ECE637 Lab report 7
Image Restoration

Name:Chengzhang Zhong

Section 1 . Minimum Mean Square Error (MMSE) Linear Filters

1. The plot of original images

Figure 1.1 img14g.tif

Figure 1.2 img14bl.tif

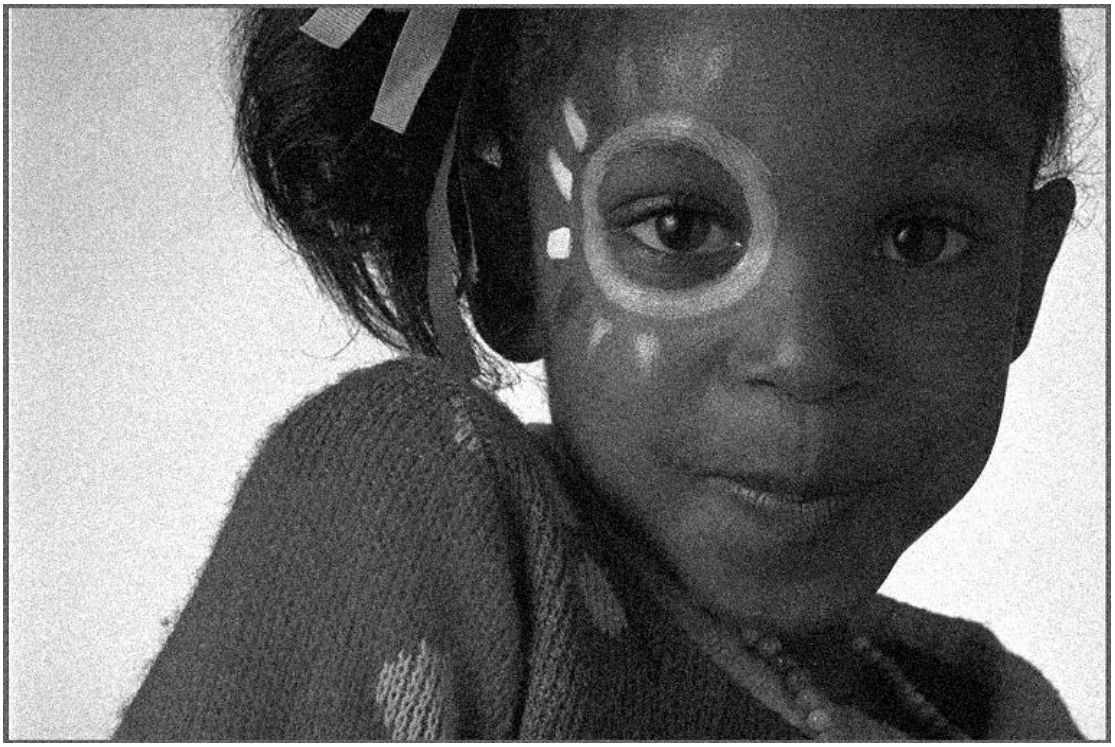

Figure 1.3 img14gn.tif

Figure 1.4 img14sp.tif

2. Output of filtered two noisy images

Figure 1.5 output image for img14bl.tif

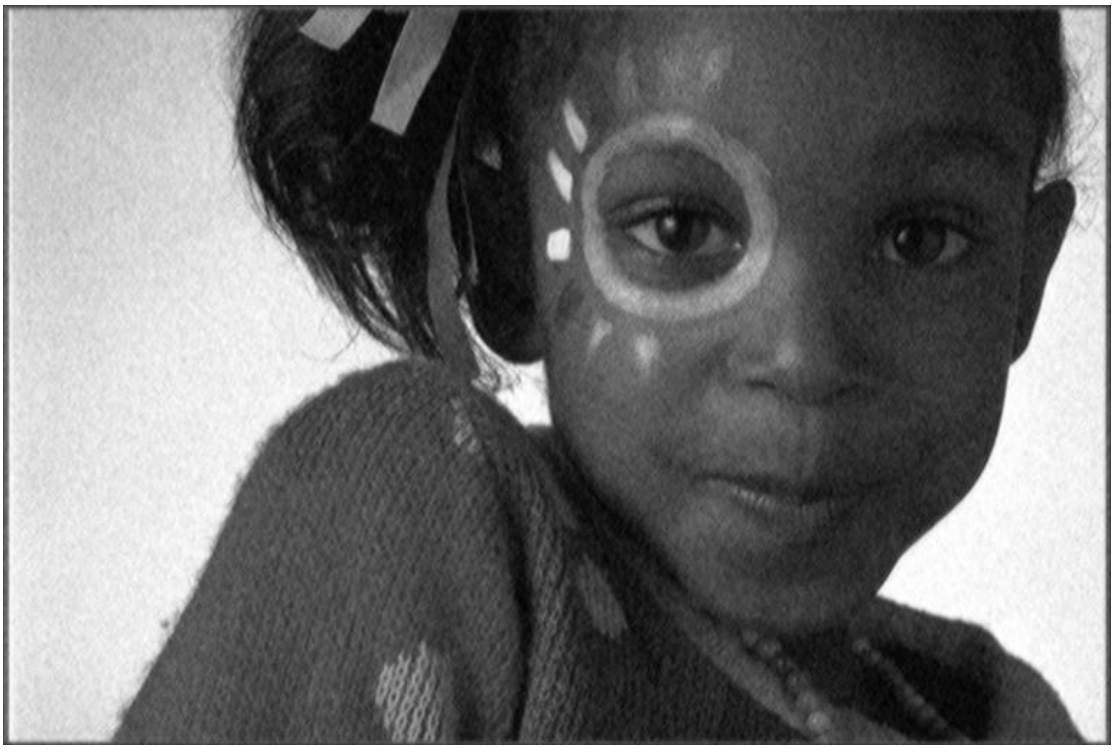

Figure 1.6 output image for img14gn.tif

Figure 1.7 output image for img14sp.tif

The matrix 7x7 of the filter:

	1	2	3	4	5	6	7
1	0.3720	0.2052	-0.9682	1.0572	0.1961	-1.0020	0.9254
2	-0.0431	0.4069	-1.2219	-0.0280	-0.6146	-1.3229	0.4024
3	-0.3541	-0.3242	-0.4810	0.3321	0.7580	-0.0871	-0.7923
4	1.1089	-2.4308	1.9317	3.7782	1.5691	-0.0701	0.0615
5	0.3791	-0.4590	-1.1045	1.2263	0.8358	-1.4710	0.3905
6	-1.0990	-0.1802	-0.2944	1.0624	-1.8928	-1.9628	0.8127
7	1.1560	0.4776	-1.7439	0.6483	0.2949	0.2604	0.3042
8							

Figure 1.8 matrix for img14bl.tif

7x7 double

	1	2	3	4	5	6	7
1	0.0165	0.0259	0.0044	0.0050	-0.0080	0.0302	-0.0259
2	-0.0055	0.0053	0.0355	0.0205	0.0464	0.0091	0.0066
3	-0.0105	-0.0125	0.0674	0.0731	0.0470	0.0290	-0.0030
4	-0.0091	-0.0153	0.0476	0.2306	0.0891	-0.0175	0.0011
5	-0.0050	-0.0222	0.0423	0.1117	0.0650	-0.0118	0.0069
6	-0.0044	0.0079	0.0307	0.0268	0.0088	-0.0063	0.0192
7	-0.0053	-0.0043	0.0154	0.0127	0.0140	0.0183	0.0054
8							

Figure 1.9 matrix for img14gn.tif

	1	2	3	4	5	6	7
1	0.0080	0.0048	-0.0016	-0.0050	0.0257	-0.0209	-0.0185
2	0.0017	-0.0016	0.0558	0.0267	0.0435	0.0214	0.0196
3	-0.0010	0.0042	0.0413	0.0968	0.0212	-0.0196	0.0199
4	-0.0014	-0.0203	0.0350	0.2652	0.1492	-0.0287	0.0083
5	0.0252	0.0023	0.0612	0.0965	0.0154	-0.0412	0.0233
6	-0.0099	-6.0704...	0.0313	0.0497	0.0143	0.0038	0.0131
7	-0.0407	0.0162	-0.0068	0.0100	0.0079	0.0129	-0.0110
8							

Figure 1.10 matrix for img14sp.tif

Session 2. Weighted Median Filter

1. Results:

Figure 2.1 matrix for img14sp.tif

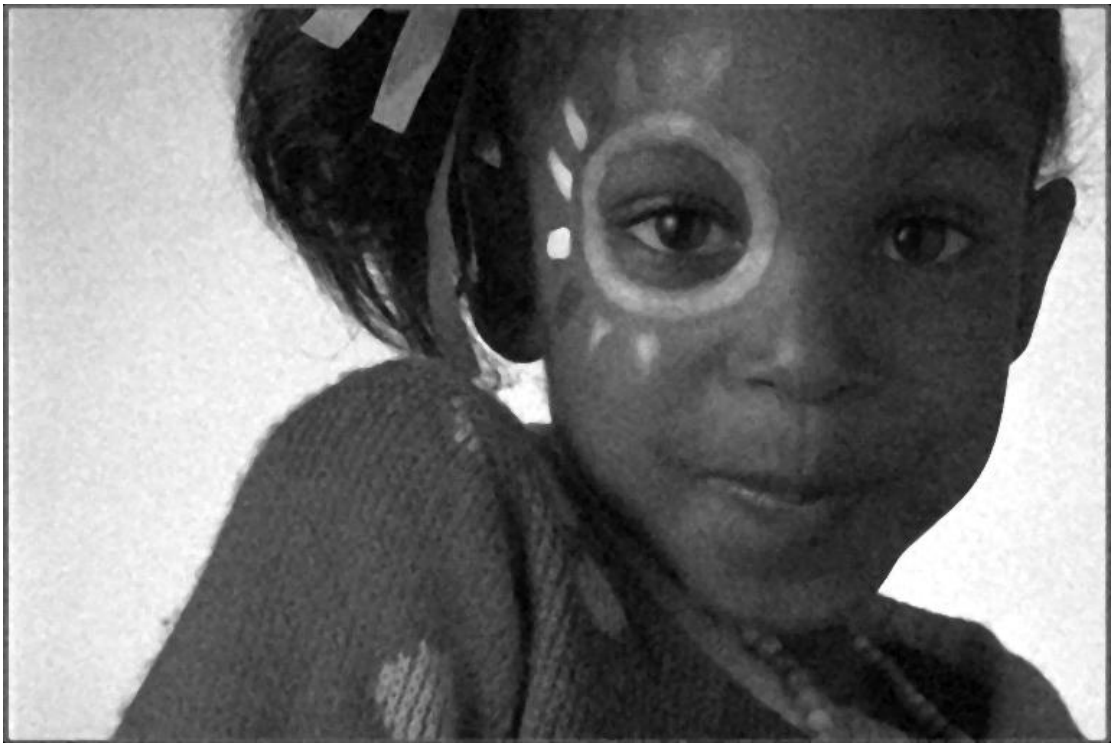

Figure 2.2 matrix for img14gn.tif

2. C Code

1.Struct:

```
struct loc
{
    int m;
    int n;
};

struct arow
{
    int weight;
    int value;
};
```

2.Filter function:

```
#include <math.h>
#include "tiff.h"
#include "allocate.h"
#include "typeutil.h"
```

```c
#include <stdio.h>
#include <stdlib.h>
#include "part2.h"



void median_weighted(struct loc pixel,unsigned char ** input, unsigned char **
output)
{

    struct arow sequence[25];
    int i,j,a=0,total = 0,add = 0;
    int n = 0;
    for (i = -2;i<3;i++)
    {
        for(j = -2;j<3;j++)
        {
            sequence[n].value = input[(pixel.m + 2)+i][(pixel.n+2)+j];
            if (i ==-2 || i==2 || j ==-2 || j==2)
            {
                sequence[n].weight = 1;
            }
            else
            {
                sequence[n].weight = 2;
            }
            n++;
        }
    }

   for (i = 0; i < 25; i++)
   {
       for (j = i + 1; j < 25; j++)
       {
           if (sequence[i].value < sequence[j].value)
           {
               a = sequence[i].value;
               sequence[i].value = sequence[j].value;
               sequence[j].value = a;

                 a = sequence[i].weight;
                 sequence[i].weight= sequence[j].weight;
               sequence[j].weight = a;
```

```
            }
        }
    }

    total = 18+16;
    for (i = 0; i < 25; i++)
    {
        for (j = i + 1; j < 25; j++)
        {
            add=add + sequence[j].weight;

        }
         if (total - add >= add)
             {
                 break;
             }
         else
         {
             add = 0;
         }
    }

    output[pixel.m][pixel.n] = sequence[i].value;
}
```

3.Mean function

```
#include <math.h>
#include "tiff.h"
#include "allocate.h"
#include "typeutil.h"
#include <stdio.h>
#include <stdlib.h>
#include "part2.h"

int main (int argc, char **argv)
{
  FILE *fp;
  struct TIFF_img input_img,result_img;

 // double **img1,**img2;
  unsigned char** img,** out_img;

  int i,j;
```

```
  struct loc pix;

  /* open image file */
  if ( ( fp = fopen ( argv[1], "rb" ) ) == NULL ) {
    fprintf ( stderr, "cannot open file %s\n", argv[1] );
    exit ( 1 );
  }

  /* read image */
  if ( read_TIFF ( fp, &input_img ) ) {
    fprintf ( stderr, "error reading file %s\n", argv[1] );
    exit ( 1 );
  }
  /* close image file */
  fclose ( fp );

  /* Allocate image of double precision floats */
  get_TIFF(&result_img, input_img.height, input_img.width, 'g');
  img = (unsigned char
**)get_img(input_img.width+4,input_img.height+4,sizeof(double));
  out_img = (unsigned char
**)get_img(input_img.width,input_img.height,sizeof(double));
  for ( i = 0; i < input_img.height+4; i++ )
  {
      for ( j = 0; j < input_img.width+4; j++ )
      {
        img[i][j] = 0;
       }
  }


  /* copy input image to a larger border image*/
  for ( i = 0; i < input_img.height; i++ )
  {
      for ( j = 0; j < input_img.width; j++ )
      {
        img[i+2][j+2] = input_img.mono[i][j];

       }
  }

  for ( i = 0; i < input_img.height; i++ )
  {
```

```
    for ( j = 0; j < input_img.width; j++ )
    {
      pix.m = i;
      pix.n = j;
      median_weighted(pix,img,out_img);
     }
}


for ( i = 0; i < input_img.height; i++ )
{
  for ( j = 0; j < input_img.width; j++ )
  {
      result_img.mono[i][j] = out_img[i][j];
  }
}


/* close green image file */
fclose ( fp );


/* open result image file */
if ( ( fp = fopen ( "result_img.tif", "wb" ) ) == NULL ) {
    fprintf ( stderr, "cannot open file color.tif\n");
    exit ( 1 );
}

/* write result image */
if ( write_TIFF ( fp, &result_img ) ) {
    fprintf ( stderr, "error writing TIFF file %s\n", argv[2] );
    exit ( 1 );
}


/* close result image file */
fclose ( fp );

/* de-allocate space which was used for the images */
free_TIFF ( &(input_img) );
free_TIFF ( &(result_img));

free_img( (void**)img );
free_img( (void**)out_img );
```

```
  return(0);
}
```