

영상처리 실제 10주차 실습_주파수영역처리

2023254015 장욱진

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

using namespace std;
using namespace cv;

void displayDFT(Mat& src)
{
    Mat image_array[2] = { Mat::zeros(src.size(), CV_32F),
    Mat::zeros(src.size(), CV_32F) };

    split(src, image_array);
    Mat mag_image;

    magnitude(image_array[0], image_array[1], mag_image);

    mag_image += Scalar::all(1);
    log(mag_image, mag_image);

    normalize(mag_image, mag_image, 0, 1, CV_MINMAX);
    imshow("DFT", mag_image);
    waitKey(0);
}

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);
}

Mat getFilter_21(Size size)
{
    Mat tmp = Mat(size, CV_32F);
    for (int i = 0; i < tmp.rows; i++) {
        for (int j = 0; j < tmp.cols; j++) {
            if (j > (tmp.cols / 2 - 10) && j < (tmp.cols / 2 + 10) && i
            > (tmp.rows / 2 + 10)) tmp.at<float>(i, j) = 0;
        }
    }
}
```

```

        else if (j > (tmp.cols / 2 - 10) && j < (tmp.cols / 2 + 10) && i
< (tmp.rows / 2 - 10)) tmp.at<float>(i, j) = 0;
        else tmp.at<float>(i, j) = 1;
    }
}
Mat toMerge[] = { tmp, tmp };
Mat filter;
merge(toMerge, 2, filter);
return filter;
}

```

```

Mat getFilter_17(Size size)
{
    Mat filter(size, CV_32FC2, Vec2f(0, 0));
    circle(filter, size / 2, 50, Vec2f(1, 1), -1);
    return filter;
}

```

```

int page17()
{
    Mat src = imread("./lenna.jpg", IMREAD_GRAYSCALE);
    Mat src_float;
    imshow("original", src);

    src.convertTo(src_float, CV_32FC1, 1.0 / 255.0);
    Mat dft_image;
    dft(src_float, dft_image, DFT_COMPLEX_OUTPUT);
    shuffleDFT(dft_image);
    Mat lowpass = getFilter_17(dft_image.size());
    Mat result;

    multiply(dft_image, lowpass, result);
    displayDFT(result);
    Mat inverted_image;
    shuffleDFT(result);
    idft(result, inverted_image, DFT_SCALE | DFT_REAL_OUTPUT);
    imshow("inverted", inverted_image);
    waitKey(0);
    return 1;
}

```

```

int page21_26()
{
    Mat src = imread("./lunar.png", IMREAD_GRAYSCALE);
    Mat src_float, dft_image;
    imshow("original", src);

    src.convertTo(src_float, CV_32FC1, 1.0 / 255.0);
    dft(src_float, dft_image, DFT_COMPLEX_OUTPUT);
    shuffleDFT(dft_image);
}

```

```

    displayDFT(dft_image);
    Mat lowpass = getFilter_21(dft_image.size());
    Mat result;

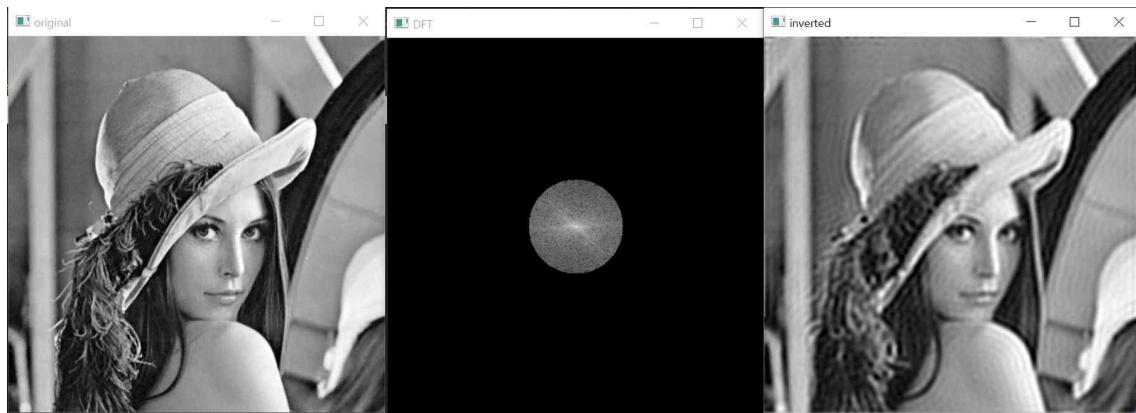
    multiply(dft_image, lowpass, result);
    displayDFT(result);
    Mat inverted_image;
    shuffleDFT(result);
    idft(result, inverted_image, DFT_SCALE | DFT_REAL_OUTPUT);
    imshow("inverted", inverted_image);
    waitKey(0);
    return 1;
}

int main()
{
    page17();
    page21_26();

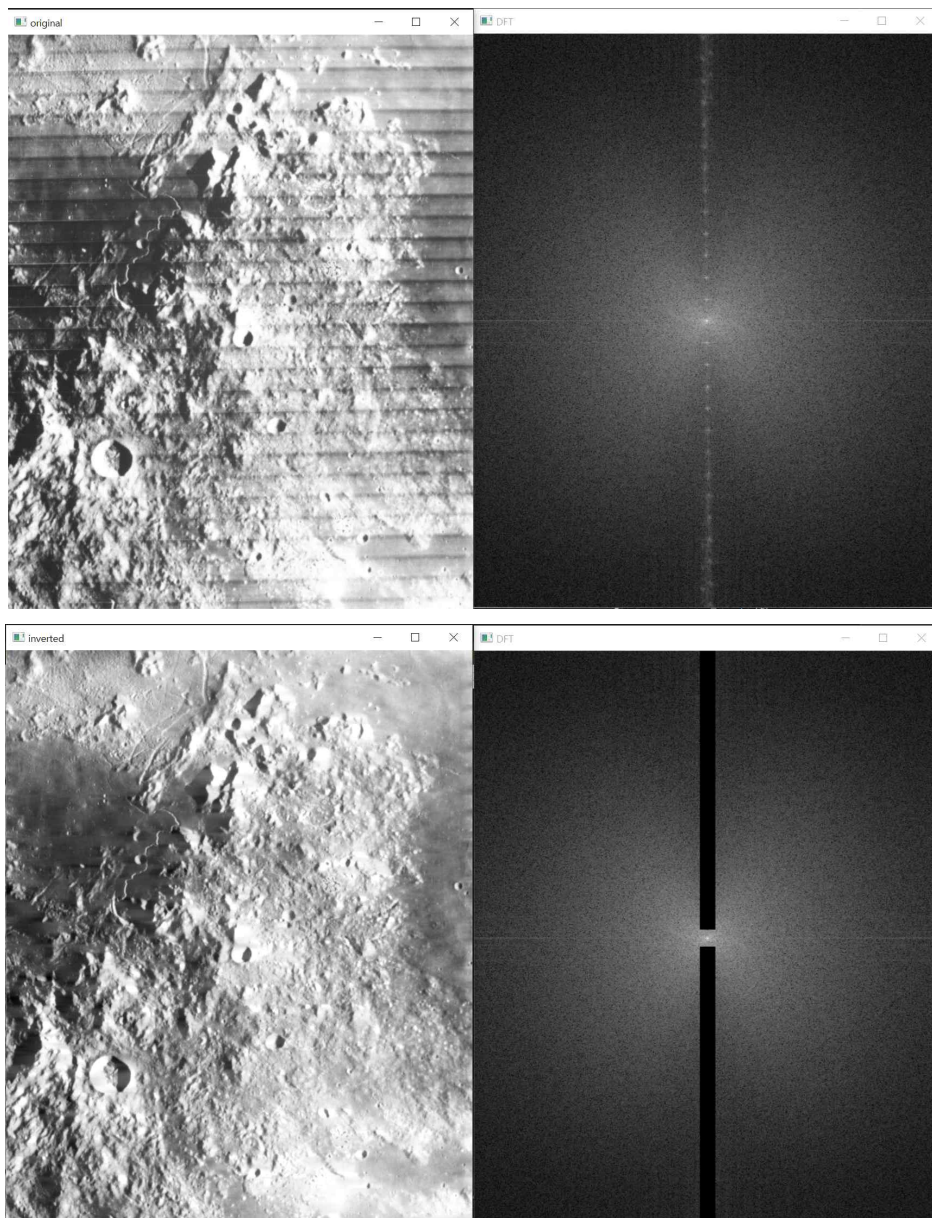
    return 0;
}

```

결과화면



<page17 결과화면>



<page21-26 결과화면>