Sub PJT 1

🎯 1. Google Mediapipe

1. 개요

얼굴, 손, 포즈 인식 등 다양한 비전 AI 기능을 안드로이드, iOS, 웹, 파이썬 등 여러 플랫폼에서 실시간으로 구동할 수 있도록 돕는 **오픈소스 크로스 플랫폼 프레임워크**

- **특징**: 복잡한 머신러닝 파이프라인을 파이프 형태(Graph)로 구성하여 효율적인 연산이 가능
- **장점**: 온디바이스 최적화가 잘 되어 있어 모바일 기기에서도 저지연 실시간 추론 가능

---

2. 주요 구성 요소

🔷 Tasks API

사용자가 복잡한 설정 없이 AI 모델을 실행할 수 있도록 제공되는 상위 수준의 인터페이스

- **Hand Landmarker**: 손가락 마디마디의 21개 3D 좌표를 추출합니다.
- **Pose Landmarker**: 신체 주요 관절 및 신체 선(Line)을 감지합니다.
- **Gesture Recognizer**: 손 모양을 분석하여 '따봉', '승리' 등의 의미를 파악합니다.

🔷 Running Modes (실행 모드)

MediaPipe는 데이터의 형태에 따라 세 가지 실행 모드를 지원

1. **IMAGE**: 단일 이미지 처리
2. **VIDEO**: 비디오 파일 프레임별 처리
3. **LIVE_STREAM (사용 중)**: 카메라 입력에 최적화된 비동기 모드, AI 추론이 진행되는 동안 UI나 카메라 프레임이 멈추지 않는 **Non-blocking** 방식

---

3. 실전 적용 테크닉

🚀 성능 최적화

- **GPU Delegation**: CPU 대신 GPU를 활용하여 연산 속도를 극대화

- **Asynchronous Callback**: `recognize_async` 함수와 콜백(Callback) 시스템을 사용하여 카메라 프레임 수(FPS)와 AI 추론 속도를 분리함으로써 부드러운 화면 전환을 유지

---

4. 개발 환경 주요 명령어 정리

📖 라이브러리 설치

```
# MediaPipe 기본 라이브러리 (추론 및 시각화용)
pip install mediapipe

# MediaPipe Model Maker (커스텀 제스처 학습용)
# *주의: 학습을 위해서는 이 패키지가 반드시 별도로 필요합니다.
pip install mediapipe-model-maker

# 이미지 처리 및 데이터 수집용 OpenCV
pip install opencv-python

# 수치 계산 및 데이터 처리용 (Numpy 등)
pip install numpy
```

✏️ 모델 학습 및 실행

```
# 데이터셋 로드
# 폴더 구조로부터 데이터를 읽어오는 명령어
data = gesture_recognizer.Dataset.from_folder(dirname=IMAGES_PATH, hp
arams=...)

# 모델 내보내기
# S24 등 모바일 기기에서 사용할 수 있는 .task 파일 생성
model.export_model()

# 작성한 추론/수집 스크립트 실행
python main.py

# 현재 설치된 라이브러리 버전 확인 (MediaPipe 버전 충돌 디버깅 시)
pip show mediapipe
```

🎯 2. OpenCV

1. 개요

OpenCV는 실시간 컴퓨터 비전을 목적으로 하는 오픈소스 라이브러리

본 프로젝트에서는 웹캠 영상을 캡처하고, AI 추론 결과를 시각화하며, 학습용 데이터셋을 생성하는 전천후 도구로 사용

향후 모델 고도화에 이미지 전처리가 필요할 경우 변환, 필터 등을 활용할 예정

---

2. 주요 활용 기술

🔷 영상 입출력 및 제어 (I/O & Control)

- `cv2.VideoCapture(0)`: 로컬 웹캠 장치에 연결하여 실시간 스트림을 생성
- `cap.read()`: 카메라로부터 프레임 단위로 이미지를 읽기
- `cv2.waitKey(1)`: 키보드 입력을 대기하며, 프레임 속도(FPS)를 조절
  - `s`: 데이터 저장 / `q`: 프로그램 종료 로직에 활용
- `cap.release()` & `cv2.destroyAllWindows()`: 프로그램 종료 시 시스템 자원을 안전 해제

🔷 이미지 전처리

- **색상 공간 변환 (`cv2.cvtColor`)**:
  - OpenCV는 기본적으로 **BGR** 순서로 이미지를 읽지만, MediaPipe는 **RGB**를 요구해서 `cv2.COLOR_BGR2RGB` 변환이 필수적
- **해상도 및 크기 확인 (`frame.shape`)**: 영상의 너비와 높이를 추출

🔷 시각화 및 UI (Visualization)

- **도형 그리기 (`cv2.circle`)**: 추출된 21개의 핸드 랜드마크를 화면에 점으로 표시하여 인식 상태를 모니터링
- **텍스트 렌더링 (`cv2.putText`)**:
  - 인식된 제스처 이름(Label) 출력
  - 실시간 FPS 및 Latency(지연 시간) 수치 표시
- **윈도우 출력 (`cv2.imshow`)**: 최종 처리된 영상을 실시간으로 사용자에게 보여줌

🔷 데이터셋 구축 (Data Engineering)

- **이미지 저장 (`cv2.imwrite`)**: `DataCollector` 클래스 내에서 특정 시점의 프레임을 `.jpg` 파일로 저장하여 커스텀 학습용 이미지 데이터베이스(DB)를 구축

---

3. 주의사항

💡 BGR vs RGB 주의

OpenCV로 이미지를 저장하거나 출력할 때는 BGR 순서를 따르지만, AI 모델(MediaPipe)에 데이터를 던질 때는 반드시 RGB로 변환해야 인식률 저하를 막을 수 있음

💡 성능 최적화

실시간성 확보를 위해 영상 처리 루프 안에서 무거운 연산을 피해야 하며, cv2.imshow 호출 빈도를 최적화하는 것이 중요

🎯 3. 26종 제스처 데이터셋 설계 및 수집 전략

1. 데이터셋 설계 개요

본 프로젝트는 발표용 제스처 제어 시스템의 높은 신뢰성을 확보하기 위해, 단순한 모양 구분을 넘어 **방향성(Orientation)**과 **상태 변화(Transition)**를 포함한 총 26종의 세분화된 라벨 체계를 구축하였습니다.

---

2. 제스처 분류 체계 (Labeling Taxonomy)

① 기본 제스처 (Standard Gestures)

일상적이고 직관적인 손 모양으로 구성됩니다.

- `None`, `Closed_Fist`, `Open_Palm`, `Pointing_Up`, `Thumb_Up`, `Thumb_Down`, `Victory`, `Okay`, `ILoveYou`, `call_hand`

② 상태 변화 대응 (Variation)

동작 사이의 모호한 구간을 메워 인식의 끊김을 방지합니다.

- `Curve_Palm`: 보자기와 주먹 사이의 'Dead Zone'을 메우기 위해 손가락을 약간 구부린 상태를 별도 라벨링.

③ 회전 및 반전 (Rotated & Flipped)

S24 카메라 각도 변화와 사용자의 다양한 손 방향에 대응합니다.

- **90° / 135° 분리**: 손을 위로 세운 상태와 옆으로 눕힌 상태를 다른 의미(기능)로 쓰기 위해 각도별로 세분화.
- **Flip (손등/손바닥)**: 같은 모양이라도 카메라가 보는 면에 따라 랜드마크 안정성이 다르므로 이를 구분하여 학습.
- `Victory_90`, `Open_Palm_90`, `Open_Palm_135`, `Open_Palm_90_Flip` 등

④ 특수 포즈 및 양손 대응 (Special Poses)

- `Son_pose_Left / Right`: 손흥민 세레머니와 같이 양손이 교차하는 포즈는 한 손 단위의 기하학적 형태가 다르므로 왼손과 오른손을 각각 별도 클래스로 학습시킨 후 로직으로 결합.

---

3. 데이터 수집 전략 (Collection Strategy)

🔷 도메인 특화 환경 (Domain Adaptation)

- **PPT 배경 수집**: 실제 발표 현장(스크린 앞)의 텍스트, 차트, 광원 노이즈를 포함하여 배경에 의한 오인식을 최소화.
- **조명 가변성**: 발표장의 저조도 환경과 프로젝터의 역광 상황을 반영하여 밝기(Brightness)와 대비(Contrast)가 다양한 데이터를 확보.

🔷 데이터 품질 관리 (Data Quality)

- **좌우 반전 금지 (No Horizontal Flip)**: 왼손과 오른손의 엄지 위치 등을 명확히 구분해야 하므로, 인위적인 반전 증강을 배제하고 직접 양손 데이터를 각각 촬영.
- **거리 및 크기 변화 (Scale Variation)**: 사용자가 무대 위에서 이동하는 상황을 고려하여, 손이 화면에 꽉 차는 근거리부터 작게 보이는 원거리까지 골고루 수집.

---

4. 효율적 수집을 위한 기술적 도구

🛠️ `DataCollector` 클래스 설계

데이터 수집의 일관성과 속도를 위해 커스텀 도구를 직접 구현하였습니다.

- **기능**:
  1. **자동 카운팅**: 폴더 내 기존 파일 개수를 파악하여 번호 덮어쓰기 방지.
  2. **실시간 저장**: 특정 키(`s`) 입력 시 즉시 해당 제스처 폴더로 이미지 저장.
  3. **데이터 정제**: 수집 단계에서 MediaPipe 랜드마크 시각화를 병행하여, 좌표가 불안정한 프레임은 즉시 제외.

---

5. 설계 의의

- **강건성(Robustness)**: 회전과 반전을 고려한 설계로 사용자가 어떤 자세를 취하든 안정적인 제어가 가능함.
- **정교함**: 90도와 135도 등 미세한 각도 차이를 데이터로 해결하려 노력함으로써 하드웨어(S24) 성능을 극한으로 활용함.