KOREA POLYTECHNIC UNIVERSITY
한국산업기술대학교

디지털 영상의 엣지 및 전경/배경 검출을 주제로 한

영상처리 프로젝트

영상처리 1팀

소프트웨어전공 2016156001 곽배준

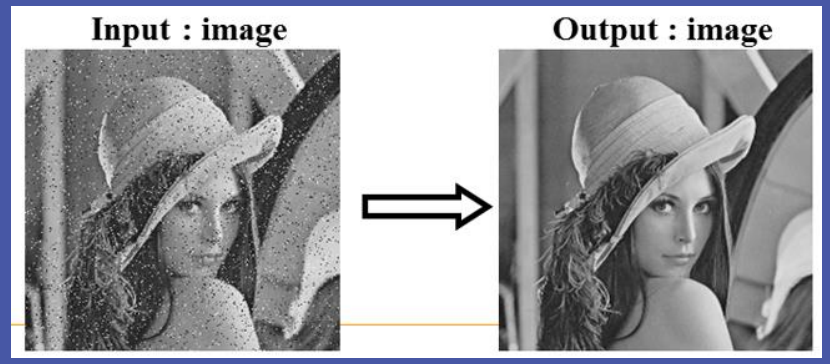
소프트웨어전공 2016156026 이형석

소프트웨어전공 2016156032 전유미

2020-2 디지털 영상처리 – 안세종 교수님

INTRO

Q . 영상처리란?



영상을 전자적으로 얻은 후,

컴퓨터로 가공/추출/변형/압축 등 원하는 기능을 넣어 처리하는 것.

2차원 함수 f(x,y)로 정의될 수 있으며 그 점에서의 밝기(색상정보)를 포함하는 것을 디지털 영상이라고 함.

신호처리가 1차원 데이터를 다루면서 아날로그를 디지털로 변환했다면,

영상처리는 2차원 데이터를 다루면서 아날로그를 디지털로 변환하는 것!

자율주행 자동차, 공장 자동화, 생체인식 시스템(특징점 활용), 주차장 자동인식 등

이 밖에도 영상조작, 물체를 측정하거나 감시하여 영상분석, 물체를 식별하는 영상인식,

크기를 축소하는 영상압축 등의 기술이 있다.

CONTENTS

디지털 영상의 엣지 및 전경/배경 검출을 주제로 한

01. 과제 수행 개요

01. 과제 수행 개요

“ 영상처리 ”

개발언어 C++과 라이브러리 OpenCV활용

UI로 통합하여 최종 결과물을 내는 것을 목표로 했으나, 일정 조율 어려움으로 영상처리 기능 구현

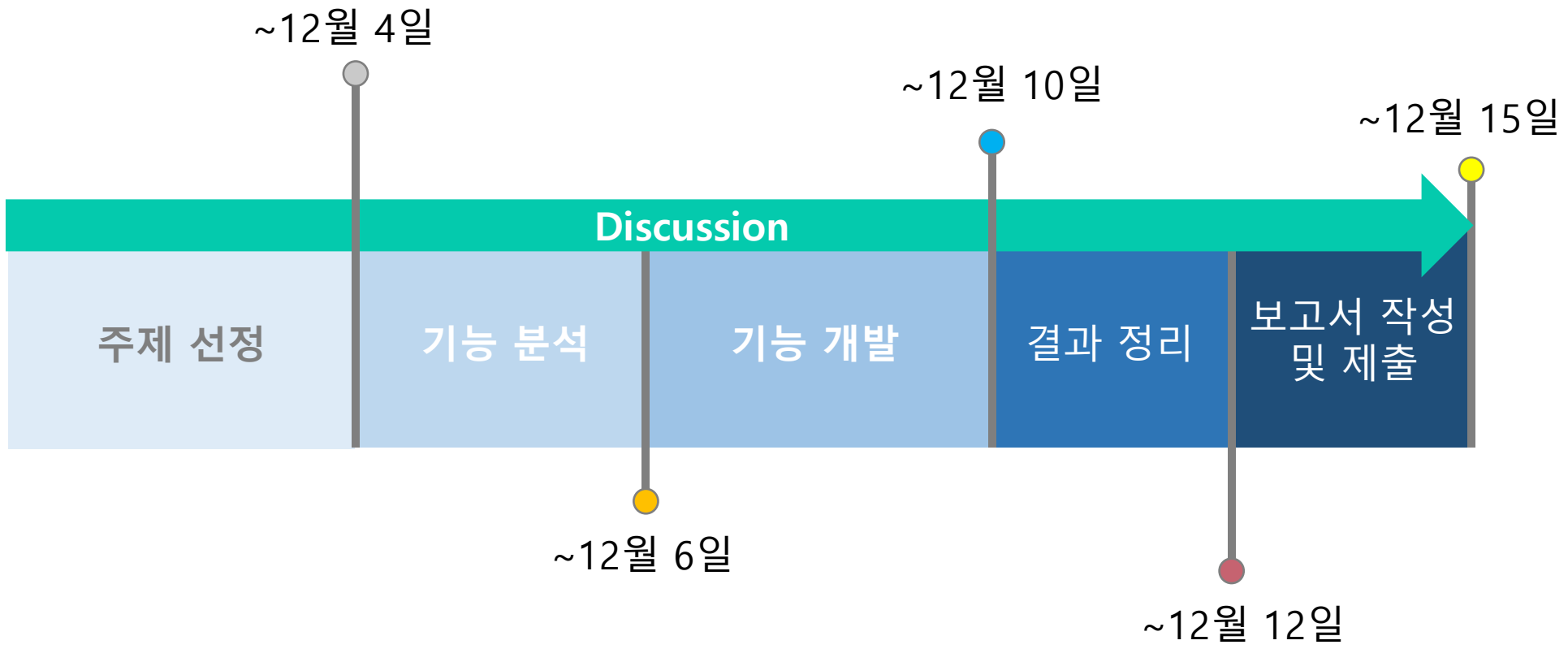
주제	**디지털 영상의 엣지 및 전경/배경 검출**
선택 이유 및 목표	강의 중 배운 <영상 분할>에 흥미를 느끼고 자료조사를 통해 주제를 선정하는 과정에서 학습 내용을 많이 다루고 있어 선택하게 되었다. 직접 기능 구현을 통해 실용적으로 활용되는 분야를 알아보고, 복습하는 시간을 갖는다.

Visual Studio

OpenCV

02. 프로젝트 수행 일정

“ Project Schedule ”



03. 팀원 역할 분담

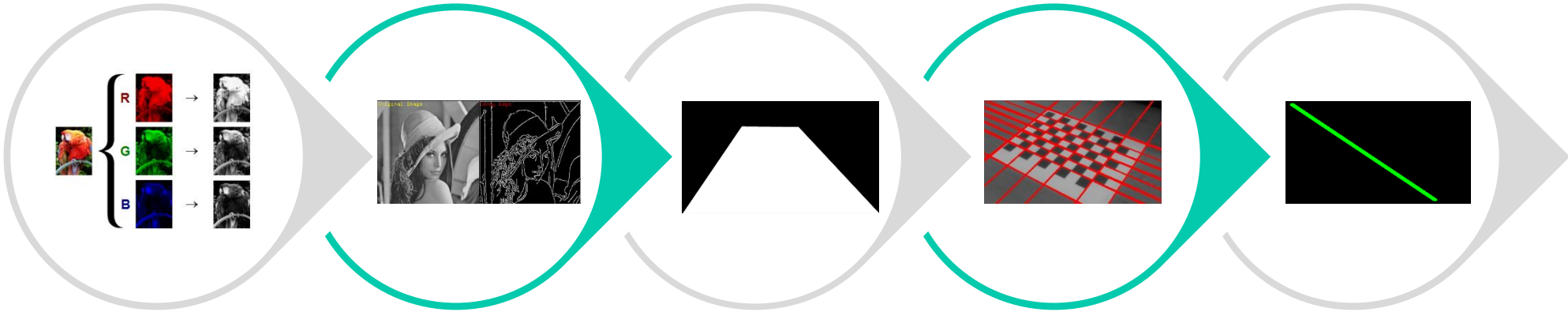
“ Team work ”

	곽배준	이형석	전유미
1차	자료 조사 및 주제 선정 & 기능 분석		
2차	보고서 설계	자율주행을 위한 직선 검출	실시간 카메라 영상 전경 검출
3차	결과 정리 및 보고서 작성		

디지털 영상의 엣지 검출을 주제로 한

02. 자율주행 자동차 영상처리

자율주행 기능 – 직선 차선 검출



1. 회색조 변환

복잡한 데이터 형식의 컬러 영상을 단순한 형태의 회색조 영상으로 변환.

2. 캐니 에지 검출

회색조 영상에서 검은색 경계선을 추출.

3. 검출 영상 마스킹

캐니 에지로 검출된 영상에서 배경은 가려지고 바닥만 보이도록 마스크 씌울 도형으로 영상에 마스킹

4. 허프 변환

윤곽선 중 직선 요소들을 검출하여 그중 가장 적합한 직선을 선택

5. 검출된 직선 시각화

허프변환으로 검출된 직선을 알아보기 쉽도록 line 함수를 통해 빨간색 선으로 시각화

1. 회색조 변환(cvtColor 함수)

회색조 변환 과정

영상은 밝기와 색상이 다른 일정한 수의 화소들로 구성되는데 카메라를 통해 들어온 컬러 영상은 다양한 색과 밝기, 위치 등을 표현하기 위해 복잡한 데이터를 가지고 있음.

따라서 단순한 형태의 회색조 영상으로 변환해야 함.

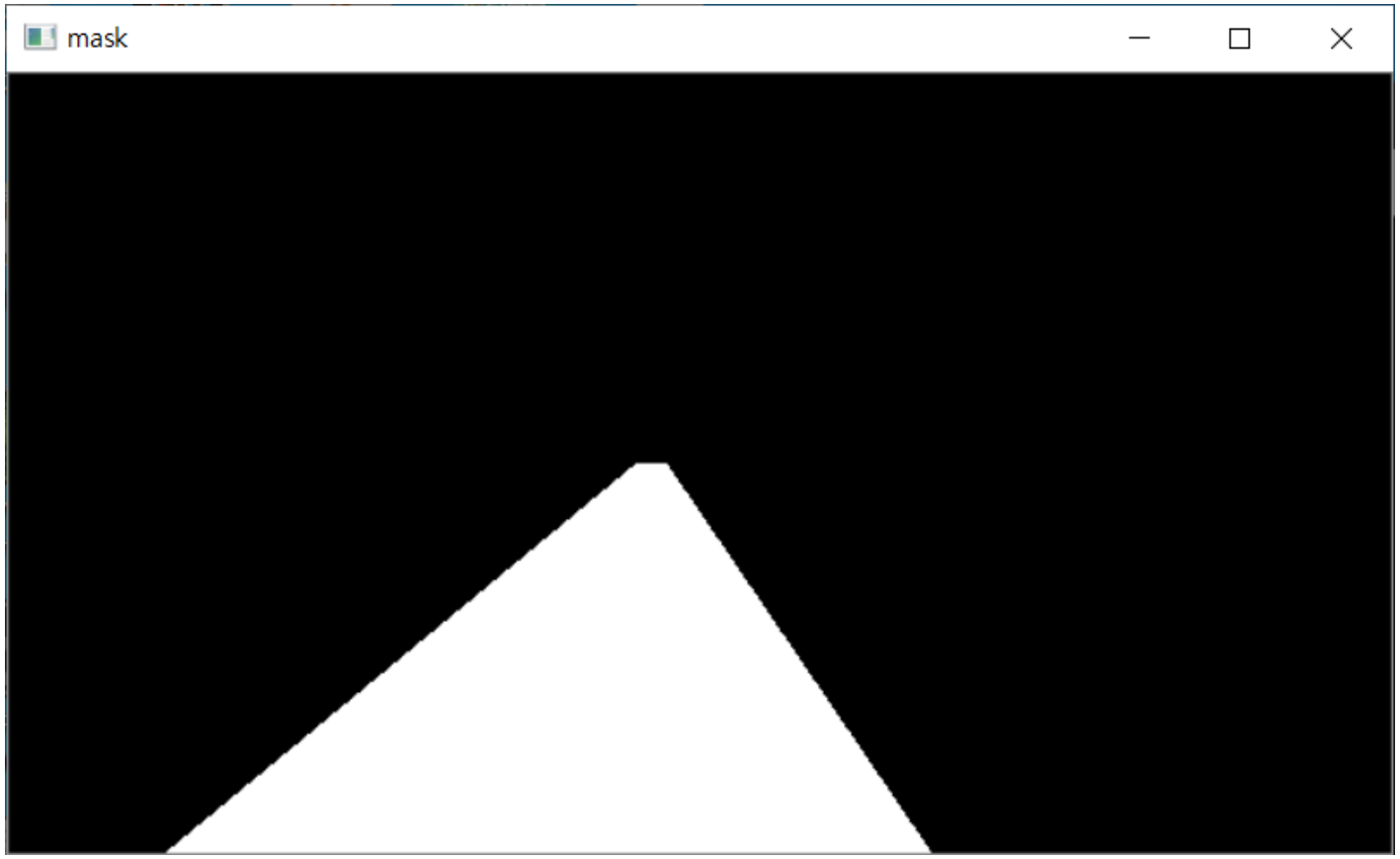
2. 캐니 에지 검출(Canny 함수)

캐니 엣지 연산자 과정

1. 잡음 억제
2. 그라디언트 계산
3. 비최대 억제
4. 히스테리시스

영상에서 흰색과 검은색 경계선을 추출하는데 서로 붙어 있는 흰색과 검은색 픽셀들이 연결되어 있는지 검사.

3. 검출 영상 마스킹(mask 매트릭스)

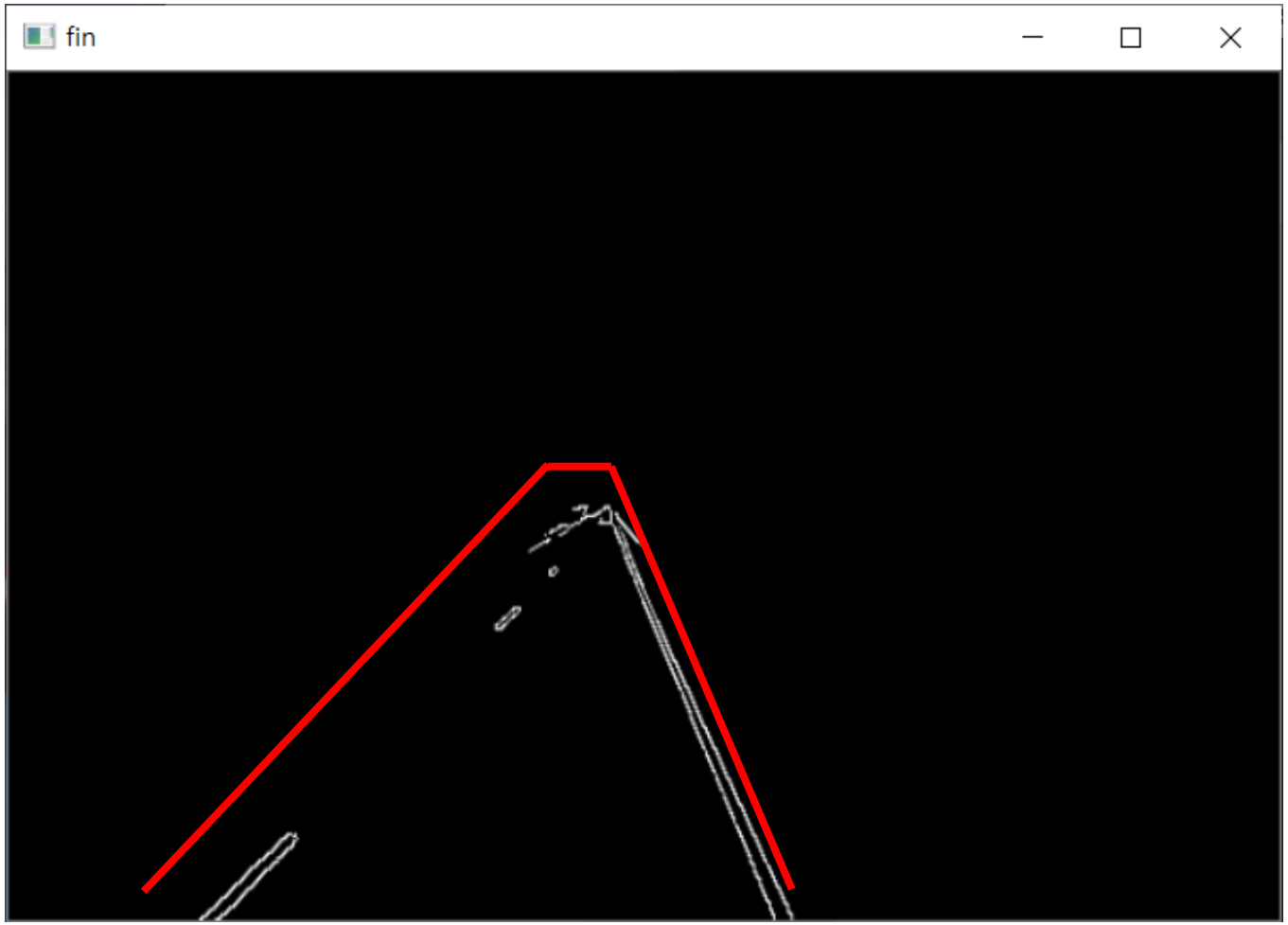


검출 영상 마스킹 과정

마스킹할 매트릭스를 생성하는 데 vector 클래스에 마스킹 할 부분의 좌표를 삽입하여 fillConvexPoly() 함수로 해당 부분을 제외한 나머지 배경을 검은색으로 만듬.

이후 bitwise_and 연산을 통해 Canny 연산자로 검출한 영상과 AND 연산을 수행

4. 허프 변환(HoughLinesP 함수)



직선 검출 허프 변환 과정

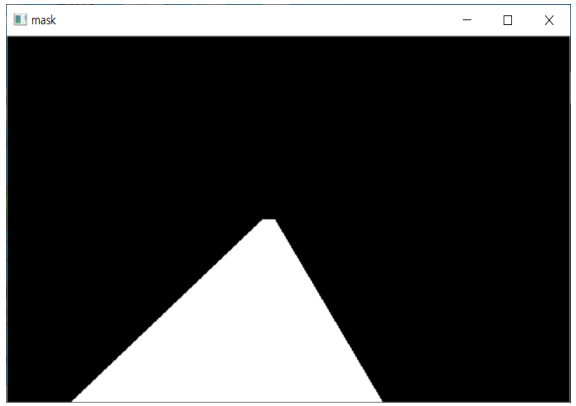
Canny 엣지 연산과 마스크 매트릭스와 AND 연산 이후, 해당 영상에 허프 변환하여 직선을 검출해야 함.

윤곽선 중 직선 요소들을 검출하여 그 중 적합한 직선을 차선으로 선택함.

4. 허프 변환(HoughLinesP 함수)

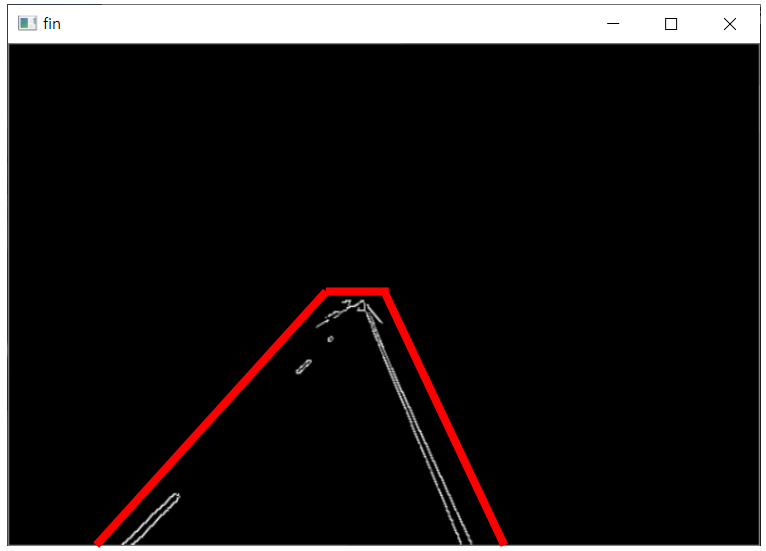
캐니 에지 결과물

AND



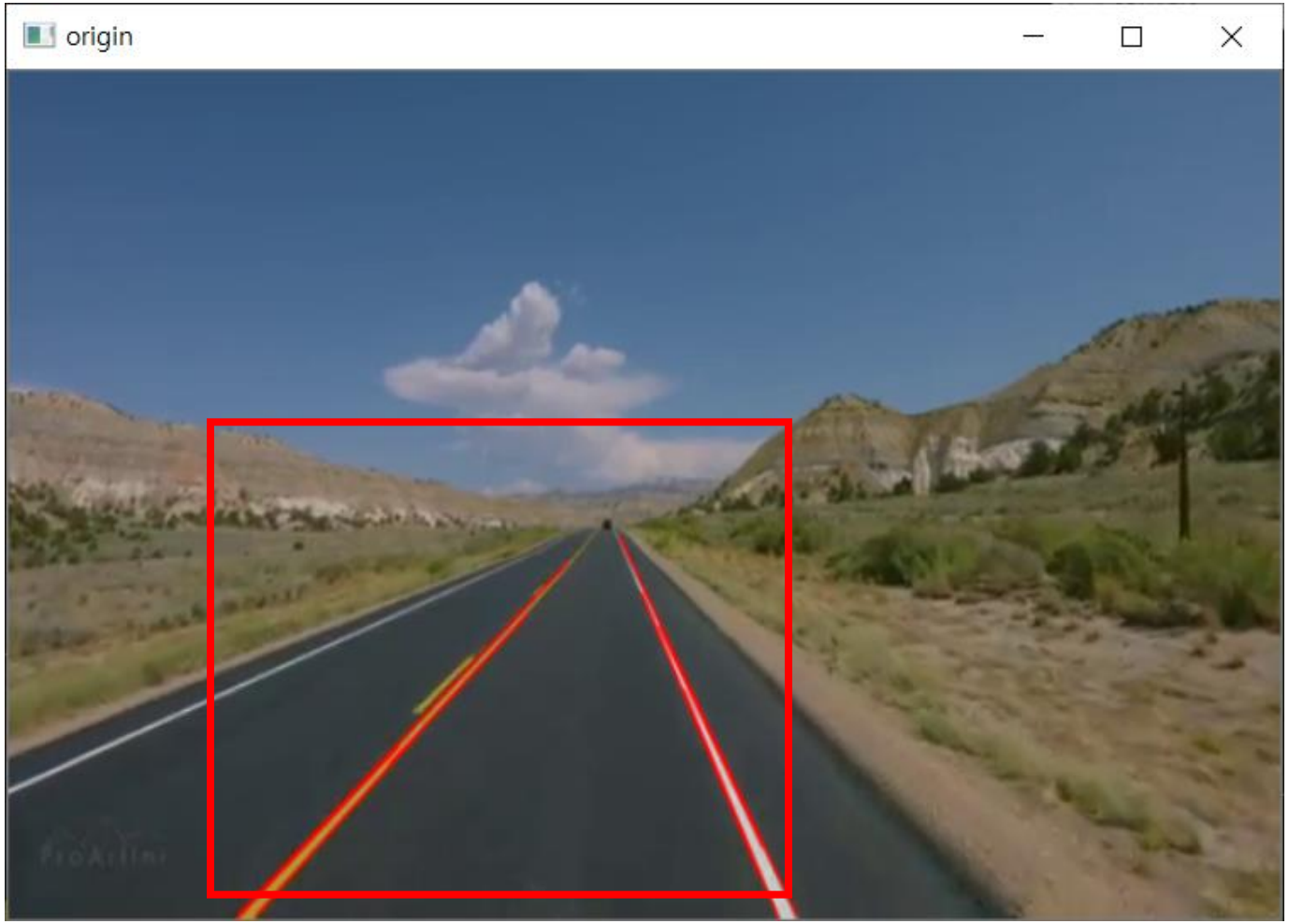
마스크

=



AND 연산 결과

5. 검출된 직선 시각화(Line 함수)



직선 시각화 과정

Canny 엣지 연산과 마스킹 이후 영상에 대해 허프 변환 후 직선을 검출한다면, 빨간색 블록 부분만 검출되어 배경은 검출되지 않고 차선만 검출될 수 있음.

코드 흐름도

```cpp
int main() {
    VideoCapture cap("D:/3-2학기/영상처리-안세종/프로젝트 영상/anew.mp4");
    VideoError(cap);
```

```cpp
cap >> frame;

cvtColor(frame, gray, COLOR_BGR2GRAY);
```

```cpp
// 캐니 에지 연산자로
//Canny(frame, dst, 100, 200);
Canny(gray, dst, 100, 200);
```

```cpp
// 마스크 처리
// 캐니 에지로 검출된 dst의 사이즈와 타입과 같은 매트릭스 선언
Mat mask = Mat::zeros(dst.size(), dst.type());

// 포인트 벡터에 선을 검출할 마스크 좌표를 삽입
vector<Point> points;
points.push_back(Point(dst.cols / 2.2, dst.rows / 2.0));
points.push_back(Point((dst.cols / 2.1), dst.rows / 2.0));
points.push_back(Point(dst.cols/1.5, dst.rows));
points.push_back(Point(dst.cols / 8.8, dst.rows));

// 마스크 포인트 부분을 제외한 모든 부분을 검은색으로 만듬
fillConvexPoly(mask, points, Scalar(255, 255, 255));
```

```cpp
// 캐니 에지 AND 마스크 영상에 허프 변환으로 직선 검출
vector<Vec4i> lines;
HoughLinesP(fin, lines, 1, CV_PI / 180, 50, 55, 5);

// 검출된 직선을 line 함수를 통해 그려준다.
for (size_t i = 0; i < lines.size(); i++) {
    Vec4i l = lines[i];
    line(frame, Point(l[0], l[1]), Point(l[2], l[3]), Scalar(0, 0, 255), 1, CV_AA);
}
```

```cpp
// 캐니 에지로 검출된 매트릭스와 마스크 AND 연산
// 결과는 fin 매트릭스에 저장
bitwise_and(dst, mask, fin);
imshow("mask", mask);
```

디지털 영상의 전경/배경 검출을 주제로 한

03. 실시간 카메라 영상 분할

영상 분할(Image Segmentation)이란?

영상 안의 화소를 의미 있는 영역(segment)으로 분할하는 것

분할의 목적은 영상의 표현을 의미 있고 해석하기 쉬운 것으로 단순화하거나 변환하는 것.
영상 분할은 특히 영상에서 물체와 경계(선, 곡선)를 찾는데 사용되곤 함.

에지(edge)를 사용하여 물체의 윤곽선을 추적하거나, 클러스터링을 이용해
영역 성장법을 사용하거나 히스토그램, 무늬, 딥러닝(deep learning)을 이용하여 분할할 수 있음.

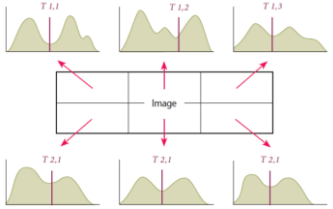
의료 영상, 위성 영상, 생체 인식, 자동 교통 시스템 등 넓은 분야에서 실용적으로 응용됨.

실시간 카메라 영상 분할 – 전경 검출

OPEN

1. 이진화

전경/배경 분할을 위해선
이진화 과정이 필수.
실시간으로 카메라로부터
받은 컬러 영상을 단순한
형태의 회색조 영상으로 변환

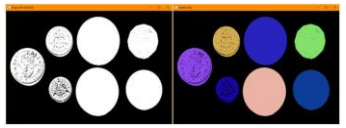


2. 적응적 이진화

입력 이미지에 따라 임계값이
스스로 다른 값을 할당할 수
있도록, 영상을 분할하여 서로
다른 임계값 선택하여 이진화

3. 전경/배경 분할(1)

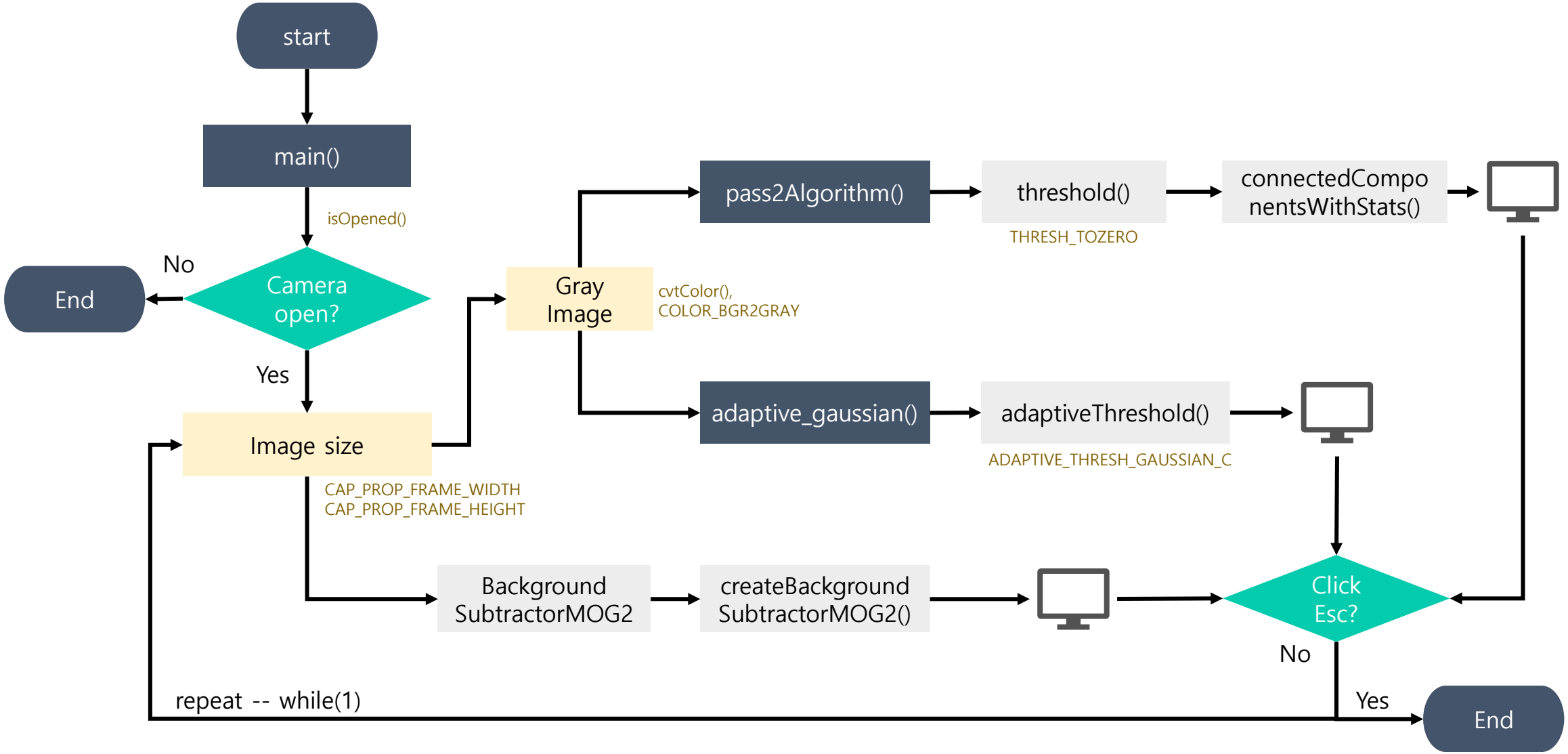
통계적 배경 이미지 제거와
픽셀 단위 베이지안 분할을 결합한,
배경 제거를 편리하게 해주는
BackgroundSubtractorMOG2을
사용하여 전경 검출



4. 전경/배경 분할(2)

4-연결을 이용한 2-패스 연결
성분 레이블링 통해 전경 검출.
움직이는 객체에 RBG 색상을
입혀 새롭게 표현

FLOW CHART


start
main()
isOpened()
Camera open?
No
End
Yes
Image size
CAP_PROP_FRAME_WIDTH
CAP_PROP_FRAME_HEIGHT
Gray Image
cvtColor(), COLOR_BGR2GRAY
pass2Algorithm()
threshold()
THRESH_TOZERO
connectedComponentsWithStats()
adaptive_gaussian()
adaptiveThreshold()
ADAPTIVE_THRESH_GAUSSIAN_C
Background SubtractorMOG2
createBackground SubtractorMOG2()
Click Esc?
No
Yes
End
repeat -- while(1)

0. 카메라 개방

OPEN

```
int main()
{
    VideoCapture capture(0);

    if (!capture.isOpened()) {
        cout << "Cannot open capture : 카메라 개방 오류 !!" << endl;
        return 0;
    }

    Size size = Size((int)capture.get(CAP_PROP_FRAME_WIDTH),
        (int)capture.get(CAP_PROP_FRAME_HEIGHT));
    cout << "Size = " << size << endl;

    waitKey(100);
```

- 라이브러리의 VideoCapture 클래스를 사용하여 영상 파일 입력 및 카메라로부터 입력을 받음
- 객체를 생성하고, 영상 파일 또는 카메라를 개방하여 비디오 프레임 획득
- 카메라 개방 오류 체크 및 영상 사이즈 지정

1. 이진화

영상 이진화 과정

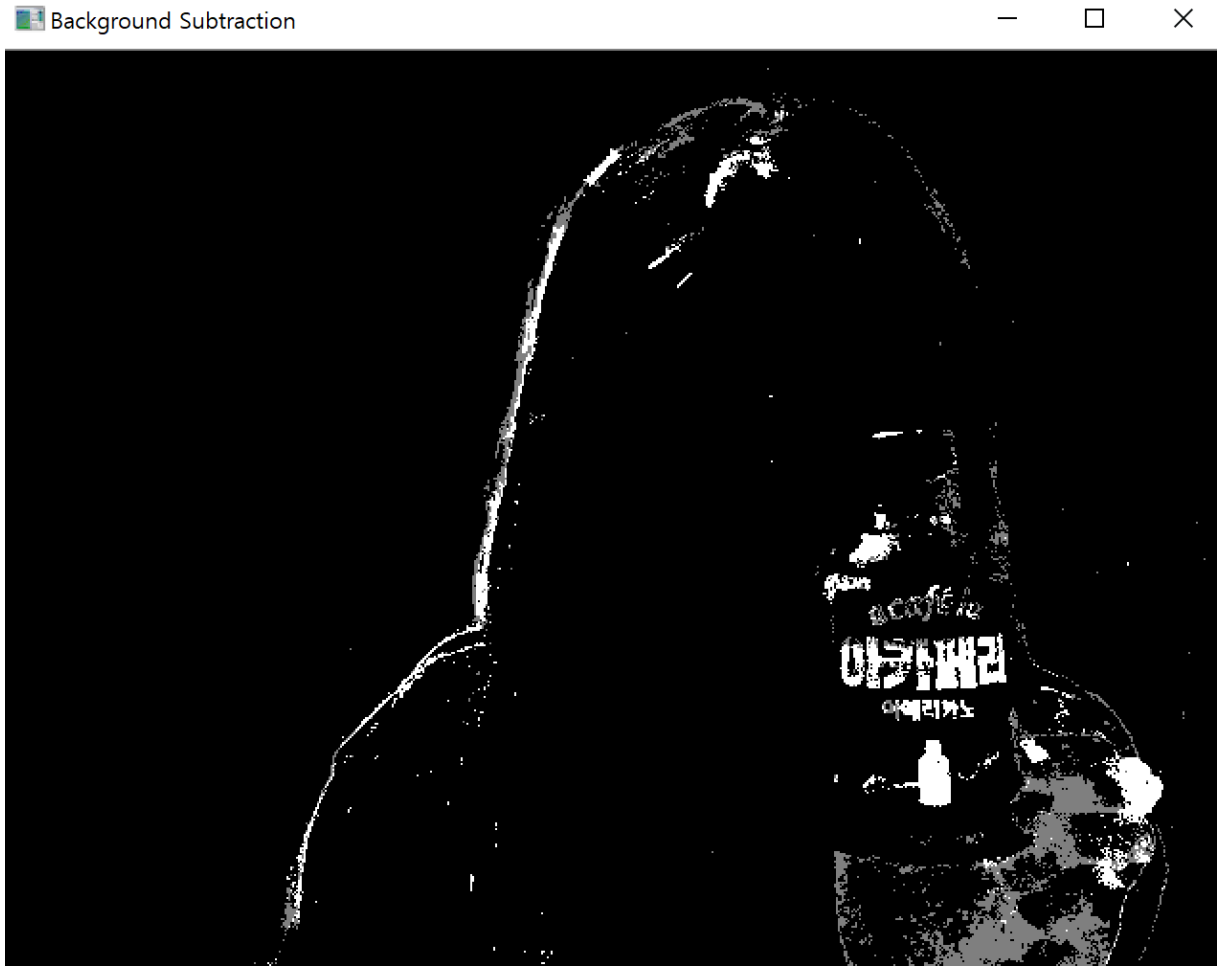
- 어떤 임계값을 정하고 이 값을 기준으로 회색조 영상을 이진 영상으로 생성
- 본 프로젝트에선 COLOR_BGR2GRAY,
- THRESH_BINARY, THRESH_TOZERO 이진화 타입 사용
- 전경/배경 검출의 기초 작업

2. 적응적 이진화

영상을 분할하여 이진화

- 영상을 분할하여 더 세세하게 이진화.
- 가중치가 가우시안인 윈도우를, 이웃 화소에 씌워서 계산한 가중치의 합을 임계값으로 하는 ADAPTIVE_THRESH
- _GAUSSIAN_C를 매개변수로 하여 adaptiveThreshold() 함수 사용
- 전경/배경 검출의 기초 작업

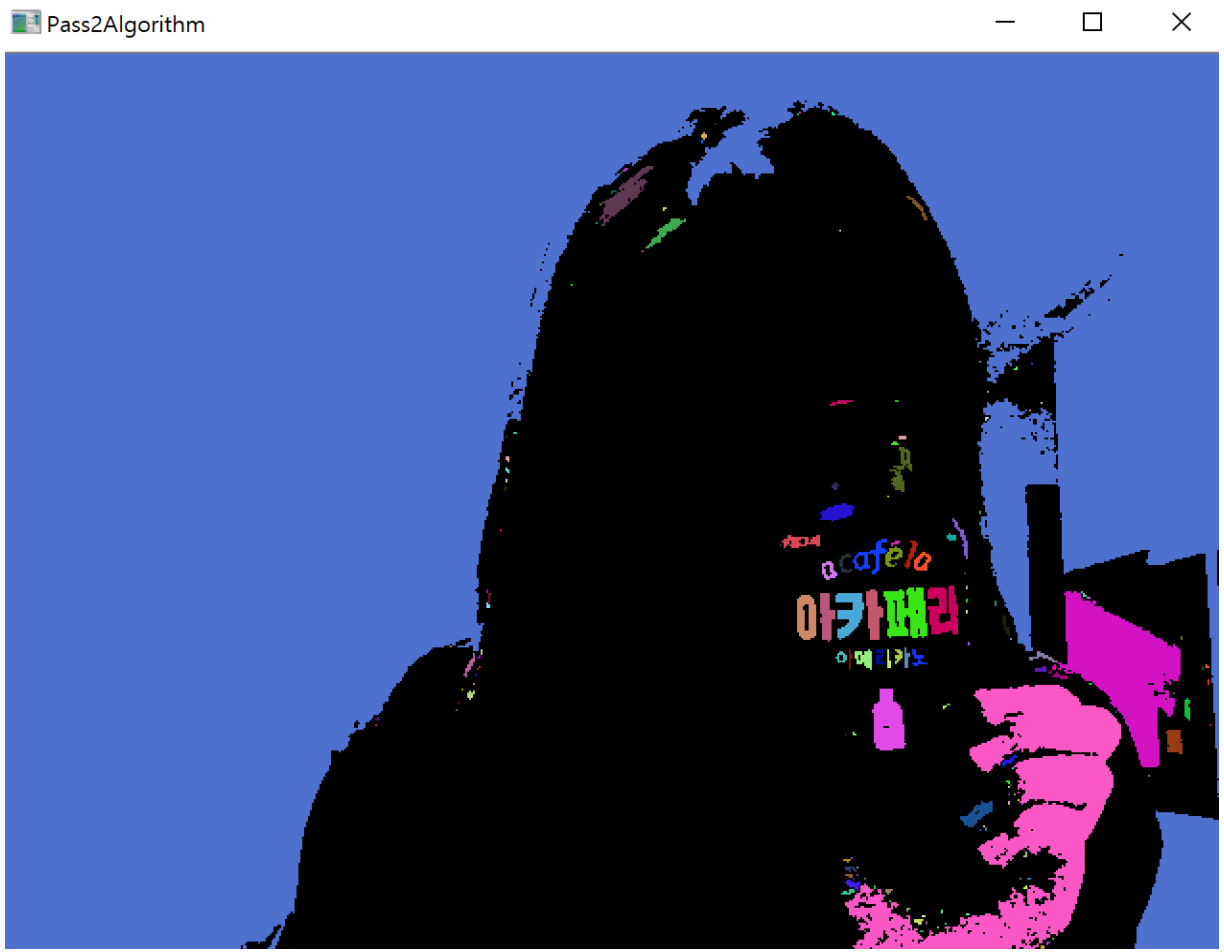
3. 전경/배경 분할(1) – BackgroundSubtractorMOG2



움직이는 전경 검출

- 가우시안 혼합 기반 전경/배경 분할 알고리즘 BackgroundSubtractor MOG2을 활용
- 전경과 배경의 차이를 이용하여 threshhold로 전경 마스크를 생성하는 방식
- 영상 내 움직이는 객체를 확인 가능

4. 전경/배경 분할(2) – 2-pass algorithm



색상 입혀 전경 검출

- 연결 성분 레이블링을 통해 2-패스 알고리즘으로 구현(4-연결)
- conntectedComponentsWithStats() 내장함수를 사용
- 컬러이미지에 사용되는 3개의 채널을 가지는 백터를 적용하여, RGB를 통해 전경에 색상을 표현함.

코드 흐름도

OPEN

카메라 개방 및 영상 얻기

```
int main()
{
    VideoCapture capture(0);

    if (!capture.isOpened()) {
        cout << "Cannot open capture : 카메라 개방 오류 !!" << endl;
        return 0;
    }

    Size size = Size((int)capture.get(CAP_PROP_FRAME_WIDTH),
        (int)capture.get(CAP_PROP_FRAME_HEIGHT));
    cout << "Size = " << size << endl;

    waitKey(100);
```

영상 프레임 별 저장 및 카운트

```
    while (1) {
        capture >> frame;
        if (frame.empty()) break;
        cout << "FrameNum = " << ++frameNum << endl;
```

그레이스케일, 이진화/적응적 이진화

```
cvtColor(frame, grayImage, COLOR_BGR2GRAY);
imshow("grayImage", grayImage);
```

```
threshold(grayImage, image, 85, 255, THRESH_TOZERO);
```

```
static void adaptive_gaussian() {
    adaptiveThreshold(grayImage, adapImage, 255, ADAPTIVE_THRESH_GAUSSIAN_C, THRESH_BINARY, 11, 2);
    imshow("Adaptive_gaussian", adapImage);
}
```

가우시안 기반 전경/배경 분할(1)

```
Mat result;
Ptr<BackgroundSubtractorMOG2> pmog;
pmog = createBackgroundSubtractorMOG2();

pmog->apply(frame, result);
imshow("Background Subtraction", result);
ckey = waitKey(30);
```

2-패스 알고리즘 전경/배경 분할(2)

```
static void pass2Algorithm() {

    threshold(grayImage, image, 85, 255, THRESH_TOZERO);

    Mat labels, centroids, img_color, stats;
    int n = connectedComponentsWithStats(image, labels, stats, centroids, 4);

    vector<Vec3b> colors(n + 1);
    colors[0] = Vec3b(0, 0, 0);
    for (int i = 1; i <= n; i++) {
        colors[i] = Vec3b(rand() % 256, rand() % 256, rand() % 256);
    }

    img_color = cv::Mat::zeros(image.size(), CV_8UC3);
    for (int y = 0; y < img_color.rows; y++) {
        for (int x = 0; x < img_color.cols; x++) {
            int label = labels.at<int>(y, x);
            img_color.at<cv::Vec3b>(y, x) = colors[label];
        }
    }

    imshow("Pass2Algorithm", img_color);
}
```

Esc 눌러 종료

```
if (ckey == 27)
    break;
```

디지털 영상의 엣지 및 전경/배경 검출을 주제로 한

04. 과제를 마치며

과제 수행 소감

곽배준	이형석	전유미
수업시간에 배운 내용을 프로젝트를 통해 더 자세히 알 수 있게 되었고, 막연하게만 알고있던 영상처리를 가까이 할 수 있던 기회였다. 실생활의 디지털 영상들을 다뤄 새롭게 표현할 수 있다는 점에 감명 받았다. 앞으로 4차 산업혁명에 있어 영상처리가 어떻게 쓰일지 관심을 가지게 됐다.	영상 처리를 배우면서 생각했던 자율 주행 자동차의 기능 중 하나인 영상 인식 기술을 직접 구현해보면서 배운 것을 응용해볼 수 있었다는 점이 인상 깊었고 배운 점을 응용해보면서 복습할 수 있는 좋은 기회였다고 생각한다.	카메라 영상의 전경/배경 검출을 손수 마스크를 씌워서 처리하는 것이 어려웠는데, OpenCV의 라이브러리를 사용하니 훨씬 간편하고 수월하게 구현할 수 있었다. 강의 중 실습에 이어 더 깊이 공부할 수 있어 좋았고, 다른 영상처리 기능도 궁금해지는 계기가 됐다.

THANK YOU

영상처리 1팀

소프트웨어전공 2016156001 곽배준

소프트웨어전공 2016156026 이형석

소프트웨어전공 2016156032 전유미

2020-2 디지털 영상처리 – 안세종 교수님