1.소스코드

#include <opencv\cv.h>

#include<opencv\highgui.h>

#include<opencv\cxcore.h>

#include <stdio.h>

#define unsigned char uchar

uchar\*\* uc\_alloc(int size\_x, int size\_y) // ��� �޸� �Ҵ� x,y

{

    uchar\*\* m;

    int i;

    if ((m = (uchar \* \*)calloc(size\_y, sizeof(uchar\*))) == NULL)

    {

        printf("uc\_alloc error 1\7\n");

        exit(0);

    }

    for (i = 0; i < size\_y; i++)

        if ((m[i] = (uchar\*)calloc(size\_x, sizeof(uchar))) == NULL) {

            printf("uc\_alloc error 2\7\n");

            exit(0);

        }

    return m;

}

void read\_ucmatrix(int size\_x, int size\_y, uchar\*\* ucmatrix, char\* filename)

{

    int i;

    FILE\* f;

    if ((fopen\_s(&f, filename, "rb")) != NULL)

    {

        printf("%s File open Error! \n", filename);

        exit(0);

    }

    for (i = 0; i < size\_y; i++)

        if (fread(ucmatrix[i], sizeof(uchar), size\_x, f) != size\_x)

        {

            printf("Data Read Error! \n");

            exit(0);

        }

    fclose(f);

}

void write\_ucmatrix(int size\_x, int size\_y, uchar\*\* ucmatrix, char\* filename)

{

    int i;

    FILE\* f;

    if ((fopen\_s(&f, filename, "wb")) != NULL)

    {

        printf("%s File open Error! \n", filename);

        exit(0);

    }

    for (i = 0; i < size\_y; i++)

        if (fwrite(ucmatrix[i], sizeof(uchar), size\_x, f) != size\_x)

        {

            printf("Data Write Error! \n");

            exit(0);

        }

    fclose(f);

}

//평균값

double average(uchar\*\* img, int size\_x, int size\_y)

{

    double sum = 0, avg;

    int i, j;

    for (i = 0; i < size\_x; i++)

    {

        for (j = 0; j < size\_y; j++)

        {

            sum += img[i][j];

        }

    }

    avg = sum / ((double)size\_x \* size\_y);

    printf("Average of Image %lf \n", avg);

    return avg;

}

//분산값 구하기

double bunsan(uchar\*\* img, int size\_x, int size\_y)

{

    double avg = average(img, size\_x, size\_y);

    double total = 0, valance = 0;

    int i, j;

    for (i = 0; i < size\_x; i++)

    {

        for (j = 0; j < size\_y; j++)

        {

            valance += (img[i][j] - avg) \* (img[i][j] - avg);

        }

    }

    total = valance / (size\_x \* size\_y);

    printf("�л��� %lf \n", total);

    printf("ǥ�������� %lf \n", sqrt(total));

    return avg;

}

void makeBinary(uchar\*\* img, uchar\*\* out, int size\_x, int size\_y, double avg)

{

    int i, j;

    for (i = 0; i < size\_x; i++)

    {

        for (j = 0; j < size\_y; j++)

        {

            if (img[i][j] > avg)

                out[i][j] = 255;      // chang value for book with avg-30

            else

                out[i][j] = 0;

        }

    }

}

void PowImg(uchar\*\* img, uchar\*\* Result, int Row, int Col, double gamma)

{

    int i, j;

    double tmp;

    for (i = 0; i < Row; i++)//��

        for (j = 0; j < Col; j++)//��

        {

            tmp = pow(img[j][i] / 255., 1 / gamma);

            if (tmp \* 255 > 255)tmp = 1;

            else if (tmp \* 255 < 0) tmp = 0;

            tmp = tmp \* 255;

            Result[j][i] = tmp;

        }

}

int main(int argc, char\* argv[]) {

    int i, j;

    double avg, gamma = 1;

    IplImage\* cvImg, \* cvImg2;

    CvSize imgSize, imgSize2;

    uchar\*\* img;

    uchar\*\* omg;

    if (argc != 5) {

        printf("Exe imgData x\_size y\_size \n");

        exit(0);

    }

    imgSize.width = atoi(argv[2]);

    imgSize.height = atoi(argv[3]);

    cvImg = cvCreateImage(imgSize, 8, 1);

    //img ,원본 / omg , 감마값 적용이미지

    img = uc\_alloc(imgSize.width, imgSize.height);

    omg = uc\_alloc(imgSize.width, imgSize.height);

    //img 이미지 읽기

    read\_ucmatrix(imgSize.width, imgSize.height, img, argv[1]);

    //평균값설정

    avg = average(img, imgSize.width, imgSize.height);

    //감마2.45 설정

    PowImg(img, omg, imgSize.width, imgSize.height, 10);

    //평균값 128로 설정

    if (avg < 128) {

        while (avg < 128) {

            gamma = gamma + 0.001;

            PowImg(img, omg, imgSize.width, imgSize.height, gamma);

            avg = average(omg, imgSize.width, imgSize.height);

        }

    }

    else if (avg > 128) {

        while (avg > 128) {

            gamma = gamma - 0.001;

            PowImg(img, omg, imgSize.width, imgSize.height, gamma);

            avg = average(omg, imgSize.width, imgSize.height);

        }

    }

    else

        //구해진 평균값에 따른 감마값을 적용시켜 이미지 추출

        PowImg(img, omg, imgSize.width, imgSize.height, gamma);

    //띄울 사진 설정

    for (i = 0; i < imgSize.width; i++)

        for (j = 0; j < imgSize.height; j++) {

            //img[j][i]일경우 원본사진이 뜨고, omg[j][i]일경우 감마작업을 한 사진이 뜬다.

            ((uchar\*)(cvImg->imageData + cvImg->widthStep \* j))[i] = img[j][i];

        }

    //감마작업한 사진을 저장한다.

    write\_ucmatrix(imgSize.width, imgSize.height, omg, argv[4]);

    //윈도우창 이름설정

    cvNamedWindow(argv[1], 0);

    //보여줄 이미지 설정

    cvShowImage(argv[1], cvImg);

    cvWaitKey(0);

    cvReleaseImage(&cvImg);

    return 0;

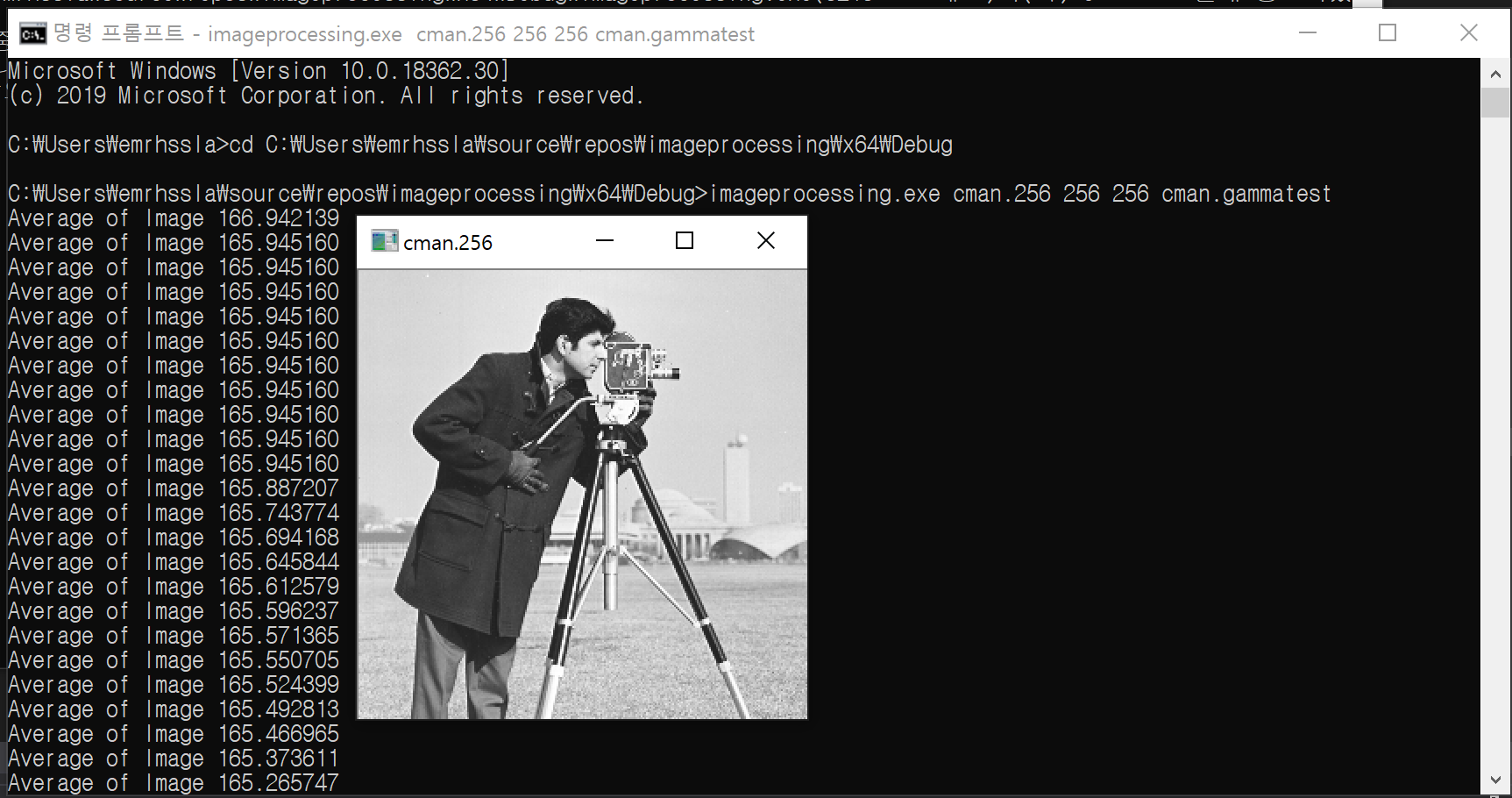
}

git주소

<https://github.com/emrhssla/imageProcessing>

2.실행결과

--원본사진 cman.256



감마작업을 진행한 사진 cman.gammatest

