컴퓨터비전 기말(2차) 프로젝트

(빨간 표지판 인식 - 주의, 경고)

201920952_김준영

컴퓨터비전 프로젝트의 목적

도로 안전과 관련하여, 표지판 인식은 자동차 운전자와 보행자 모두에게 필수적인 요소이다. 빨간색 표지판은 주의, 경고, 금지 등 중요한 정보를 전달하는 데 사용되어, 운전자가 필수적으로 인지하고 반응할 수 있도록 사용되고 있다. 이러한 표지판 인식으로 도로 안전을 유지하고 교통사고를 예방하기 위해서 이 프로젝트를 주제로 선정하였다. 이 기술은 자율 주행 자동차, 스마트 도시 인프라, 그리고 향상된 운전 보조 시스템 개발에 중요한 기여할 수 있고, 빨간색 표지판을 정확하게 인식하고 해석함으로써, 운전자의 반응 시간을 단축하고 잠재적인 위험 상황을 대비하기 위해 영상이나 동영상에서 빨간색 표지판을 강조하여 사고를 예방하고자 한다.

순서 : opencv로 표지판 영역 검출 -> 좌표를 .txt로 저장한 후 -> c에서 .txt를 불러와 표지판 영역를 확대함.

opencv 알고리즘

1. 빨간색 객체를 더 정확히 인지하기 위해 BGR 색상 공간 -> HSV 색상 공간으로 변경
2. 빨간색 픽셀을 감지하기 위한 마스크 생성
3. 마스크에 이진 침식과 이진 팽창을 적용하여 노이즈 경계를 제거.
4. 침식 및 팽창 처리된 마스크에서 윤곽선(contour)를 찾는다.
5. 윤관선 검출이 된 객체의 위치를 사각형으로 표시하고 강조를 위해 화살표와 영역 회전 및 확대
6. 사각형으로 표시된 부분이 따로 crop되어 영상으로 나옴.
7. 사용자가 'q'를 누르면 모든 창 종료.

c 알고리즘

1. opencv에서 구한 표지판의 좌표를 불러온다.
2. 그 좌표부분을 1.5배 확대시킨 뒤 원래 입력영상에 넣는다.

opencv 코드 - 이미지

```
#include <opencv2/opencv.hpp>
using namespace cv;
using namespace std;

int main(int argc, char* argv[]) {
    // 입력 영상 불러오기
    Mat image = imread(argv[1]);
```

```
    // 입력 영상을 HSV 색상 공간으로 변환
    Mat hsvImage; // HSV의 변환된 이미지 저장을 위해 객체 선언
    // 색상 공간 변환 함수 사용. BGR보다 더 정확하게 찾기 위함.
    cvtColor(image, hsvImage, COLOR_BGR2HSV); // 입력 영상, 출력 영상, 색상 변환 유형

    // 빨간색을 감지하기 위한 범위 설정. 색상, 채도, 명도 조절. 빨간색을 효과적으로 찾기 위해
       두 가지의 범위로 나눠서 설정. opencv에서 hsv 색상 값이 원의 양 끝에 걸쳐 있음.
    Scalar lowerRed1 = Scalar(0, 150, 50);
    Scalar upperRed1 = Scalar(5, 255, 255);
    Scalar lowerRed2 = Scalar(170, 150, 100);
    Scalar upperRed2 = Scalar(180, 255, 255);

    // 두 영역의 빨간색 마스크 생성
    inRange : 입력영상, 색상 하한 범위, 색상 상한 범위, 결과 마스크 영상
    Mat mask1, mask2;
    inRange(hsvImage, lowerRed1, upperRed1, mask1);
    inRange(hsvImage, lowerRed2, upperRed2, mask2);

    // 두 마스크 합치기
    Mat mask = mask1 | mask2;

    // 추가적인 색상 필터링: 채도와 명도를 기준으로
    Mat filteredMask;
    inRange(hsvImage, Scalar(0, 100, 50), Scalar(180, 255, 255), filteredMask);

    // 마스크와 필터링된 마스크의 비트 AND 연산
    Mat finalMask = mask & filteredMask;

    // 이진 침식(salt 제거)
    int erosionSize = 1; // 침식 크기 설정
    // 침식에 유용한 함수이용. element에 저장후 침식됨.
    Mat element = getStructuringElement(MORPH_RECT, Size(2 * erosionSize + 1, 2 *
erosionSize + 1), Point(erosionSize, erosionSize));
    erode(finalMask, finalMask, element);

    // 이진 팽창(pepper 제거)
    int dilationSize = 1; // 팽창 크기 설정
    // 팽창에 유용한 함수이용. element에 저장후 팽창됨.
    Mat element1 = getStructuringElement(MORPH_RECT, Size(2 * dilationSize + 1, 2 *
dilationSize + 1), Point(dilationSize, dilationSize));
    dilate(finalMask, finalMask, element1);

    vector<vector<Point>> contours; // 윤곽선을 저장할 벡터 선언
    findCountours : 윤곽선 찾아내는 함수.
    // 영상, 찾아낸 윤곽선 저장 변수, 윤곽선 검출모드(가장 바깥쪽), 메모리 사용 감소
```

```
findContours(finalMask, contours, RETR_EXTERNAL, CHAIN_APPROX_SIMPLE);
ofstream coordFile("coordinates.txt"); // .txt 파일을 불러옴.
const double MIN_AREA = 400.0; // 최소 영역 범위 설정.

int index = 0;
for (const auto& contour : contours) {
    if (contourArea(contour) < MIN_AREA) // 너무 작은 객체는 무시. 표지판 영역만
        continue;
    Rect boundingBox = boundingRect(contour); // 직사각형 모양 계산
    // 찾은 표지판을 영상에 빨간색으로 표시. 선 두께는 2
    rectangle(image, boundingBox, Scalar(0, 0, 255), 2);

    // 꼭짓점 좌표 계산
    int x1 = boundingBox.x;
    int y1 = boundingBox.y;
    int x2 = boundingBox.x + boundingBox.width;
    int y2 = boundingBox.y + boundingBox.height;

    coordFile << x1 << " " << y1 << " " << x2 << " " << y2 << endl;

    // 바운딩 박스의 왼쪽 변 중간점
    Point middleLeft(boundingBox.x, boundingBox.y + boundingBox.height / 2);

    // 화살표의 시작점 (왼쪽 변에서 왼쪽으로 50픽셀)
    Point arrowStart(middleLeft.x - 60, middleLeft.y);

    // 화살표 그리기(영상, 시작점, 화살표의 위치, 노란색, 화살표의 크기는 4.5크기로)
    arrowedLine(image, arrowStart, middleLeft, Scalar(0, 255, 255), 4.5);
    // 회전 및 확대할 영역 추출
    Mat zoomedArea = image(boundingBox);

    // 회전 행렬 생성
    Point2f center = Point2f(zoomedArea.cols / 2.0f, zoomedArea.rows / 2.0f);
    Mat rotationMatrix = getRotationMatrix2D(center, 0, 1.0); // 확대 비율 1.0

    // 영역 회전 및 확대 적용
    Mat rotatedZoomedArea;
    warpAffine(zoomedArea, rotatedZoomedArea, rotationMatrix, zoomedArea.size());

    // 회전 및 확대된 이미지 표시
    string windowName = "Rotated and Zoomed Area " + to_string(index++);
    imshow(windowName, rotatedZoomedArea);
}
imshow("빨간 표지판 인식", image); // 빨간 표지판 부분 보여주기
if (waitKey(0) == 'q') // q를 누르면 종료
```

```
        destroyAllWindows(); // 모든 창을 닫기
    return 0;
}
```

opencv - 동영상

```
#include <opencv2/opencv.hpp>
using namespace cv;
using namespace std;

int main(int argc, char *argv[]) {
    // 동영상 파일 열기
    VideoCapture video(argv[1]);

    Mat image1;
    while (1) {
        // 프레임을 읽기
        video >> image1;

        // 동영상이 끝나면 중단
        if (image1.empty()) {
            break;
        }

        // HSV 색상 공간으로 변환
        Mat hsvImage;
        cvtColor(image1, hsvImage, COLOR_BGR2HSV);

        // 빨간색 범위 조정
        Scalar lowerRed1 = Scalar(0, 150, 100);
        Scalar upperRed1 = Scalar(5, 255, 255);
        Scalar lowerRed2 = Scalar(170, 150, 100);
        Scalar upperRed2 = Scalar(180, 255, 255);

        // 빨간색 마스크 생성
        Mat mask1, mask2;
        inRange(hsvImage, lowerRed1, upperRed1, mask1);
        inRange(hsvImage, lowerRed2, upperRed2, mask2);

        // 두 마스크 합치기
        Mat mask = mask1 | mask2;

        // 추가적인 색상 필터링: 채도와 명도를 기준으로
        Mat filteredMask;
        inRange(hsvImage, Scalar(0, 100, 50), Scalar(180, 255, 255), filteredMask);
```

```
        // 마스크와 필터링된 마스크의 비트 AND 연산
        Mat finalMask = mask & filteredMask;

        // 이진 침식(salt 제거)
        int erosionSize = 1; // 침식 크기 설정
        Mat element = getStructuringElement(MORPH_RECT, Size(2 * erosionSize + 1, 2 *
erosionSize + 1), Point(erosionSize, erosionSize));
        erode(finalMask, finalMask, element);

        // 이진 팽창(pepper 제거)
        int dilationSize = 1; // 팽창 크기 설정
        Mat element1 = getStructuringElement(MORPH_RECT, Size(2 * dilationSize + 1, 2 *
dilationSize + 1), Point(dilationSize, dilationSize));
        dilate(finalMask, finalMask, element1);

        // 빨간색 영역을 바운딩 박스로 표시하고 화살표 그리기
        vector<vector<Point>> contours;
        findContours(finalMask, contours, RETR_EXTERNAL, CHAIN_APPROX_SIMPLE);

        const double MIN_AREA = 500.0; // 최소 영역 범위 설정
        for (const auto &contour : contours) {
            if (contourArea(contour) < MIN_AREA) 너무 작은 객체는 무시. 표지판 영역만
                continue;
            Rect boundingBox = boundingRect(contour); 직사각형 모양 계산

            // 찾은 표지판을 영상에 빨간색으로 표시. 선 두께는 2
            rectangle(image1, boundingBox, Scalar(0, 0, 255), 2);

            // 바운딩 박스의 왼쪽 변 중간점
            Point middleLeft(boundingBox.x, boundingBox.y + boundingBox.height / 2);

            // 화살표의 시작점 (왼쪽 변에서 왼쪽으로 50픽셀)
            Point arrowStart(middleLeft.x - 50, middleLeft.y);

            // 화살표 그리기
            arrowedLine(image1, arrowStart, middleLeft, Scalar(0, 255, 255), 2); // 노란색
화살표
        }
         imshow("빨간 표지판 검출", image1);
         if(waitKey(1000/30) == 'q')
             break;
    }
    return 0;
}
```

c - 이미지

```c
#include <stdio.h>
#include <stdlib.h>

// 구조체 정의: 이미지 내의 특정 영역을 나타냄
typedef struct {
    int x1, y1, x2, y2;
} Area;

int main(int argc, char *argv[]) {
    char M, N; // 이미지 파일 형식을 위한 문자 변수
    int XX, YY, MAX, x, y; // 이미지의 크기, 최대값 및 루프 카운터
    FILE *file1, *file2; // 파일 포인터
    int image1[1000][1000], temp[1500][1500]; // 이미지 배열 및 임시 배열

    // 좌표를 읽어올 파일 열기
    FILE *file = fopen("coordinates.txt", "r");
    Area areas[100]; // 확대할 영역을 저장할 배열
    int num_areas = 0; // 영역의 수
    int x1, y1, x2, y2; // 좌표 변수

    // 첫 번째 파일에서 이미지 데이터 읽기
    file1 = fopen(argv[1], "r");
    fscanf(file1, "%c%c", &M, &N); // 파일 형식 읽기
    fscanf(file1, "%d%d", &XX, &YY); // 이미지 크기 읽기
    fscanf(file1, "%d", &MAX); // 최대 픽셀 값 읽기

    // 이미지 배열 채우기
    for (y = 0; y < YY; y++)
        for (x = 0; x < XX; x++)
            fscanf(file1, "%d", &image1[y][x]);

    // 좌표 파일에서 영역 정보 읽기
    while (fscanf(file, "%d %d %d %d", &x1, &y1, &x2, &y2) == 4) {
        areas[num_areas].x1 = x1;
        areas[num_areas].y1 = y1;
        areas[num_areas].x2 = x2;
        areas[num_areas].y2 = y2;
        num_areas++;

        if (num_areas >= 100) {
            break;
        }
    }
```

```
    fclose(file);

    // 각 지정된 영역에 대한 처리
    for (int i = 0; i < num_areas; i++) {
        int width = areas[i].x2 - areas[i].x1;
        int height = areas[i].y2 - areas[i].y1;

        // 영역 확대 처리( 1.5배 )
        for (y = 0; y < height * 1.5; y++) {
            for (x = 0; x < width * 1.5; x++) {
                int origX = areas[i].x1 + x / 1.5;
                int origY = areas[i].y1 + y / 1.5;
                temp[y][x] = image1[origY][origX];
            }
        }

        // 임시 배열에서 원본 이미지로 확대된 영역 복사
        for (y = 0; y < height * 1.5; y++) {
            for (x = 0; x < width * 1.5; x++) {
                image1[areas[i].y1 + y][areas[i].x1 + x] = temp[y][x];
            }
        }
    }

    // 결과 파일에 이미지 데이터 쓰기
    file2 = fopen(argv[2], "w");
    fprintf(file2, "%c%c\n", M, N);
    fprintf(file2, "%d %d\n", XX, YY);
    fprintf(file2, "%d\n", MAX);

    // 표지판이 확대된 이미지로 출력
    for (y = 0; y < YY; y++) {
        for (x = 0; x < XX; x++) {
            fprintf(file2, "%4d", image1[y][x]);
        }
        fprintf(file2, "\n");
    }

    return 0;
}
```

이미지1 명령실행화면

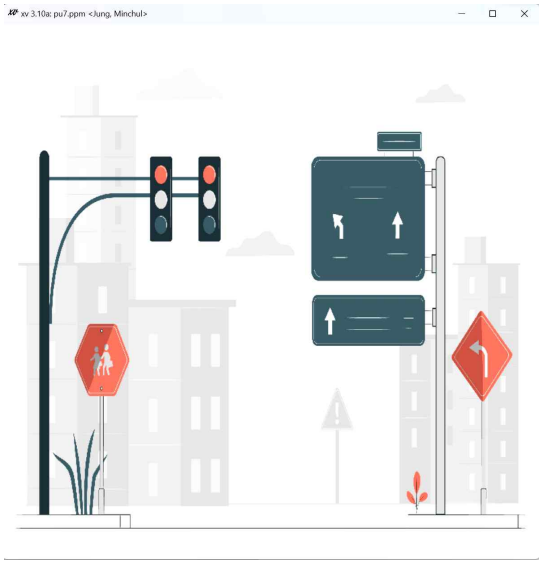
```
[kimjunyoung@raspberry4 ~]g++ pan1.cpp `pkg-config opencv4 --libs --cflags`
[kimjunyoung@raspberry4 ~]a.out pu3.ppm
```

이미지2 명령실행화면

```
[kimjunyoung@raspberry4 ~]g++ pan1.cpp `pkg-config opencv4 --libs --cflags`
[kimjunyoung@raspberry4 ~]a.out pu7.ppm
```

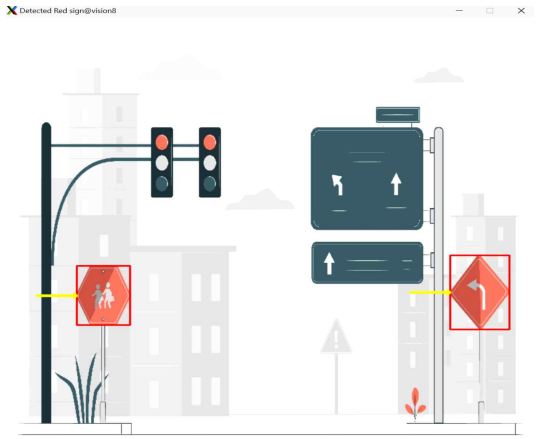
이미지1 입력영상

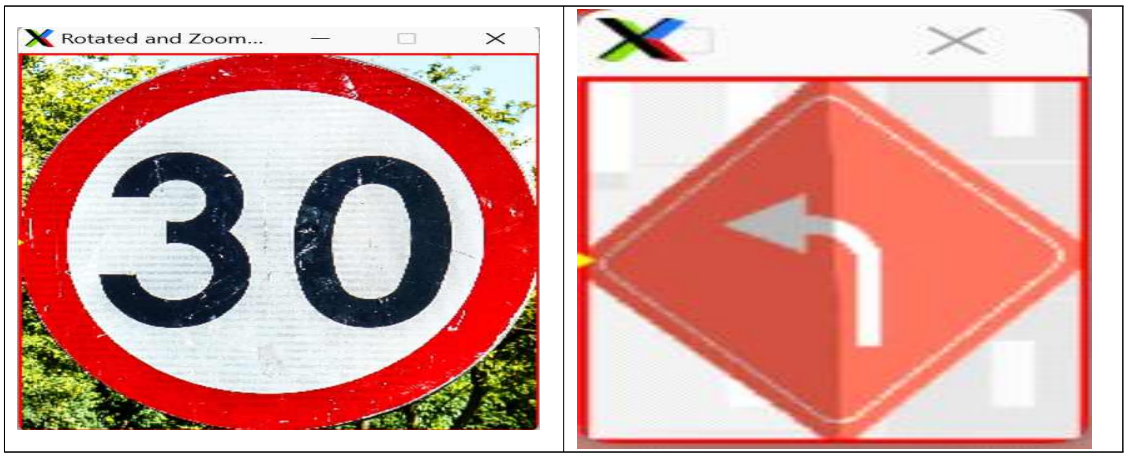
이미지2 입력영상



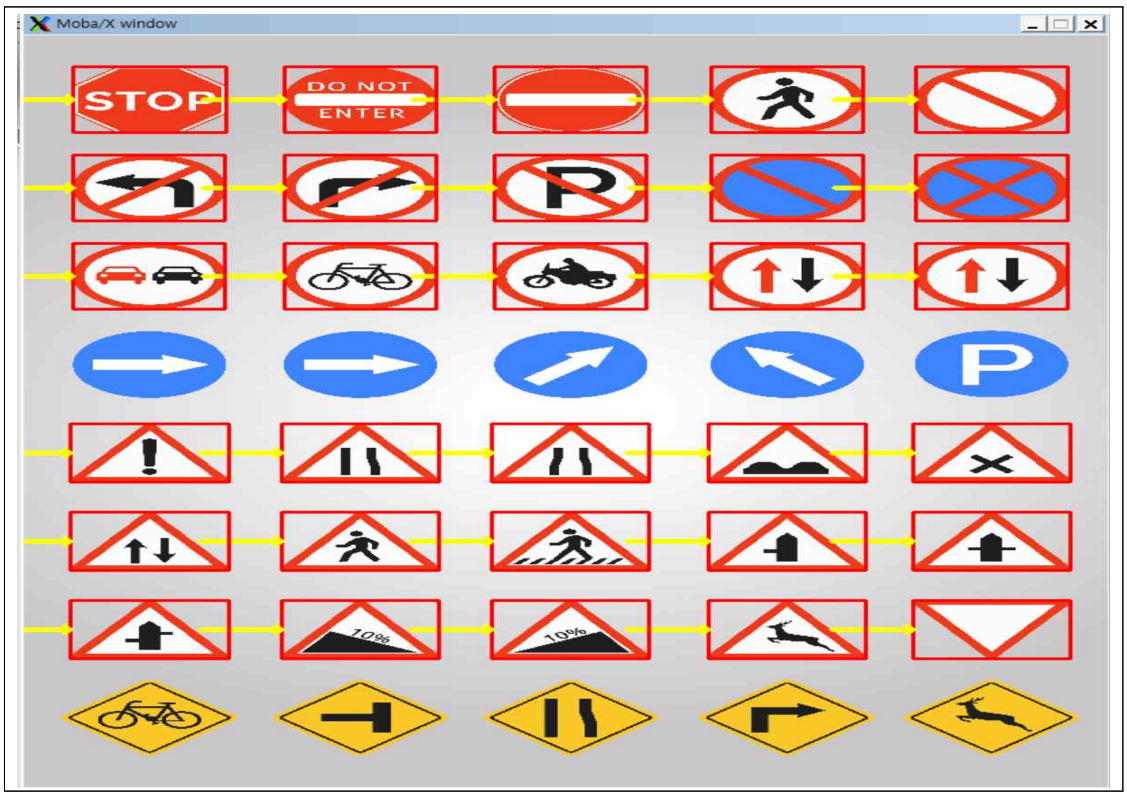
이미지1 출력영상

이미지2 출력영상





표지판만 있는 경우의 영상



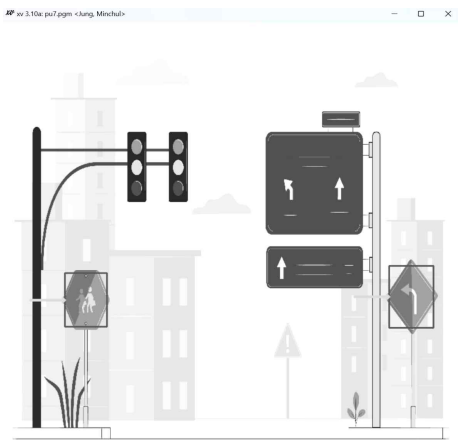
동영상1 명령실행화면

```
[kimjunyoung@raspberry4 ~]g++ pan2.cpp `pkg-config opencv4 --libs --cflags`
[kimjunyoung@raspberry4 ~]a.out pu7.mp4
```

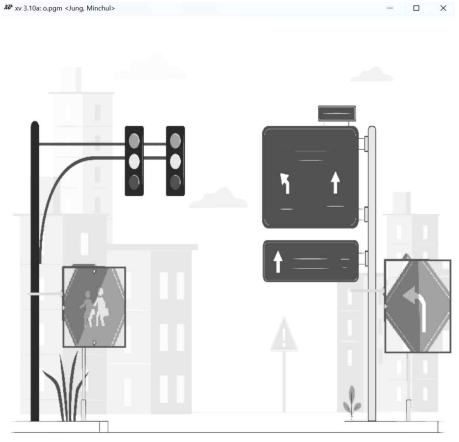
c에서의 표지판 확대

```
[kimjunyoung@raspberry4 ~]gcc largehw.c
[kimjunyoung@raspberry4 ~]a.out pu7.pgm o.pgm
[kimjunyoung@raspberry4 ~]xv o.pgm
```

입력 영상



출력 영상



설명

빨간색 표지판을 인식해야 하므로 처음에는 RGB로 빨간색을 찾으려 했는데 생각대로 검출이 제대로 이루어지지 않아서 has로 색상 공간으로 바꿔주었다. 검출하기 전, 먼저 이진 침식을 하여 salt 부분을 제거하고 pepper 부분을 제거하기 위해서 이진 팽창을 하였다. 이진 침식과 팽창은 객체의 형태를 변경하고 잡신호를 제거하는 역할을 한다. 이를 통해 객체 주위를 깨끗하게 하여 표지판 인식이 가능했다. 그 후 빨간 부분의 윤곽선을 검출하여 그 주위에 좌표를 찾고 주위에 직사각형으로 표시했다. 추가로 강조하기 위해 화살표 모양을 넣어주었다. c로 넘어와서 그 좌표값이 들어있는 .txt파일을 불러와 객체 좌표에 부분을 1.5배 표지판을 확대하여 원래 영상에 넣어주었다. 이를 통해 객체를 검출하여 표시하고 확대까지 진행하여 프로젝트를 완성했다.

의의와 한계

프로젝트를 통해 색깔로 검출할 수 있고 윤곽선을 통해 객체 검출이 가능함을 알 수 있었고, 영상 확대를 통해 특정 객체를 강조할 수 있다는 것을 알았다. opencv에 서의 픽셀값 위치를 c에서 어떻게 사용해야 할지 시간이 오래 걸렸는데, ofstream coordFile(" coordinates.txt ")라는 명령어를 통해 값 전달이 가능했다. 이 프로젝트가 다른 영상에서 빨간색 옷이라든지 빨간 차도 인식이 되므로 약간의 오류가 있다. 색깔과 윤곽선으로만 인식을 하다 보니 모든 것을 고려하기는 힘들지만, 표지판의 특성, 글자 등의 다양한 데이터가 있다면 자율주행이나 내비게이션 빨간 표지판 강조 등등을 사전에 미리 인지하여 사고 예방하는 데 쓰일 수 있지 않을까 싶다.