6.Digital image Processing

Image Handling and File I/O

Dr.Weerayuth Suanpaga
(D.ENG RS&GIS)

Department of Civil Engineering
Faculty of Engineering , Kasetsart University
Bangkok, Thailand

https://pirun.ku.ac.th/~fengwys/ram/ge749/lect/lect6.pdf

typedef

```
typedef unsigned char      u_char;
typedef unsigned short     u_short;
u_char   pixel;
u_char   *line_image;
u_short  *radar_line_image;
u_short  *rada_pixel;
```

Allocate memory to a line image

```
#include<stdio.h>
#include<stdlib.h>
typedef unsigned char               u_char;
typedef unsigned short              u_short;
#define     W           256
u_char      *img1_alloc( int w);
void        img1_free( u_char * img );
main()
{

      u_char            *img;
      u_short           *short_img;
      float             *float_img;
      int               i;

      img = img1_alloc( W );
      for(i=0;i<W;i++)
            img[i] = i;
      img1_free( img );
/*
      short_img = (u_short * )
      img1_alloc( W*sizeof( u_short ) );
      float_img = (float * ) img1_alloc( W*sizeof( float ) );
*/
}
```

```
u_char    *img1_alloc( int w)
{
          u_char    *line_img;

          line_img = malloc(  w );
          return  line_img;
}

void      img1_free( u_char * img )
{
          free( img );
}
```

3

Allocate memory to a 1-channel image

```
#include<stdio.h>
#include<stdlib.h>
typedef unsigned char    u_char;
typedef unsigned short   u_short;
#define     W            256
#define     H            256
u_char      **img2_alloc( int w, int h );
void        img2_free( u_char **img, int h );
main()
{
      u_char  **img;
      u_short **short_img;
      float   **float_img;
      int     i, j;

      img = img2_alloc( W, H );
      if( img == NULL ){
            printf("Not enough memory¥n"); exit(1);
      }
      for(i=0;i<H;i++)
            for(j=0;j<W;j++)
                  img[i][j] = j;
      printf("img = %d¥n", img );
      img2_free( img, h );
/*
      short_img = (u_short ** ) img2_alloc( W*sizeof( u_short ), H );
      float_img = (float ** ) img2_alloc( W*sizeof( float ), H );
*/

}
```

```
u_char      **img2_alloc( int w, int h )
{
      u_char  **img;
      int     i,j;

      if( (img = malloc( sizeof( u_char * ) * h ) )
      == NULL ){
            return NULL;
      }
      for(i=0;i<h;i++){
            img[i] = malloc( w );
            if( img[i] == NULL ){
                  for(j=0;j<i;j++)

            free( img[i] );
                  free( img );
                  return NULL;
            }
      }
      return img;
}

void  img2_free( u_char **img, int h )
{
      int     i;
      for(i=0;i<h;i++)
            free( img[i] );
}
```

4

Allocate memory to multi channel image

```
#include<stdio.h>
#include<stdlib.h>
typedef unsigned charu_char;
typedef unsigned short          u_short;
#define     W           256
#define     H           256
#define     NCHAN       3
u_char      ***img_alloc( int w, int h, int nchan );
void        img_free( u_char ***img, int h, int nchan );
u_char      **img2_alloc( int w, int h );
void        img2_free( u_char **img, int h );
main()
{
    u_char***img;
    u_short             ***short_img;
    float   ***float_img;
    int     i, j, ichan;

    img = img_alloc( W, H, NCHAN );
    if( img == NULL ){
            printf("Not enough memory¥n"); exit(1);
    }
    for(ichan=0;ichan<NCHAN;ichan++)
            for(i=0;i<H;i++)
                    for(j=0;j<W;j++)
                            img[ichan][i][j] = j;
    img_free( img, H, NCHAN );
/*  short_img = (u_short *** ) img_alloc( W*sizeof( u_short ),
    H, NCHAN);
    float_img = (float *** ) img_alloc( W*sizeof( float ), H,
    NCHAN);         */
}
```

```
u_char      ***img_alloc( int w, int h, int nchan
{
    u_char***img;
    int     ichan,j;

    if( (img = malloc( sizeof( u_char ** ) *
    nchan ) ) == NULL ){
            return NULL;
    }
    for(ichan=0;ichan<nchan;ichan++){
            img[ichan] = img2_alloc( w, h );
            if( img[ichan] == NULL ){
                    for(j=0;j<ichan;j++)
                            img2_free( img[j], h );
                    free( img );
                    return NULL;
            }
    }
    return img;
}

void  img_free( u_char ***img, int h, int nchan )
{
    int     i;
    for(i=0;i<nchan;i++)
            img2_free( img[i], h );
}
```

5

FILE I/O

We can open/create files, read /write from/to file through a pointer to FILE.

FILE *fopen(const char *filename, const char *mode);

int fclose(FILE *stream);

Text mode I/O: fprintf, fscanf

```
#include<stdio.h>
 void main ( void ) {
    FILE *fp;
    double x = 3.14;

    fp = fopen( “data.txt”, “wt” );
    if( fp = = NULL ) {
            fprintf(stderr, “file open error for
writing¥n”);
            return 1;
    }
    fprintf(fp,”lf”, x );
    fclose( fp );
}
```

```
#include<stdio.h>
void main ( void ) {
    FILE *fp;
    double x;

    fp = fopen( “data.txt”, “rt” );
    if( fp = = NULL ) {
            fprintf(stderr, “file open error for
reading¥n”);
            return 1;
    }
    fscaff(fp,”lf”, &x );
    printf(“x = %lf¥n”, x );
    fclose( fp );
}
```

FILE I/O -2

Binary FILE I/O

size_t fwrite(const void*buffer, size_t size, size_t count, FILE *stream);

size_t fread(void *buffer, size_t size, size_t count, FILE *stream);

int fseek(FILE *stream, long offset, int origin);

```
#include<stdio.h>
 void main ( void ) {
    FILE *fp;
    u_char *line_img;
 .......
    fp = fopen( "test.img", "wb" );
    if( fp = = NULL ) {
            fprintf(stderr, "file open error for writing¥n");
            return 1;
    }
    fwrite( line_img, w, 1, fp  );
    fclose( fp );
}
```

```
#include<stdio.h>
 void main ( void ) {
    FILE *fp;
    u_char *line_img;
    .......
    fp = fopen( "test.img", "rb" );
    if( fp = = NULL ) {
            fprintf(stderr, "file open error for reading ¥n");
            return 1;
    }
    fread( line_img, w, 1, fp  );
    fclose( fp );
}
```


7

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

typedef unsigned char       u_char;
typedef unsigned short      u_short;
#define W       256
#define         H               256

u_char          **img2_alloc(int w,int h);
void img2_free(u_char**img,int h);

void main()
{
        u_char  **img;
        int                     i,j;
        FILE  *fp;
        char  *fname="testimg3.txt";


        img=img2_alloc(W,H);
        if(img==NULL){
                printf("Not enough memory\n");
                exit(1);
        }

        for(i=0;i<H;i++)
                for(j=0;j<W;j++)
                                                //img[i][j]=H-i;

                img[i][j]=W/2+W/2*sin((double)j/W*22./7.);

        fp=fopen(fname,"wb");

        if (fp==NULL){
                printf("Can not open this image file\n");
                exit(0);
        }


        for (i=0;i<H;i++)
                fwrite(img[i],W,1,fp);

        fclose(fp);

        printf("***** write image file name testimg.img complete\n*****");
        img2_free(img,H);
}

u_char          **img2_alloc(int w,int h)
{
        u_char          **img;
        int                             i,j;

        if((img=malloc(sizeof(u_char*)*h))==NULL){
                return NULL;
        }

        for(i=0;i<h;i++){
                img[i]=malloc(w);
                if(img[i]==NULL){
                        for(j=0;j<i;j++)
                                free(img[i]);
                free(img);
                return NULL;
                }
        }
        return img;
}
void img2_free(u_char**img,int h)
{
        int             i;
        for(i=0;i<h;i++)
                free(img[i]);

}
```

image operation 1

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
typedef unsigned char     u_char;
typedef unsigned short    u_short;
#define       W           256
#define       H           256
#define       NCHAN       3
void img_setup( u_char ***img, int w, int h, int nchan );
int   image_write( char *fname, u_char ***img, int w, int h,
      int nchan);
u_char        ***img_alloc( int w, int h, int nchan );
void img_free( u_char ***img, int h, int nchan );
u_char        **img2_alloc( int w, int h );
void img2_free( u_char **img, int h );
main()
{

      u_char  ***img;

      img = img_alloc( W, H, NCHAN );
      if( img == NULL ){
            printf("Not enough memory¥n"); exit(1);
      }

      img_setup( img, W, H, NCHAN );
      image_write( "test.img", img, W, H, NCHAN);

      img_free( img, H, NCHAN );
}
```

```
#define       PI 3.14159265
void img_setup( u_char ***img, int w, int h, int nchan )
{
      int       i, j, ichan;
      for(ichan=0;ichan<nchan;ichan++){
            printf("%d¥n",ichan);
            for(i=0;i<h;i++) {
                  for(j=0;j<w;j++) {
                        img[ichan][i][j] =
sin((double)j/(w*(1+ichan*0.5))*2.*PI*(10.0*i/h))*128+128;
                  }
            }
      }
      printf("img_setup end¥n");
}
int   image_write( char *fname, u_char ***img, int w, int h,
      int nchan)
{
      FILE    *fp;
      int       i, ichan;

      if((fp=fopen(fname,"wb")) == NULL ){
            fprintf(stderr,"image_write:file open error");
            return          1;
      }
      printf("file open end¥n");
      for(ichan=0;ichan<nchan;ichan++)
            for(i=0;i<h;i++)
                  fwrite( img[ichan][i], w, 1, fp );
                  fclose(fp );
      return 0;
}
```


image operation 2

```
#include<stdio.h>
#include<stdlib.h>
typedef unsigned char     u_char;
#define       W           256
#define       H           256
#define       NCHAN       3
#define       LUT_SIZE    256

void lut_setup( u_char *lut );
void image_conv( u_char ***img, int w, int h, int nchan,
      u_char *lut );
int   image_read( char *fname, u_char ***img, int w, int h,
      int nchan);
/* other prototype declarations here     */
main()
{
      u_char  ***img;
      u_char  lut[LUT_SIZE];

      lut_setup( lut );
      img = img_alloc( W, H, NCHAN );
      if( img == NULL ){
            printf("Not enough memory¥n"); exit(1);
      }
      image_read( "test.img", img, W, H, NCHAN);
      image_conv( img, W, H, NCHAN, lut );
      image_write( "test2.img", img, W, H, NCHAN);

      img_free( img, H, NCHAN );

}
```

```
void lut_setup( u_char *lut )
{
      int       i;
      for(i=0;i<LUT_SIZE;i++)
            lut[i] = LUT_SIZE-i-1;
}
void image_conv( u_char ***img, int w, int h, int nchan,
      u_char *lut )
{
      int       i, j, ichan;
      for(ichan=0;ichan<nchan;ichan++)
          for(i=0;i<h;i++)
            for(j=0;j<w;j++)
                img[ichan][i][j] = lut[img[ichan][i][j]];
}
int   image_read( char *fname, u_char ***img, int w, int h, int
      nchan)
{
      FILE    *fp;
      int       i, ichan;
      if((fp=fopen(fname,"rb")) == NULL ){
            fprintf(stderr,"image_read:file open error");
            return          1;
      }
      printf("image_read: file open end¥n");
      for(ichan=0;ichan<nchan;ichan++){
            for(i=0;i<h;i++){
                  fread( img[ichan][i], w, 1, fp );
            }
      }
      fclose(fp );
      return 0;
}
```

Structure

```
struct IMAGE  {
  int w, h, nchan;
  unsigned char ***data;
};
struct IMAGE  image;
image.w = 1024;
image.h = 1024;
struct IMAGE  *imagep;
image = malloc( sizeof( struct IMAGE ) );
image->w = 1024;
image->h = 1024;
```


typedef 2

```
struct IMAGE_STRUCT{
  int w, h, nchan;
u_char ***data;
};

typedef struct IMAGE_STRUCT  IMAGE;

IMAGE *image;
image = malloc( sizeof ( IMAGE ) );
image->w = 1024;
image->h = 1024;
image->nchan = 3;
image->data = image_alloc(image->w ,
  image->h, image->nchan );
```

Exercise

- Confirm the output image of “Image Operation 1” using ENVI.
- Make your own color pattern
- Prepare a 1 channel image and apply “Image Operation 2”
- Make a Pseude Color Example

Reference:

Assoc.Prof.Dr.HONDA Kiyoshi, Lecture Note .RS and GIS Field of study,School of Engineering and Technology ,AIT Thailand.2005

Thank you for your kind attention