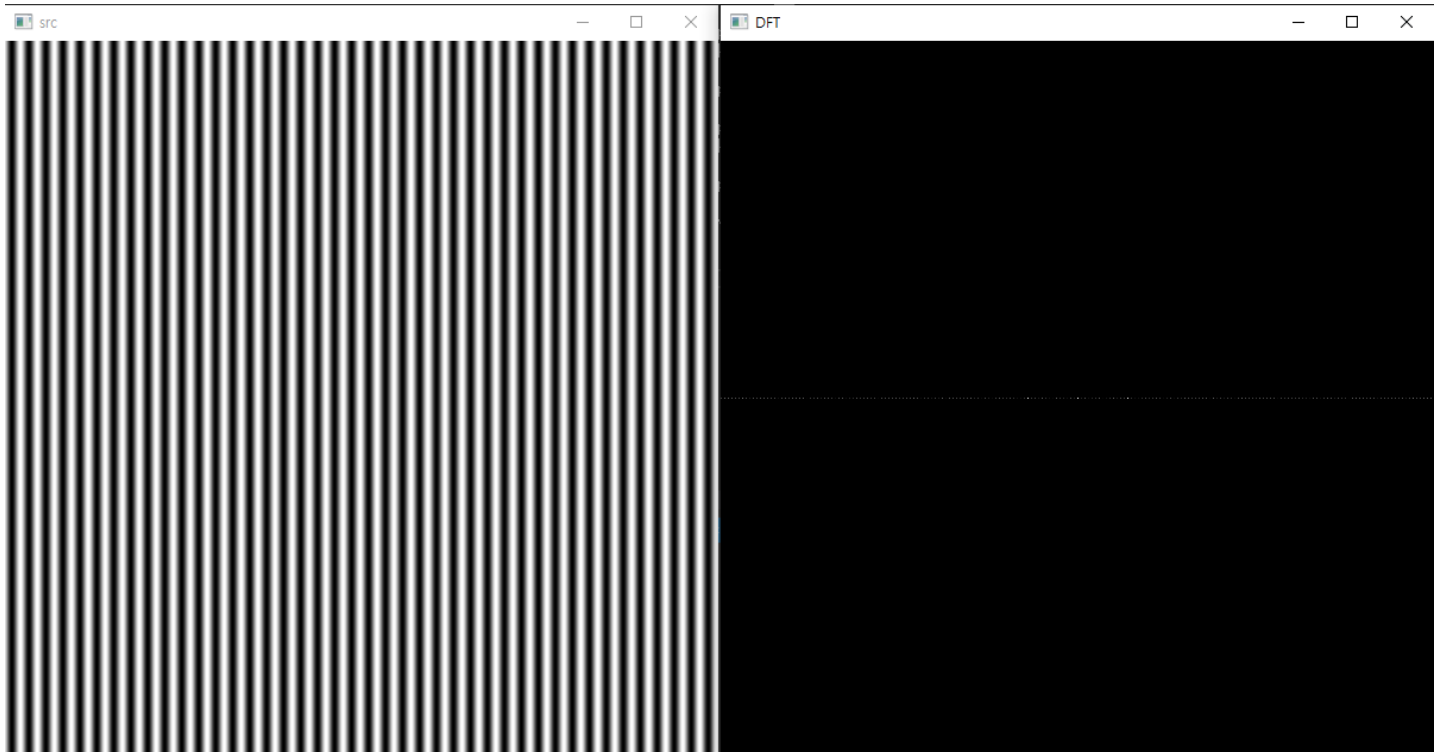
영상처리 실제 - 10주차 과제

2023254009 최현동

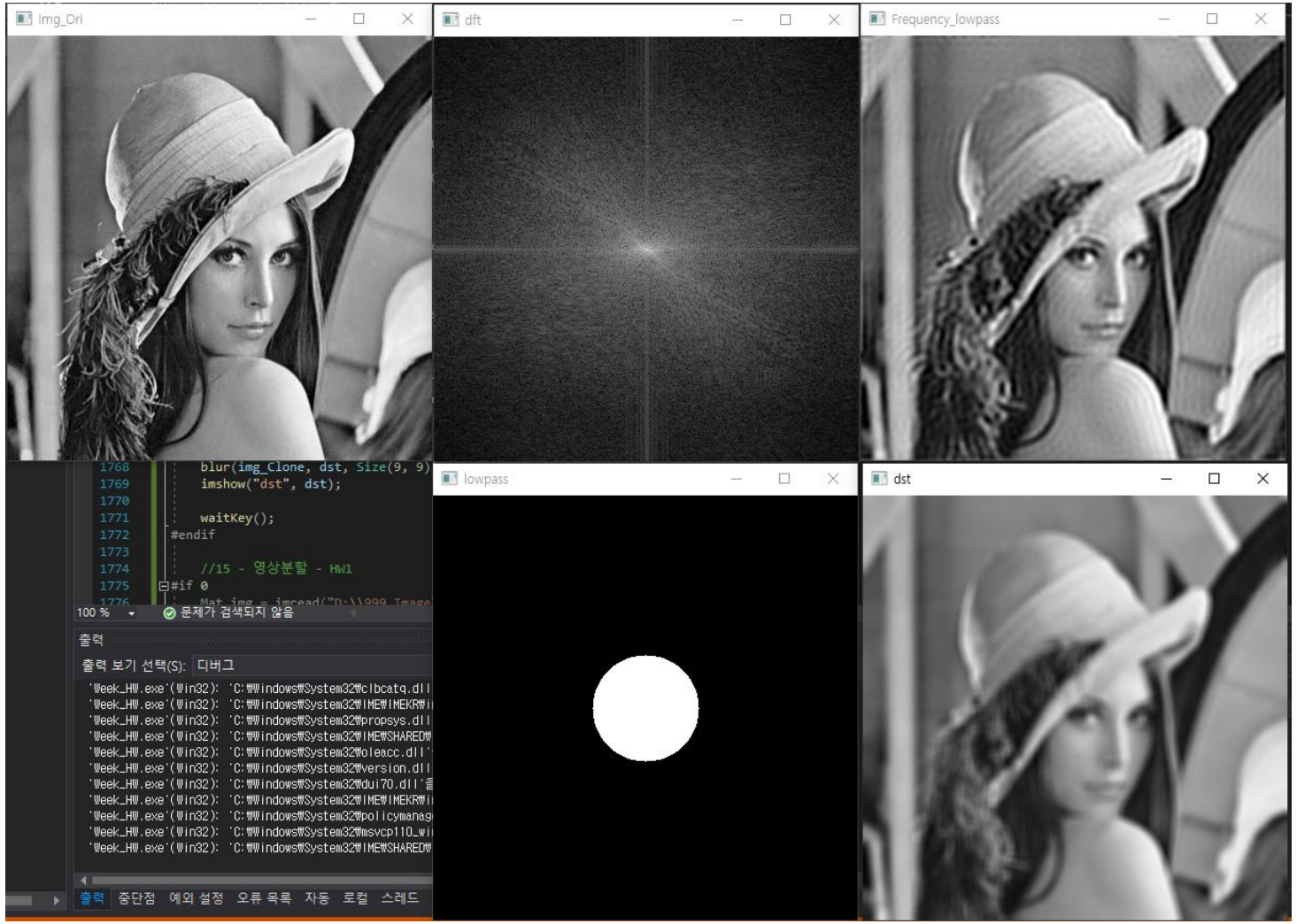
: 14 – 주파수영상처리 – HW1

```
void displayDFT(Mat& src)
{
    Mat image_array[2] = { Mat::zeros(src.size(), CV_32F), Mat::zeros(src.size(), CV_32F) };
    // ① DFT 결과 영상을 2개의 영상으로 분리한다.
    split(src, image_array);

    Mat mag_image;
    // ② 푸리에 변환 계수들의 절대값을 계산한다.
    magnitude(image_array[0], image_array[1], mag_image);

    // ③ 푸리에 변환 계수들은 상당히 크기 때문에 로그 스케일로 변환한다.
    // 0값이 나오지 않도록 1을 더해준다.
    mag_image += Scalar::all(1);
    log(mag_image, mag_image);

    // ④ 0에서 255로 범위로 정규화한다.
    normalize(mag_image, mag_image, 0, 1, NORM_MINMAX);
    imshow("DFT", mag_image);
}

void shuffleDFT(Mat& src)
{
    int cX = src.cols / 2;
    int cY = src.rows / 2;

    Mat q1(src, Rect(0, 0, cX, cY));
    Mat q2(src, Rect(cX, 0, cX, cY));
    Mat q3(src, Rect(0, cY, cX, cY));
    Mat q4(src, Rect(cX, cY, cX, cY));

    Mat tmp;
    q1.copyTo(tmp);
    q4.copyTo(q1);
    tmp.copyTo(q4);
    q2.copyTo(tmp);
    q3.copyTo(q2);
    tmp.copyTo(q3);
}
    //14 - 주파수영역 처리 - HW1
#if 1
    Mat src = imread("D:\\999.Image\\image.jpg", IMREAD_GRAYSCALE);
    Mat src_float;

    // 그레이스케일 영상을 실수 영상으로 변환한다.
    src.convertTo(src_float, CV_32FC1, 1.0 / 255.0);
    Mat dft_image;
    dft(src_float, dft_image, DFT_COMPLEX_OUTPUT);
    shuffleDFT(dft_image);
    displayDFT(dft_image);
    imshow("src", src);
    waitKey();
#endif
```



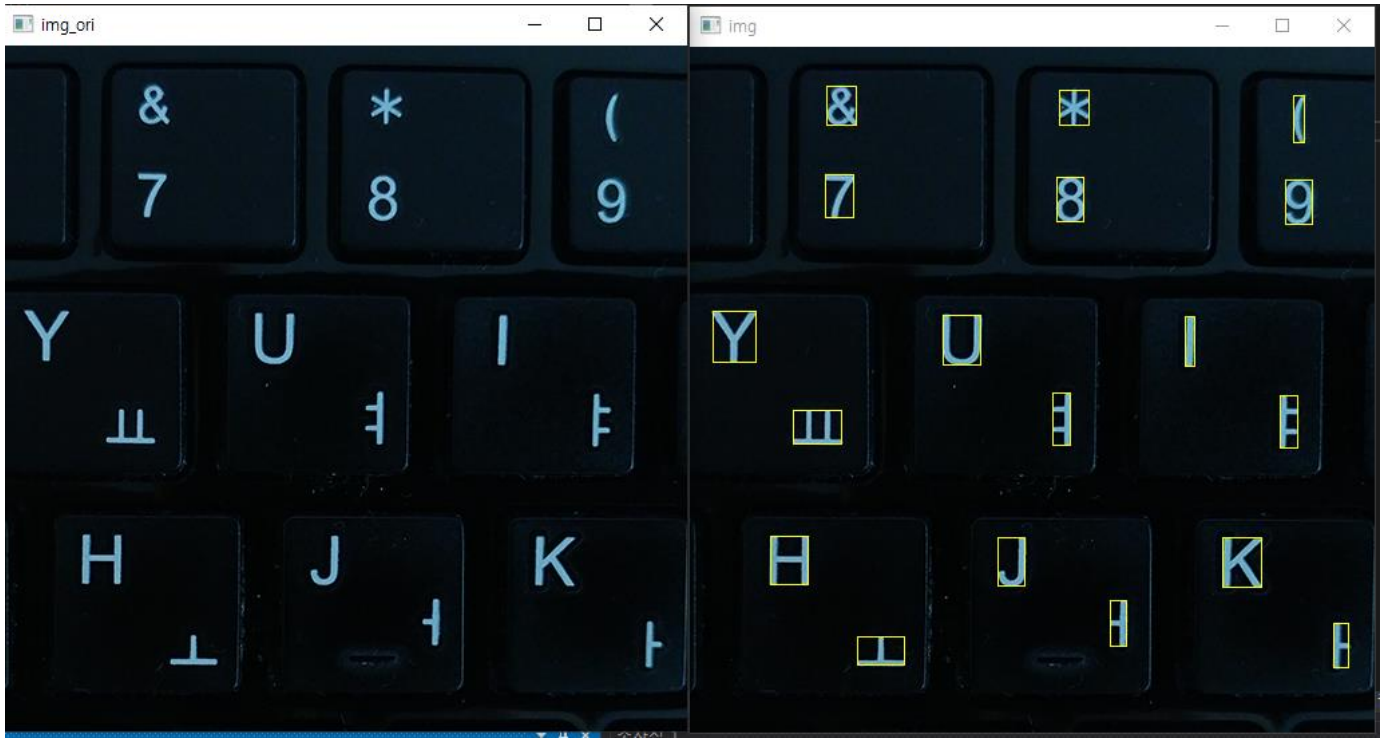
: 14 – 주파수영상처리 – HW2

```
    //14 - 주파수영역 처리 - HW2
#if 1
    Mat img = imread("D:\\999.Image\\lenna.jpg", IMREAD_GRAYSCALE);
    imshow("Img_Ori", img);
    Mat img_Clone;
    img_Clone = img.clone();

    Mat src_float;
    img.convertTo(src_float, CV_32FC1, 1.0 / 255.0);
    Mat dft_image;
    dft(src_float, dft_image, DFT_COMPLEX_OUTPUT);
    shuffleDFT(dft_image);
    displayDFT(dft_image, "dft");

    Mat lowpass = getFilter_Circle(dft_image.size());
    displayDFT(lowpass, "lowpass");
    Mat result;

    multiply(dft_image, lowpass, result);

    Mat inverted_image;
    shuffleDFT(result);
    idft(result, inverted_image, DFT_SCALE | DFT_REAL_OUTPUT);
    imshow("Frequency_lowpass", inverted_image);


    Mat dst;
    blur(img_Clone, dst, Size(9, 9));
    imshow("dst", dst);

    waitKey();
#endif
```



: 15 – 영상분할 – HW1

```cpp
//15 - 영상분할 - HW1
#if 1
Mat img = imread("D:\\999.Image\\keyboard.bmp", IMREAD_COLOR);
if (img.empty())
{
    cerr << "Image load failed!" << endl;
    return -1;
}
imshow("img_ori", img);
Mat gray;
cvtColor(img, gray, COLOR_BGR2GRAY);

GaussianBlur(gray, gray, Size(3, 3), 0, 0, BORDER_DEFAULT);

Mat bin;
threshold(gray, bin, 200, 255, THRESH_BINARY | THRESH_OTSU);

vector<vector<Point>> contours;
findContours(bin, contours, RETR_EXTERNAL, CHAIN_APPROX_SIMPLE);

for (int i = 0; i < (int)contours.size(); i++)
{
    Rect  mr = boundingRect(contours[i]);

    rectangle(img, mr, Scalar(0, 255, 255), 1);
}

imshow("img", img);
waitKey();
#endif
```



: 15 – 영상분할 – HW2

```cpp
    //15 - 양상분할 - HW2
#if 1
    Mat img = imread("D:\\999.Image\\shape.bmp", IMREAD_COLOR);
    if (img.empty())
    {
        cerr << "Image load failed!" << endl;
        return -1;
    }
    Mat img_Clone;
    img_Clone = img.clone();

    Mat gray;
    cvtColor(img_Clone, gray, COLOR_BGR2GRAY);

    Mat bin;
    threshold(gray, bin, 250, 255, THRESH_BINARY_INV);

    vector<vector<Point>> contours;
    findContours(bin, contours, RETR_EXTERNAL, CHAIN_APPROX_SIMPLE);

    drawContours(img_Clone, contours, -1, Scalar(0, 0, 0), 3);
    imshow("img", img);
    imshow("img_Clone", img_Clone);
    waitKey();
#endif
```