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

using namespace cv;
using namespace std;

void filter_embossing();                                //엠보싱필터링
void emb_change(int pos, void* userdata);               //트랙바

void gaussian();                                        //가우시안필터링
void gau_change(int pos, void* userdata);               //트랙바

void mean();                                            //평균값필터링
void mean_change(int pos, void* userdata);              //트랙바

void unsharp_mask();                                    //언샤프마스크필터링
void unsharp_change(int pos, void* userdata);           //트랙바

void filter_bilateral();                                //잡음추가,양방향필터링
void bilateral_change(int pos, void* userdata);         //트랙바

void filter_median();                                   //잡음추가,미디언필터링
void median_change(int pos, void* userdata);            //트랙바

void sobel_edge();                                      //마스크기반에지검출
void sobel_change(int pos, void* userdata);             //트랙바

void canny_edge();                                      //캐니에지검출
void canny_change(int pos, void* userdata);             //트랙바

void hough_lines();                                     //허프변환직선검출
void lines_change(int pos, void* userdata);             //트랙바

void hough_circles();                                   //허프변환곡선검출
void circles_change(int pos, void* userdata);           //트랙바

void adaptive();                                        //적응형이진화
void adaptive_change(int pos, void* userdata);          //트랙바

void erode_dilate();                                    //이진영상의 침식과팽창
void open_close();                                      //이진영상의 열기와 닫힘

void labeling_stats();                                  //이진영상 레이블링

void contours_hier();                                   //외곽선검출
void hier_change(int pos, void* userdata);              //트랙바

int main(void)                  //메인문
{
    filter_embossing();         //엠보싱필터링
    //gaussian();               //가우시안필터링
    //mean();                   //평균값필터링
    //unsharp_mask();           //언샤프필터링

    //filter_bilateral();       //잡음추가,양방향필터링
    //filter_median();          //잡음추가,미디언필터링

    //sobel_edge();             //마스크기반 에지검출
    //canny_edge();             //캐니에지검출
    //hough_lines();            //허프변환직선검출
    //hough_circles();          //허프변환곡선검출

    //adaptive();               //적응형이진화
    //erode_dilate();           //이진영상의 침식과 팽창
    //open_close();             //이진영상의 열기와 닫힘

    //labeling_stats();         //레이블링
    //contours_hier();          //외곽선검출

    return 0;
}
```

1번

엠보싱필터링, 가우시안필터링,평균값 필터링, 언샤프마스크필터링 수행

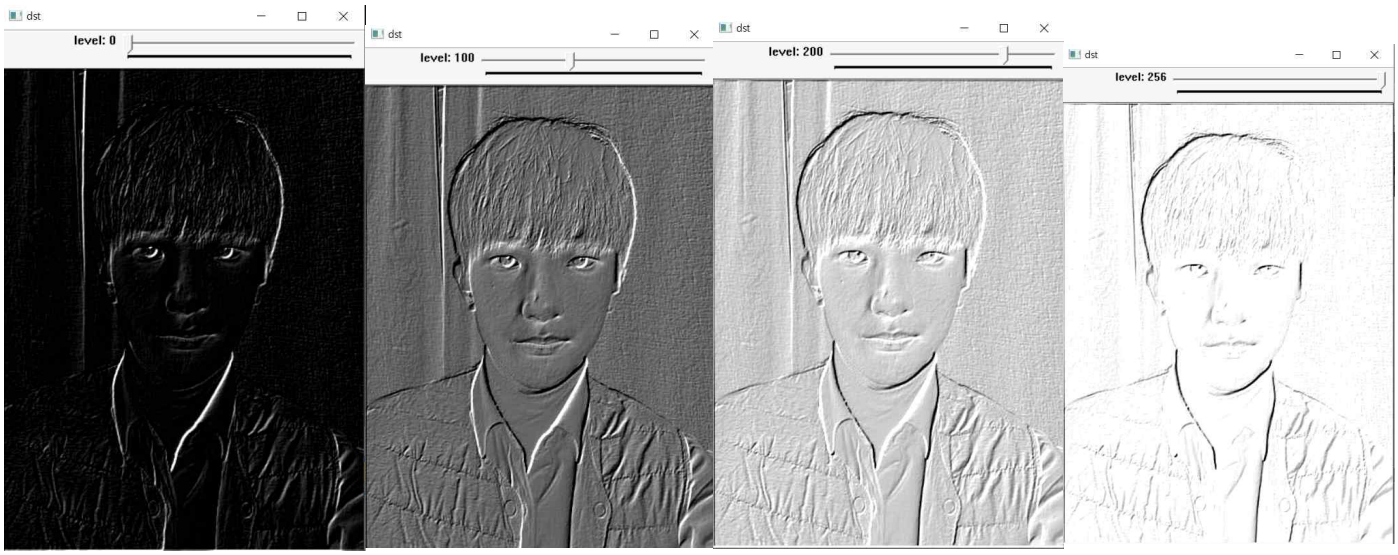
1)엠보싱 필터링

```
void filter_embossing()                                          //엠보싱필터링
{
    Mat src = imread("my.bmp", IMREAD_GRAYSCALE);                //나의 얼굴 흑백 불러오기
    namedWindow("dst");                                          //트랙바 윈도우 창 이름
    createTrackbar("level", "dst", 0, 256, emb_change, (void*)&src);   //트랙바 생성함수, 트랙바값이 256까지

    waitKey();
}
void emb_change(int pos, void* userdata)                         //엠보싱필터링 트랙바
{
    Mat src = *(Mat*)userdata;                                   //void*타입 인자 userdata를
                                                                 //Mat*타입으로 형변환한 후 src변수로 참조
    float data[] = { -1, -1, 0, -1, 0, 1, 0, 1, 1 };             //3x3그기의 엠보싱 필터 마스크 행렬
    Mat emboss(3, 3, CV_32FC1, data);                            //위 행렬의 emboss생성

    Mat dst;                                                     //출력이름생성
    filter2D(src, dst, -1, emboss, Point(-1, -1), pos);          //엠보싱필터 수행, 필터링 결과 영상에 변화하는pos값을 더함

    imshow("dst", dst);                                          //출력
}
```

결과

level

0 100 200 256



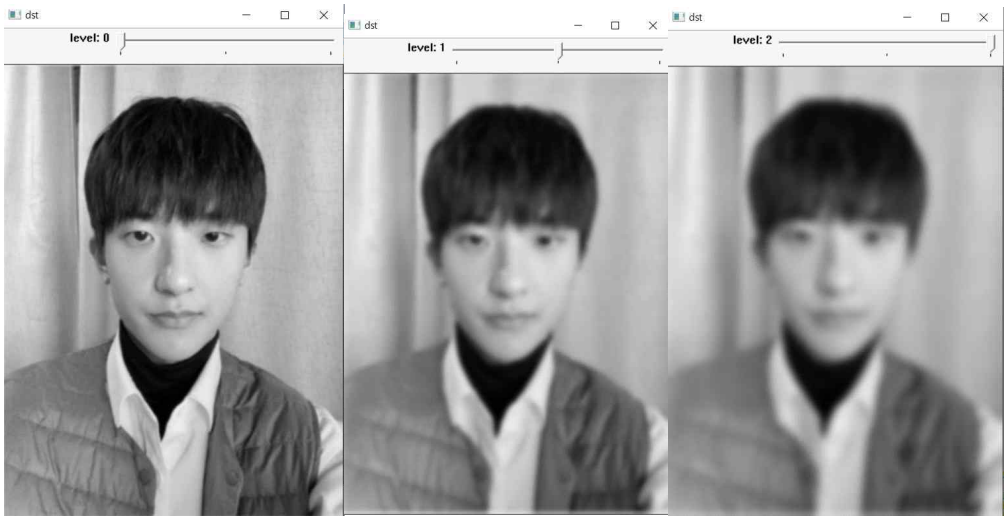
2)가우시안필터링

```
void gaussian()                                                  //가우시안필터링
{
    Mat src = imread("my.bmp", IMREAD_GRAYSCALE);                //나의 얼굴 흑백 불러오기
    namedWindow("dst");                                          //트랙바 윈도우창 이름
    createTrackbar("level", "dst", 0, 2, gau_change, (void*)&src);   //트랙바 생성함수,트랙바 5까지 변환

    waitKey();
}
void gau_change(int pos, void* userdata)                         //트랙바
{
    Mat src = *(Mat*)userdata;                                   //void*타입 인자 userdata를
                                                                 //Mat*타입으로 형변환한 후 src변수로 참조
    if (pos % 2 == 0)pos = pos + (pos + 1);                      //pos가 짝수일때 실행 (0일때 1, 2일때 5로 변환)
    else pos = pos + 2;                                          //홀수일때 순서에 맞게하기위해 2더함(1일때 3으로 변환)
    Mat dst;                                                     //dst변수 지정
    GaussianBlur(src, dst, Size(), (double)pos);                 //src영상에 가우시안필터링 실행후 dst에 저장

    imshow("dst", dst);                                          //dst 실행
}
```

결과

level

0(sigma=1) 1(sigma=3) 2(sigma=5)



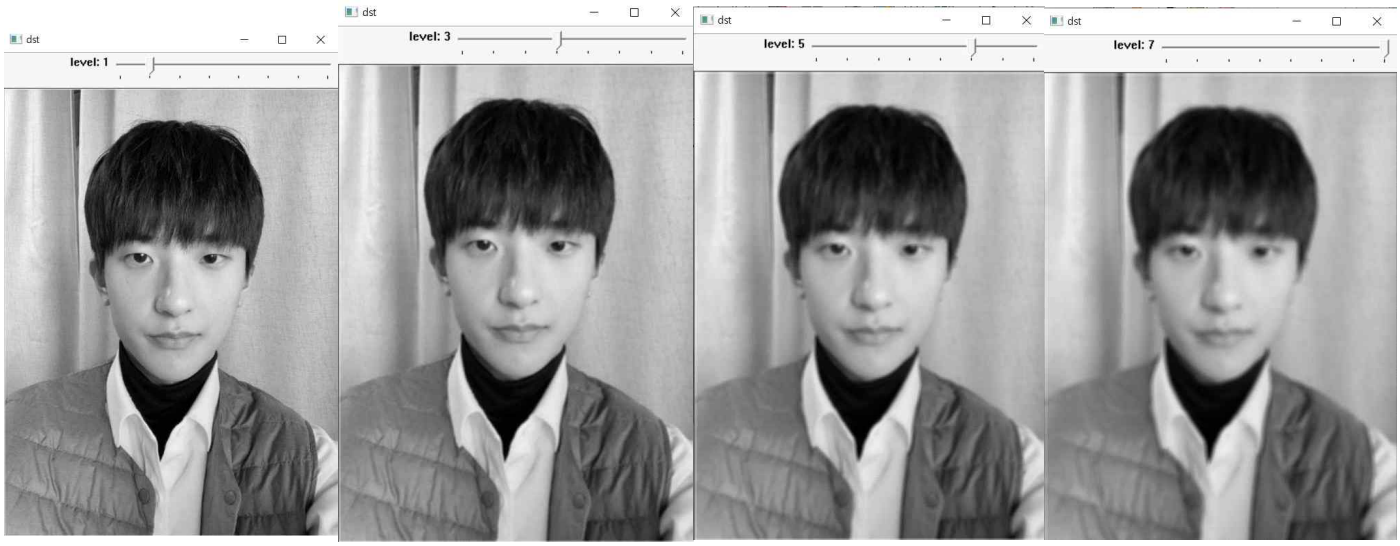
3)평균값필터링

```
void mean()                                                    //평균값필터링
{
    Mat src = imread("my.bmp", IMREAD_GRAYSCALE);              //나의 얼굴 흑백 불러오기
    namedWindow("dst");                                        //트랙바 윈도우창 이름
    createTrackbar("level", "dst", 0, 7, mean_change, (void*)&src);   //트랙바 생성함수, 트랙바 7까지 변환

    waitKey();
}
void mean_change(int pos, void* userdata)
{
    Mat src = *(Mat*)userdata;                                 //void*타입 인자 userdata를
                                                               //Mat*타입으로 형변환한 후 src변수로 참조
    Mat dst;                                                   //dst변수 지정
    blur(src, dst, Size(pos, pos));                            //posXpos크기의 평균값필터 마스크 이용하여 블러링 수행

    imshow("dst", dst);                                        //dst실행
}
```

결과

level

1 3



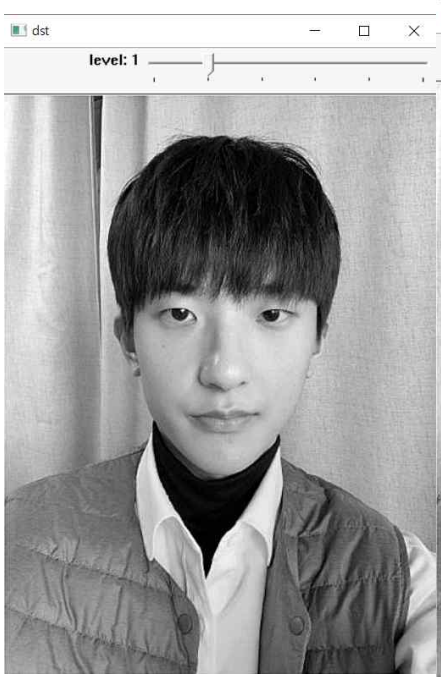
4)언샤프마스크필터링

```
void unsharp_mask()                                                    //언샤프필터링
{
    Mat src = imread("my.bmp", IMREAD_GRAYSCALE);                      //나의 얼굴 흑백 불러오기
    namedWindow("dst");                                                //트랙바 윈도우창 이름
    createTrackbar("level", "dst", 0, 5, unsharp_change, (void*)&src); //트랙바 생성함수, 트랙바 5까지 변환

    waitKey();
    destroyAllWindows();
}
void unsharp_change(int pos, void* userdata)                           //트랙바
{
    Mat src = *(Mat*)userdata;                                         //void*타입 인자 userdata를
                                                                       //Mat*타입으로 형변환한 후 src변수로 참조

    Mat blurred;                                                       //blurred변수 지정
    GaussianBlur(src, blurred, Size(), pos);                           //가우시안 필터 이용한 블러링 영상을 blurred에 저장

    float alpha = 1.f;                                                 //언샤프 마스크 필터링 수행
    Mat dst = (1 + alpha) * src - alpha * blurred;

    imshow("dst", dst);
}
```

결과

level

1



3

5

2번

영상에 적당한 잡음 첨가하고, 양방향필터링, 미디언필터링 수행

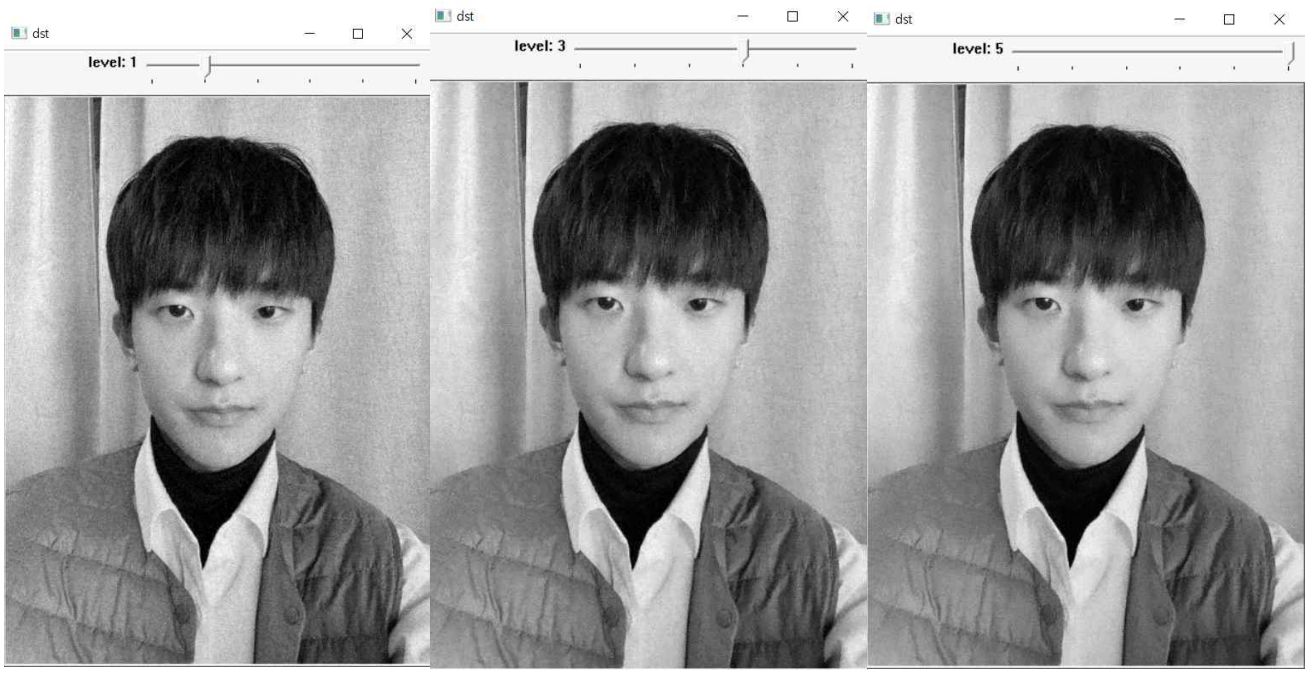
1) 잡음첨가, 양방향필터링

```
void filter_bilateral()                                          //양방향필터링
{
    Mat src = imread("my.bmp", IMREAD_GRAYSCALE);                //나의 얼굴 흑백으로 불러오기

    Mat noise(src.size(), CV_32SC1);                             //잡음 추가
    randn(noise, 0, 10);                                         //표준편차가 10인 잡음
    Mat src1;
    add(src, noise, src1, Mat(), CV_8U);                         //노이즈 더한걸 src1에 저장
    namedWindow("dst");                                          //트랙바 윈도우창 이름
    createTrackbar("level", "dst", 0, 5, bilateral_change, (void*)&src1);   //트랙바 생성 함수, 트랙바값 5까지

    waitKey();

    destroyAllWindows();
}
void bilateral_change(int pos, void* userdata)                   //트랙바
{
    Mat src1 = *(Mat*)userdata;                                  //void*타입 인자 userdata를
                                                                 //Mat*타입으로 형변환한 후 src1변수로 참조
    Mat dst;
    bilateralFilter(src1, dst, -1, pos+5, pos);                  //색공간의 표준편차 pos+5,좌표공간 표준편차 pos의 양방향 필터링
    imshow("dst", dst);                                          //결과창띄움
}
```

결과

level

1



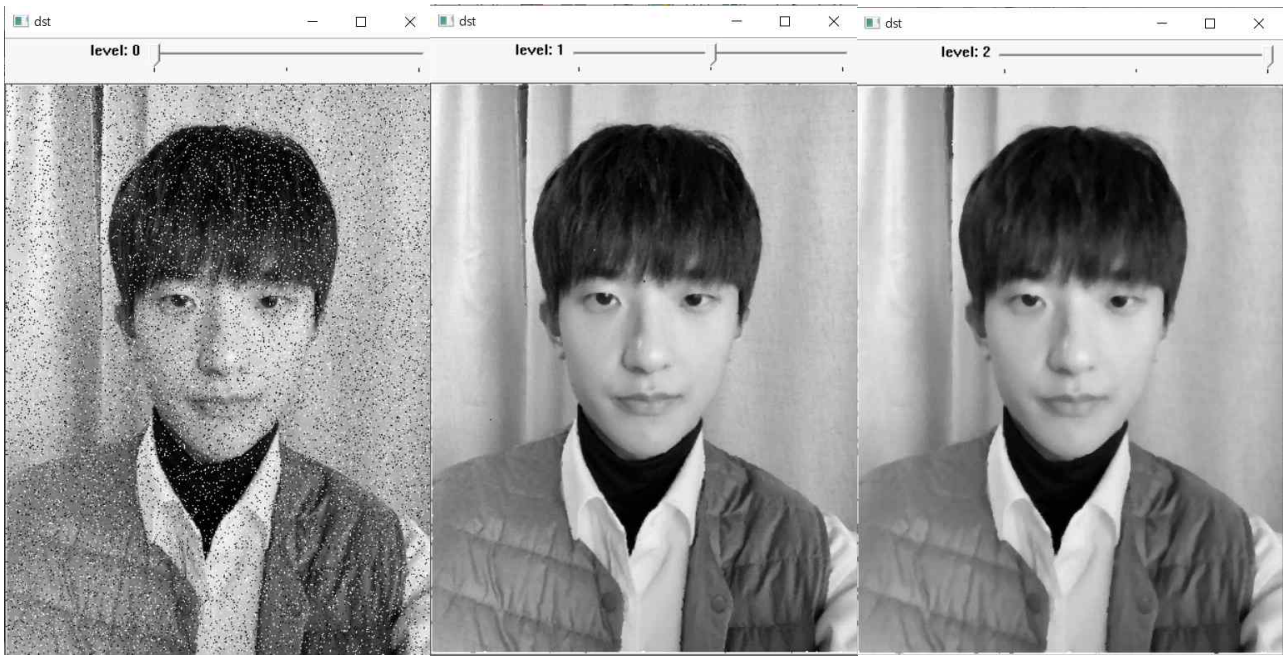
2) 잡음첨가, 미디언필터링

```
void filter_median()                                                    //미디언 필터링
{
    Mat src = imread("my.bmp", IMREAD_GRAYSCALE);                       //나의 얼굴 흑백 불러오기

    int num = (int)(src.total() * 0.1);                                 //src영상에서 10%해당하는 픽셀 값을 0또는 255로 설정(소금&후추 잡음)
    for (int i = 0; i < num; i++) {
        int x = rand() % src.cols;
        int y = rand() % src.rows;
        src.at<uchar>(y, x) = (i % 2) * 255;
    }

    namedWindow("dst");                                                 //트랙바 윈도우 창 이름
    createTrackbar("level", "dst", 0, 2, median_change, (void*)&src);   //트랙바 생성 함수, 트랙바값 2까지
    waitKey();
    destroyAllWindows();
}
void median_change(int pos, void* userdata)                             //트랙바
{
    Mat src = *(Mat*)userdata;                                          //void*타입 인자 userdata를
                                                                        //Mat*타입으로 형변환한 후 src1변수로 참조
    if (pos%2==0)pos = pos + (pos+1);                                   //pos가 짝수일때 실행 (0일때 1, 2일때 5로 변환)
    else pos = pos +2;                                                  //홀수일때 순서에 맞게하기위해 2더함(1일때 3으로 변환)

    Mat dst;
    medianBlur(src, dst, pos);                                          //크기가 pos인 미디안 필터 실행
    imshow("dst", dst);                                                 //결과 띄우기
}
```

결과

level

0 1



3번

마스크기반 에지검출, 캐니에지검출, 허프변환직선검출, 허프변환곡선검출

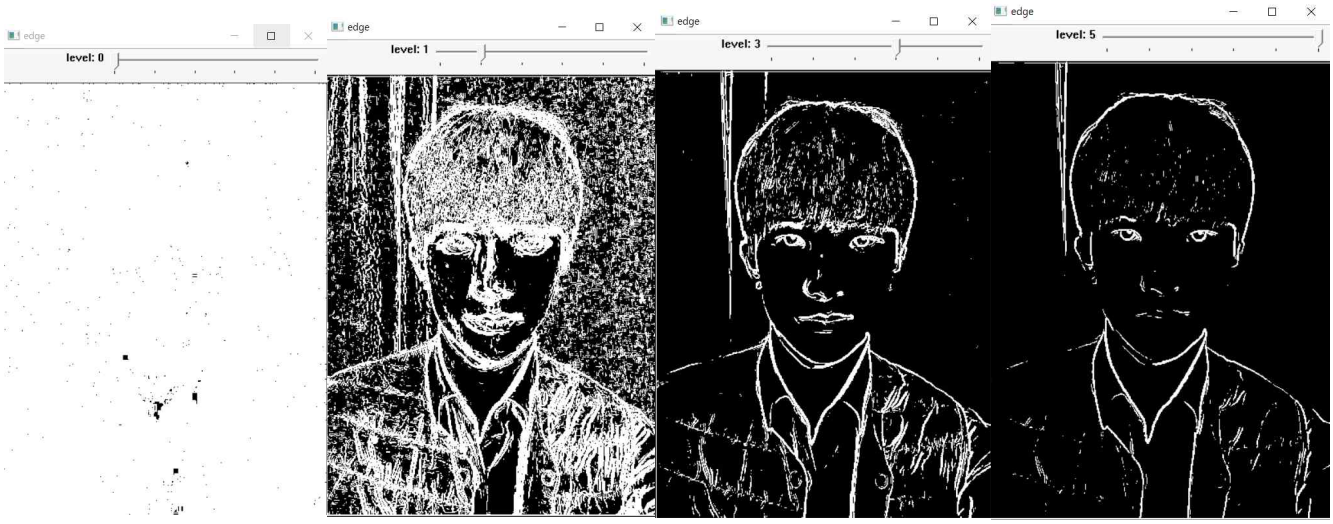
1)마스크기반 에지검출

```
void sobel_edge()                                                   //마스크기반 에지검출
{
    Mat src = imread("my.bmp", IMREAD_GRAYSCALE);                   //나의 얼굴 흑백 불러오기

    Mat dx, dy;                                                     //dx,dy변수 지정
    Sobel(src, dx, CV_32FC1, 1, 0);                                 //x축 방향으로 1차 편미분 구하여 dx행렬에 저장
    Sobel(src, dy, CV_32FC1, 0, 1);                                 //y축 방향으로 1차 편미분 구하여 dy행렬에 저장

    Mat fmag, mag;
    magnitude(dx, dy, fmag);                                        //dx,dy행렬로부터 그래디언트 크기 계산하여 fmag에 저장
    fmag.convertTo(mag, CV_8UC1);                                   //실수형 행렬 fmag를 그레이스케일형식으로 변환하여 mag에 저장
    namedWindow("edge");                                            //트랙바 윈도우창 이름
    createTrackbar("level", "edge", 0, 5, sobel_change, (void*)&mag);   //트랙바 생성 함수, 트랙바 값 5까지

    imshow("src", src);
    waitKey();
    destroyAllWindows();
}
void sobel_change(int pos, void* userdata)
{
    Mat mag = *(Mat*)userdata;
    Mat edge = mag > pos*30;                                        //에지 판별을 위한 그래디언트 크기 임계값을
                                                                    //pos*30으로 설정하여 에지 판별
                                                                    //행렬 edge의 원소값은 mag행렬 원소 값이 pos*30보다 크면 255,
                                                                    //작으면0으로 설정
    imshow("edge", edge);                                           //결과 출력
}
```

결과

level

0 1 3 5



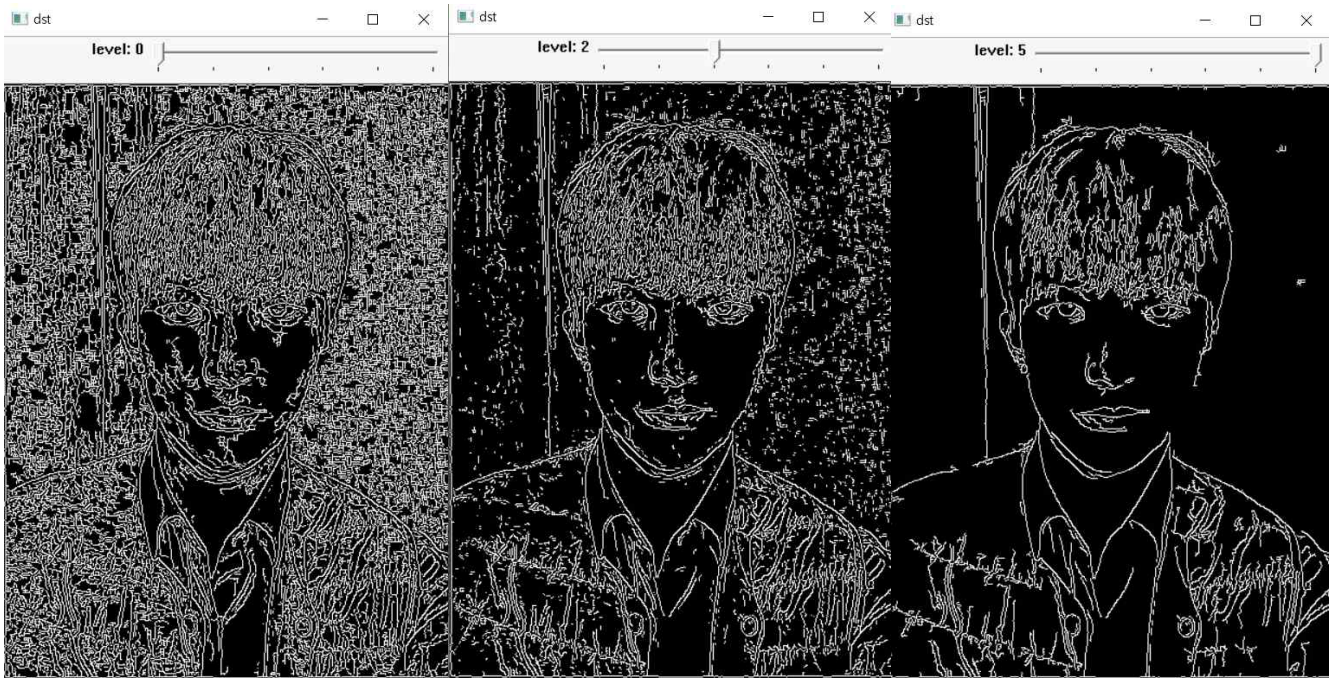
2)캐니에지검출

```
void canny_edge()                                                    //캐니에지검출
{
    Mat src = imread("my.bmp", IMREAD_GRAYSCALE);                    //나의 얼굴 흑백 불러오기

    namedWindow("dst");                                              //트랙바 윈도우창 이름
    createTrackbar("level", "dst", 0, 5, canny_change, (void*)&src); //트랙바 생성 함수, 트랙바값 5까지

    imshow("src", src);                                              //원본 보기

    waitKey();
    destroyAllWindows();
}
void canny_change(int pos, void* userdata)                           //트랙바
{
    Mat src = *(Mat*)userdata;                                       //void*타입 인자 userdata를
                                                                     //Mat*타입으로 형변환한 후 src1변수로 참조
    Mat dst;
    Canny(src, dst, 50, pos * 30);                                   //낮은 임계값을 50, 높은 임계값을 pos*30으로 설정하여 캐니 에지 검출

    imshow("dst", dst);                                              //결과 출력
}
```

결과

level

0



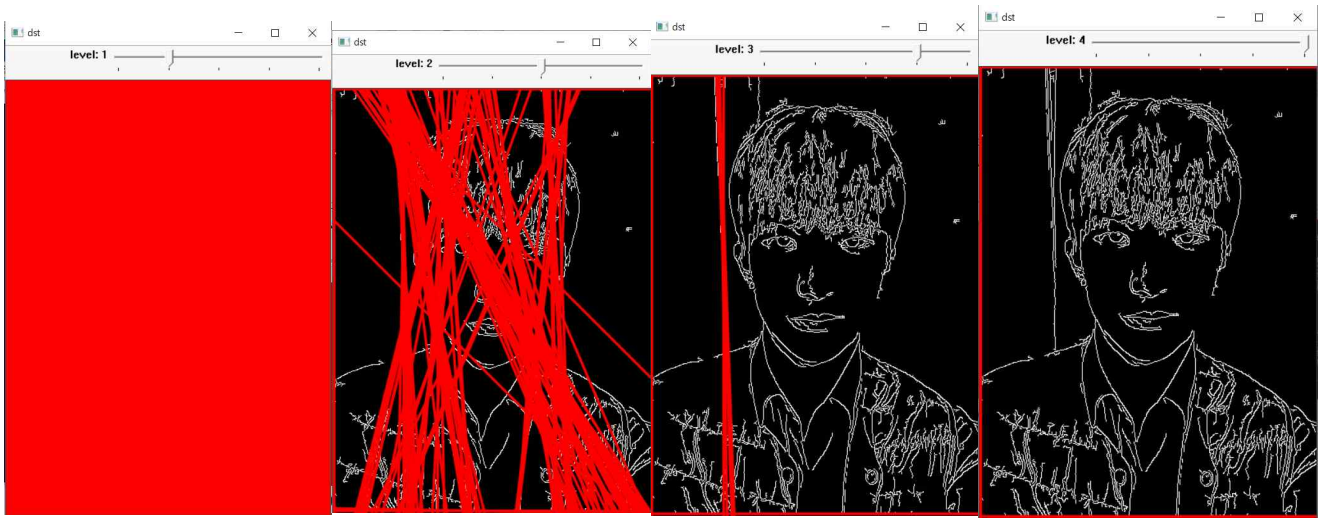
3)허프변환직선검출

```
void hough_lines()                                                  //허프변환직선검출
{
    Mat src = imread("my.bmp", IMREAD_GRAYSCALE);                   //나의 얼굴 흑백 불러오기

    Mat edge;
    Canny(src, edge, 50, 150);                                      //캐니에지검출기를 이용하여 구한 에지 영상 edge에 저장
    namedWindow("dst");                                             //트랙바 윈도우창 이름
    createTrackbar("level", "dst", 0, 4, lines_change, (void*)&edge);   //트랙바 생성 함수, 트랙바 값 4까지

    imshow("src", src);
    waitKey(0);
    destroyAllWindows();
}
void lines_change(int pos, void* userdata)                          //트랙바
{
    Mat edge = *(Mat*)userdata;                                     //void*타입 인자 userdata를
                                                                    //Mat*타입으로 형변환한 후 src1변수로 참조
    vector<Vec2f> lines;
    HoughLines(edge, lines, 1, CV_PI / 180, pos * 50);              //rho(픽셀단위),theta(라디안 단위)의 값을 line에 저장

    Mat dst;
    cvtColor(edge, dst, COLOR_GRAY2BGR);                            //edge를 BGR 3채널 컬러 영상으로 변환하여 dst에 저장

    for (size_t i = 0; i < lines.size(); i++) {                     //line 개수만큼 for문 반복
        float rho = lines[i][0], theta = lines[i][1];
        float cos_t = cos(theta), sin_t = sin(theta);
        float x0 = rho * cos_t, y0 = rho * sin_t;
        float alpha = 1000;

        Point pt1(cvRound(x0 - alpha * sin_t), cvRound(y0 + alpha * cos_t));    //pt1에x0에서 멀리 떨어져있는 직선상의 점좌표 저장
        Point pt2(cvRound(x0 + alpha * sin_t), cvRound(y0 - alpha * cos_t));    //pt2에y0에서 멀리 떨어져있는 직선상의 점좌표 저장
        line(dst, pt1, pt2, Scalar(0, 0, 255), 2, LINE_AA);         //검출된 직선을 두께가 2인 빨간색 실선으로 그림
    }


    imshow("dst", dst);                                             //결과 출력
}
```

결과

level

1 2 3 4



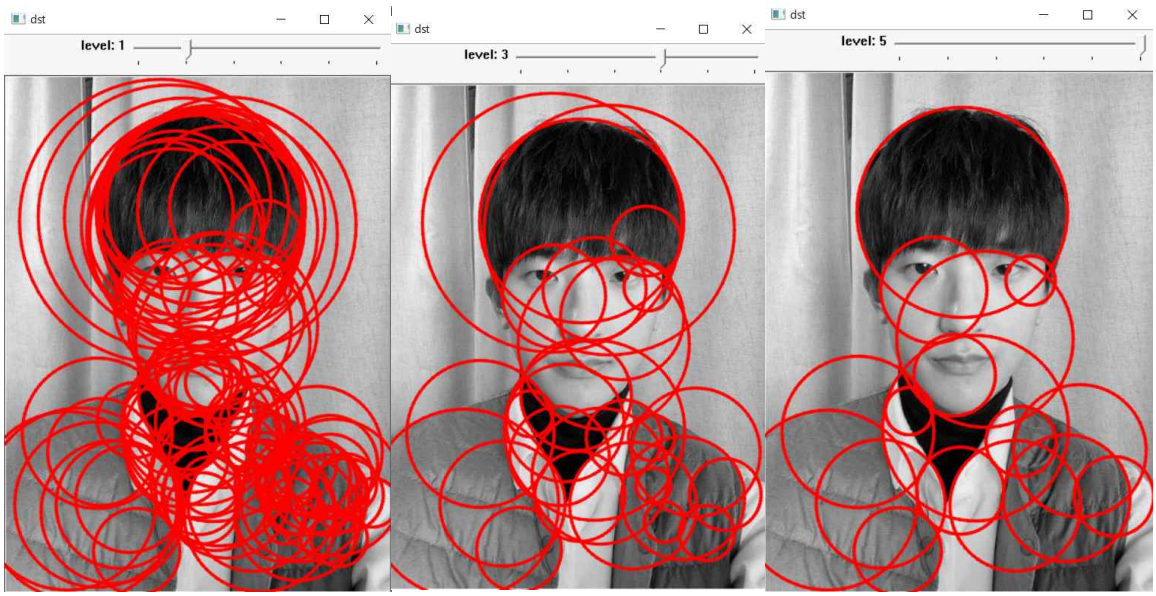
4)허프변환곡선검출

```
void hough_circles()     //허프변환곡선
{
    Mat src = imread("my.bmp", IMREAD_GRAYSCALE);                          //나의 얼굴 흑백 불러오기

    Mat blurred;
    blur(src, blurred, Size(3, 3));                                        //잡음제거, blurred에 저장
    namedWindow("dst");                                                    //트랙바 윈도우 창 이름
    createTrackbar("level", "dst", 0, 5, circles_change, (void*)&blurred); //트랙바 생성 함수, 트랙바 값 5 까지

    waitKey(0);
    destroyAllWindows();
}
void circles_change(int pos, void* userdata)                               //트랙바
{
    Mat blurred = *(Mat*)userdata;                                         //void*타입 인자 userdata를
                                                                           //Mat*타입으로 형변환한 후 blurred변수로 참조
    Mat src = imread("my.bmp", IMREAD_GRAYSCALE);                          //트랙바함수 안에서 사용하기위해 불러옴
    vector<Vec3f> circles;
    HoughCircles(blurred, circles, HOUGH_GRADIENT, 1, pos*10, 150, 30);    //원검출,축적배율 크기 동일하게,원중심거리pos*10보다 작으면 검출x
                                                                           //캐니에지검출 높은임계값 150,낮은 임계값 30으로 설정
                                                                           //검출후 circles에 저장
    Mat dst;
    cvtColor(src, dst, COLOR_GRAY2BGR);                                    //입력영상을 3채널 컬러영상으로 변환

    for (Vec3f c : circles) {                                              //검출된 원을 빨간색으로 그림
        Point center(cvRound(c[0]), cvRound(c[1]));
        int radius = cvRound(c[2]);
        circle(dst, center, radius, Scalar(0, 0, 255), 2, LINE_AA);
    }


    imshow("dst", dst);
}
```

결과

level

1 3 5



4번

적응형이진화, 이진영상의 침식과 팽창, 이진영상의 열기와 닫힘

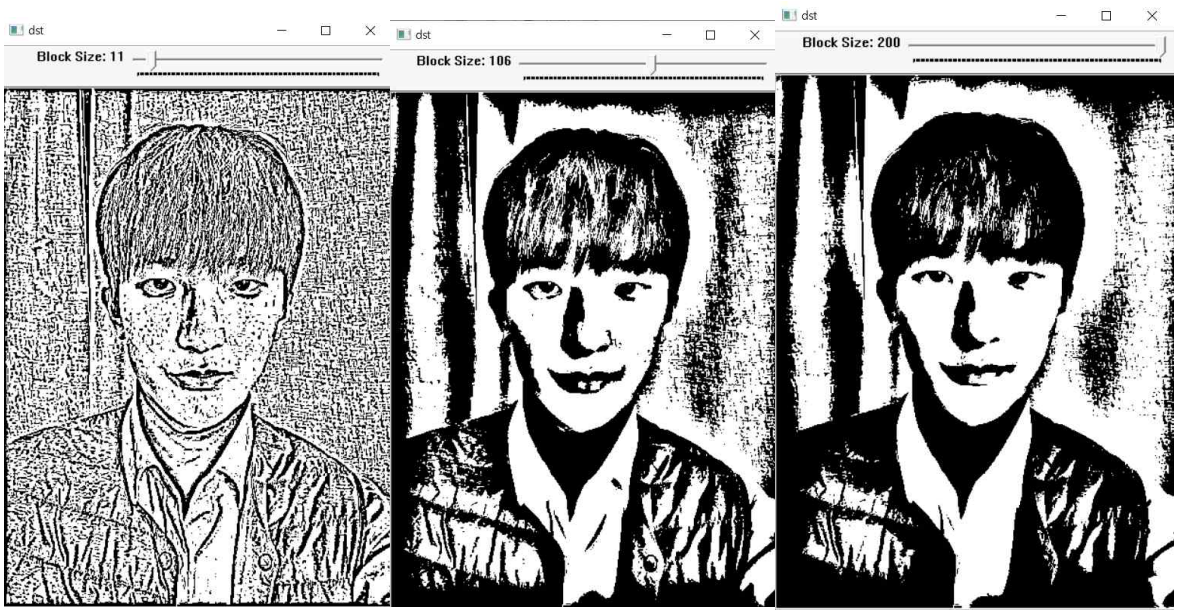
1)적응형 이진화

```
void adaptive()                                                          //적응형이진화
{
    Mat src = imread("my.bmp", IMREAD_GRAYSCALE);                        //나의 얼굴 흑백 불러오기

    imshow("src", src);

    namedWindow("dst");                                                  //트랙바 윈도우창 이름
    createTrackbar("Block Size", "dst", 0, 200, adaptive_change, (void*)&src);  //트랙바 생성함수, 트랙바값 200까지
    setTrackbarPos("Block Size", "dst", 11);                             //트랙바 초기 위치 11로 설정

    waitKey(0);

}

void adaptive_change(int pos, void* userdata)                            //트랙바
{
    Mat src = *(Mat*)userdata;                                           //void*타입 인자 userdata를
                                                                         //Mat*타입으로 형변환한 후 src변수로 참조

    int bsize = pos;
    if (bsize % 2 == 0) bsize--;                                         //bsize값이 짝수이면 1빼서 홀수로
    if (bsize < 3) bsize = 3;                                            //bsize값이 3보다 작으면 3으로 설정

    Mat dst;
    adaptiveThreshold(src, dst, 255, ADAPTIVE_THRESH_GAUSSIAN_C, THRESH_BINARY,  //트랙바에서 설정한 블록크기를 이용하여 적응형 이진화 수행
        bsize, 2);                                                       //가우시안 가중 평균 사용 블록 평균에서 5 뺀 값을 임계값으로 사용

    imshow("dst", dst);                                                  //결과 출력
}
```

결과

level

11 106 200



2)이진영상의 침식과 팽창

```
void erode_dilate()                                          //이진영상 침식,팽창
{
    Mat src = imread("my.bmp", IMREAD_GRAYSCALE);            //나의 얼굴 흑백 불러오기

    Mat bin;
    threshold(src, bin, 0, 255, THRESH_BINARY | THRESH_OTSU);   //입력영상을 오츠알고리즘으로 자동 이진화 수행 bin에 저장

    Mat dst1, dst2;
    erode(bin, dst1, Mat());                                 //bin영상에 3x3 정방형 구조 요소를 이용하여 침식 연산 수행
    dilate(bin, dst2, Mat());                                //bin영상에 3x3 정방형 구조 요소를 이용하여 팽창 연산 수행

    imshow("src", src);                                      //원본
    imshow("bin", bin);                                      //이진화영상
    imshow("erode", dst1);                                   //침식수행한 영상
    imshow("dilate", dst2);                                  //팽창 수행한 영상

    waitKey();
    destroyAllWindows();
}
```

결과

원본

이진화영상

침식수행한 영상

팽창수행한 영상

3)이진영상의 열기와 닫힘

```
void open_close()                                                    //이진영상 열기,닫기
{
    Mat src = imread("my.bmp", IMREAD_GRAYSCALE);                    //나의 얼굴 흑백 불러오기

    Mat bin;
    threshold(src, bin, 0, 255, THRESH_BINARY | THRESH_OTSU);        //입력영상을 오츠알고리즘으로 자동 이진화 수행 bin에 저장

    Mat dst1, dst2;
    morphologyEx(bin, dst1, MORPH_OPEN, Mat());                      //열기연산
    morphologyEx(bin, dst2, MORPH_CLOSE, Mat());                     //닫기연산

    imshow("src", src);                                              //원본
    imshow("bin", bin);                                              //이진화영상
    imshow("opening", dst1);                                         //열기연산 영상
    imshow("closing", dst2);                                         //닫기연산 영상

    waitKey();
    destroyAllWindows();
}
```

결과

원본

이진화영상

열기연산영상

닫기연산 영상

5번

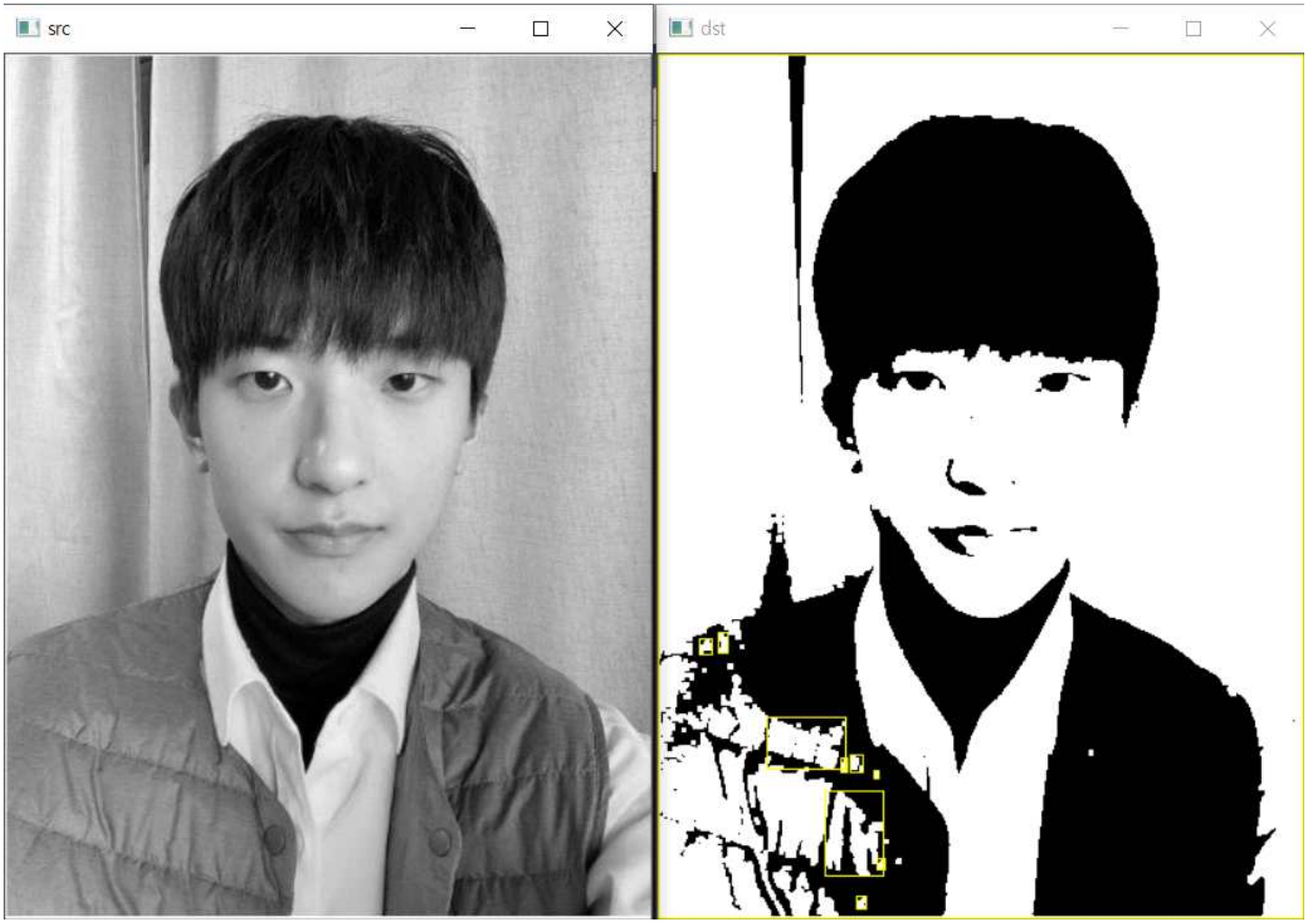
이진영상의 레이블링, 외곽선검출

1)이진영상의 레이블링

```
void labeling_stats()                                                    //레이블링
{
    Mat src = imread("my.bmp", IMREAD_GRAYSCALE);                        //나의 얼굴 흑백 불러오기

    Mat bin;
    threshold(src, bin, 0, 255, THRESH_BINARY | THRESH_OTSU);            //입력영상을 오츠알고리즘으로 자동 이진화 수행 bin에 저장
    Mat dst1;
    morphologyEx(bin, dst1, MORPH_OPEN, Mat());                          //열기연산 dst1에 저장

    Mat labels, stats, centroids;
    int cnt = connectedComponentsWithStats(dst1, labels, stats, centroids);    //레이블링 수행,각 객체 영역의 통계정보를 추출

    Mat dst;
    cvtColor(dst1, dst, COLOR_GRAY2BGR);                                 //3채널 컬러 영상 형식으로 변환하여 dst에 저장

    for (int i = 1; i < cnt; i++) {                                      //배경 제외 흰색 객체 영역에 대해서만 for반복문 수행
        int* p = stats.ptr<int>(i);

        if (p[4] < 20) continue;                                         //객체 픽셀수가 20보다 작으면 잡음간주 무시

        rectangle(dst, Rect(p[0], p[1], p[2], p[3]), Scalar(0, 255, 255));    //검출된 객체에 바운딩 박스를 노란색으로 그림
    }


    imshow("src", src);
    imshow("dst", dst);

    waitKey();
    destroyAllWindows();
}
```

결과

원본 **결과**



2)외곽선검출

```
void contours_hier()                                                    //외곽선검출
{
    Mat src = imread("my.bmp", IMREAD_GRAYSCALE);                       //나의 얼굴 흑백 불러오기
    namedWindow("dst1");                                                //트랙바 윈도우 이름
    createTrackbar("Block Size", "dst1", 0, 200, hier_change, (void*)&src);    //트랙바 생성 함수 , 트랙바 값 200까지
    setTrackbarPos("Block Size", "dst1", 11);                           //트랙바 11부터 시작

    waitKey(0);
    destroyAllWindows();
}
void hier_change(int pos, void* userdata)                               //트랙바
{
    Mat src = *(Mat*)userdata;                                          //void*타입 인자 userdata를
                                                                        //Mat*타입으로 형변환한 후 src1변수로 참조
    int bsize = pos;
    if (bsize % 2 == 0) bsize--;                                        //값이 짝수일때 -1감소
    if (bsize < 3) bsize = 3;                                           //값이 3보다  크면 3으로 설정

    Mat dst;
    adaptiveThreshold(src, dst, 255, ADAPTIVE_THRESH_GAUSSIAN_C, THRESH_BINARY,
        bsize, 2);

    vector<vector<Point> > contours;
    vector<Vec4i> hierarchy;
    findContours(dst, contours, hierarchy, RETR_CCOMP, CHAIN_APPROX_SIMPLE);    //hierarchy인자를 전달하여 계층 정보를 받음

    Mat dst1;
    cvtColor(dst, dst1, COLOR_GRAY2BGR);

    for (int idx = 0; idx >= 0; idx = hierarchy[idx][0]) {                  //0외곽선부터 시작 계층 정보의 다음 외곽선을 이동하면서 for문 실행
        Scalar c(rand() & 255, rand() & 255, rand() & 255);
        drawContours(dst1, contours, idx, c, -1, LINE_8, hierarchy);        //hierarchy정보를 전달하여 외곽선 그림 선두께-1로 설정(외곽선 내부 채움)
    }
    imshow("dst1", dst1);
}
```

결과

level

11

91

200