#1.Assignment

Term Project : Imager Processing

제출일	2023.06.21	**전공**	인공지능학과
과목	영상처리	**학번**	202176019
담당교수	김용훈	**이름**	윤혜주

1. 목차

2. 연습문제

1.1 연습문제 17

다음의 그림과 같이 캐니에지 알고리즘에서 이중 임계값을 트랙바로 만들어서 두 개의 임계값을 조절하여 에지를 검토하도록 프로그램을 작성하시오.

해당 문제에서는 캐니에지 알고리즘과 트랙바를 작성하여 임계값을 조절할 수 있게 만드는 게 문제의 목적이다.

캐니 에지 검출이란 John F. Canny에 의해 개발된 알고리즘으로 블러링을 통한 노이즈 제거 후 화소 기울기의 강도와 방향 검출을 한다. 이후 비최대치 억제를 하고 이력 임계값으로 edge를 결정한다.

1.1.1. 가우시안 블러링

블러링은 불필요한 잡음을 제거하기 위해서 수행하는 것이기 때문에 마스크의 크기를 다르게 하거나 다른 필터링을 적용해도 무관하다. 가우시안 필터링은 가우시안 분포를 마스크의 계수로 사용하여 회선을 수행하는 것을 말한다.

1.1.2. 화소 기울기(gradiant)의 강도와 방향 검출(소벨마스크)

화소 기울기 검출에는 가로 방향과 세로 방향의 소벨 마스크로 회선을 적용하고, 회선이 완료된 행렬을 이용해서 화소 기울기의 크기와 방향을 계산한다. 기울기의 방향은 4개방향(0, 45, 90, 135)으로 근사하여 단순화한다. 기울기의 방향과 edge의 방향은 수직을 이루어야 한다.

소벨 마스크는 edge 추출을 위한 가장 대표적인 1차 미분 연산이다. 수직, 수평 방향의 edge도 잘 추출하며, 중심화소 차분 비중을 높였기 때문에 대각선 방향의 edge도 잘 검출한다.

1.1.3. 비최대치 억제(non-maximum suppression)

비최대치 억제는 현재 화소가 이웃하는 화소들보다 크면 edge로 보존하고, 그렇지 않으면 edge가 아닌 것으로 간주해서 제거한다. 기울기의 방향에 있는 두 개의 화소를 비교 대상으로 선택한다. 현재 화소와 선택된 두 화소의 edge 강도를 비교하여 최대치가 아니면 억제되고, 최대치인 것만 edge로 결정한다. 잘못된 edge를 제거하는 쉬운 방법 중에 하나가 임계값을 설정하고, edge의 강도가 이 임계값보다 작으면 edge에서 제외한다. 임계값이 높으면 실제 edge도 제거될 수 있으며, 임계값이 낮으면 잘못된 edge를 제거하지 못하는 문제가 생길 수 있다.

1.1.4. 캐니 알고리즘

캐니 알고리즘은 잘못된 edge를 제거하고, 정확한 edge만을 검출하여 edge가 끊어지는 것을 방

지하는 방법으로 이력 임계값 방법을 사용한다. 두 개의 임계값을 사용해서 edge의 이력을 추적하여 edge를 결정하는 방법이다. 각 화소에서 높은 임계값보다 크면 edge 추적을 시작한다. 추적을 시작하면 추적하지 않은 이웃 화소들을 대상으로 낮은 임계값보다 큰 화소를 edge로 결정하는 방식이다.

1.1.5. 코드

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

using namespace std;
using namespace cv;

Mat img, gray, edge;
int th1 = 50, th2 = 50;

void onTrackbar1(int , void*) {
    Canny(gray, edge, th1,th1*2 ,3);
    imshow("canny edge", edge);
}
void onTrackbar2(int, void*) {
    Canny(gray, edge, th2,th2*2, 3);
    imshow("canny edge", edge);
}
int main() {
    img = imread("color_space.jpg", 1);
    CV_Assert(img.data);
    cvtColor(img, gray, COLOR_BGR2GRAY);

    namedWindow("canny edge", WINDOW_NORMAL);
    createTrackbar("th1", "canny edge", &th1, 255, onTrackbar1);
    createTrackbar("th2", "canny edge", &th2, 255, onTrackbar2);

    waitKey(0);
    return 0;
}
```

전체 코드

```
Mat img, gray, edge;
int th1 = 50, th2 = 50;
```

전역변수로 Mat img, gray, edge를 선언해주었다. 이 객체들은 이미지 데이터를 저장할 때 사용된다. Int형으로 th1과 th2를 선언하고 각각 50으로 초기화 해주었다. 이 값들은 Canny 엣지 검출 알고리즘에서 임계값을 담당한다.

```
int main() {
    img = imread("color_space.jpg", 1);
    CV_Assert(img.data);
```

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

    namedWindow("canny edge", WINDOW_NORMAL);
    createTrackbar("th1", "canny edge", &th1, 255, onTrackbar1);
    createTrackbar("th2", "canny edge", &th2, 255, onTrackbar2);

    waitKey(0);
    return 0;
}
```

main문에서는 imread를 활용해 이미지를 읽어들인다. 1로 하여 컬러 이미지로 받아와서 img에 저장한다. CV_Assert는 오류가 날 상황을 대비하여 선언해주었다. cvtColor는 img에 저장된 이미지를 그레이변환을 하여 gray에 저장해준다.

nameWindow함수를 사용하여 "canny edge"라는 이름의 윈도우를 생성하였다. WINDOW_NORMAL은 사용자가 창 크기를 조정할 수 있게 해준다. 그리고 createTrackbar 함수를 사용하여 "th1"과 "th2"라는 두 개의 트랙바를 생성한다. 이 트랙바들을 이용하여 임계값을 조정할 수 있다. 여기서 255는 maxVal로 0부터 255 사이에서 값을 가질 수 있도록 설정하는 것이다. 트랙바의 값이 변경될 때 onTrackbar1과 onTrackbar2 함수를 호출한다.

```
void onTrackbar1(int , void*) {
    Canny(gray, edge, th1,th1*2 ,3);
    imshow("canny edge", edge);
}
void onTrackbar2(int, void*) {
    Canny(gray, edge, th2,th2*2, 3);
    imshow("canny edge", edge);
}
```

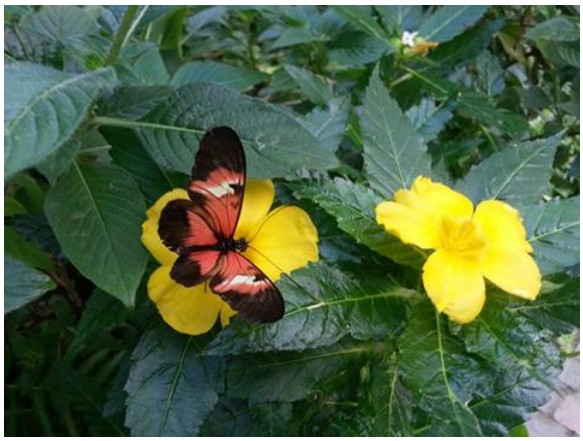
onTrackbar1과 onTrackbar2 함수를 정의해주었다. 이 함수들은 Canny 함수를 호출하여 gray 이미지에서 edge를 검출한다. edge를 검출하는데 th1과 th2보다 높은 값의 픽셀은 강한 edge로 분류되고, 두 번째 임계값인 th1*2와 th2*2보다 낮은 값의 픽셀은 약한 edge로 분류된다. 세 번째 매개변수는 edge 검출에 사용되는 커널 크기를 말한다.

onTrackbar1 함수는 수평 방향의 edge를 검출하고, onTrackbar2는 수직 방향의 edge를 검출해준다. 각각의 th1과 th2 변수들은 해당 방향들의 edge를 이미지에서 검출한다.

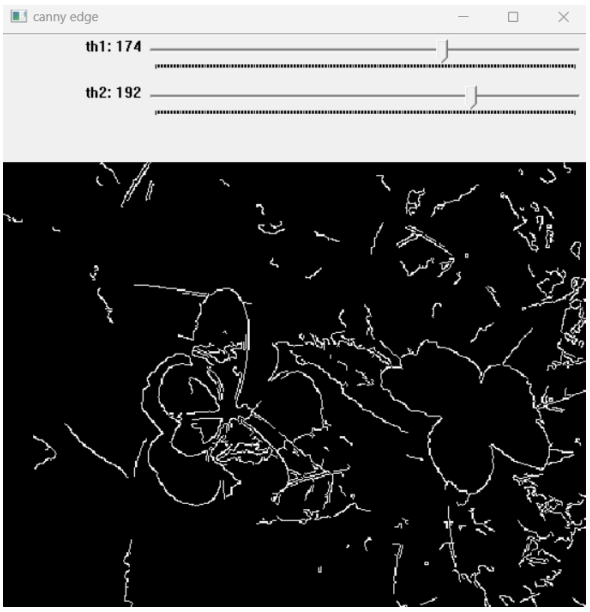
imshow함수를 사용하여 'canny edge"라는 윈도우에 "edge" 이미지를 표시해준다. 여기서 윈도우는 main문에서 작성한 namedWindow함수로 생성된 것이다.

여기서 Canny는 OpenCV에서 정의된 함수를 활용하였다.

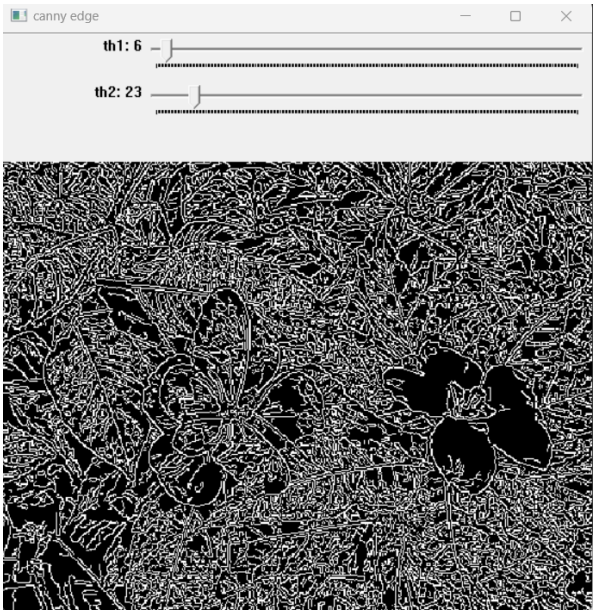
1.1.6. 영상처리 결과



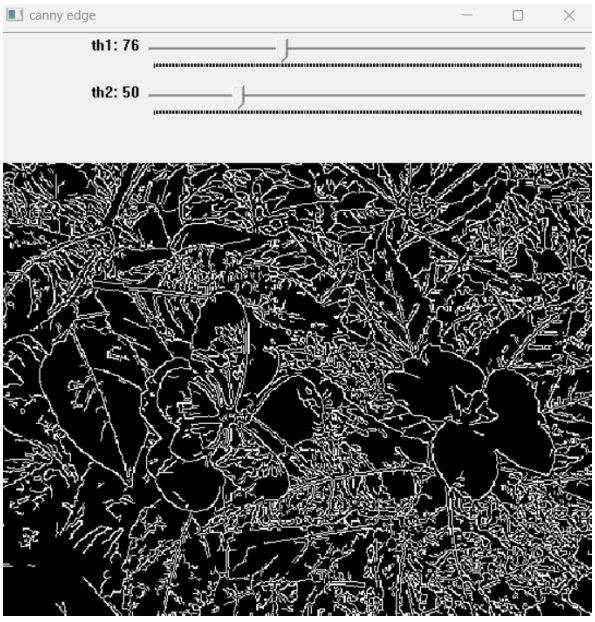
원본 이미지



#th1은 174, th2는 192로 edge 검출



th1은 6, th2는 23



th1은 76, th2는 50

각각의 임계값을 조정하였을 때 edge의 검출을 확인할 수 있다.

1.2 연습문제 18

동전 영상에서 동전 객체를 검출하기 위한 전처리를 다음과 같이 수행하시오. 1) 명암도 영상 변환 -> 2) 가우시안 블러링 -> 3) 이진화 -> 4) 모폴로지 열림 연산

해당 문제는 이미지를 불러온 후 명암도 영상 그러니까 그레이화를 해준 후 가우시안 블러링을

적용해준다. 그리고 이진화를 통해서 0과 255의 값을 가지게 한 후 모폴로지 열림 연산을 수행한다.

1.2.1. 명암도 영상 변환

명암도 영상의 변환의 경우 이미지를 불러올 때 IMREAD_GRAYCOLOR로 불러와도 되고 컬러로 불러드린 이미지를 cvtColor를 활용해 명암도 영상으로 변경할 수 있다.

1.2.2. 가우시안 블러링

가우시안 블러링은 위의 17번 문제에서 서술하였기에 생략한다.

1.2.3. 이진화

이진화란 픽셀의 값을 0 또는 255의 값으로 변환하는 연산을 말한다. threshold함수를 활용하여 최소값, 최대값을 지정한 후 THRESH_BINARY를 통해 이진화를 시켜주는 것이 일반적 방법이다.

1.2.4. 모폴로지 열림 연산

모폴로지는 영상의 객체들의 형태를 분석하고 처리하는 기법이다. 영상의 경계, 골격, 블록 등의 형태를 표현하는데 필요한 요소를 추출한다. 정리하자면, 영상 내에 존재하는 객체의 형태를 조금씩 변환시킴으로써 영상 내에서 불필요한 잡음을 제거하거나 객체를 뚜렷하게 하여 필요한 요소를 추출한다.

열림 연산은 침식 연산을 먼저 수행하고, 바로 팽창 연산을 수행하는 것을 말한다. 침식 연산은 객체의 크기는 축소되고 배경은 확장되어 노이즈 제거에 유용하다. 팽창 연산은 객체의 크기는 확대되고 배경은 축소된다. 따라서 열림 연산은 침식 연산으로 인해 객체는 축소되고, 배경 부분의 잡음들은 제거된다. 그리고 축소되었던 객체들이 팽창 연산으로 인해서 다시 원래 크기로 돌아간다.

1.2.5. 코드

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

using namespace std;
using namespace cv;

int main() {
	Mat img = imread("coins.jpg", 1);
	CV_Assert(img.data);

	// 1) 명암도 영상 변환
	Mat gray;
	cvtColor(img, gray, COLOR_BGR2GRAY);
```

```
        // 2) 가우시안 블러링
        Mat blur;
        GaussianBlur(gray, blur, Size(3, 3), 0);

        // 3) 이진화
        Mat binary;
        threshold(blur, binary, 55, 255, THRESH_BINARY); //60을 해보니 원 안에 점 생김

        // 4) 모폴로지 열림 연산
        Mat morp;
        Matx <uchar, 3, 3> mask;
        mask << 0, 1, 0,
                1, 1, 1,
                0, 1, 0;
        morphologyEx(binary, morp, MORPH_OPEN, mask);

        imshow("img", img);
        imshow("gray", gray);
        imshow("blur", blur);
        imshow("binary", binary);
        imshow("mophology", morp);

        waitKey(0);
        return 0;
}
```

전체 코드

OpenCV에 내장되어 있는 함수들을 사용하여 구현하였다.

```
        Mat img = imread("coins.jpg", 1);
        CV_Assert(img.data);

        // 1) 명암도 영상 변환
        Mat gray;
        cvtColor(img, gray, COLOR_BGR2GRAY);
```

img객체에 imread를 통해 이미지를 컬러 영상으로 불러온 후 저장한다. 그리고 gray 객체에 cvtColor에 COLOR_BGR2GRAY를 입력하여 영상을 그레이화 해주었다.

```
// 2) 가우시안 블러링
        Mat blur;
        GaussianBlur(gray, blur, Size(3, 3), 0);
```

OpenCV 내장함수 GaussianBlur를 통해 노이즈를 제거해준다. Size(3,3)은 블러 커널의 크기를 나타내고, 마지막 0은 표준 편차를 의미한다. 0을 설정하면 자동으로 표준편차를 계산해준다.

```
      // 3) 이진화
        Mat binary;
        threshold(blur, binary, 55, 255, THRESH_BINARY); //60을 해보니 원 안에 점 생김
```

이진화로 threshold 함수를 이용하여 blur 영상을 이진화해준다. 임계값을 55로 설정해주었는데

60을 넘어가면 영상 내 잡음 또는 동전이 일그러진 형태가 된다. 이진화한 영상을 binary에 저장해준다.

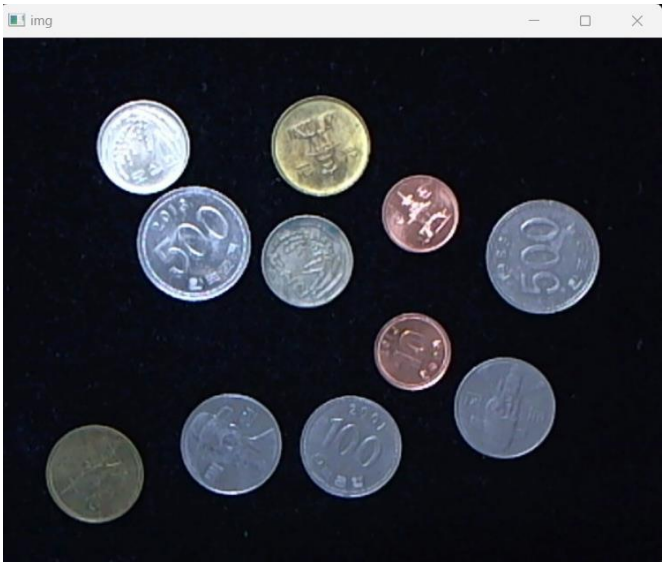
```
// 4) 모폴로지 열림 연산
        Mat morp;
        Matx <uchar, 3, 3> mask;
        mask << 0, 1, 0,
                1, 1, 1,
                0, 1, 0;
        morphologyEx(binary, morp, MORPH_OPEN, mask);
```

morphologyEx 함수를 사용하여 binary 영상에 모폴로지 열림 연산을 수행해준다. MORPH_OPEN은 모폴로지 열림을 의미한다. 이 연산을 수행하면 잡음을 제거하여 동전의 형태를 더 뚜렷하게 해준다. Mask에는 3*3 크기의 마스크를 사용하여 열림 연산을 수행해준다. Morp에 연산 결과를 저장해준다.

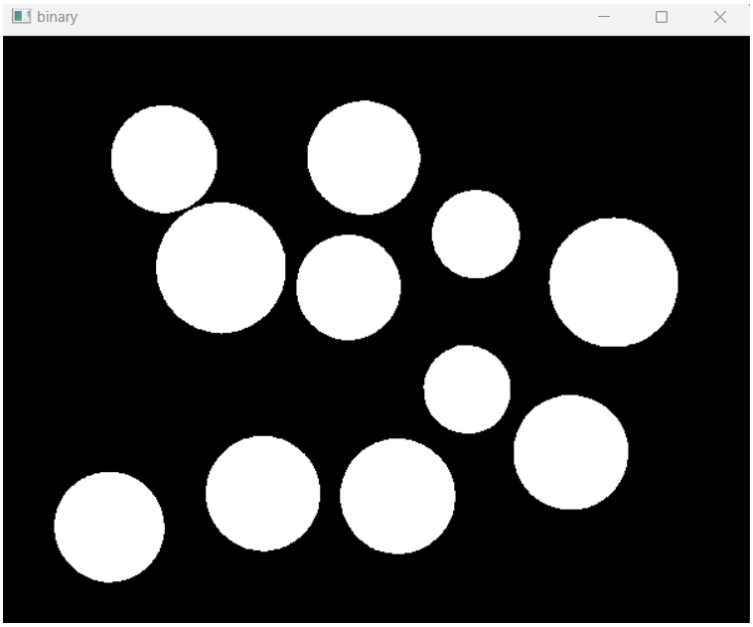
```
        imshow("img", img);
        imshow("gray", gray);
        imshow("blur", blur);
        imshow("binary", binary);
        imshow("mophology", morp);
```

imshow함수들을 이용하여 영상의 결과를 확인한다.

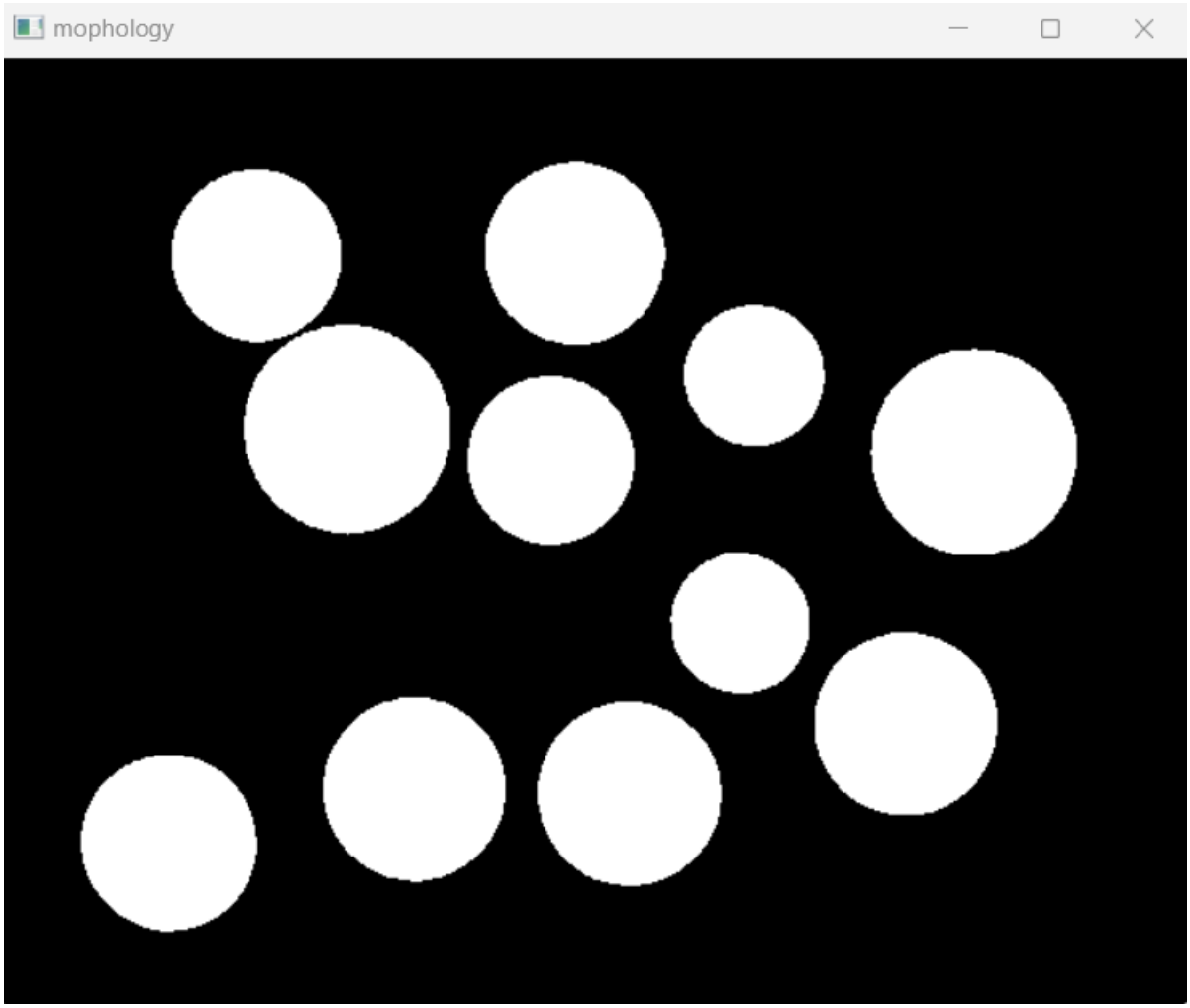
1.2.6. 영상처리 결과



#원본 이미지와 명암도 영상 변환을 해준 영상



가우시안 블러링 영상 (좌), 이진화 영상 (우)



모폴로지 열림 연산 영상

1.3 연습문제 19

예제 _7.4.1과 예제 _7.4.2에서 구현한 침식 연산과 팽창 연산 함수 erosion()와 dilation()는 소스 내용이 거의 동일하다. 두 함수를 참고하여 하나로 통일해서 morphology()함수로 구현하시오.

```
void erosion(Mat img, Mat& dst, Mat mask) {
        dst = Mat(img.size(), CV_8U, Scalar(0));
        if (mask.empty()) mask = Mat(3, 3, CV_8UC1, Scalar(1));

        Point h_m = mask.size() / 2;
        for (int i = h_m.y; i < img.rows-h_m.y; i++) {
                for (int j = h_m.x; j < img.cols-h_m.x; j++) {
                        Point start = Point(j, i) - h_m;
                        bool check = check_match(img, start, mask, 0);
                        dst.at<uchar>(i, j) = (check) ? 255 : 0;
                }
        }
}
```

#예제 7.4.1의 erosion() 함수 : 입력 이미지에 대해 침식 연산을 수행

```
void dilation(Mat img, Mat& dst, Mat mask) {
        dst = Mat(img.size(), CV_8U, Scalar(0));
        if (mask.empty()) mask = Mat(3, 3, CV_8UC1, Scalar(1));

        Point h_m = mask.size() / 2;
        for (int i = h_m.y; i < img.rows - h_m.y; i++) {
                for (int j = h_m.x; j < img.cols - h_m.x; j++) {
                        Point start = Point(j, i) - h_m;
                        bool check = check_match(img, start, mask, 1);
                        dst.at<uchar>(i, j) = (check) ? 0 : 255;
                }
        }
}
```

#예제 7.4.2의 dilation() 함수 : 입력 이미지에 팽창 연산을 수행

침식과 팽창의 설명은 위의 18에서 간략히 설명했기에 생략한다.

1.3.1. 코드

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

using namespace std;
using namespace cv;

bool check_match(Mat img, Point start, Mat mask, int mode = 0) {
```

```
        for (int u = 0; u < mask.rows; u++) {
                for (int v = 0; v < mask.cols; v++) {
                        Point pt(v, u);
                        int m = mask.at<uchar>(pt);
                        int p = img.at<uchar>(start + pt);

                        bool ch = (p == 255);
                        if (m == 1 && ch == mode) return false;
                }
        }
        return true;
}
void erosion(Mat img, Mat& dst, Mat mask) {
        dst = Mat(img.size(), CV_8U, Scalar(0));
        if (mask.empty()) mask = Mat(3, 3, CV_8UC1, Scalar(1));

        Point h_m = mask.size() / 2;
        for (int i = h_m.y; i < img.rows-h_m.y; i++) {
                for (int j = h_m.x; j < img.cols-h_m.x; j++) {
                        Point start = Point(j, i) - h_m;
                        bool check = check_match(img, start, mask, 0);
                        dst.at<uchar>(i, j) = (check) ? 255 : 0;
                }
        }
}
void dilation(Mat img, Mat& dst, Mat mask) {
        dst = Mat(img.size(), CV_8U, Scalar(0));
        if (mask.empty()) mask = Mat(3, 3, CV_8UC1, Scalar(1));

        Point h_m = mask.size() / 2;
        for (int i = h_m.y; i < img.rows - h_m.y; i++) {
                for (int j = h_m.x; j < img.cols - h_m.x; j++) {
                        Point start = Point(j, i) - h_m;
                        bool check = check_match(img, start, mask, 1);
                        dst.at<uchar>(i, j) = (check) ? 0 : 255;
                }
        }

}
void morpholy(Mat img, Mat& dst, Mat mask, bool isErosion) {
        dst = Mat(img.size(), CV_8U, Scalar(0));
        if (mask.empty()) mask = Mat(3, 3, CV_8UC1, Scalar(1));

        Point h_m = mask.size() / 2;
        for (int i = h_m.y; i < img.rows - h_m.y; i++) {
                for (int j = h_m.x; j < img.cols - h_m.x; j++) {
                        Point start = Point(j, i) - h_m;
                        bool check = check_match(img, start, mask, isErosion ?0:1);
                        dst.at<uchar>(i, j) = (check) ? (isErosion ? 255 : 0) : (isErosion ?
0:255);
                }
        }
}
int main() {
```

```
        Mat img = imread("morph_test.jpg", 0);
        CV_Assert(img.data);
        Mat th_img, dst1, dst2,dst3,dst4;
        threshold(img, th_img, 128, 255, THRESH_BINARY);

        uchar data[] = { 0,1,0,
                                        1,1,1,
                                        0,1,0 };
        Mat mask(3, 3, CV_8UC1, data);
        erosion(th_img, dst1, (Mat)mask);
        dilation(th_img, dst2, (Mat)mask);

        morpholy(th_img, dst3, (Mat)mask, true);//Erosion
        morpholy(th_img, dst4, (Mat)mask, false);//Dilation

        imshow("img", img);
        imshow("erosion", dst1);
        imshow("dilation", dst2);

        imshow("m_erosion", dst3);
        imshow("m_dilation", dst4);
        waitKey(0);
        return 0;

}
```

#전체 코드

비교를 위해 erosion()함수와 dilation 함수도 구현하였다. 해당 코드에서는 OpenCV에 내장된 함수가 아닌 직접 회선을 수행하는 과정을 담고 있다.

```
bool check_match(Mat img, Point start, Mat mask, int mode = 0) {
        for (int u = 0; u < mask.rows; u++) {
                for (int v = 0; v < mask.cols; v++) {
                        Point pt(v, u);
                        int m = mask.at<uchar>(pt);
                        int p = img.at<uchar>(start + pt);

                        bool ch = (p == 255);
                        if (m == 1 && ch == mode) return false;
                }
        }
        return true;
}
```

해당 check_match 함수는 마스크 원소와 마스크 범위 입력화소 간의 일치 여부를 확인하는 함수이다. 특정 픽셀과 마스크의 해당 위치에서의 값이 일치하는지 확인하고 일치 여부에 따라 mode 값에 따라 true와 false를 반환한다. Erosion과 dilation 모두 활용되었고 morpholy 함수에도 이용된다.

```
void morpholy(Mat img, Mat& dst, Mat mask, bool isErosion) {
```

```
        dst = Mat(img.size(), CV_8U, Scalar(0));
        if (mask.empty()) mask = Mat(3, 3, CV_8UC1, Scalar(1));

        Point h_m = mask.size() / 2;
        for (int i = h_m.y; i < img.rows - h_m.y; i++) {
                for (int j = h_m.x; j < img.cols - h_m.x; j++) {
                        Point start = Point(j, i) - h_m;
                        bool check = check_match(img, start, mask, isErosion ?0:1);
                        dst.at<uchar>(i, j) = (check) ? (isErosion ? 255 : 0) : (isErosion ?
0:255);
                }
        }
}
```

Morpholy 함수는 입력 이미지에 모폴로지 변환 연산을 수행한다. isErosion이 true인 경우 침식 연산을 수행하고 false인 경우 팽창 연산을 수행하게 작성하였다.

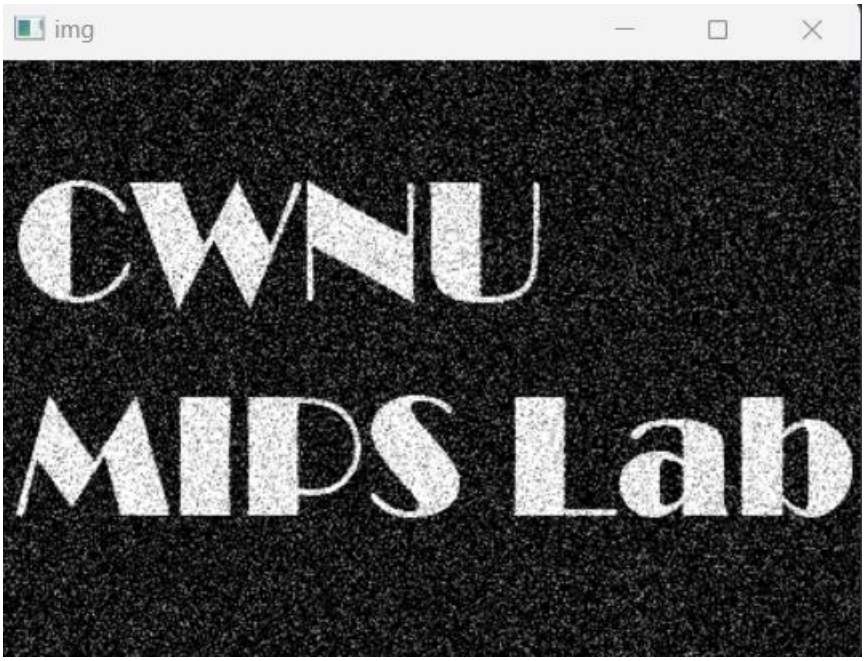
```
int main() {
        Mat img = imread("morph_test.jpg", 0);
        CV_Assert(img.data);
        Mat th_img, dst1, dst2,dst3,dst4;
        threshold(img, th_img, 128, 255, THRESH_BINARY);

        uchar data[] = { 0,1,0,
                                        1,1,1,
                                        0,1,0 };
        Mat mask(3, 3, CV_8UC1, data);
        erosion(th_img, dst1, (Mat)mask);
        dilation(th_img, dst2, (Mat)mask);

        morpholy(th_img, dst3, (Mat)mask, true);//Erosion
        morpholy(th_img, dst4, (Mat)mask, false);//Dilation

        imshow("img", img);
        imshow("erosion", dst1);
        imshow("dilation", dst2);

        imshow("m_erosion", dst3);
        imshow("m_dilation", dst4);
        waitKey(0);
        return 0;
}
```

Main 문에서는 imread를 통해 이미지를 명암도 영상으로 가져온다. 그리고 threshold함수를 이용해 이진화 영상으로 만들어 th_img에 저장한다.

그리고 data[] 배열에 마스크를 저장한다. 그리고 Mat으로 선언된 mask에 데이터를 입력해주고 morpholy 함수를 호출하는데 true, false를 구분하여 각각 erosion과 dilation을 수행하여 dst3와 dst4에 저장해준다.

Imshow를 해줄 때는 m_erosion, m_dilation으로 표시하여 예제 7.4.1과 7.4.2에 구현된 결과와 비교하여 준다.

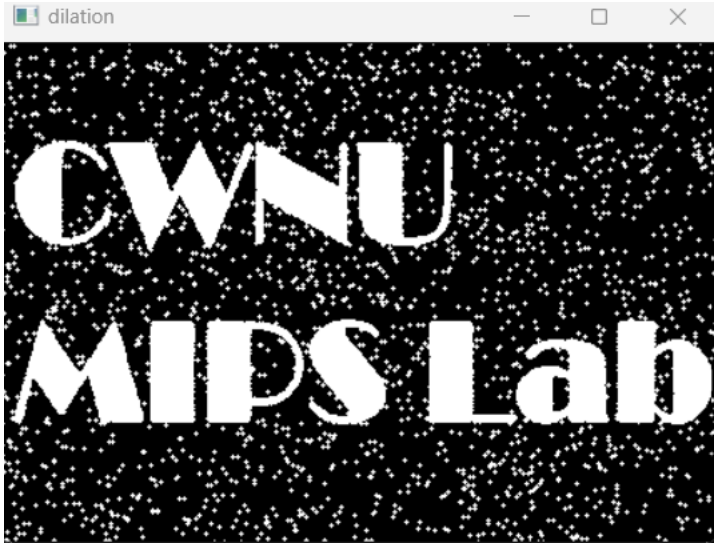
1.3.2. 영상처리 결과



원본 이미지 : 배경과 객체에 노이즈가 많이 분포

예제 7.4.1의 erosion() (좌), morphology()로 구현한 erostion

- 원본 영상과 비교하여 배경의 노이즈가 제거된 것을 확인할 수 있다. 객체 내부의 노이즈는 증가하였다.



예제 7.4.2의 dilation() (좌), morphology()로 구현한 dilation

- 원본 영상과 비교하여 배경의 노이즈는 증가하였다. 객체 내부의 노이즈는 감소한 것을 확인 할 수 있다.

3. 결론 및 기타

OpenCV를 활용하여 캐니edge 알고리즘과 트랙바 작성과 가우시안 블러링, 이진화, 모폴로지 등의 영상처리 활용법을 혼자서 학습하는 시간이 되었던 거 같다. OpenCV를 활용하니 간단하게 코드로 구현할 수 있다는 장점이 있다는 걸 느꼈다. 회선 같은 부분은 사용자가 직접 구현을 해야 한다는 필요성도 느끼게 된 거 같다. 과제를 통해 영상처리의 모폴로지 변환에 대해서는 더 자세하게 알 수 있는 시간이 되었던 거 같다.