hw1_보고서

20기 인턴 송수민

목차

1. 컴퓨터 비전

1.1. 개요

인공지능의 한 분야

머신 러닝과 신경망을 활용해 컴퓨터와 시스템이 이미지, 비디오 및 기타 시각적 입력에서 의미 있는 정보를 추출

1.2. 컴퓨터 비전의 기본 처리 과정

1) 데이터 수집 및 준비

데이터 수집: 카메라, 비디오 등으로부터 시각적 데이터를 획득

데이터 전처리: 수집된 이미지의 노이즈를 제거하고, 영상의 대비를 조절하여 선명도를 높이는 등, 컴퓨터가 분석하기 적절한 형태로 데이터를 정제

2) 특징 추출

이미지에서 객체를 인식하기 위해 필요한 특징을 추출한다. 예를 들어, 객체의 윤곽선, 모서리, 색상, 질감 등과 같은 수치적 데이터를 추출하여 데이터로 변환하는 과정이다.

3) 모델 학습

추출된 특징들을 사용하여 딥러닝 모델을 학습시킨다.

이 과정에서 대량의 시각 데이터(이미지)를 기계에 학습시켜 패턴을 인식하고, 객체를 식별하는 방법을 배우도록 한다.

4) 객체 인식 및 분석

학습이 완료된 모델은 새로운 이미지를 입력받았을 때, 이미지 속에서 특정 객체를 정확하게 식별하고 분류한다.

예를 들어, 꽃 사진을 보고 해당 꽃의 종을 식별하거나, 사진 속 사람 수를 파악하는 등의 작업을 수행한다.

5) 최종 판단 및 반응

인식된 정보를 기반으로 시스템은 상황에 맞는 판단을 내리고 필요한 반응을 한다.

1.3. 컴퓨터 비전의 응용 분야

자율주행 자동차: 도로의 차선, 보행자, 신호등 등을 인식하여 안전한 주행을 지원.

의료 영상 분석: X-ray, MRI, CT 영상에서 질병이나 종양을 자동으로 탐지.

보안 및 감시 시스템: 얼굴 인식, 이상 행동 감지, 침입자 탐지 등에 활용.

산업 자동화: 제조업에서 불량품 검출, 품질 관리, 로봇 비전.

스마트 시티 및 IoT: 교통 흐름 분석, 군중 밀집 상황 모니터링.

1.4. 컴퓨터 비전의 기술적 과제

데이터 부족 문제: 딥러닝 모델은 대량의 데이터가 필요한데, 특정 분야의 데이터는 제한적일 수 있음.

연산 비용: 대규모 영상 데이터를 처리하기 위해 고성능 GPU와 메모리가 필요.

환경 변화에 대한 민감성: 조명, 날씨, 각도 등에 따라 인식률이 크게 달라질 수 있음.

프라이버시 이슈: 얼굴 인식 등은 개인정보 보호와 충돌할 수 있음.

1.5. 컴퓨터 비전의 미래 전망

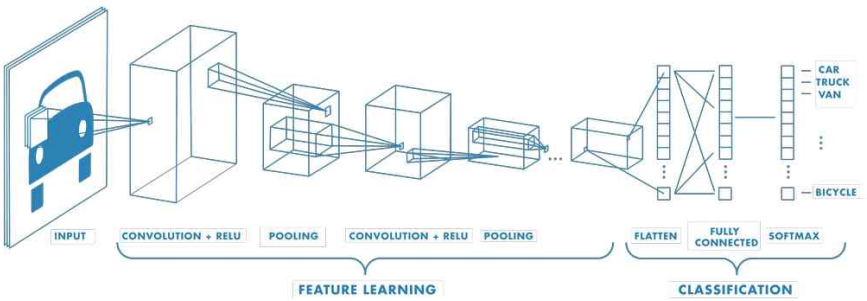
멀티모달 인공지능: 텍스트, 음성, 영상 데이터를 결합하여 더 정밀한 분석 가능.

실시간 처리 기술 발전: 5G/6G 통신과 함께 초저지연 환경에서의 영상 인식 확산.

엣지 컴퓨팅: 클라우드가 아닌 디바이스 자체에서 영상 분석을 수행하여 보안과 효율성 강화.

생성형 AI와 결합: 단순 인식이 아닌, 새로운 영상 생성 및 시뮬레이션에도 활용.

1.6. 컴퓨터 비전에서 활용되는 주요 딥러닝 모델



1) CNN (Convolutional Neural Network, 합성곱 신경망)
이미지 처리에 가장 널리 사용되는 구조.
합성곱 연산을 통해 이미지의 지역적 특징(윤곽선, 모서리, 패턴)을 자동으로 추출.
대표적 모델: LeNet, AlexNet, VGGNet, ResNet, EfficientNet 등.

2) RNN (Recurrent Neural Network, 순환 신경망)
주로 연속된 데이터(시퀀스) 처리에 강점.
영상 속 연속된 프레임(비디오 분석, 행동 인식)에서 시간적 패턴을 학습하는 데 활용.
CNN과 결합해 비디오 행동 인식 등에 적용.

3) Vision Transformer (ViT)
원래 텍스트 처리에 쓰이던 Transformer 구조를 이미지 인식에 적용.
이미지를 작은 패치로 나누어 전역적인 관계(global context)를 학습.
대용량 데이터와 GPU 자원이 있을 때 CNN보다 뛰어난 성능을 보이기도 함.

2. opencv
2.1 개요
오픈 소스 컴퓨터 비전 라이브러리 중 하나
실시간 이미지 처리에 중점을 두었음

2.2 opencv의 특징
1) 오픈소스 라이브러리
무료로 사용할 수 있으며, 전 세계 개발자 커뮤니티가 활발하게 유지·보수.
다양한 운영체제(Windows, Linux, macOS, Android, iOS)에서 사용 가능.

2) 광범위한 기능 지원
이미지 처리(필터링, 색 공간 변환, 에지 검출 등).
객체 인식 및 추적(얼굴 인식, 보행자 검출 등).
머신 러닝과 딥러닝 프레임워크와의 연동(TensorFlow, PyTorch 등).
카메라 캘리브레이션, 3D 재구성, 증강현실(AR) 기능.

3) 고속 처리
C/C++ 기반으로 개발되어 성능이 우수하며, GPU(CUDA, OpenCL) 가속 지원.
실시간 이미지·영상 처리에 최적화.

4) 풍부한 언어 바인딩
C++, Python, Java 등 다양한 언어에서 사용 가능.
특히 Python 바인딩(PyOpenCV)이 교육용 및 연구용으로 널리 활용.

5) 광범위한 활용 분야
자율주행, 로봇 비전, 산업 자동화, 의료 영상 분석, 보안 시스템 등 여러 분야에서 실질적으로 사용.

2.3 opencv의 주요 함수
1) 영상 읽기 / 출력 / 저장
- 이미지 읽기
cv::Mat img = cv::imread("image.jpg", cv::IMREAD_COLOR);
컬러 영상 읽기

cv::Mat gray = cv::imread("image.jpg", cv::IMREAD_GRAYSCALE);
흑백 영상 읽기

cv::Mat unchanged = cv::imread("image.jpg", cv::IMREAD_UNCHANGED);
원본 그대로 읽기

지원 포맷: JPG, PNG, BMP, TIFF 등
주의: 이미지 로드 실패 시 img.empty()가 true

- 영상 출력
cv::imshow("Window", img);
지정한 윈도우에 영상 출력

cv::waitKey(0);
키 입력 대기 (0 = 무한 대기, n = nms)

- 영상 저장
cv::imwrite("output.jpg", img);
영상 파일로 저장 (JPG, PNG 등)

2) 영상 생성 / 메모리 관리
cv::Mat newImg(rows, cols, CV_8UC3, cv::Scalar(0,0,0));
새 컬러 영상 생성, 초기값 검정색

cv::Mat copyImg = img.clone();
영상 깊은 복사

img.release();
영상 메모리 해제

- 자료형 예시
CV_8UC1 : 8비트 1채널 (흑백)
CV_8UC3 : 8비트 3채널 (컬러)
cv::Scalar(B,G,R) : 채널별 초기값 지정

3) 산술 연산
- 두 영상 간 연산
cv::add(src1, src2, dst); 덧셈
cv::subtract(src1, src2, dst); 뺄셈
cv::multiply(src1, src2, dst); 곱셈
cv::divide(src1, src2, dst); 나눗셈

- 단일 영상 연산
cv::add(src, cv::Scalar(value), dst); 상수 더하기
cv::bitwise_not(src, dst); 영상 반전

4) 이진화 / 컬러 변환 / 필터링

이진화

cv::threshold(src, dst, thresh, maxval, type);

type: cv::THRESH_BINARY, cv::THRESH_BINARY_INV,
cv::THRESH_TRUNC, cv::THRESH_TOZERO

컬러 변환

cv::cvtColor(src, dst, cv::COLOR_BGR2GRAY); 컬러 → 흑백

cv::cvtColor(src, dst, cv::COLOR_BGR2HSV); BGR → HSV

블러링

cv::GaussianBlur(src, dst, ksize, sigmaX); 가우시안 블러

cv::medianBlur(src, dst, ksize); 미디언 블러

cv::blur(src, dst, ksize); 단순 평균 블러

5) 에지 검출 / 변환

cv::Sobel(src, dst, ddepth, dx, dy, ksize); Sobel 에지

cv::Laplacian(src, dst, ddepth, ksize); Laplacian 에지

cv::Canny(src, edges, threshold1, threshold2); Canny 에지

cv::convertScaleAbs(src, dst); 스케일 변환 및 절대값

6) 도형 / 텍스트 삽입

도형

cv::rectangle(img, pt1, pt2, color, thickness); 사각형

cv::circle(img, center, radius, color, thickness); 원

cv::line(img, pt1, pt2, color, thickness); 선

텍스트 삽입

```
cv::putText(
    img,                          // 텍스트를 넣을 영상
    "Hello OpenCV",               // 출력 문자열
    cv::Point(50, 50),            // 시작 좌표 (좌측 하단 기준)
    cv::FONT_HERSHEY_SIMPLEX,// 폰트 종류
    1.0,                          // 폰트 크기 배율
```

```
    cv::Scalar(255,0,0),       // 글자 색상 (BGR)
    2,                          // 글자 두께
    cv::LINE_AA,                // 선 형태
    false                       // 원점 기준 (좌측 하단 = false)
);
```

설명: 영상에 텍스트를 추가 가능, 위치, 폰트, 크기, 색상, 두께 등 조절 가능

7) 영상 기하 변환

cv::resize(src, dst, cv::Size(width, height)); 영상 크기 조절

cv::warpAffine(src, dst, M, dsize); 이동, 회전, 스케일 변환

cv::flip(src, dst, flipCode); 상하/좌우 반전 (0=세로, 1=가로, -1=둘 다)

8) 형태학적 연산

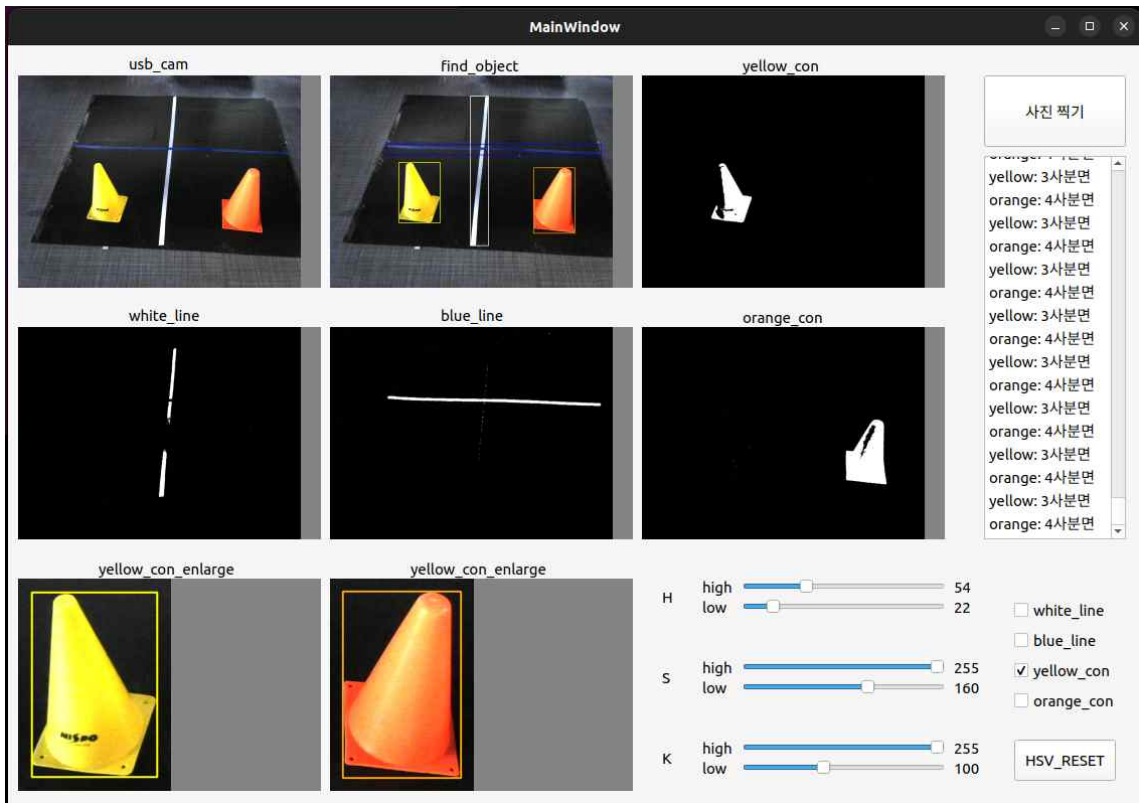
cv::erode(src, dst, kernel); 침식

cv::dilate(src, dst, kernel); 팽창

cv::morphologyEx(src, dst, op, kernel); 오프닝/클로징 등

3. hw1

3.1. 실행 결과



- 기능 설명

1) 사진 찍기 버튼 클릭 시 웹캠으로부터 이미지를 가져옴

2) 오른쪽 아래 4개의 체크박스 선택 시 원하는 색상의 hsv값 변경 가능

3) 리셋 버튼 클릭 시 모든 hsv값을 초깃값으로 리셋함

3.2. 코드 설명

1) 생성자

hsv값을 조정하는 슬라이더의 범위를 hsv의 범위와 동일하게 0-179, 0-255, 0-255로 설정함

ui에 있는 각각의 버튼과 슬라이더들과 함수를 연결함

카메라를 열고 만약 열지 못한 경우 cmd를 통해서 사용자에게 알려줌

각 색상에 대한 hsv 기본값을 저장함

타이머와 함수를 연결해서 hsv값 변경 시 자동으로 바운더리 박스와 바이너리 이미지를 생성 후 ui에 보여줌

체크 박스를 초기화 함

2) 이미지 업데이트 함수
이미지가 있을 때만 작동하며 타이머와 연결되어서 자동으로 30ms마다 호출됨
이미지를 hsv이미지로 변환함
바이너리 이미지를 생성하고 ui에 띄우기 위해 업데이트 마스크 함수를 호출함
바운딩 박스를 그리기 위해 원본 이미지를 복사한 이미지를 하나 생성한 후
바운딩 박스와 위치를 확인해주는 바운딩 박스 함수를 호출함
위 함수에서 바운더리 박스를 그린 이미지를 rgb이미지로 변환 후 ui에 띄워줌
슬라이더로 조정한 hsv값을 슬라이더 옆에 있는 라벨에도 업데이트 함

3) 마스크 업데이트 함수
바이너리 이미지를 생성하고 ui에 띄우기 위한 함수로 이미지 업데이트 함수에서 호출 됨
이미지 업데이트 함수에서 호출할 때 어떤 색상의 바이너리 이미지를 만들지와 어느 ui애 띄울지를 value 값으로 줘서 호출하므로 원하는 색상의 바이너리 이미지를 생성함
ui에 띄우기 전 필터와 침식 팽창을 적용해서 전처리를 함
전처리 후 라벨에 바이너리 이미지를 띄움

4) 바운딩 박스 함수
마스크 함수와 같이 이미지 함수에서 호출되며 위에서 생성한 마스크 이미지와 복사된 원본 이미지, 어떤 색상을 바운더리 박스를 그릴지를 변수로 받음

외각선은 바이너리 이미지의 각 포인터들의 백터의 집합이므로 백터를 저장할 배열을 선언함
복사된 이미지와 탐지된 외각선과 가장 바깥 외각선과 꼭짓점을 저장하는 변수를 선언함
만약 바이너리 이미지가 존재하지 않을 시 종료하며 노랑과 오렌지일 경우

콘이 없다고도 출력 후 종료한다.

바이너리 이미지로 나온 모든 포인터들의 백터를 더해서 바운더리 박스를 그릴 것이므로 일정 크기 이하의 포인터들은 노이즈로 판단하고 저장하지 않음
만약 저장된 이미지가 없을 경우 위와 같이 종료함

cv::Rect bbox = cv::boundingRect(filteredPoints);를 통해서 포인터 백터들의 집합에서 외곽선을 추출해서 왼쪽 상단의 xy, 가로, 세로 길이로 저장함
이후 색상별 바운더리 박스를 생성 후 이미지에 저장한다.

콘일 경우 위치를 판단해야 함
만약 존재하지 않을 시 종료함
이미지의 중심점의 위치를 가져옴
파란 선과 하얀 선이 선이 아니라 면으로 위치를 판단하기 위한 허용 범위를 설정함
파란 선 기준으로 위, 아래, 파란선 위를 판단하고 하얀 선 기준으로 좌우, 하얀선 위를 판단한다.
최종적으로 위치를 좌표 평면 위 9개의 위치 + 없음, 총 10개의 위치로 나타낸다.

바운딩 박스 주변 여유 픽셀을 설정하고 여유 영역을 설정한 박스를 생성
생성한 박스 만큼의 이미지를 바운더리 박스를 그린 이미지에서 추출해서 새로운 ui에 띄움

5) 키보드 이벤트 처리 함수
스페이스 키로도 사진을 찍을 수 있도록 사진찍기 함수랑 연결하였음

6) 사진 찍기 함수
카메라를 못 불러온 경우 종료하고 이미지를 카메라로부터 불러오고 불러온 이미지가 없을 시 종료, 이미지가 있을 시 이미지를 ui애 띄워주는 함수를 호출함

7) 이미지 띄워주는 함수
이미지를 rgb로 변환 후 ui에 이미지를 띄워줌

8) 슬라이더 함수
각각의 슬라이더와 hsv값을 조정하도록 설정하며 사용자가 체크한 색상의 hsv 값만 조정한다.

9) 체크박스 클릭 처리 함수
체크박스 클릭 시 하나만 선택 가능하도록 선택한 체크박스 이외의 체크박스를 비활성화 해주고 슬라이더를 현재 저장되어 있는 hsv값으로 동기화를 해준다.

10) hsv 리셋
클릭 시 설정된 hsv값을 기본값으로 되돌리고 초기화 시 선택된 체크박스를 흰색으로 설정한다.