:**
  Azure API 연동을 위한 인스턴스를 생성하여 나중에 사용합니다.

4.2. 주요 메서드

- `is_fully_inside_frame`**:**
  주어진 바운딩 박스가 프레임 내부에 완전히 존재하는지 확인합니다.
- `generate_color`**:**
  객체의 고유 ID에 따라 임의의 색상을 생성하여, 시각적 구분을 위해 사용합니다.
- `estimate_face_area`**:**
  객체의 키포인트 데이터를 기반으로 얼굴 영역을 추정합니다.
  - 얼굴에 해당하는 키포인트(인덱스 0~4)를 선택하고, 유효한 포인트를 확인합니다.
  - 바운딩 박스와 비교하여 얼굴 영역을 계산하며, 확장 비율을 적용합니다.
- `apply_face_blur`**:**
  추정된 얼굴 영역에 대해 Gaussian Blur를 적용하여 익명화(blur 처리)합니다.
- `detect_and_track`**:**
  핵심 동작 메서드로, 영상 스트림에서 사람을 탐지하고 추적합니다.
  - **모델 추적:**
    YOLO의 `track` 메서드를 호출하여 스트림 형태로 결과를 순회합니다.
  - **Azure API 준비:**
    결과 처리 전에 Azure API 클라이언트 세션을 시작합니다.
  - **프레임 및 박스 처리:**
    각 결과(프레임)에서 탐지된 객체의 바운딩 박스 및 키포인트 데이터를 추출합니다.
  - **새 객체 처리:**
    새로운 객체가 감지되면, 얼굴 영역을 추정하고 크롭 이미지와 전체 프레임 이미지를 저장합니다.
  - **비동기 작업:**
    저장된 이미지를 기반으로 Azure API를 호출하고, 분석 결과를 처리하기 위해 `asyncio.gather`를 사용하여 병렬로 처리합니다.
  - **디스플레이:**
    블러 처리된 얼굴과 바운딩 박스, 객체 ID를 프레임에 표시하여 화면에 출력하며, 종료 키(예: 'q' 혹은 's')에 따른 동작을 처리합니다.
  - **세션 종료:**
    모든 처리가 완료되면 Azure API 세션을 종료합니다.
- `process_person`**:**
  개별 객체에 대해 Azure API를 호출하여 성별과 연령대 예측 결과를 받습니다.
  - **예측 임계치:**
    확률이 30% 이상인 결과만 고려하며, 매핑 테이블을 통해 'young', 'adult', 'old', 'male', 'female' 등으로 변환합니다.
  - **서버 전송:**
    최종 분석 결과와 관련 이미지를 백엔드 서버로 전송하기 위해 `send_data_to_server` 메서드를 호출합니다.

- `send_data_to_server`:
  분석 결과를 백엔드 서버에 HTTP POST 요청으로 전송합니다.
  - **폼 데이터 구성:**
    CCTV ID, 탐지 시간, 객체 ID, 성별, 연령대 등의 정보를 `FormData`로 구성합니다.
  - **이미지 파일 첨부:**
    이미지 파일이 존재할 경우 multipart/form-data 형식으로 첨부하여 전송합니다.
  - **비동기 HTTP 호출:**
    `aiohttp.ClientSession`을 통해 비동기 POST 요청을 수행하며, 응답 결과를 출력합니다.
- `save_cropped_person`:
  감지된 사람의 이미지를 두 가지 형태로 저장합니다.
  - **크롭된 이미지:**
    바운딩 박스 영역을 추출하여 저장합니다.
  - **전체 프레임 이미지:**
    전체 프레임에서 해당 객체가 포함된 영역은 선명하게, 그 외는 블러 처리하여 저장합니다.
  - **얼굴 블러 처리:**
    추정된 얼굴 영역에 대해서도 추가 블러 처리를 적용합니다.
  - **파일명 구성:**
    타임스탬프와 객체 ID를 포함하여 고유한 파일명으로 저장합니다.

---

5. 비동기 및 멀티태스킹 활용

- **aiohttp 및 asyncio:**
  비동기 방식으로 Azure API 호출과 서버 전송을 처리하여, 실시간 스트림 처리 시 병목 현상을 줄입니다.
- **비동기 파일 입출력:**
  `aiofiles`를 사용하여 이미지 파일을 비동기적으로 읽어들임으로써 I/O 효율을 높입니다.

---

6. 시각적 디스플레이 및 사용자 상호작용

- **OpenCV 활용:**
  각 프레임에 탐지된 객체에 대해 바운딩 박스, 객체 ID, 얼굴 블러 처리 결과를 표시합니다.
- **키보드 이벤트:**
  'q' 키를 누르면 프로세스를 종료하며, 's' 키를 누르면 일시정지 후 재개할 수 있도록 구현되어 있습니다.

결론

이 코드는 CCTV 영상에서 사람을 실시간으로 탐지하고, 추적한 후 Azure API를 통해 성별과 연령대를 예측하여 분석 결과를 백엔드 서버에 전송하는 복합적인 파이프라인을 구현합니다. 비동기 처리와 OpenCV 기반 이미지 처리, YOLO 모델 활용, 그리고 Azure API 연동이 유기적으로 결합되어 있으며, CCTV 모니터링 및 분석 시스템에 적합한 구조로 설계되어 있습니다.