```
1
2 #include <math.h>
 3 #include "tiff.h"
4 #include "allocate.h"
5 #include "randlib.h"
6 #include "typeutil.h"
8 void error(char *name);
9
10 // sort array
11 void sort(int *pixel, int *weight, int length)
12 {
13
       int i, j;
14
       int v;
15
       for(i = 0; i < length - 1; i ++)
16
           for(j = i+1; j < length; j ++)
17
                if(pixel[i] < pixel[j])</pre>
18
19
                {
20
                    v = pixel[i];
21
                    pixel[i] = pixel[j];
22
                    pixel[j] = v;
23
24
                    v = weight[i];
25
                    weight[i] = weight[j];
26
                    weight[j] = v;
27
                }
28
           }
29 }
30
   // input a pixel in a image, output the weighted median
32 int WMfilter(struct TIFF_img input_img, int x, int y)
33 | {
       int pixel[25]; // pixels by 5*5 filter
34
35
       int weight[25] = {1,1,1,1,1,1,2,2,2,1,1,2,2,2,1,1,2,2,2,1,1,1,1,1,1,1}; //
   weighted factors
36
       int i, j, k=0;
       // int istar; // output
37
       // get 5*5 window from image
38
39
       for (i=x-2; i<=x+2; i++)
40
41
           for (j=y-2; j<=y+2; j++)
42
43
                pixel[k] = input_img.mono[i][j];
44
                k = k + 1;
45
           }
46
       // sort the pixel and weight
47
48
       sort(pixel,weight,25);
49
       // find i*
       int temp = 0;
50
51
       for (i=0; i<25; i++)
52
53
           temp = temp + weight[i];
54
           if (temp >= 17)
55
           {
56
                break;
57
           }
58
       return pixel[i];
59
```

localhost:4649/?mode=clike 1/3

2021/4/5

```
60 }
 61
 62
 63
 64 int main (int argc, char **argv)
 65 {
        FILE *fp;
 66
 67
        struct TIFF_img input_img, output_img;
 68
        // double **img1,**img2;
 69
        int32_t i,j;
 70
 71
        printf("processstart");
 72
 73
        if ( argc != 2 ) error( argv[0] );
 74
 75
        /* open image file */
        if ( ( fp = fopen ( argv[1], "rb" ) ) == NULL ) {
 76
 77
        fprintf ( stderr, "cannot open file %s\n", argv[1] );
 78
        exit ( 1 );
 79
        }
 80
        /* read image */
 81
        if ( read_TIFF ( fp, &input_img ) ) {
 82
 83
        fprintf ( stderr, "error reading file %s\n", argv[1] );
 84
        exit ( 1 );
 85
 86
        /* close image file */
 87
 88
        fclose (fp);
 89
        /* check the type of image data */
 90
 91
        if ( input_img.TIFF_type != 'g' ) {
        fprintf ( stderr, "error: image must be grey scale\n" );
 92
 93
        exit ( 1 );
 94
        }
 95
        /* Allocate image of double precision floats */
 96
 97
        // img1 = (double **)get img(input img.width,input img.height,sizeof(double));
 98
        // img2 = (double **)get_img(input_img.width,input_img.height,sizeof(double));
99
        /* set up structure for output achromatic image */
100
101
        /* to allocate a full color image use type 'c' */
        get_TIFF ( &output_img, input_img.height, input_img.width, 'g' );
102
103
104
105
        // copy image component to double array
106
        for ( i = 0; i < input_img.height; i++ )</pre>
107
        for ( j = 0; j < input_img.width; j++ ) {
108
        img1[i][j] = input_img.mono[i][j];
109
        */
110
111
        /* Filter image */
112
113
        for ( i = 2; i < input_img.height-2; i++ )</pre>
        for ( j = 2; j < input_img.width-2; j++ ) {
114
        output_img.mono[i][j] = WMfilter(input_img,i,j);
115
116
        }
117
118
        /*fill in boundary pixels*/
        for (i = 0; i < 2; i++)
119
```

localhost:4649/?mode=clike 2/3

2021/4/5

```
for (j = 2; j < 2; j++) {
120
121
        output_img.mono[i][j] = 0;
122
123
        for ( i = input_img.height-2; i < input_img.height; i++ )</pre>
124
        for ( j = input img.width-2; j < input img.width; j++ ) {
125
        output_img.mono[i][j] = 0;
126
127
        for (i = 0; i < 2; i++)
        for ( j = input_img.width-2; j < input_img.width; j++ ) {</pre>
128
129
        output img.mono[i][j] = 0;
130
        }
131
        for ( i = input_img.height-2; i < input_img.height; i++ )</pre>
132
        for (j = 2; j < 2; j++) {
133
        output_img.mono[i][j] = 0;
134
        }
135
136
        /* open output image file */
        if ( ( fp = fopen ( "output.tif", "wb" ) ) == NULL ) {
137
138
        fprintf ( stderr, "cannot open file output.tif\n");
139
        exit ( 1 );
140
        }
141
        /* write output image */
142
143
        if ( write_TIFF ( fp, &output_img ) ) {
144
        fprintf ( stderr, "error writing TIFF file %s\n", argv[2] );
145
        exit ( 1 );
146
147
148
        /* close output image file */
149
        fclose (fp);
150
151
        /* de-allocate space which was used for the images */
152
        free_TIFF ( &(input_img) );
153
        free_TIFF ( &(output_img) );
154
        //free_TIFF ( &(color_img) );
155
156
        //free img( (void**)img1 );
157
        //free img( (void**)img2 );
158
159
        printf("processsuccess");
160
        return(0);
161
162 }
163
164 void error(char *name)
165 {
166
        printf("usage: %s image.tiff \n\n",name);
        printf("this program reads in a 24-bit color TIFF image.\n");
167
        printf("It then horizontally filters the green component, adds noise,\n");
168
169
        printf("and writes out the result as an 8-bit image\n");
170
        printf("with the name 'green.tiff'.\n");
171
        printf("It also generates an 8-bit color image,\n");
        printf("that swaps red and green components from the input image");
172
173
        exit(1);
174 }
175
```

localhost:4649/?mode=clike 3/3