디지털 영상처리 연구실 연구보고서

정지우

기본 주요클래스 및 함수

<영상 파일>

<C++>

해더파일 읽기 -> 픽셀데이터 수직반전된 형태로 읽기

=> BITMAPFILEHEADER → BITMAPINFOHEADER → RGBQUAD → Image data (pixel value)

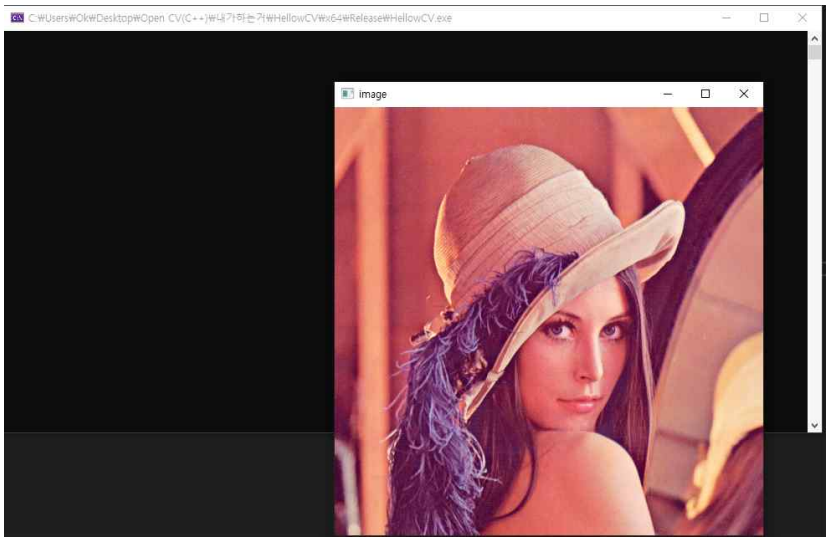
```
#include <stdio.h>
#include <Windows.h>

BYTE* readBMP(BITMAPFILEHEADER* hf, BITMAPINFOHEADER* hinfo, char filename[])
{
    int width = 0, height = 0;
    FILE* f = NULL;
    f = fopen(filename, "rb");
    if (f == NULL)
    {
        printf("fail to read file");
        return NULL;
    }
    fread(hf, sizeof(BITMAPFILEHEADER), 1, f);
    fread(hinfo, sizeof(BITMAPINFOHEADER), 1, f);
    //fread(hRGB, sizeof(RGBQUAD), 256, f);

    width = hinfo->biWidth;
    height = hinfo->biHeight;

    printf("Width :%d \t Height :%d\t Image Size :%d\n", width, height, hinfo->biSizeImage

    BYTE* InputImg = NULL;

    InputImg = (BYTE*)malloc(sizeof(BYTE) * hinfo->biSizeImage);
    fread(InputImg, sizeof(BYTE), hinfo->biSizeImage, f);

    fclose(f);

    if (hf->bfType == 0x4D42)
    {
        printf("bmp format image\n");
    }
    else
    {
        printf("wrong file type input\n");
    }

    return InputImg;
}
```

<Open CV>

```
Mat imread(const String& filename, int flags = IMREAD_COLOR);
```

- filename 불러올 영상 파일 이름
- flags 영상 파일 불러오기 옵션 플래그. ImreadModes 열거형 상수를 지정합니다.
- 반환값 불러온 영상 데이터(Mat 객체)

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

using namespace cv;
using namespace std;

int main()
{
    Mat img;
    img = imread("lenna.bmp");

    if (img.empty()) {
        cerr << "Image load failed!" << endl;
        return -1;
    }

    namedWindow("image");
    imshow("image", img);

    waitKey();

    return 0;
}
```

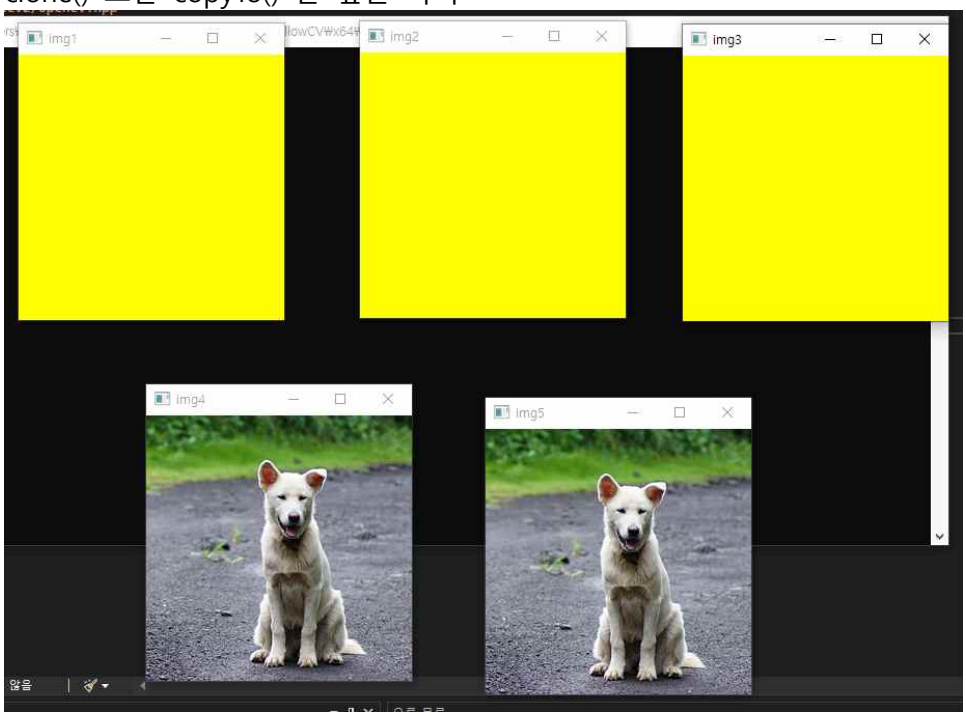


<복사 방법>

```
int main()
{
    Mat img1 = imread("dog.bmp");

    Mat img2 = img1;
    Mat img3;
    img3 = img1;

    Mat img4 = img1.clone();
    Mat img5;
    img1.copyTo(img5);

    img1.setTo(Scalar(0, 255, 255));    // yellow

    imshow("img1", img1);
    imshow("img2", img2);
    imshow("img3", img3);
    imshow("img4", img4);
    imshow("img5", img5);

    waitKey();
    destroyAllWindows();

    return 0;
}
```

'='를 사용한 복사는 **얕은 복사**

clone() 또는 copyTo() 는 깊은 복사



<특정 색 Tracking>

```
int main() {
    // 비디오 캡처 객체 생성 (카메라 사용 시)
    VideoCapture cap(0); // 0은 첫번째 카메라 사용한다는 의미
    if (!cap.isOpened()) {
        cout << "Error: Could not open the camera." << endl;
        return -1;
    }

    Mat frame, hsv, mask, result; //Mat는 행렬을 저장하고 관리할 수 있는 클래스
    while (true) {
        cap >> frame; // 한 프레임의 정지 영상을 받아와서 frame에 저장
        if (frame.empty()) {
            cout << "Error: Could not capture frame." << endl;
            break;
        }

        // BGR 이미지를 HSV 이미지로 변환
        cvtColor(frame, hsv, COLOR_BGR2HSV);
```

(HSV : 색상, 채도, 밝기)

RGB에서 바로 처리하지 않고 HSV 색상 공간으로 변경하여 처리하는게 유리한 이유

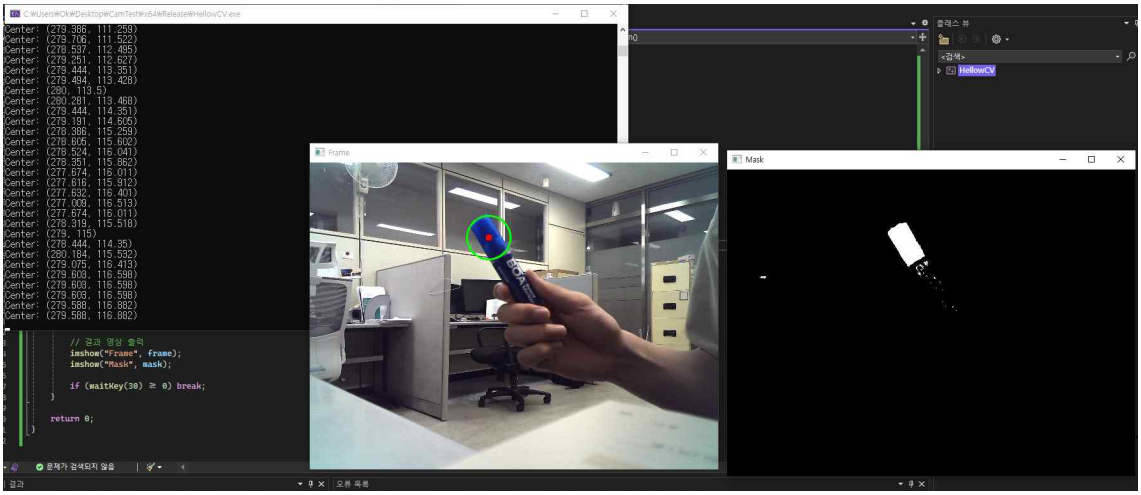
- RGB는 밝기와 색상이 분리되지 않기 때문에, 색상과 채도와 밝기가 분리되어 있는 HSV가 영상처리하는데 유리합니다.

```
        // 파란색 범위를 정의
        Scalar lower_blue(100, 150, 50);
        Scalar upper_blue(140, 255, 255);

        // 파란색에 해당하는 부분을 마스크로 생성하고
        // inRange()는 범위 안의 부위를 1(흰색) 0(검은색) 이진화해서 저장한다.
        inRange(hsv, lower_blue, upper_blue, mask);
```

```
        // 파란색 물체의 컨투어(외곽선) 찾기
        vector<vector<Point>> contours; //2차원 배열 만드는법
        // mask에서 외곽선을 찾아서 contours에 저장
        findContours(mask, contours, RETR_EXTERNAL, CHAIN_APPROX_SIMPLE);
```

```
    Point2f center;
    float radius;
    float max_area = 0;
    Point2f max_center;
    float max_radius;

    for (size_t i = 0; i < contours.size(); i++) {
        float area = contourArea(contours[i]);
        // 너무 작은 컨투어는 무시
        if (area < 500) {
            continue;
        }

        // 가장 큰 컨투어 찾기
        if (area > max_area) {
            max_area = area;
            // 컨투어(외각선)의 최소 외접 원을 계산
            minEnclosingCircle(contours[i], max_center, max_radius);
        }
    }
```

```
        // 가장 큰 파란색 물체의 중심 좌표 표시
        if (max_area > 0) {
            // frame(원본)에 초록색 외접원 그리기 , 2는 두께
            circle(frame, max_center, (int)max_radius, Scalar(0, 255, 0), 2);
            // frame(원본)에 빨간색 중심점 표시, -1은 채워라 라는 뜻
            circle(frame, max_center, 5, Scalar(0, 0, 255), -1);

            // 중심 좌표 출력
            cout << "Center: (" << max_center.x << ", " << max_center.y << ")" << endl;
        }

        // 결과 영상 출력
        imshow("Frame", frame);
        imshow("Mask", mask);

        if (waitKey(30) >= 0) break;
    }

    return 0;
}
```


C:₩Users₩Ok₩Desktop₩CamTest₩x64₩Release₩HellowCV.exe
Center: (279.386, 111.259)
Center: (279.706, 111.522)
Center: (278.537, 112.485)
Center: (279.251, 112.627)
Center: (279.444, 113.351)
Center: (279.494, 113.428)
Center: (280., 113.5)
Center: (280.281, 113.468)
Center: (279.444, 114.351)
Center: (279.191, 114.605)
Center: (278.986, 115.259)
Center: (278.605, 115.602)
Center: (278.524, 116.041)
Center: (278.351, 115.862)
Center: (277.674, 116.011)
Center: (277.616, 115.912)
Center: (277.632, 116.401)
Center: (277.008, 116.513)
Center: (277.674, 116.011)
Center: (278.319, 115.518)
Center: (279., 115)
Center: (279.444, 114.35)
Center: (280.184, 115.532)
Center: (279.075, 116.413)
Center: (279.603, 116.598)
Center: (279.565, 116.598)
Center: (279.603, 116.598)
Center: (279.588, 116.882)
Center: (279.508, 116.882)
// 결과 영상 출력
imshow("Frame", frame);
imshow("Mask", mask);
if (waitKey(30) >= 0) break;
return 0;
문제가 검색되지 않음
결과
오류 목록
클래스 뷰
<검색>
HellowCV
Frame
Mask