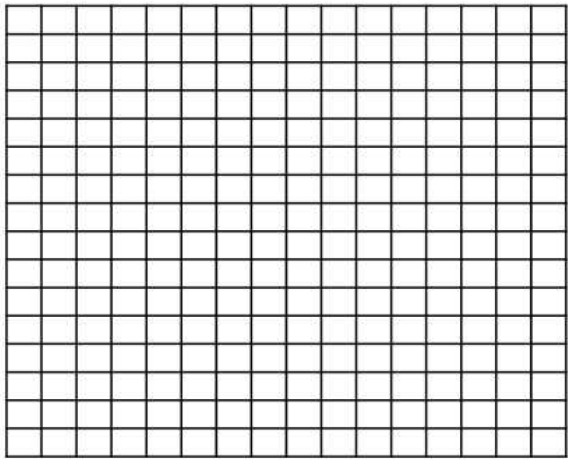
DSP Lab. Week 11
FFT – 2D

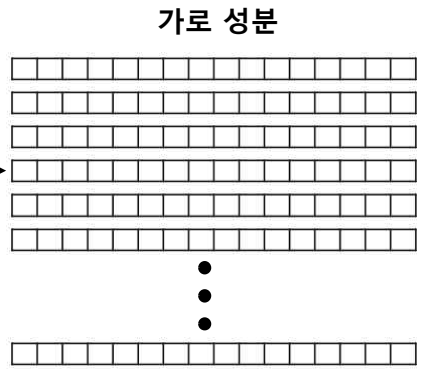
Kyuheon Kim
Media Lab. Rm567
kyuheonkim@khu.ac.kr
Last update : September 2, 2019

2D FFT

$M \times N$ Image

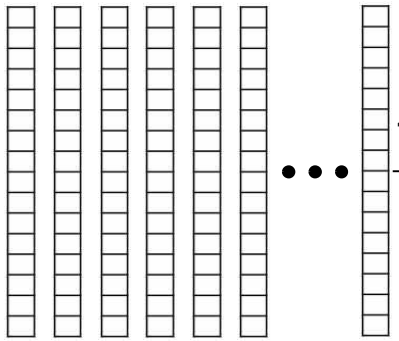


Pixel



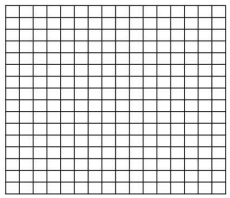
가로 성분

1D-FFT



세로 성분

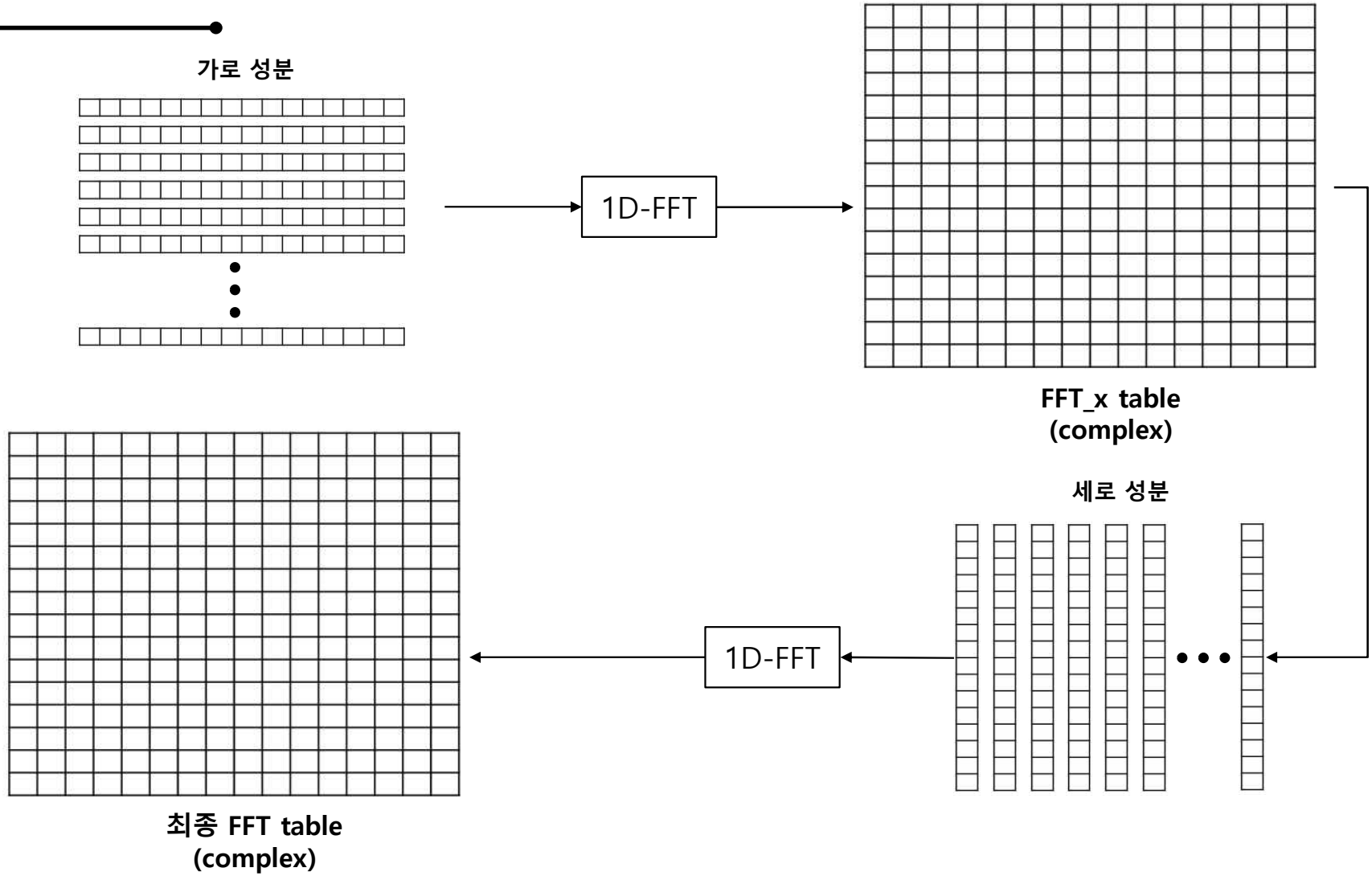
1D-FFT



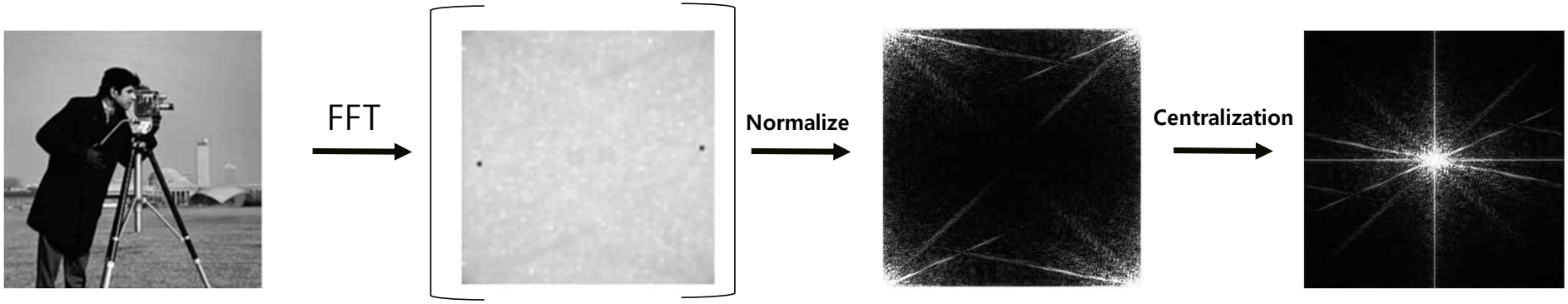
2D FFT table

- For FFT, $M = 2^m$, $N = 2^n$ $(\boldsymbol{m > 1, n > 1})$

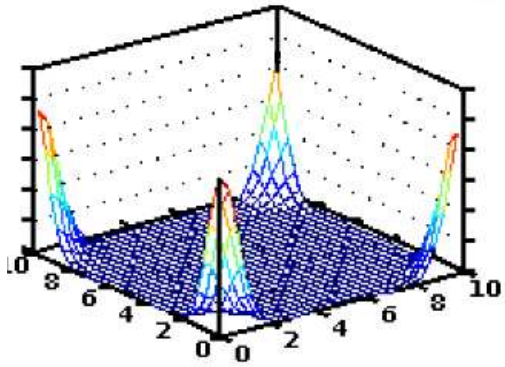
2D FFT

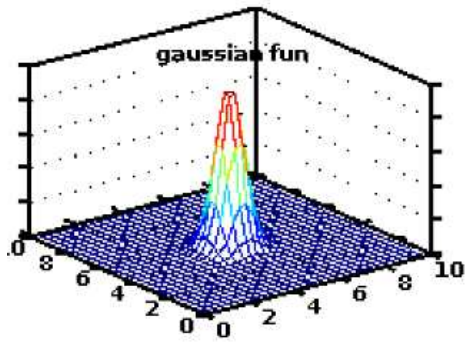

가로 성분
1D-FFT
FFT_x table
(complex)
세로 성분
1D-FFT
최종 FFT table
(complex)

2D FFT – Frequency Domain



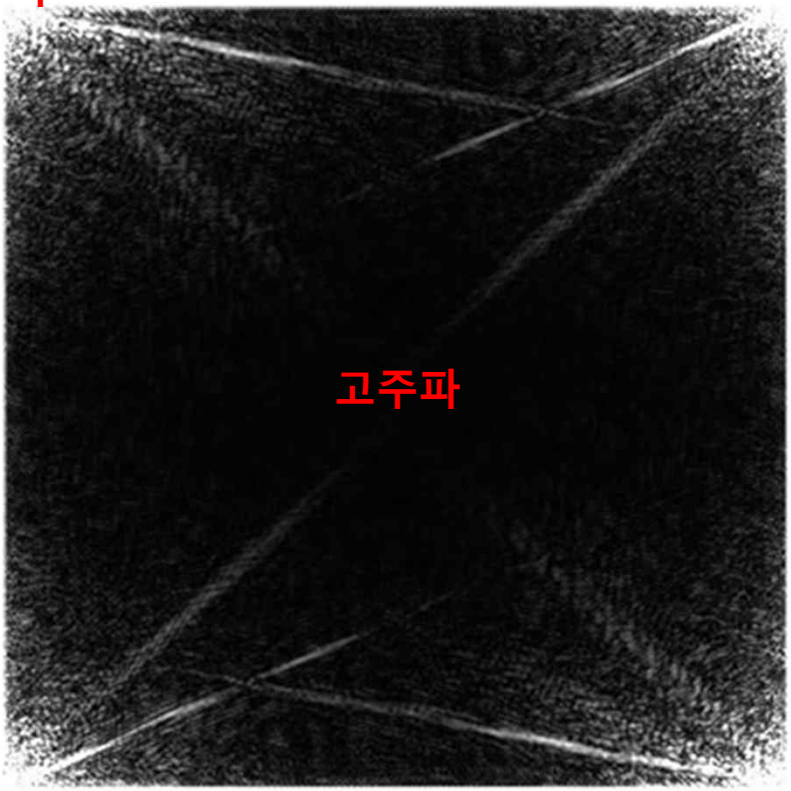
신호 크기가 너무 크다면, $10bg\ (|X|+1)$과 같이 양자화
(Quantization)



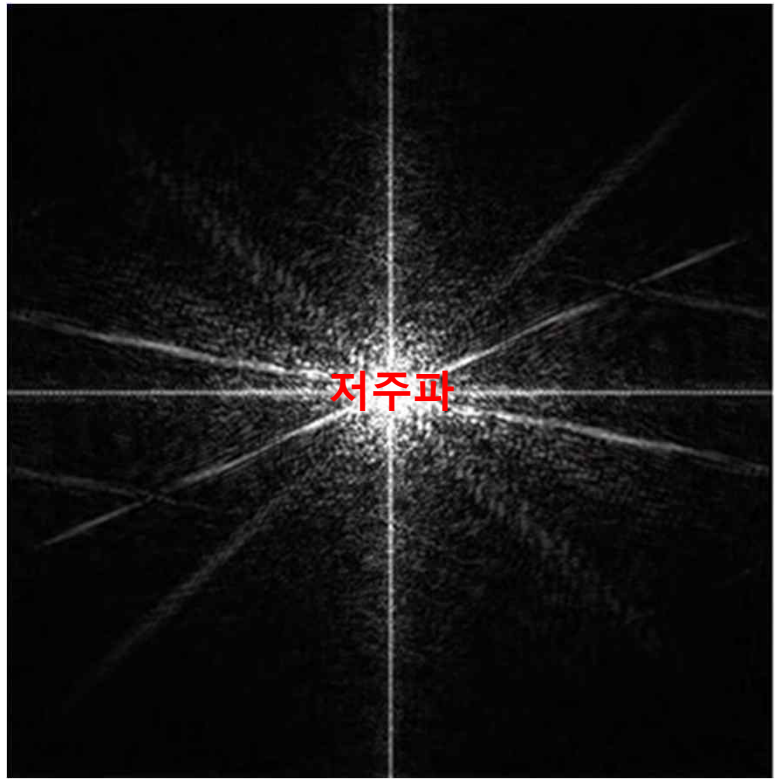


2D FFT – Frequency Domain

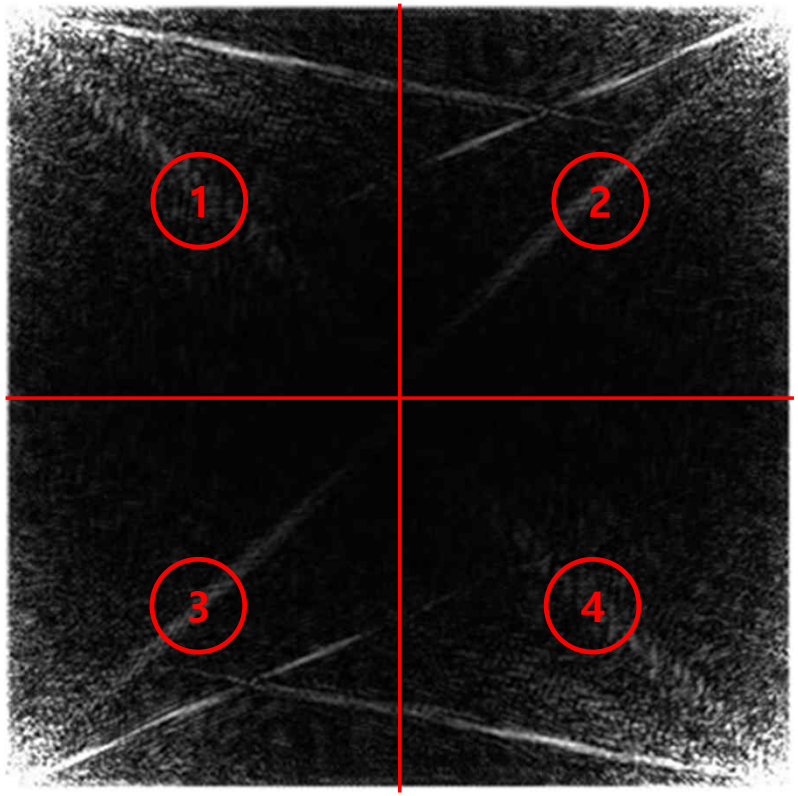
저주파

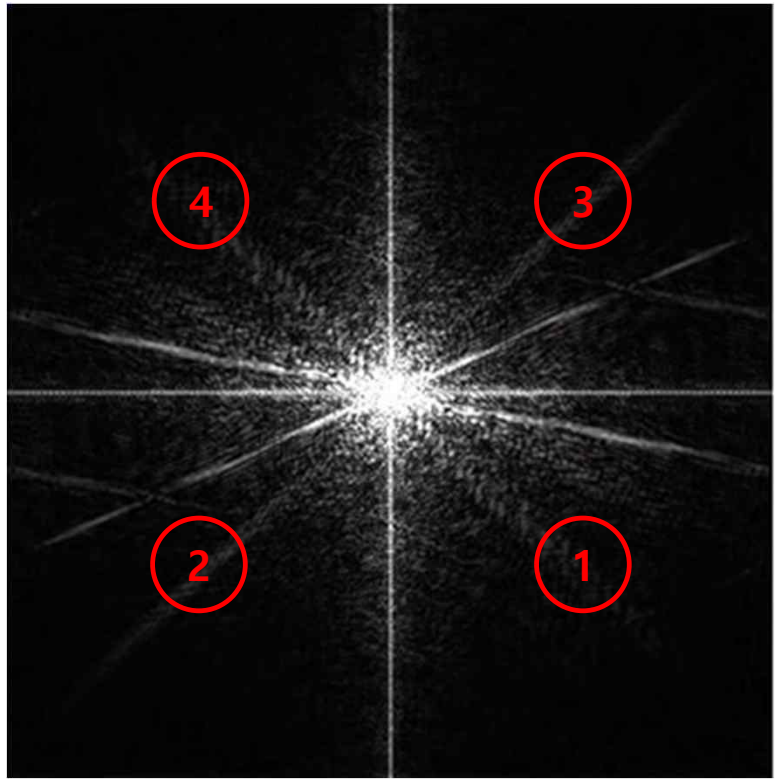


고주파



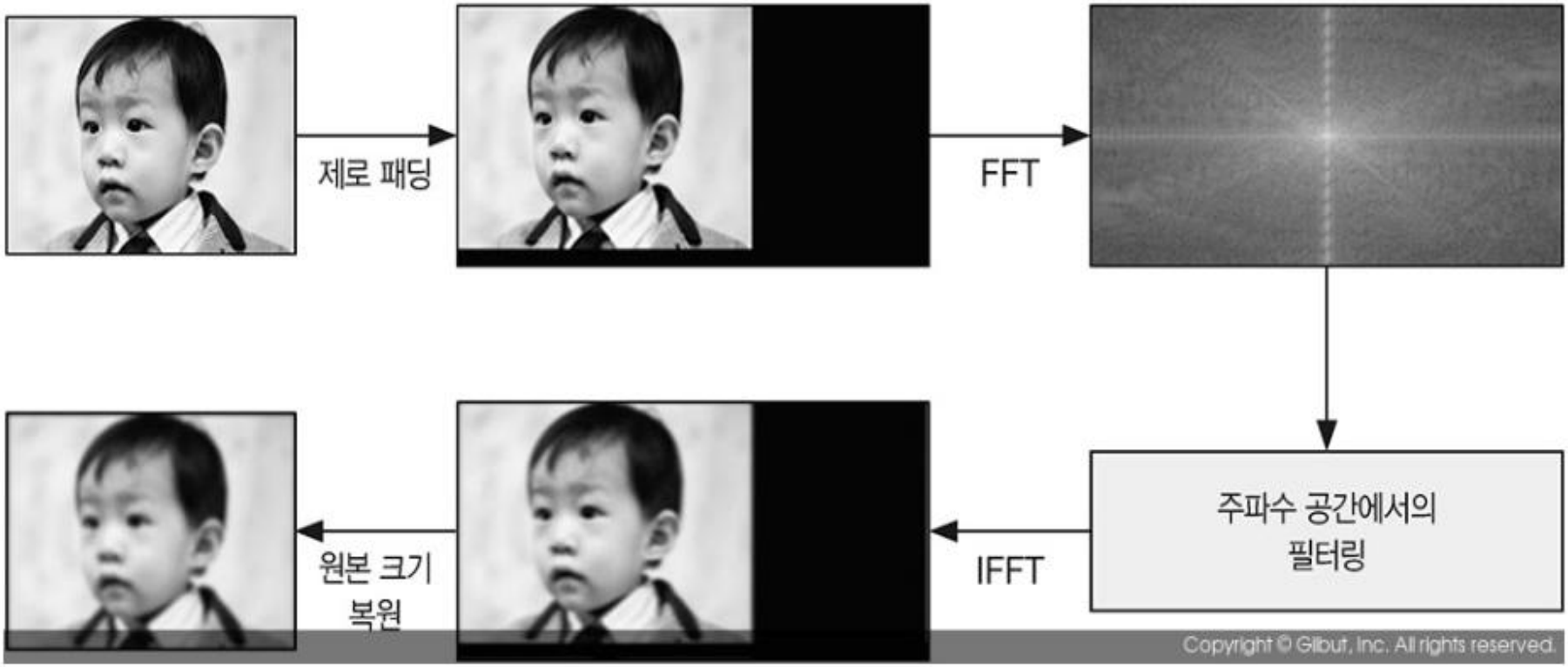
2D FFT – Frequency Domain



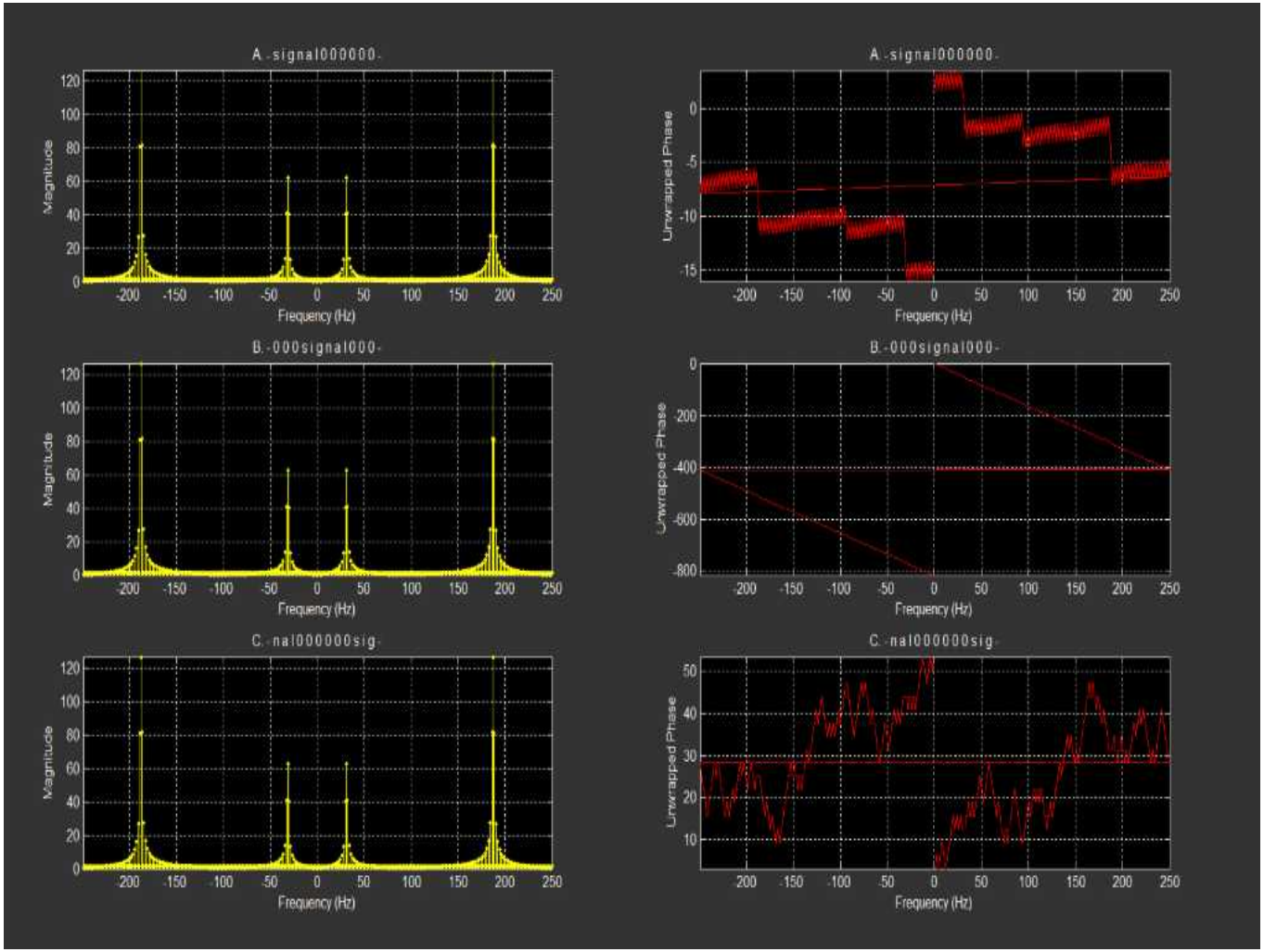


2D FFT- Zero Padding

- For FFT, $M \neq 2^m$, $N \neq 2^n$



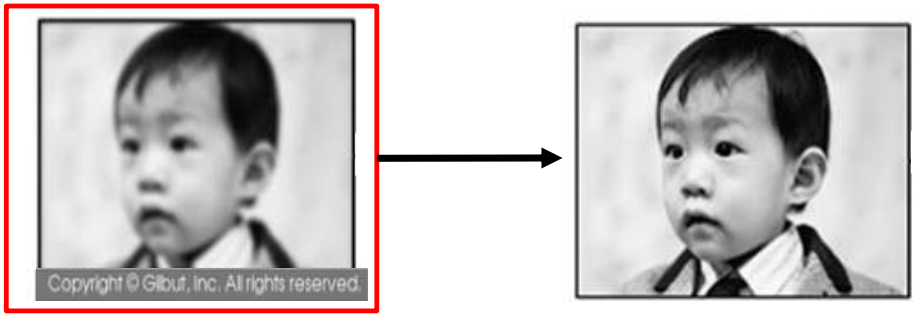
2D FFT- Zero Padding



원본 이미지

Phase 성분 역할

2D FFT



❖ Another FFT Algorithm - SRFFT, FHT, QFT, DITF … , **BUT only special size**

❖ Sharpening - Adding High Frequency

0	-1	0
-1	5	-1
0	-1	0

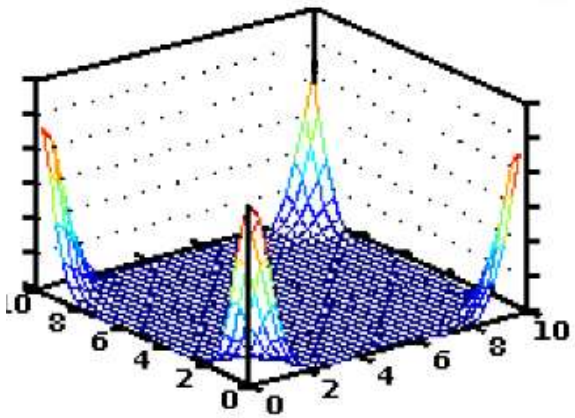
-1	-1	-1
-1	9	-1
-1	-1	-1

1	-2	1
-2	5	-2
1	-2	1

Sharpening Mask

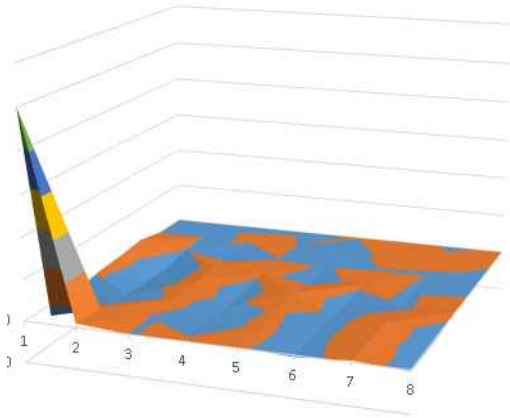
DFT, DCT, FFT

DFT



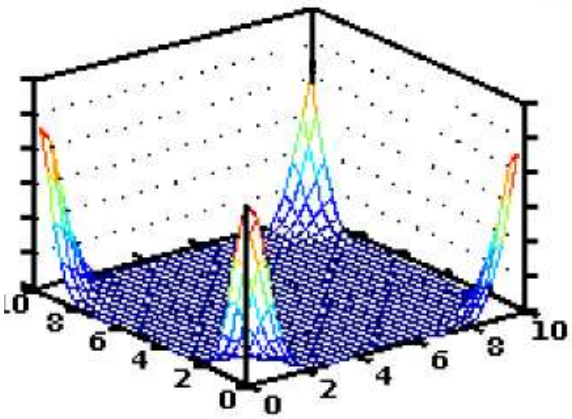
$$F[u,v] = \sum_{y=0}^{N-1}\sum_{x=0}^{M-1} f(x,y)e^{-j2\pi(\frac{ux}{M}+\frac{vy}{N})}$$

DCT



$$F(u,v) = \frac{2C(u)C(v)}{\sqrt{MN}}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\cos\frac{(2i+1)\cdot u\pi}{2M}\cdot\cos\frac{(2j+1)\cdot v\pi}{2N}\cdot f(i,j)$$

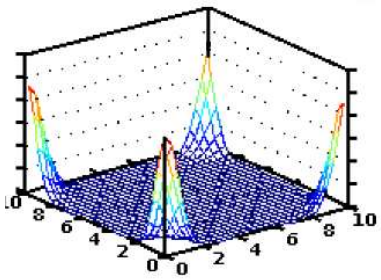
FFT

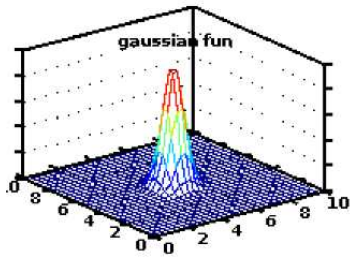


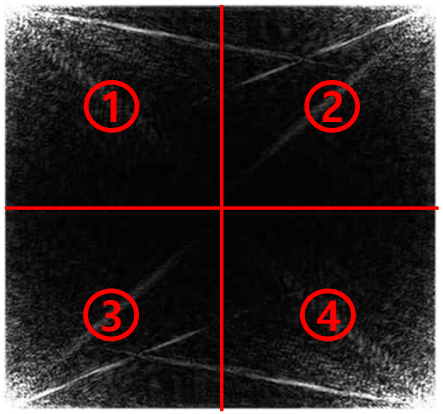
$$X[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r]W_N^{2kr} + W_N^k \sum_{r=0}^{\frac{N}{2}-1} x[2r+1]W_N^{2kr}$$

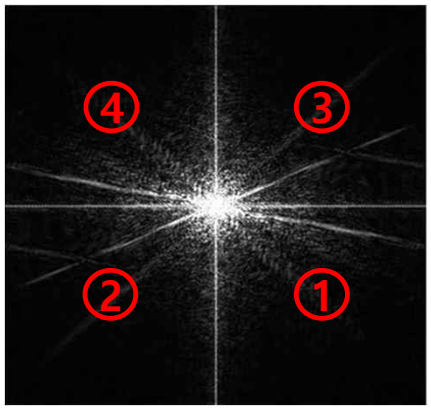
DFT, DCT, FFT

DFT, FFT

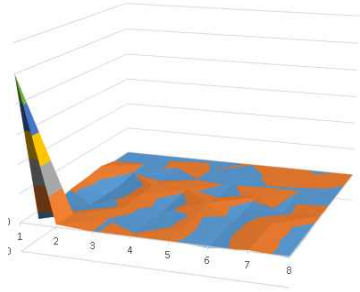


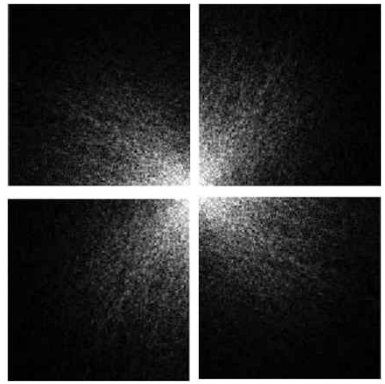






DCT





FFT code

```
void FFT2Radix(double* Xr, double* Xi, double* Yr, double* Yi, int nN, bool bInverse)
{

    double T, Wr, Wi;

    if (nN <= 1) return;

    for (int i = 0; i < nN; i++) {
        Yr[i] = Xr[i];
        Yi[i] = Xi[i];
    }

    int j = 0, k = 0;
    for (int i = 1; i < (nN - 1); i++) {
        k = nN / 2;
        while (k <= j) {
            j = j - k;
            k = k / 2;
        }
        j = j + k;
        if (i < j) {
            T = Yr[j];
            Yr[j] = Yr[i];
            Yr[i] = T;

            T = Yi[j];
            Yi[j] = Yi[i];
            Yi[i] = T;
        }
    }

    double Tr, Ti;
    int iter, j2, pos;
    k = nN >> 1;
    iter = 1;
    while (k > 0) {
        j = 0;
        j2 = 0;

        for (int i = 0; i < nN >> 1; i++) {
            Wr = cos(2.*PI*(j2*k) / nN);
            if (bInverse == 0)
                Wi = -sin(2.*PI*(j2*k) / nN);
            else
                Wi = sin(2.*PI*(j2*k) / nN);

            pos = j + (1 << (iter - 1));

            Tr = Yr[pos] * Wr - Yi[pos] * Wi;
            Ti = Yr[pos] * Wi + Yi[pos] * Wr;

            Yr[pos] = Yr[j] - Tr;
            Yi[pos] = Yi[j] - Ti;

            Yr[j] += Tr;
            Yi[j] += Ti;

            j += 1 << iter;
            if (j >= nN) j = ++j2;
        }
        k >>= 1;
        iter++;
    }
    if (bInverse) {
        for (int i = 0; i < nN; i++) {
            Yr[i] /= nN;
            Yi[i] /= nN;
        }
    }
}
```

FFT code

```
void FFT2D(uchar** img, double** OutputReal, double** OutputImag, int nW, int nH) // 1D fft를 이용하여 2D FFT를 구현
{
    int x, y;
    double *dRealX, *dImagX;
    double *dRealY, *dImagY;

    dRealX = new double[nW];
    dImagX = new double[nW];
    dRealY = new double[nW];
    dImagY = new double[nW];

    for (y = 0; y < nH; y++) {
        for (x = 0; x < nW; x++) {
            dRealX[x] = img[y][x]; //
            dImagX[x] = 0.;
        }

        FFT2Radix(dRealX, dImagX, dRealY, dImagY, nW, false);

        for (x = 0; x < nW; x++) {
            OutputReal[y][x] = dRealY[x];
            OutputImag[y][x] = dImagY[x];
        }
    }
    delete[] dRealX;
    delete[] dImagX;
    delete[] dRealY;
    delete[] dImagY;

    dRealX = new double[nH];
    dImagX = new double[nH];
    dRealY = new double[nH];
    dImagY = new double[nH];

    for (x = 0; x < nW; x++) {
        for (y = 0; y < nH; y++) {
            dRealX[y] = OutputReal[y][x];
            dImagX[y] = OutputImag[y][x];
        }

        FFT2Radix(dRealX, dImagX, dRealY, dImagY, nH, false);

        for (y = 0; y < nH; y++) {
            OutputReal[y][x] = dRealY[y];
            OutputImag[y][x] = dImagY[y];

        }
    }
    delete[] dRealX;
    delete[] dImagX;
    delete[] dRealY;
    delete[] dImagY;
}
```

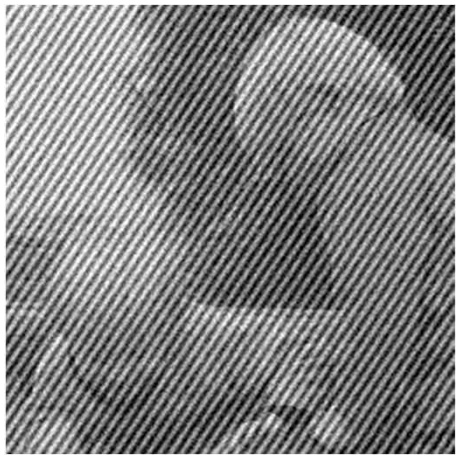
FFT code

```
void FFT2Dinverse(double** InputReal, double** InputImag, double** OutputDouble, int nW, int nH)
{
    int x, y;
    double* dRealX, *dImagX;
    double* dRealY, *dImagY;
    double** OutputReal, **OutputImag;

    OutputReal = new double*[nH];
    OutputImag = new double*[nH];
    for (int i = 0; i < nH; i++) {
        OutputReal[i] = new double[nW];
        OutputImag[i] = new double[nW];
    }

    dRealX = new double[nW];
    dImagX = new double[nW];
    dRealY = new double[nW];
    dImagY = new double[nW];

    for (y = 0; y < nH; y++) {
        for (x = 0; x < nW; x++) {
            dRealX[x] = InputReal[y][x];
            dImagX[x] = InputImag[y][x];
        }

        FFT2Radix(dRealX, dImagX, dRealY, dImagY, nW, true);

        for (x = 0; x < nH; x++) {
            OutputReal[y][x] = dRealY[x];
            OutputImag[y][x] = dImagY[x];
        }
    }
    delete[] dRealX;
    delete[] dImagX;
    delete[] dRealY;
    delete[] dImagY;

    dRealX = new double[nH];
    dImagX = new double[nH];
    dRealY = new double[nH];
    dImagY = new double[nH];

    for (x = 0; x < nW; x++) {
        for (y = 0; y < nH; y++) {
            dRealX[y] = OutputReal[y][x];
            dImagX[y] = OutputImag[y][x];
        }

        FFT2Radix(dRealX, dImagX, dRealY, dImagY, nH, true);

        for (y = 0; y < nH; y++) {
            OutputReal[y][x] = dRealY[y];
            OutputImag[y][x] = dImagY[y];

        }
    }
    delete[] dRealX;
    delete[] dImagX;
    delete[] dRealY;
    delete[] dImagY;

    for (y = 0; y < nH; y++) {
        for (x = 0; x < nW; x++) {
            OutputDouble[y][x] = OutputReal[y][x];
        }
    }

    for (int i = 0; i < nH; i++) {
        delete[] OutputReal[i];
        delete[] OutputImag[i];
    }
    delete[] OutputReal;
    delete[] OutputImag;
}
```

FFT code (Normalize)

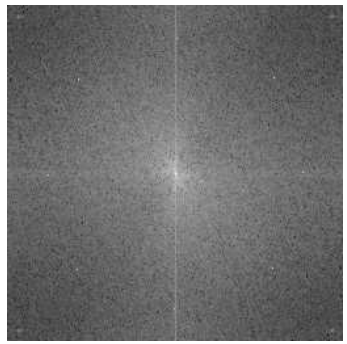
```
void DNormalize2D(double **p1, uchar **p2, int nW, int nH)
{
    // 정규화 함수
    int x, y;
    double min = 9999.;
    double max = -9999.;
    // 최대와 최소를 미리 초기화
    double val; // 최대와 최소 비교를 통해 최대 또는 최소를 새로 초기화
    for (y = 0; y < nH; y++)
        for (x = 0; x < nW; x++) {
            val = p1[y][x];
            if (val > max) max = val;
            if (val < min) min = val;
        }
    if (max == min) { //최대와 최소가 같다면, 모든 값이 같으므로 0으로 초기화
        for (y = 0; y < nH; y++)
            for (x = 0; x < nW; x++)
                p2[y][x] = 0;
        return;
    }
    //0~255사이의 값으로 설정
    double dfactor = 255 / (max - min);
    for (y = 0; y < nH; y++)
        for (x = 0; x < nW; x++)
            p2[y][x] = (uchar)((p1[y][x] - min) * dfactor);
}
```

Assignment



노이즈가 포함된 사진을 필터링하여 원본과 유사하게 복원하라.

[보고서 제출 시]
1. 원본의 주파수 영역 사진 (가운데 저주파 정렬)
2. 노이즈 영상의 주파수 영역 사진 (가운데 저주파 정렬)
3. 복원된 사진
4. 프로그래밍 파일



Assignment Rule

"KLAS에 제출할 때 다음 사항을 꼭 지켜주세요"

1. 파일명 : "Lab00_요일_대표자이름.zip"
Ex) Lab01_목_홍길동.zip **(압축 툴은 자유롭게 사용)**

2. 제출 파일 **(보고서와 프로그램을 압축해서 제출)**
   - 보고서 파일 (hwp, word): 이름, 학번, 목적, 변수, 알고리즘(순서), 결과 분석, 느낀 점
   - 프로그램

DSP 실험 보고서

과제 번호	Lab01	제출일	2019.09.02
학번/이름	20xxxxxxx 홍길동 20xxxxxxx 푸리에		

1. 목적	
2. 변수	
3. 알고리즘	
4. 결과분석	
5. 느낀 점