차량 검출 프로그램

2023254005 이찬희

2023.11.12

문제 해결 방법

-이미지를 스무싱, 이진화, 모폴로지, 라벨링, 블롭 등을 사용하여 건물, 나무, 차량 별도로 구분하고 차량을 검출하는 것을 하고 싶었으나, 해당 객체들을 분류하는 것을 하지 못하였고, 여러 방법을 해봐도 실력이 모자라 차량을 검출하는 방법에 도달하지 못했습니다.

-제 실력이 모자라 배운 것으로는 어렵다 판단하여,
OpenCV에서 제공하는 Haar Cascade 분류기를 사용하였습니다.

-Haar Cascade 분류기를 사용하여도 바로 차량이 검출되는 것은 아니였기에 차량이 검출이 되도록 이미지 전처리 하는 과정이 필요했습니다.

-최종적으로는 이미지를 전처리하여 Haar Cascade 분류기를 통해 차량을 검출하는 과정으로 코드를 진행하였습니다.

사용된 코드 요약

1. 이미지 불러오기
2. 이미지의 그레이스케일 변환
3. 이미지 평활화 (Equalize Histogram)
4. 모폴로지 연산 수행 (Morphology Operations)
   4.1. 모폴로지 연산 수행 (MORPH_OPEN)
   4.2. 모폴로지 연산 수행 (MORPH_CLOSE)
5. Haar Cascade 파일 로드
6. 차량 검출
   6.1. 차량 검출 수행 (detectMultiScale)
      6.1.1. 이미지 스케일링(factor) 설정
      6.1.2. 미니멈 이웃(neighbor) 간의 최소 거리 설정
      6.1.3. 검출된 객체의 최소 크기 설정
7. 검출된 차량 처리
   7.1. 검출된 차량 사각형 좌표를 벡터에 저장
8. 결과 표시 및 저장
   8.1. 이미지에 검출된 차량 표시 (rectangle)
   8.2. 검출된 차량의 좌표와 크기를 파일에 저장

사용된 코드

```cpp
#include <opencv2/opencv.hpp>
#include <iostream>
#include <fstream>

using namespace cv;
using namespace std;

// 비교 함수 정의
bool compareRectSize(const Rect& a, const Rect& b) {
    return (a.width * a.height) > (b.width * b.height);
}

int main() {
    // 차량 검출을 위한 Haar Cascade 파일 경로
    String cascadePath = cv::samples::findFile("../Project/haarcascade_car.xml");

    // 이미지 열기
    String imagePath = cv::samples::findFile("../image/cars/test6.png");

    // 이미지 불러오기
    Mat image = imread(imagePath);

    if (image.empty()) {
        cerr << "이미지를 불러올 수 없습니다." << endl;
        return -1;
    }

    // Haar Cascade 분류기 초기화
    CascadeClassifier carCascade;
    if (!carCascade.load(cascadePath)) {
        cerr << "Haar Cascade 파일을 불러올 수 없습니다." << endl;
        return -1;
    }

    // 그레이스케일로 변환
    Mat gray;
    cvtColor(image, gray, COLOR_BGR2GRAY);

    // 이미지 평활화
    equalizeHist(gray, gray);

    // 모폴로지 연산
    Mat kernel = getStructuringElement(MORPH_ELLIPSE, Size(5, 5));
    Mat kernel2 = getStructuringElement(MORPH_ELLIPSE, Size(3, 3));
    Mat opening, closing;
    morphologyEx(gray, opening, MORPH_OPEN, kernel);
    morphologyEx(opening, closing, MORPH_CLOSE, kernel2);

    // 차량 검출
    vector<Rect> cars;
    carCascade.detectMultiScale(closing, cars, 1.1, 1, 0, Size(100, 100));

    // 이미지 파일 이름에서 확장자 제거
    size_t lastDotPos = imagePath.find_last_of(".");
    string imageNameWithoutExtension = imagePath.substr(0, lastDotPos);

    // 결과를 출력할 텍스트 파일 생성
    string outputFilePath = imageNameWithoutExtension + ".txt";
    ofstream outputFile(outputFilePath);

    // 크기순으로 정렬
    sort(cars.begin(), cars.end(), compareRectSize);

    // 최대 5개까지만 출력
    size_t numCarsToPrint = min(cars.size(), static_cast<size_t>(5));

    // 검출된 차량 표시 및 좌표 기록
    for (size_t i = 0; i < numCarsToPrint; ++i) {
        Rect car = cars[i];
        rectangle(image, car, Scalar(0, 255, 0), 1);

        // 좌표 및 크기를 텍스트 파일에 탭으로 구분하여 기록
        outputFile << i + 1 << '₩t' << car.x << '₩t' << car.y << '₩t' << car.width << '₩t' << car.height << endl;
    }

    // 파일 닫기
    outputFile.close();

    // 결과 출력
    imshow("Detected Cars", image);
    waitKey(0);

    return 0;
}
```

Haar Cascade 분류기

```
int main() {
    // 차량 검출을 위한 Haar Cascade 파일 경로
    String cascadePath = cv::samples::findFile("../Project/haarcascade_car.xml"); // 여기에 haarcascade_cars.xml 파일의 경로를 넣어주세요.

    // 차량 검출
    vector<Rect> cars;
    carCascade.detectMultiScale(closing, cars, 1.1, 1, 0, Size(100, 100));
```

-Haar Cascade 분류기는 객체 검출을 위한 머신 러닝 분류기입니다. 주로 얼굴, 눈, 차량 등과 같은 특정 객체를 빠르게 검출하기 위해 사용됩니다.

-OpenCV 라이브러리에서 다양하게 미리 학습된 Haar Cascade 파일이 제공됩니다. 그 중 차량 객체 검출 라이브러리인 haarcascade_car.xml 사용하였습니다.

-carCascade라는 CascadeClassifier 객체를 사용하여 이미지에서 차량을 검출합니다.

-detectMultiScale 함수는 이미지에서 다양한 크기의 객체를 검출할 수 있는 Multi-Scale 검출 방법을 사용합니다.

-매개변수는 다음과 같습니다.

(image, 검출 좌표 입력될 vector, 이미지스케일링(factor), 이웃간 최소거리, 0(일반적인 경우), 검출된 객체의 최소 크기)

이미지 평활화

```
// 이미지 평활화
equalizeHist(gray, gray);
```

-이미지 평활화는 이미지의 히스토그램을 조절하여 전체적인 대비를 향상시키는 이미지 처리방법으로, 평활화를 통해 이미지의 픽셀값 분포를 조절하여 어두운 부분과 밝은 부분의 차이를 줄이고, 뚜렷한 특징을 도드라지게 만드는 특징이 있습니다.

Morphology

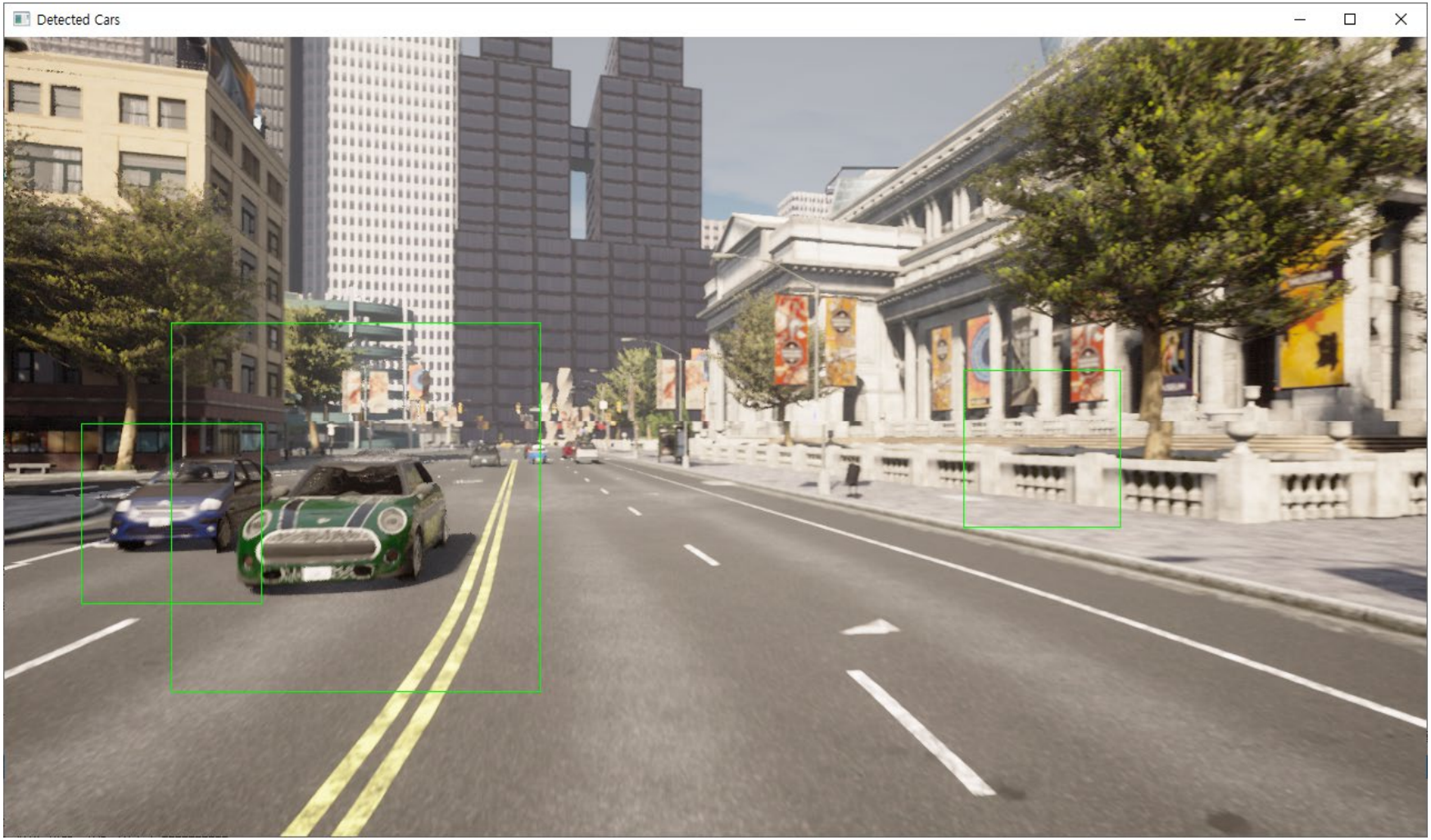
```
// 모폴로지 연산
Mat kernel = getStructuringElement(MORPH_ELLIPSE, Size(5, 5));
Mat kernel2 = getStructuringElement(MORPH_ELLIPSE, Size(3, 3));
Mat opening, closing;
morphologyEx(gray, opening, MORPH_OPEN, kernel);
morphologyEx(opening, closing, MORPH_CLOSE, kernel2);
```

- Morphology 열림연산 : 주변보다 밝은 노이즈를 제거하는데 효과적이며, 맞닿아 있는 것처럼 보이는 독립된 개체를 분리하거나 돌출된 모양을 제거하는데 효과적입니다.

- Morphology 닫힘연산 : 주변보다 어두운 노이즈를 제거하는데 효과적이며 끊어져 보이는 개체를 연결하거나 구멍을 메우는 데 효과적입니다.

- 열림 연산만을 사용했을 때보다, 열림 연산 후 닫힘 연산을 진행하였을 때, Haar Cascade 분류기를 통한 차량 검출이 더 잘되어 이렇게 코드를 구성하게 되었습니다.

결과(test1.png)

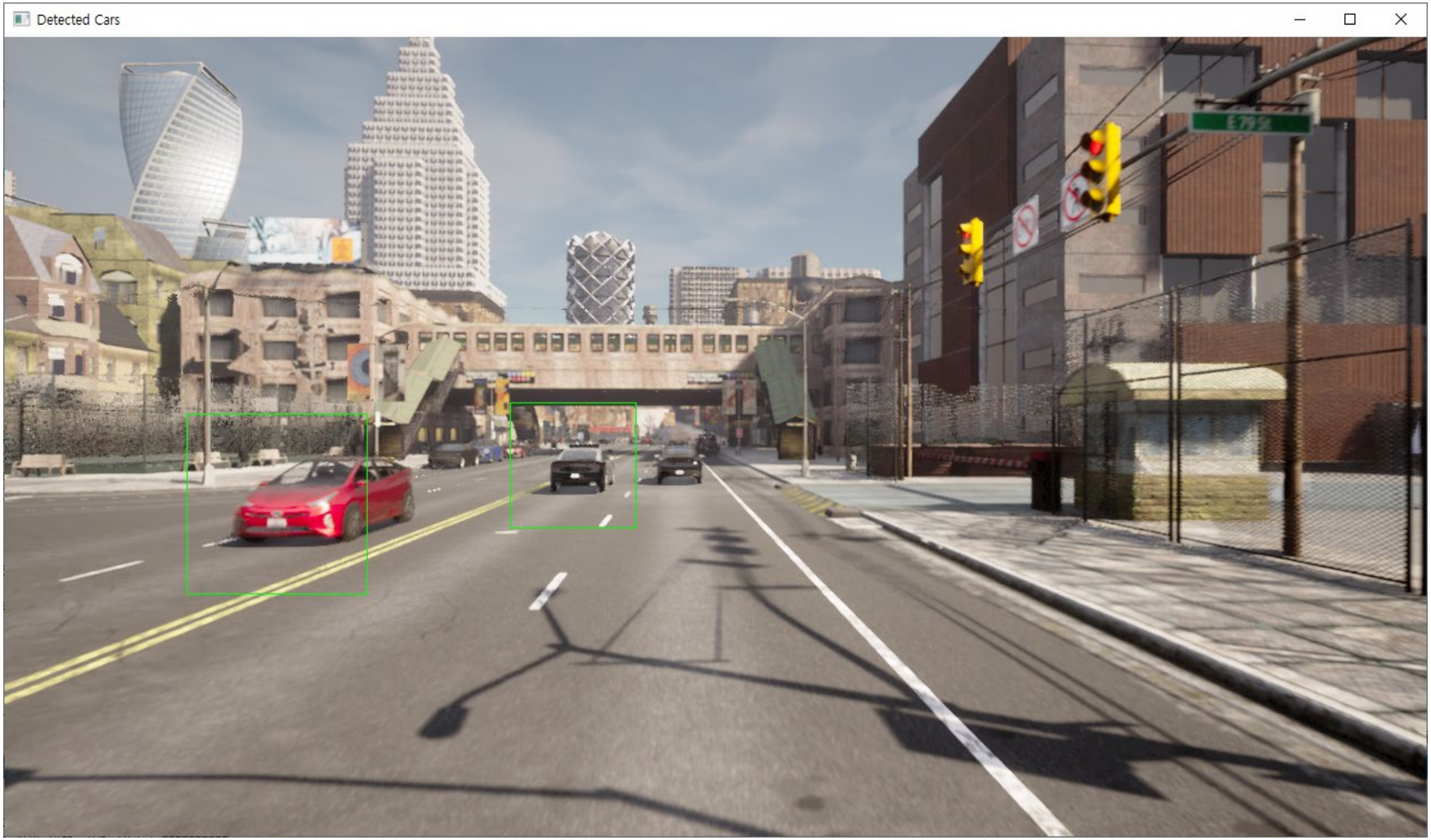


결과(test2.png)

결과(test3.png)

결과(test4.png)

결과(test5.png)



결과(test6.png)

결론

-해당 코드로 차량을 검출하기는 하나 test2, test4 이미지에서 차량이 아닌 것을 차량이라 검출하기도 하며, 또한 차량을 검출하지 못하는 상황도 생겼습니다.

-구성한 코드에서 값 튜닝만 가지고는 더 나아지기는 어려워보이며, 이미지 전처리 과정이 더 필요해보입니다.
다만 haarCascade 분류기가 어떠한 방식으로 차량을 인식하는지에 대한 것을 알 수 없기에, 여러 방법에 대한 시도가 필요할 것으로 생각됩니다.