**영상처리 HW2**

**16010980 이우석**



< 코드 내용 >

1. **IP.H**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* file - ip.h \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* typedefs \*/

typedef unsigned char \*image\_ptr; // 배열

typedef double \*double\_ptr;

typedef struct

{

unsigned char r,g,b;

} pixel;

typedef pixel \*pixel\_ptr;

typedef struct

{

int width; // 넓이

int height; // 높이

float \*x\_data;

float \*y\_data;

} mesh;

typedef struct

{

double re;

double im;

} COMPLEX;

typedef COMPLEX \*complex\_ptr;

typedef struct

{

int x;

int y;

} POINT;

typedef struct

{

POINT P;

POINT Q;

int dx, dy;

float length;

long length\_squared;

} LINE;

typedef struct

{

POINT P;

POINT Q;

} LINE\_SEGMENT;

typedef struct

{

int number; /\* number of segments to follow \*/

LINE\_SEGMENT line[100];

char \*filename; /\* name of file holding the line list \*/

} LINE\_LIST;

/\* defines \*/

#define PI 3.14159265358979323846

#define CLIP(val, low, high) {if(val<low) val=low; if(val>high) val=high;} // val 이 low 보다 작으면 val 에 low 값을 저장. val 이 high 보다 크면 val 에 high 값을 저장.

#define CLAMP(val, low, high) ((val<low) ? low : ((val>high) ? high : val))

#define MAX(A,B) ((A) > (B) ? (A) : (B))

#define MIN(A,B) ((A) < (B) ? (A) : (B))

#define IP\_MALLOC(X) malloc(X) // malloc 함수 - 메모리 할당

#define IP\_FREE(X) free(X) // free 함수 - 메모리 해제

#define PBM 4 // '4' 를 PBM 형식으로 인식.

#define PGM 5 // '5' 를 PGM 형식으로 인식.

#define PPM 6 // '6' 을 PPM 형식으로 인식.

1. **Iplib.c**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* File: iplib.c \*

\* \*

\* Desc: general purpose image processing routines \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <malloc.h>

#include <stdio.h>

#include <stdlib.h>

#include "ip.h"

image\_ptr read\_pnm(char \*filename, int \*rows, int \*cols, int \*type);

int getnum(FILE \*fp);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Func: read\_pnm \*

\* \*

\* Desc: reads a portable bitmap file \*

\* \*

\* Params: filename - name of image file to read \*

\* rows - number of rows in the image \*

\* cols - number of columns in the image \*

\* type - file type \*

\* \*

\* Returns: pointer to the image just read into memory \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

image\_ptr read\_pnm(char \*filename, int \*rows, int \*cols, int \*type)

{

int i; /\* index variable \*/

int row\_size; /\* size of image row in bytes \*/

int maxval; /\* maximum value of pixel \*/

FILE \*fp; /\* input file pointer \*/

int firstchar, secchar; /\* first 2 characters in the input file \*/

image\_ptr ptr; /\* pointer to image buffer \*/

unsigned long offset; /\* offset into image buffer \*/

unsigned long total\_size; /\* size of image in bytes \*/

unsigned long total\_bytes; /\* number of total bytes written to file \*/

float scale; /\* number of bytes per pixel \*/

/\* open input file \*/

if((fp = fopen(filename, "rb")) == NULL) // 예외 처리. 문제가 생길 시 프로그램이 에러를 출력하고 종료.

{

printf("Unable to open %s for reading\n",filename);

exit(1);

}

firstchar = getc(fp); // 영상이 PPM 파일인지 확인하기 위해.

secchar = getc(fp); // 영상이 PBM 인지, PGM 인지, PPM 인지 확인하기 위해.

if(firstchar != 'P') // 첫글자가 'P' 가 아니면, 잘못된 입력임을 출력하고 프로그램을 종료.

{

printf("You silly goof... This is not a PPM file!\n");

exit(1);

}

\*cols = getnum(fp); // 가로 길이를 받음.

\*rows = getnum(fp); // 세로 길이를 받음.

\*type = secchar - '0'; // 변수 형식이 char 인 secchar 를 type 에 정수로 바꿔서 넣기 위해.

switch(secchar)

{

case '4': /\* PBM \*/

scale = 0.125;

maxval = 1; // PBM 형식은 0 과 1 뿐 이기 때문에.

break;

case '5': /\* PGM \*/

scale = 1.0;

maxval = getnum(fp); // 헤더의 마지막은 밝기 레벨의 최대값이므로.

break;

case '6': /\* PPM \*/

scale = 3.0;

maxval = getnum(fp); // 마찬가지.

break;

default : /\* Error \*/

printf("read\_pnm: This is not a Portable bitmap RAWBITS file\n");

exit(1);

break;

}

row\_size = (\*cols) \* scale;

total\_size = (unsigned long) (\*rows) \* row\_size; // 전체 크기 계산.

ptr = (image\_ptr) IP\_MALLOC(total\_size); // 전체 크기만큼 메모리 할당.

if(ptr == NULL) // 예외 처리. ptr == NULL 이면, 메모리 할당이 정상적으로 이루어지지 않았음을 의미하기 때문에.

{

printf("Unable to malloc %lu bytes\n",total\_size);

exit(1);

}

total\_bytes=0;

offset = 0;

for(i=0; i<(\*rows); i++) // 마지막 행까지 반복.

{

total\_bytes+=fread(ptr+offset, 1, row\_size, fp); // ptr 배열에 1 byte (0 ~ 255) 크기의 값을 파일로부터 한 행씩 입력받음.

offset += row\_size; // 다음 행의 첫 offset 을 지정함.

}

if(total\_size != total\_bytes) // total\_size 와 total\_bytes 가 같은 지 확인함으로써, 파일로부터 데이터를 불러오는 데에 문제가 없었는지 확인.

{

printf("Failed miserably trying to read %ld bytes\nRead %ld bytes\n",

total\_size, total\_bytes);

exit(1);

}

fclose(fp);

return ptr; // ptr 의 첫 offset 주소 반환.

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Func: getnum \*

\* \*

\* Desc: reads an ASCII number from a portable bitmap file header \*

\* \*

\* Param: fp - pointer to file being read \*

\* \*

\* Returns: the number read \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int getnum(FILE \*fp)

{

char c; /\* character read in from file \*/

int i; /\* number accumulated and returned \*/

do

{

c = getc(fp); // 파일로부터 문자 c 를 입력 받음.

}

while((c==' ') || (c=='\t') || (c=='\n') || (c=='\r'));

if((c<'0') || (c>'9')) // 예외 처리.

if(c == '#') /\* chew off comments \*/

{

while(c == '#') // 주석을 무시하기 위해.

{

while(c != '\n') // 줄 바꿈이 나올 때까지 모두 무시.

c = getc(fp);

c = getc(fp);

}

}

else

{

printf("Garbage in ASCII fields\n"); // 예외 처리. '#', "0 ~ 9" 문자가 아닌 경우, 에러 출력 후, 프로그램 종료.

exit(1);

}

i=0;

do

{

i=i\*10+(c-'0'); /\* convert ASCII to int \*/

c = getc(fp);

}

while((c>='0') && (c<='9'));

return i; // 숫자로 변환된 i 반환

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Func: write\_pnm \*

\* \*

\* Desc: writes out a portable bitmap file \*

\* \*

\* Params: ptr - pointer to image in memory \*

\* filename \_ name of file to write image to \*

\* rows - number of rows in the image \*

\* cols - number of columns in the image \*

\* magic\_number - number that defines what type of file it is \*

\* \*

\* Returns: nothing \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void write\_pnm(image\_ptr ptr, char \*filename, int rows,

int cols, int magic\_number)

{

FILE \*fp; /\* file pointer for output file \*/

long offset; /\* current offset into image buffer \*/

long total\_bytes; /\* number of bytes written to output file \*/

long total\_size; /\* size of image buffer \*/

int row\_size; /\* size of row in bytes \*/

int i; /\* index variable \*/

float scale; /\* number of bytes per image pixel \*/

switch(magic\_number)

{

case 4: /\* PBM \*/

scale = 0.125;

break;

case 5: /\* PGM \*/

scale = 1.0;

break;

case 6: /\* PPM \*/

scale = 3.0;

break;

default : /\* Error \*/

printf("write\_pnm: This is not a Portable bitmap RAWBITS file\n");

exit(1);

break;

}

/\* open new output file \*/

if((fp=fopen(filename, "wb")) == NULL)

{

printf("Unable to open %s for output\n",filename);

exit(1);

}

/\* print out the portable bitmap header \*/ // 헤더 작성

fprintf(fp, "P%d\n%d %d\n", magic\_number, cols, rows);

if(magic\_number != 4)

fprintf(fp, "255\n");

row\_size = cols \* scale;

total\_size = (long) row\_size \*rows; // 전체 크기 계산.

offset = 0;

total\_bytes = 0;

for(i=0; i<rows; i++)

{

total\_bytes += fwrite(ptr+offset, 1, row\_size, fp); // ptr 배열로부터 1 byte (0 ~ 255) 크기의 값을 파일에 한 행씩 입력함.

offset += row\_size; // 다음 행의 첫 offset 을 지정함.

}

if(total\_bytes != total\_size) // total\_size 와 total\_bytes 가 같은 지 확인함으로써, 파일에 데이터를 쓰는 데에 문제가 없었는지 확인.

printf("Tried to write %ld bytes...Only wrote %ld\n",

total\_size, total\_bytes);

fclose(fp);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Func: pnm\_open \*

\* \*

\* Desc: opens a pnm file and determines rows, cols, and maxval \*

\* \*

\* Params: rows- pointer to number of rows in the image \*

\* cols - pointer number of columns in the image \*

\* maxval - pointer to max value \*

\* filename - name of image file \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

FILE \*pnm\_open(int \*rows, int \*cols, int \*maxval, char \*filename)

{

int firstchar, secchar;

float scale;

unsigned long row\_size;

FILE \*fp;

if((fp = fopen(filename, "rb")) == NULL) // 예외 처리.

{

printf("Unable to open %s for reading\n",filename);

exit(1);

}

firstchar = getc(fp); // 영상이 PPM 파일인지 확인하기 위해.

secchar = getc(fp); // 영상이 PBM 인지, PGM 인지, PPM 인지 확인하기 위해.

if(firstchar != 'P') // 첫글자가 'P' 가 아니면, 잘못된 입력임을 출력하고 프로그램을 종료.

{

printf("You silly goof... This is not a PPM file!\n");

exit(1);

}

\*cols = getnum(fp); // 가로 길이를 받음.

\*rows = getnum(fp); // 세로 길이를 받음.

switch(secchar)

{

case '4': /\* PBM \*/

scale = 0.125;

\*maxval = 1;

break;

case '5': /\* PGM \*/

scale = 1.0;

\*maxval = getnum(fp);

break;

case '6': /\* PPM \*/

scale = 3.0;

\*maxval = getnum(fp);

break;

default : /\* Error \*/

printf("read\_pnm: This is not a Portable bitmap RAWBITS file\n");

exit(1);

break;

}

row\_size = (\*cols) \* scale; // 한 행의 전체 바이트 크기.

return fp;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Func: read\_mesh \*

\* \*

\* Desc: reads mesh data into a mesh structure \*

\* \*

\* Params: filename - name of input mesh file \*

\* \*

\* Returns: mesh structure storing width, height, x data and y data \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

mesh \*read\_mesh(char \*filename)

{

FILE \*fp;

mesh \*mesh\_data;

int width, height, mesh\_size;

/\* open mesh file for input \*/

if((fp = fopen(filename, "rb")) == NULL)

{

printf("Unable to open mesh file %s for reading\n", filename);

exit(1);

}

mesh\_data = malloc(sizeof(mesh)); // mesh\_data 메모리 할당.

/\* read dimensions of mesh \*/ // 넓이, 높이를 파일로부터 입력받음.

fread(&width, sizeof(int), 1, fp);

fread(&height, sizeof(int), 1, fp);

mesh\_data->width = width; // mesh\_data 구조체 내에 있는 멤버변수 width에 read\_mesh 함수 내에 선언된 width 변수 값을 저장.

mesh\_data->height = height; // mesh\_data 구조체 내에 있는 멤버변수 height에 read\_mesh 함수 내에 선언된 height 변수 값을 저장.

mesh\_size = width \* height; // mesh\_size 를 저장.

/\* allocate memory for mesh data \*/

mesh\_data->x\_data = malloc(sizeof(float) \* mesh\_size);

mesh\_data->y\_data = malloc(sizeof(float) \* mesh\_size);

/\* mesh\_data 메모리 할당 후, 파일로부터 불러온 데이터 저장. \*/

fread(mesh\_data->x\_data, sizeof(float), mesh\_size, fp);

fread(mesh\_data->y\_data, sizeof(float), mesh\_size, fp);

return(mesh\_data);

}

1. **List2\_1.c**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* File: arithlut.c \*

\* \*

\* Desc: This program performs arithmetic point operations via LUTs \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

#include <string.h>

#include <malloc.h>

#include "ip.h"

#define operation(VALUE) ((float) VALUE \* 1.9)

extern void write\_pnm(image\_ptr ptr, char filein[], int rows,

int cols, int magic\_number);

extern image\_ptr read\_pnm(char \*filename, int \*rows, int \*cols,

int \*type);

int main(int argc, char \*argv[])

{

char filein[100]; /\* name of input file \*/

char fileout[100]; /\* name of output file \*/

int rows, cols; /\* image rows and columns \*/

unsigned long i; /\* counting index \*/

unsigned long bytes\_per\_pixel; /\* number of bytes per image pixel \*/

unsigned char LUT[256]; /\* array for Look-up table \*/

image\_ptr buffer; /\* pointer to image buffer \*/

unsigned long number\_of\_pixels; /\* total number of pixels in image \*/

int temp; /\* temporary variable \*/

int type; /\* what type of image data \*/

/\* set input filename and output file name \*/

if(argc == 3)

{

strcpy(filein, argv[1]);

strcpy(fileout, argv[2]);

}

else

{

printf("Input name of input file\n");

gets(filein); // 영상 파일 이름을 입력 받음.

printf("\nInput name of output file\n");

gets(fileout); // 새로 만들어질 맵핑될 영상 파일 이름을 입력 받음.

printf("\n");

}

buffer = read\_pnm(filein, &rows, &cols, &type); // 입력된 영상을 읽는 함수를 수행.

/\* initialize Look-up table \*/

for(i=0; i<256; i++)

{

temp = operation(i); // temp 임시 변수에 i \* 1.9 값을 저장. temp 는 정수형이기 때문에 소수점 이하의 값을 버림.

CLIP(temp, 0, 255); // 맵핑된 후의 밝기 레벨(L) 값이 0 보다 작거나, 255 보다 크면 안되기 때문에.

LUT[i] = temp; // LUT[i] 에 temp 값을 초기화.

}

/\* determine bytes\_per\_pixel, 3 for color, 1 for gray-scale \*/

if(type == PPM) // 한 픽셀당 bytes 크기를 지정함. PPM 은 3 bytes 그 이외에는 1 byte 로 설정함.

bytes\_per\_pixel = 3;

else

bytes\_per\_pixel = 1;

number\_of\_pixels = bytes\_per\_pixel \* rows \* cols; // PPM 의 경우, 총 픽셀 수 라기보다는 총 바이트 크기와 같음.

/\* process image via the Look-up table \*/

for(i=0; i<number\_of\_pixels; i++) // 모든 픽셀에 대해 Look-up Table 을 이용하여 밝기 레벨(L)을 새롭게 맵핑.

buffer[i] = LUT[buffer[i]];

write\_pnm(buffer, fileout, rows, cols, type);

IP\_FREE(buffer); // 메모리 해제

return 0;

}