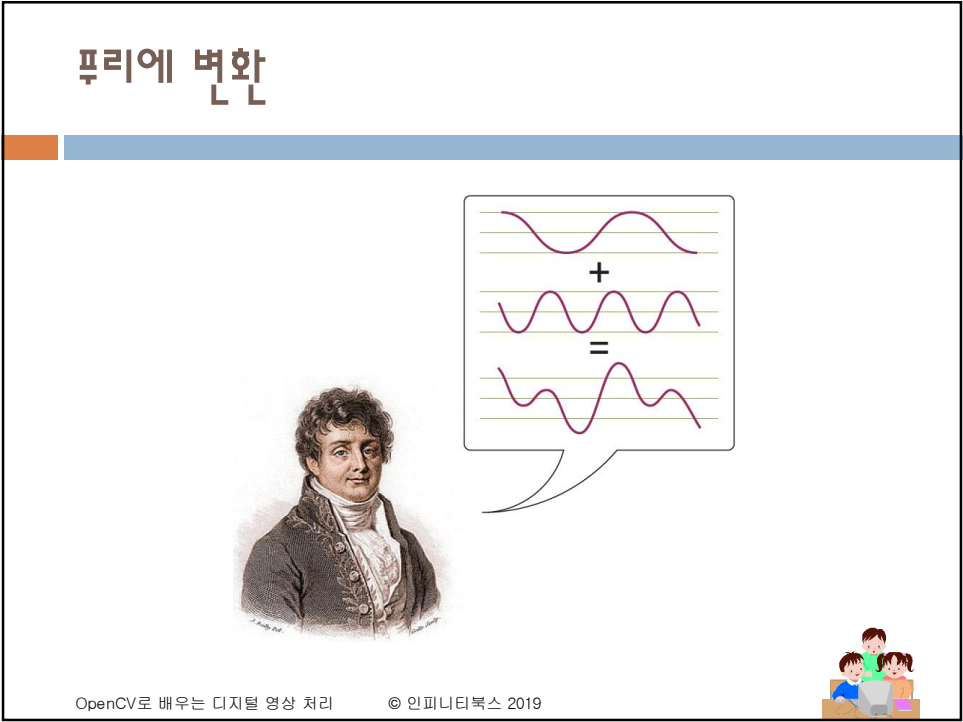
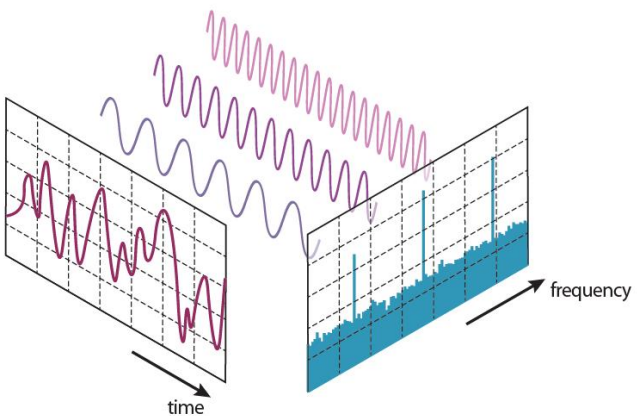


1




2

푸리에 변환



출처: 위키 백과

OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019



3

공간 주파수

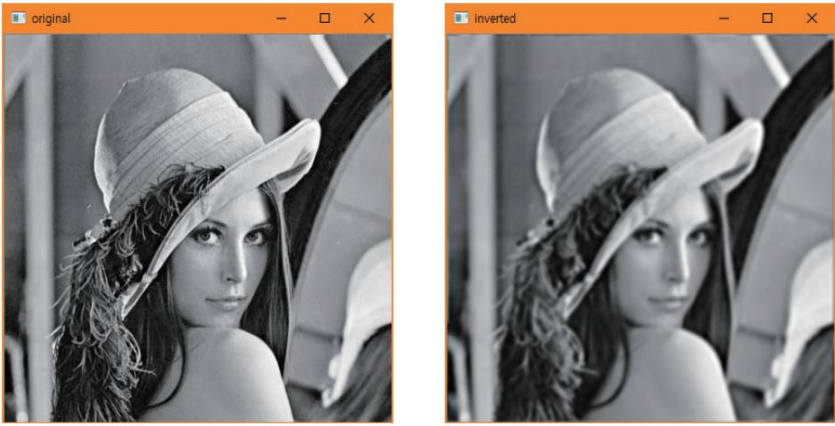


OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019



4

고주파와 저주파

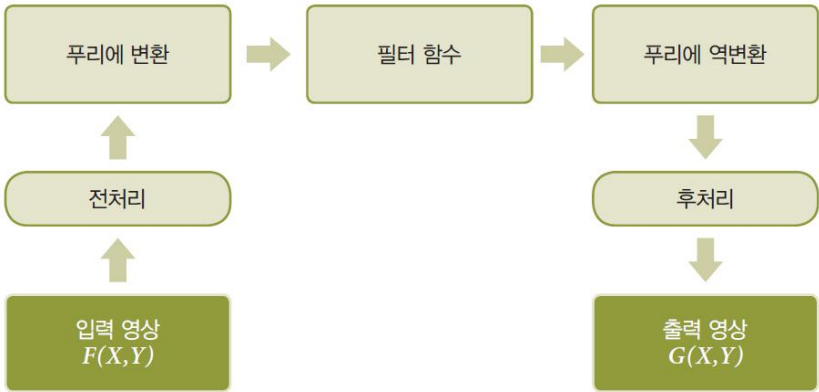


OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019



5

주파수 영역 처리



OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019



6

푸리에 변환

The diagram illustrates the Fourier Transform process. On the left, a complex, jagged signal is shown. An arrow labeled 'FT' points to a stack of several sine waves of different frequencies and amplitudes. To the right, three individual sine waves are shown, each labeled $a(x)$ on the vertical axis and x on the horizontal axis. Arrows from these sine waves point to a summation symbol Σ . From the summation symbol, an arrow points to a reconstructed signal graph, which is a smooth curve labeled $a(x)$ on the vertical axis and x on the horizontal axis. The vertical axis of the reconstructed signal has tick marks at 1, -1, and -2.

OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019

7

푸리에 변환

푸리에 변환	$a(x) \rightarrow \alpha(f), \theta(f)$
역푸리에 변환	$\alpha(f), \theta(f) \rightarrow a(x)$

OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019

8

푸리에 변환

$$A(f) = \int_{-\infty}^{+\infty} a(x)e^{-\Omega\pi} dx$$

$$a(x) = \int_{-\infty}^{+\infty} A(f)e^{+\Omega\pi} df$$

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



9

이산 푸리에 변환

$$F(k, l) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i, j)e^{-i2\pi\left(\frac{ki}{N} + \frac{lj}{N}\right)}$$

여기서 $e^{ix} = \cos x + i \sin x$ 이다.

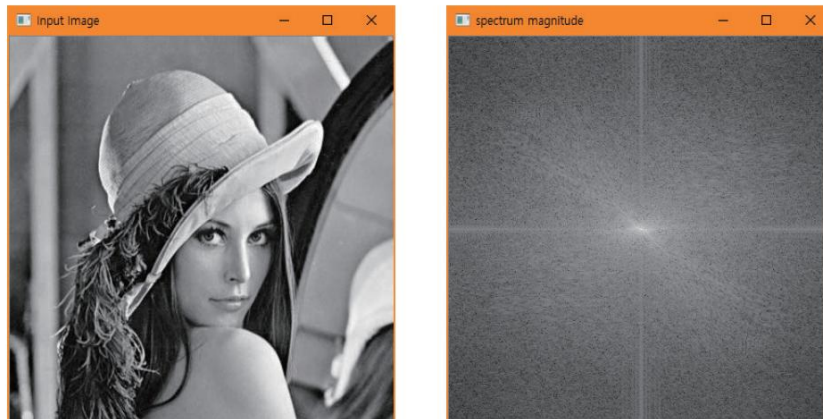
OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



10

이산 푸리에 변환



OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



11

그레이스케일 영상의 DFT 계산하기

```
#include "opencv2/opencv.hpp"
#include <iostream>
using namespace std;
using namespace cv;

int main()
{
    Mat src = imread("d:/lenna.jpg", IMREAD_GRAYSCALE);
    Mat src_float;
    Mat dft_image;

    // ① 그레이스케일 영상을 실수 영상으로 변환한다.
    src.convertTo(src_float, CV_32FC1, 1.0 / 255.0);

    // ② DFT를 수행한다.
    dft(src_float, dft_image, DFT_COMPLEX_OUTPUT);
    return 1;
}
```

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



12

DFT를 화면에 출력하기

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

13

DFT를 화면에 출력하기

```
int main()
{
    Mat src = imread("d:/lenna.jpg", CV_LOAD_IMAGE_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);
    displayDFT(dft_image);
    return 1;
}
```



14

실행 결과



OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



15

서플링



OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



16

서프링

```
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);
}
```

OpenCV로 배우는 디지털 영상 처리 © 한파다컴퓨터스 2019

17

서프링

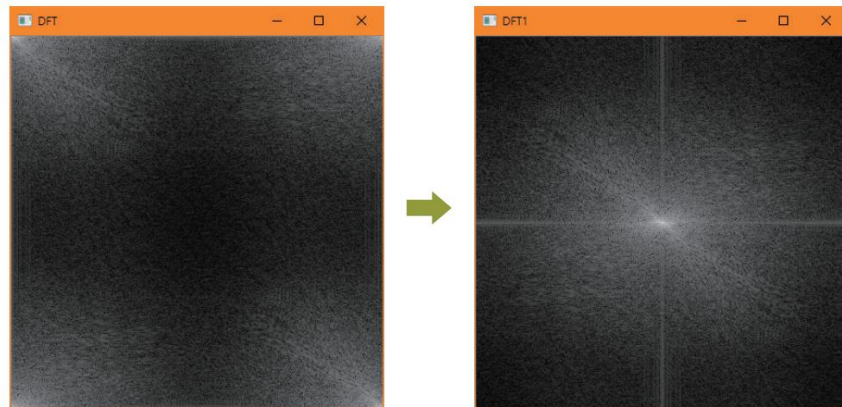
```
int main()
{
    Mat src = imread("d:/lenna.jpg", CV_LOAD_IMAGE_GRAYSCALE);
    Mat src_float;
    Mat dft_image;

    // 그레이스케일 영상을 실수 영상으로 변환한다.
    src.convertTo(src_float, CV_32FC1, 1.0 / 255.0);
    dft(src_float, dft_image, DFT_COMPLEX_OUTPUT);
    shuffleDFT(dft_image);
    displayDFT(dft_image);
    return 1;
}
```

OpenCV로 배우는 디지털 영상 처리 © 한파다컴퓨터스 2019

18

실행 결과



OpenCV로 배우는 디지털 영상 처리

© 인피니티박스 2019



19

역변환

```
int main()
{
    Mat img = imread("d:/lenna.jpg", IMREAD_GRAYSCALE);

    Mat img_float, dft1, inversedft, inversedft1;
    img.convertTo(img_float, CV_32F);
    dft(img_float, dft1, DFT_COMPLEX_OUTPUT);

    // 역변환을 수행한다.
    idft(dft1, inversedft, DFT_SCALE | DFT_REAL_OUTPUT);

    inversedft.convertTo(inversedft1, CV_8U);
    imshow("invertedfft", inversedft1);

    imshow("original", img);
    waitKey(0);
    return 0;
}
```

OpenCV로 배우는 디지털 영상 처리

© 인피니티박스 2019



20

실행 결과



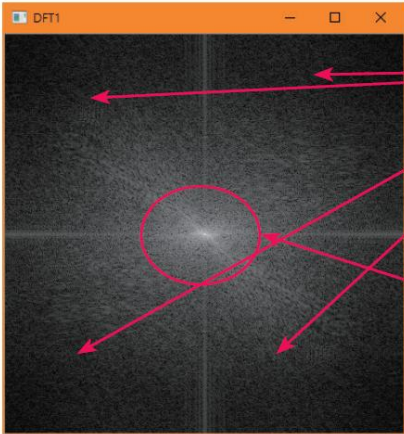
The image shows two side-by-side windows. The left window, titled 'original', displays a grayscale photograph of a woman wearing a wide-brimmed hat. The right window, titled 'invertedfft', displays the same grayscale photograph. A green arrow points from the 'original' window to the 'invertedfft' window, indicating a transformation process.

OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019




21

주파수 필터링



The image shows a window titled 'DFT1' containing a frequency spectrum plot. The plot is a grayscale image with a central bright spot and a dark background. A red circle is drawn around the central bright spot. Four red arrows point from the text labels '고주파' (High Frequency) and '저주파' (Low Frequency) to the plot. The '고주파' label has three arrows pointing to the top, top-right, and bottom-right corners of the plot. The '저주파' label has one arrow pointing to the center of the plot, which is inside the red circle.

OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019




22

저주파 통과 필터

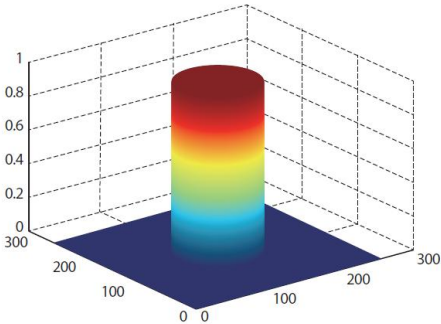
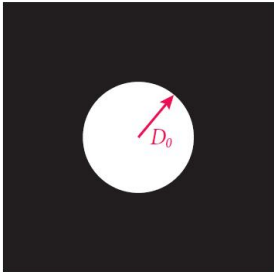
$$H(u, v) = \begin{cases} 1 & D(u, v) \leq D_0 \\ 0 & D(u, v) > D_0 \end{cases}$$
$$D(u, v) = \sqrt{u^2 + v^2}$$

OpenCV로 배우는 디지털 영상 처리 © 인피니티박스 2019




23

저주파 통과 필터



OpenCV로 배우는 디지털 영상 처리 © 인피니티박스 2019



24

자주파 통과 필터

```
void shuffleDFT(Mat& src) { ... }
void displayDFT(Mat& src) { ... }

// 원형 필터를 만든다.
Mat getFilter(Size size)
{
    Mat filter(size, CV_32FC2, Vec2f(0, 0));
    circle(filter, size / 2, 50, Vec2f(1, 1), -1);
    return filter;
}

int main()
{
    Mat src = imread("d:/lenna.jpg", IMREAD_GRAYSCALE);
    Mat src_float;
    imshow("original", src);
```

OpenCV로 배우는 디지털 영상 처리 © 한파다컴퓨터스 2019

25

자주파 통과 필터

```
// 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);

Mat lowpass = getFilter(dft_image.size());
Mat result;

// 원형 필터와 DFT 영상을 서로 곱한다.
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;
```

} OpenCV로 배우는 디지털 영상 처리 © 한파다컴퓨터스 2019

26

실행 결과

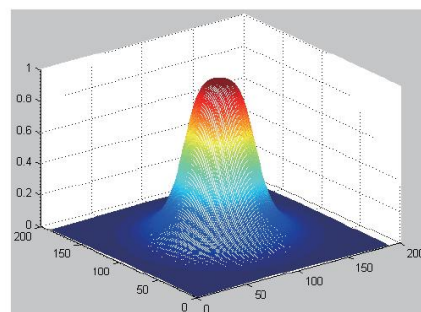


27

버터워스 필터 버전

$$H(u,v) = \frac{1}{1 + (\sqrt{2} - 1) [D(u,v)/D_0]^{2n}}$$

$$D(u,v) = \sqrt{u^2 + v^2}, \quad n = 1, 2, \dots$$



OpenCV로 배우는 디지털 영상 처리 © 인피니티박스 2019

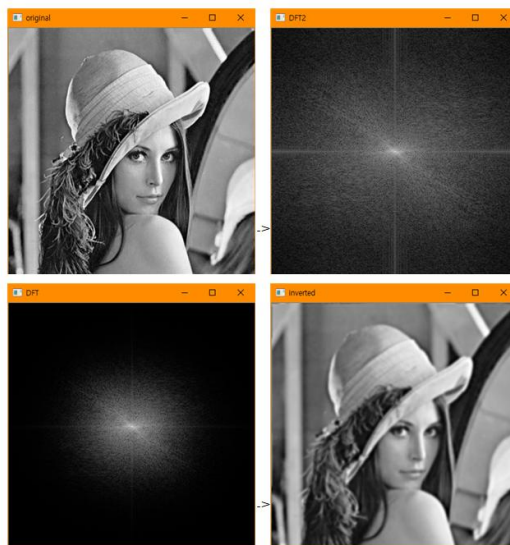
28

```
// 버터워스 필터를 만든다.
Mat getFilter(Size size)
{
    Mat tmp = Mat(size, CV_32F);
    Point center = Point(tmp.rows / 2, tmp.cols / 2);
    double radius;
    double D = 50;
    double n = 2;

    for (int i = 0; i < tmp.rows; i++) {
        for (int j = 0; j < tmp.cols; j++) {
            radius = (double)sqrt(pow((i - center.x), 2.0) +
pow((double)(j - center.y), 2.0));
            tmp.at<float>(i, j) = (float)
                (1 / (1 + pow((double)(radius / D), (double)(2 *
n)))));
        }
    }
    Mat toMerge[] = { tmp, tmp };
    Mat filter;
    merge(toMerge, 2, filter);
    return filter;
}
```

29

실행 결과



OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019

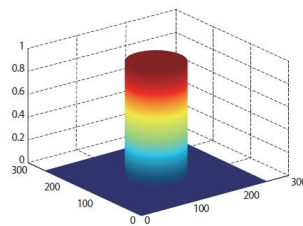
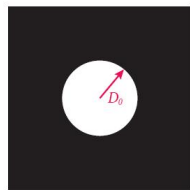


30

고주파 통과 필터

$$H(u, v) = \begin{cases} 0 & D(u, v) \leq D_0 \\ 1 & D(u, v) > D_0 \end{cases}$$

$$D(u, v) = \sqrt{u^2 + v^2}$$



OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



31

저주파 통과 필터

// 원형 필터를 만든다.

Mat getFilter(Size size)

```
{
    Mat filter = Mat::ones(size, CV_32FC2);
    circle(filter, size / 2, 10, Vec2f(0, 0), -1);
    return filter;
}
```

int main()

```
{
    Mat src = imread("d:/lenna.jpg", IMREAD_GRAYSCALE);
    Mat src_float;
    imshow("original", src);

    // 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);
}
```

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



32

자주파 통과 필터

```
Mat highpass = getFilter(dft_image.size());
Mat result;

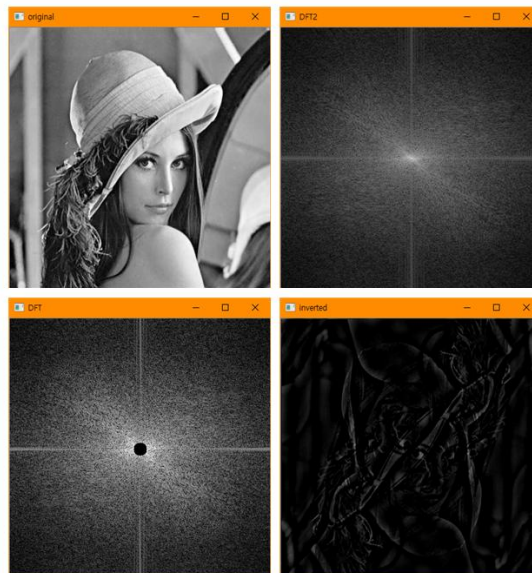
// 원형 필터와 DFT 영상을 서로 곱한다.
multiply(dft_image, highpass, result);
displayDFT(result);

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

OpenCV로 배우는 디지털 영상 처리 © 김파라디콕스 2019

33

실행 결과



OpenCV로 배우는 디지털 영상 처리 © 김파라디콕스 2019



34

주기적인 패턴 제거



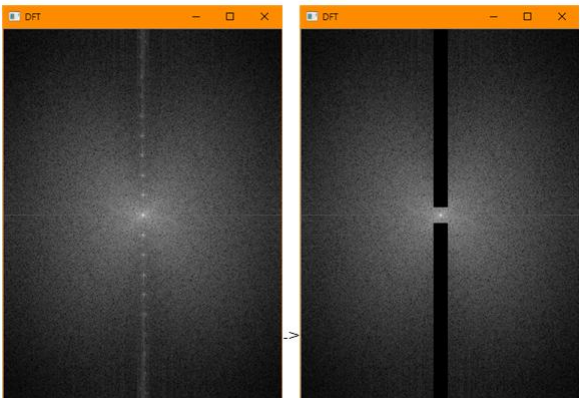
The image shows two side-by-side windows. The left window, titled 'original', displays a grayscale image of a textured surface with a prominent vertical periodic pattern. The right window, titled 'inverted', shows the same image after the periodic pattern has been removed, resulting in a smoother texture. A small arrow points from the original image to the inverted image.

OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019



35

주기적인 패턴 제거



The image shows two side-by-side windows. The left window, titled 'DFT', displays the Discrete Fourier Transform (DFT) of the original image, showing a bright central peak and a vertical line of periodicity. The right window, also titled 'DFT', shows the DFT after the periodic pattern has been removed, resulting in a smoother texture. A small arrow points from the original DFT image to the processed DFT image.

OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019



36

```

// 필터를 만든다.
Mat getFilter(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;
}

```

37

```

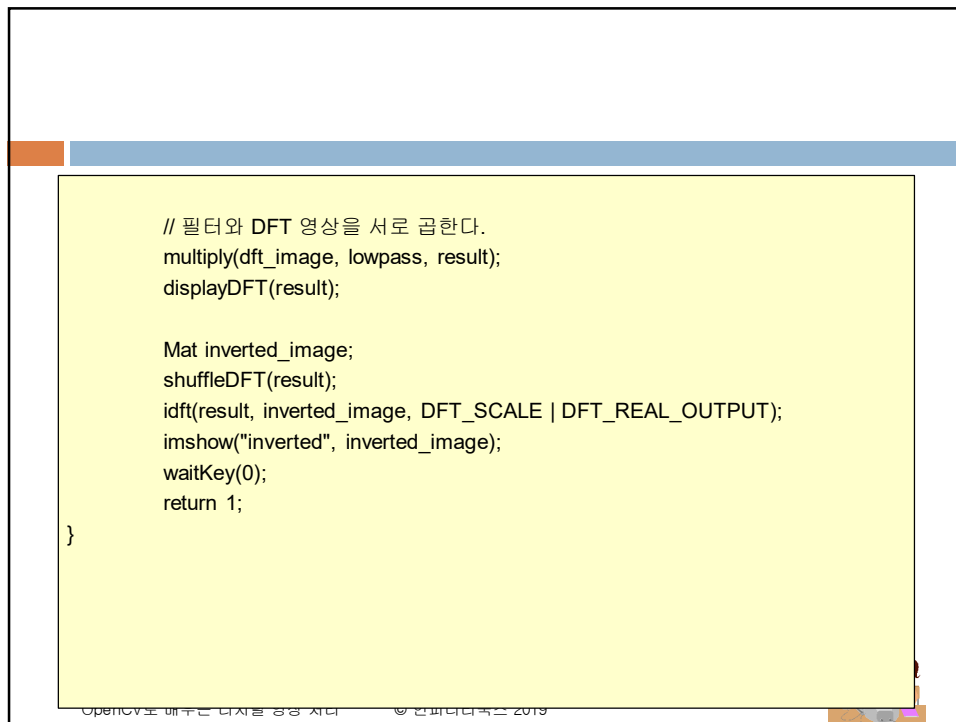
int main()
{
    Mat src = imread("d:/lunar.png", IMREAD_GRAYSCALE);
    Mat src_float, dft_image;
    imshow("original", src);

    // 그레이스케일 영상을 실수 영상으로 변환한다.
    src.convertTo(src_float, CV_32FC1, 1.0 / 255.0);
    dft(src_float, dft_image, DFT_COMPLEX_OUTPUT);
    shuffleDFT(dft_image);
    displayDFT(dft_image);

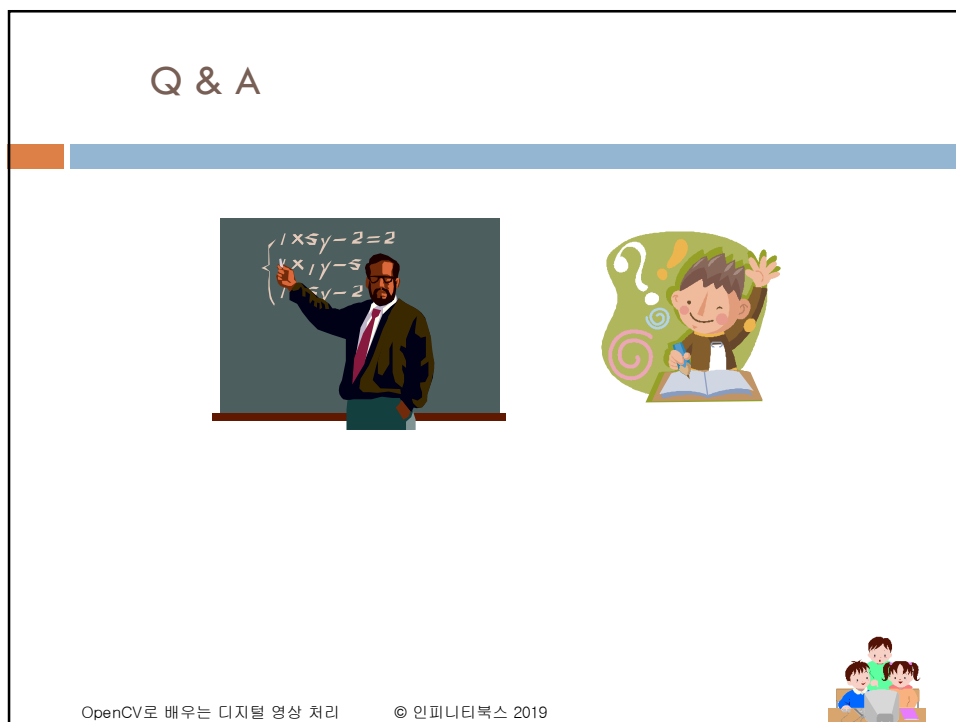
    Mat lowpass = getFilter(dft_image.size());
    Mat result;
}

```

38



39



40