HW1 보고서

로빛 20기 인턴 모시온

2025407006 로봇학부

목차

1. 개요
   1-1. 목표
   1-2. 기본 개념
   1-3. 요약

2. 코드 분석
   2-1. 헤더 코드
   2-2. 소스 코드

3. 실행 결과
   3-1. 바이너리 이미지
   3-2. 가우시안 필터 여부

4. 참조 링크

1. 개요

1-1. 목표

본 과제는 제시된 예제를 참고 및 InRange 함수를 사용하여 각각의 공을 분류한 후 바이너리 이미지로 전환하여 출력.
이에 추가적으로 가우시안 필터를 적용하며 필터의 유무가 이미지에 도출되는 결과를 확인 및 분석.

<지시사항에 따라 모든 문체는 개조식으로 구성>

1-2. 기본 개념

(1) InRange

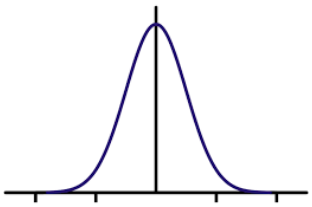
범위를 설정하여 지정 범위 내의 색상을 1(흰색) 또는 255, 아닌 색을 모두 0(검은색)으로 출력하는 바이너리 함수

예시

inRange(<사진>, <최하단 범위 1>, <최상단 범위 2>, 출력 이미지);

(2) Gaussianblur

주어진 프레임을 흐리게할 때 사용
오른쪽의 가우시안 필터 적용하여
평균을 통한 중앙 픽셀값 변경



<커널 크기 기준이므로 중앙 픽셀이 명백>
<커널 크기는 무조건 (홀수*홀수)로서 정의>

예시

GaussianBlur(<사진>, <커널 크기>, <테두리 픽셀 처리 상수>)

(3) Scalar

OpenCV 내의 Class
프레임의 픽셀 값을 정의하는 용도
RGB의 순서와 반대로 BGR 순서 정의

예시

Scalar(B, G , R, <투명도>);
또는

Scalar(<밝기>); -> 그레이스케일 용도

(4) Mat

행렬의 약자(matrix), 다차원 배열을 opencv에 표현하는 Class.

각각 빈 행렬, 지정된 행렬을 표현할 수 있다.

본 과제에서는 이미지를 저장하기 위한 용도로서 사용.

예시

Mat <행렬 변수>

또는

Mat <행렬 변수>(<가로>, <세로>, <행렬 범위>)

<행렬 범위>

이름	의미	값	범위
CV_8U	unsigned char (uchar) 8-bit unsigned integer	0	0 ~ 255
CV_8S	signed char (schar) 8-bit signed integer	1	-128 ~ 127
CV_16U	unsigned short (ushort) 16-bit unsigned integer	2	0 ~ 65535
CV_16S	signed short (short) 16-bit signed integer	3	-32768 ~ 32767

1-3. 요약

과제를 수행하기 위한 기본적인 개념을 조사 및 활용

2. 코드 분석

2-1. 헤더 코드

(강의 제공자 "김근형"선배님의 조언을 얻어 ros2 내의 opencv를 활용)

`#include <opencv2/opencv.hpp>`

ROS2의 패키지 다운로드 시 포함된 opencv 헤더를 호출

`#include <string>`

과제 사진은 문자열 경로로 지정되기 때문에 불러오기 위해 헤더 호출

`using namespace cv;`

opencv 내의 여러 Class 호출에 따라 필요한 전처리 수식언 사전 언급

`class HW1`

Class는 HW1로 선언하며, 이후 소스 코드에서 이를 활용하여 객체 생성

(public)

`HW1(const string& path);`

사진 경로를 받으며 초기화하는 생성자

`void progress();`

코드의 실행을 위한 각각의 함수 호출하는 함수

(private)

`string path_img;`

사진 경로를 문자열로 저장

`Mat raw_img;`

초기 프레임 행렬을 저장할 변수

`Mat hsv_img;`

BGR의 이미지를 HSV로 변환하여 저장할 행렬 변수

`Scalar lower_red1, upper_red1;`

빨간색의 색체 구간을 저장할 변수

`Scalar lower_red2, upper_red2;`

채도가 옅은 빨간색만을 저장할 변수

`Scalar lower_green, upper_green;`

초록색 색체 구간 저장 변수

`Scalar lower_blue, upper_blue;`

파란색 색체구간 저장 변수

`void bgrtohsv();`

색상 기준표를 HSV기준으로 하는 opencv에 맞춰 hsv변환하는 함수

`void colorfilter();`

Scalar와 inRange를 통한 색체 추출 및 이진화 함수

`void gaussianfilter();`

블러 필터 처리 함수

`Mat result_r;`

두 영역으로 구분되는 적색 결합 행렬

`Mat g_img;`

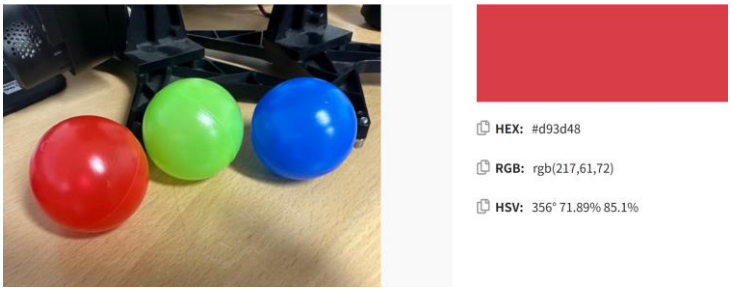
녹색 저장 행렬 변수

`Mat b_img;`
청색 저장 행렬 변수

2-2. 소스 코드

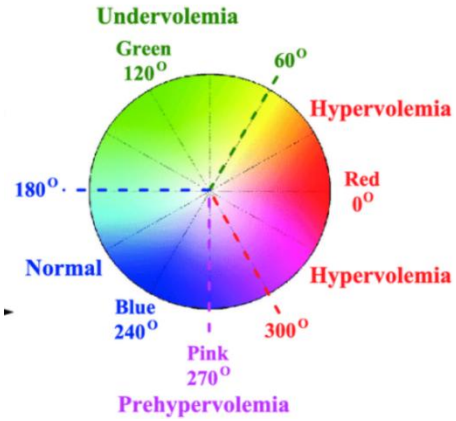
`#include "../include/hw1.hpp"`
HW1 Class 활용을 위해 헤더 호출

`raw_img = imread(path_img, IMREAD_COLOR);`
문자열 경로 기반 사진을 path_img에 불러와 저장
(imread(<이미지 경로>, <이미지 저장 방식(흑백, 컬러, 구분 없음)>))
(IMREAD_COLOR = 24bit의 3가지 색 채널(BGR) 컬러 저장)



참조: https://ko.rakko.tools/tools/64/#google_vignette

위 링크를 참고하여 각각의 색상을 검출 및 Scalar 기준치 정의
`lower_red1 = Scalar(0, 50, 50);`
`upper_red1 = Scalar(10, 255, 255);`
`lower red2 = Scalar(160, 100, 100);`
Scalar를 통해 hsv색상 필터 구간 생성
적색 계열 – 10 ~ 160
채도 - 50 ~ 255
밝기 - 50 ~ 255



`lower_red2 = Scalar(160, 100, 100);`
`upper_red2 = Scalar(180, 255, 255);`
적색 계열 – 160 ~ 180
채도 - 50 ~ 255
밝기 - 50 ~ 255
위를 통해 적색의 특수 색체 구간을 포함
이후 녹색 및 청색 또한 동일 과정

```
void HW1::bgrtohsv()
{
    cvtColor(raw_img, hsv_img, COLOR_BGR2HSV);
}
```

Class의 메서드 함수 bgrtohsv 정의

cvtColor 함수를 통하여 raw_img를 HSV로 전환하여 이를 hsv_img에 저장

```
void HW1::colorfilter()
```

색상 필터 적용을 위한 함수 정의

```
Mat r_img1, r_img2;
```

특이 경우로서 두 구간으로 분류되어 정의되는 적색 프레임 저장 변수 선언

```
inRange(hsv_img, lower_red1, upper_red1, r_img1);
```

inRange함수 사용하여 이진화 시작

사전 지정된 색체 범위에 따라 이진화된 프레임를 r_img1에 저장

```
inRange(hsv_img, lower_red2, upper_red2, r_img2);
```

다른 부분의 적색 이진화 프레임을 r_img2에 저장

```
inRange(hsv_img, lower_green, upper_green, g_img);
```

녹색 색체 구간에 따라 이진화된 프레임을 g_img에 저장

```
inRange(hsv_img, lower_blue, upper_blue, b_img);
```

청색 또한 이와 동일

```
result_r = r_img1 + r_img2;
```

각각 서로다른 두 구간으로 정의되는 적색을 합치기 위한 식

각각은 Mat으로 생성한 행렬이므로 + 연산자로 덧셈하여 겹치기가 가능

이를 통해 인간이 이해하는 적색의 범주로 이진화 결과값을 result_r에 저장

```
imshow("raw", raw_img);
```

각각의 행렬(프레임)을 윈도우로 출력(기본 이미지)

```
imshow("red", result_r);
```

전체 적색 이진화 행렬(프레임) 출력

```
imshow("green", g_img);
```

녹색 이진화 행렬(프레임) 출력

```
imshow("blue", b_img);
```

청색 이진화 행렬(프레임)출력

```
void HW1::progress()
```

각각의 생성된 함수들을 순서대로 시행하는 함수

```
int main(int argc, char** argv)
```

초기에 처리해야할 기본 이미지(raw_img)경로 지정 및 초기 시행

```
string img_path = "/home/mso/Desktop/20th_intern_mosion/OpenCV/day1/hw1/hw1_package/hw1.png"
```

문자열로 저장된 사진 경로를 img_path에 저장

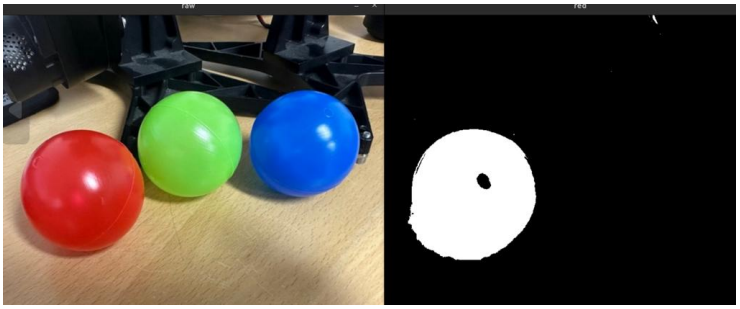
```
processor.progress();
```

마지막으로 progress함수를 호출함으로서 모든 동작 구동

3. 실행 결과

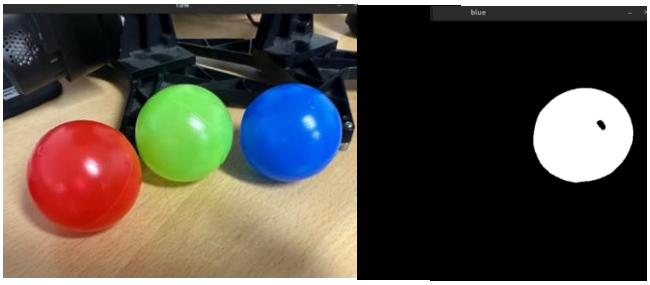
다음 오른쪽의 사진이 제공된 기본 이미지이다.

3-1. 바이너리 이미지



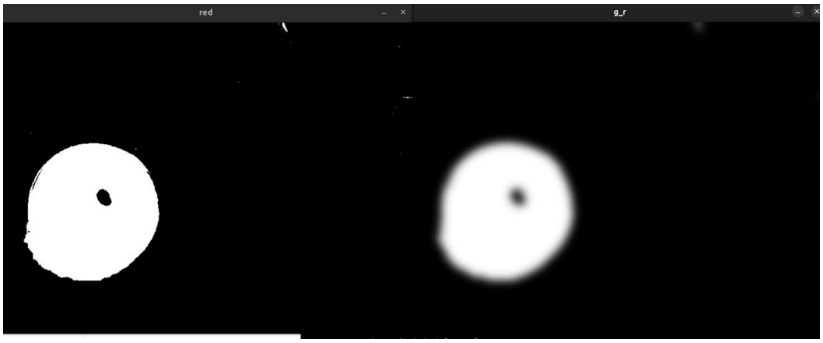
적색 추출 및 이진화 결과
우측 상단의 노이즈가 식별됨

녹색 추출 및 이진화 결과
공 우측 하단의 노이즈가 식별됨

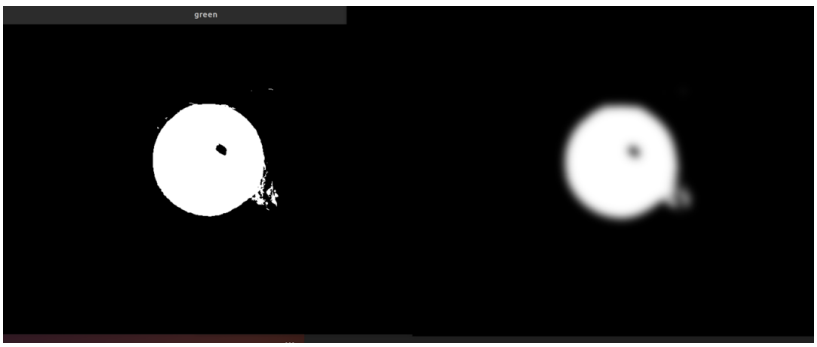


청색 추출 및 이진화 결과
육안으로 식별되는 노이즈는 없음

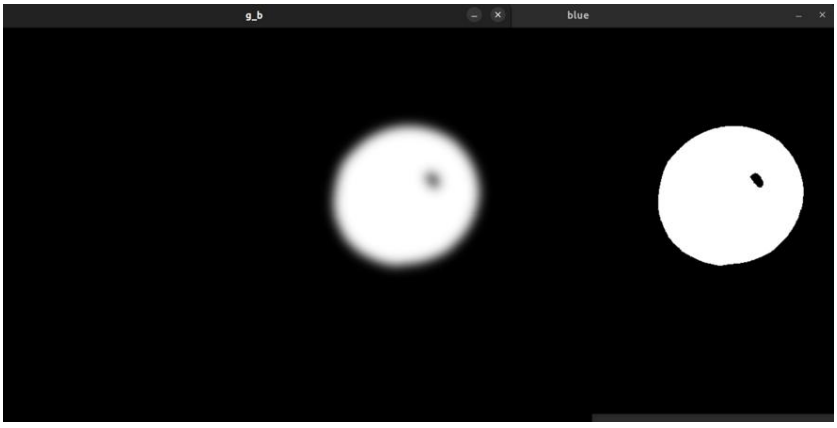
3-2. 가우시안 필터 여부(51*51, 0)



적색 이진화 사진의 경우, 우측 상단에 눈에 띄던 노이즈가 사라짐



녹색 이진화 사진 또한, 공 우측 상단 미세한 노이즈가 제거됨.
허나, 미세하지 않은 노이즈는 완전 제거가 되지 않음



청색 이진화의 경우, 특이점은 없음

결과

이로서 가우시안의 필터의 실용성에 대해 확인 가능하였다.
규모가 작고 비중이 작은 노이즈일수록 평균값 저장에 따라 소규모 노이즈 제거에 탁월함을 확인할 수 있었다.

4. 참조 링크

가우시안 필터

https://diyver.tistory.com/67

inRange 및 Scalar 사용

https://diyver.tistory.com/98