二、实验内容和原理

1、图像平移

公式如下。

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{9}$$

2、图像翻转

公式如下。

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{10}$$

当Sx = 1，且Sy = -1时实现绕x轴的镜像变换；当Sx = -1，且Sy = 1时实现绕y轴的镜像变换。

3、图像剪切

公式如下。

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & d_x & 0 \\ d_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{11}$$

当 $dx \neq 0, dy = 0$ 时实现沿x轴的剪切；当 $dx = 0, dy \neq 0$ 时实现沿y轴的剪切。

4、图像旋转

公式如下。

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{12}$$

θ 是旋转的角度。

5、图像大小变换

公式如下。

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} c & 0 & 0 \\ 0 & d & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{13}$$

6、插值

在图像的旋转（rotation）和大小变换（scale）中，有效像素点的个数会增加，但是由于原始图像像素点个数有限，因此操作后会出现空洞的问题，所以我们需要对其插值。

插值有两种，一种是线性插值，即对目标像素点做反变换，在原始图像中进行计算获得最可能的像素值；另一种是最近邻插值，就是选取最靠近反变换的像素点的像素值作为结果值。

选取靠近(x,y)的四个点，解决下面这个方程就可以得到目标像素值的插值值。

$$\begin{bmatrix} x_1 & y_1 & x_1y_1 & 1 \\ x_2 & y_2 & x_2y_2 & 1 \\ x_3 & y_3 & x_3y_3 & 1 \\ x_4 & y_4 & x_4y_4 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} \tag{14}$$

在本次实验中，如果我们选择像素点的上下左右四个点作为确定点，那么我们所需要的解就等于d，就能退化为g1, g2, g3, g4 的加权平均值。还可以推广得到，也可以增加到周围的8个点的加权平均。但实验中线性插值效果差距不大，线性插值的锐度会大于最近邻插值的锐度。

$$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & -1 & 0 & 1 \\ -1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} \tag{15}$$

$$g(0,0) = d = \frac{g_1 + g_2 + g_3 + g_4}{4} \tag{16}$$

三、实验步骤与分析

1、读入灰度图

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

typedef struct FILEHEADER{
    unsigned int   bfSize;          // Size of Figure
    unsigned short bfReserved1;     // Reserved1, must be 00
    unsigned short bfReserved2;     // Reserved2, must be 00
    unsigned int   bfOffBits;       // Offsets from header to info
}BMPFILEHEADER;

typedef struct INFOHEADER{
    unsigned int   biSize;          // Size of this structure
    unsigned int   biWidth;         // Width
    unsigned int   biHeight;        // Height
    unsigned short biPlanes;        // always is 1
    unsigned short biBitCount;      // Kind of BMP, bit/pixel = 1 4 8
16 24 32
```

```
    unsigned int   biCompression;
    unsigned int   biSizeImage;
    unsigned int   biXPelsPerMeter;
    unsigned int   biYPelsPerMeter;
    unsigned int   biClrUsed;
    unsigned int   biClrImportant;
}BMPINFOHEADER;

typedef struct YUVSPACE{
    unsigned char Y;
    unsigned char U;
    unsigned char V;
}YUV;

int main(){
    FILE *infp,*trfp;

    unsigned short bftype;
    // head includes head information of BMP file.
    BMPFILEHEADER head;
    // info includes figure information of BMP file.
    BMPINFOHEADER info;

    // read F1.bmp
    if((infp=fopen("RGB2YUV.bmp","rb")) == NULL){
        printf("OPEN FAILED!\n");
        return 0;
    }

    // bftype must be 0x4d42.
    fread(&bftype,1,sizeof(unsigned short),infp);
    if(bftype != 0x4d42){
        printf("This figure is not BMP!\n");
        Sleep(1000);
        return 0;
    }

    // read HEADER
    fread(&head,1,sizeof(BMPFILEHEADER),infp);
    fread(&info,1,sizeof(BMPINFOHEADER),infp);

    // We takes 24 bitcount bmp for experiment item.
    if(info.biBitCount != 24){
        printf("This BMP is not 24 biBitCount!\n");
        Sleep(1000);
```

```
        return 0;
    }
//////////////////////////////////////////////////////////////////////
//////////////
    // size, width, height.
    int size = head.bfSize / 3;
    int width = info.biWidth;
    int height = info.biHeight;

    // define array.
    YUV yuv[height][width];

    // read information.
    fread(yuv,size,sizeof(YUV),infp);
    fclose(infp);


```

2、Translation

```
//////////////////////////////////////////////////////////////////////
//////////////
    // define x0, y0.
    int x0 = 100;
    int y0 = 100;

    // Translation.
    int newwidth  = info.biWidth + x0;
    int newheight = info.biHeight + y0;
    int newsize   = newwidth * newheight;

    // define array.
    YUV tr[newheight][newwidth];

    int i,j;

    // Initialization.
    for(i=0;i<=newheight-1;i++){
        for(j=0;j<=newwidth-1;j++){
            tr[i][j].Y = 0;
        }
    }

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

```
25          for(j=0;j<width;j++){
26              tr[i+x0][j+y0].Y = yuv[i][j].Y;
27          }
28      }
29
30  ////////////////////////////////////////////////////////////////////
    //////////////
31      // Output BMP.
32      if((trfp = fopen("Translation.bmp","wb")) == NULL){
33          printf("Create image FAILED!\n");
34          Sleep(1000);
35          return 0;
36      }
37
38      BMPFILEHEADER newhead;
39      // info includes figure information of BMP file.
40      BMPINFOHEADER newinfo;
41
42      newhead = head;
43      newinfo = info;
44
45      newhead.bfSize = newsize * 3;
46      newinfo.biHeight = newheight;
47      newinfo.biWidth  = newwidth;
48
49      // Write header information and data into file.
50      fwrite(&bftype,1,sizeof(unsigned short),trfp);
51      fwrite(&newhead,1,sizeof(BMPFILEHEADER),trfp);
52      fwrite(&newinfo,1,sizeof(BMPINFOHEADER),trfp);
53
54      for(i=0;i<=newheight-1;i++){
55          for(j=0;j<=newwidth-1;j++){
56              fwrite(&tr[i][j].Y,1,sizeof(unsigned char),trfp);
57              fwrite(&tr[i][j].Y,1,sizeof(unsigned char),trfp);
58              fwrite(&tr[i][j].Y,1,sizeof(unsigned char),trfp);
59          }
60      }
61
62      fclose(trfp);
63      printf("Output Image successfully!\n");
64  ////////////////////////////////////////////////////////////////////
    //////////////
65
```

3、Mirror

```
    // Mirror operation.
    for(i=0;i<height;i++){
        for(j=0;j<width;j++){
            mi[i][width-1-j].Y = yuv[i][j].Y;
        }
    }
```

4、Shear

```
    float d = x0 * 1.0000 / (height-1);

    // Shear Operation.
    for(i=0;i<height;i++){
        for(j=0;j<width;j++){
            sh[i+y0][j+(int)(d*i)].Y = yuv[i][j].Y;
        }
    }
```

5、插值

```
int lip(YUV yuv[512][512], int x, int y){
    int ans = (yuv[y-1][x-1].Y + yuv[y+1][x+1].Y + yuv[y+1][x-1].Y + yuv[y-1][x+1].Y);
    ans = ans + yuv[y][x-1].Y + yuv[y][x+1].Y + yuv[y+1][x].Y + yuv[y-1][x].Y;
    ans = ans / 8;
    return ans;
}
```

6、Rotation

```
    // caculate cos(theta) and sin(theta).
    double cosa = cos(theta);
    double sina = sin(theta);

    // Rotation's changing constants.
    double const1= - width * cosa / 2.0 - height * sina / 2.0 + newwidth / 2.0;
    double const2=   width * sina / 2.0 - height * cosa / 2.0 + newheight / 2.0;

    // Filling the holes changing constants.
```

```
    double const3= - newwidth * cosa / 2.0 + newheight * sina / 2.0 +
width / 2.0;
    double const4= - newwidth * sina / 2.0 - newheight * cosa / 2.0 +
height / 2.0;

    // two temp.
    double xtemp;
    double ytemp;

    int x1,y1;

    // Rotation.
    for(i=0;i<height;i++){
      for(j=0;j<width;j++){
          xtemp =    j * 1.00000 * cosa + i * 1.00000 * sina +
const1;
          ytemp =   -j * 1.00000 * sina + i * 1.00000 * cosa +
const2;

          x1 = (int)(xtemp + 0.5);
          y1 = (int)(ytemp + 0.5);

          rot[y1][x1].Y = yuv[i][j].Y;
        }
    }

    // Fill the holes.
    for(i=1;i<newheight;i++){
        for(j=1;j<newwidth;j++){
            xtemp =   j * cosa - i * sina + const3;
            ytemp =   j * sina + i * cosa + const4;

            x1 = (int)(xtemp+0.5);
            y1 = (int)(ytemp+0.5);

            if(rot[i][j].Y == 255)
                if(x1>=1 && x1<width){
                if(y1>=1 && y1<height){
                  //rot[i][j].Y = yuv[y1][x1].Y;  // nearest
interpolation.
                    rot[i][j].Y = lip(yuv,x1,y1); // linear
interpolation.
                }
            }
        }
```

```
    }

////////////////////////////////////////////////////////////////////////////
//////////////
```

7、Scale

```
////////////////////////////////////////////////////////////////////////////
//////////////

    // Translation.
    int newwidth,newheight;
    int newsize;

    int x0 = 600;
    int y0 = 0;

    newwidth  = width  + x0;
    newheight = height + y0;
    newsize   = newwidth * newheight;

    // define array.
    YUV sc[newheight][newwidth];

    int i,j;

    for(i=0;i<newheight;i++){
        for(j=0;j<newwidth;j++){
            sc[i][j].Y = 255;
        }
    }

    double x_pre;
    double y_pre;
    int x1;
    int y1;
    double dw = (width - 1)  *1.00000 / (newwidth - 1);
    double dh = (height - 1) *1.00000 / (newheight - 1);
    printf("%lf %lf \n",dw,dh);

    for(i=0;i<=height-1;i++){
      for(j=0;j<=width-1;j++){
          x_pre =  j * 1.00000 / dw;
          y_pre =  i * 1.00000 / dh;
          x1 = (int)(x_pre + 0.5);
```

```
            y1 = (int)(y_pre + 0.5);
            sc[y1][x1].Y = yuv[i][j].Y;
        }
    }

    for(i=0;i<newheight;i++){
        for(j=0;j<newwidth;j++){
            x_pre =  j * dw;
            y_pre =  i * dh;

            x1 = (int)(x_pre + 0.5);
            y1 = (int)(y_pre + 0.5);
            if(sc[i][j].Y == 255)
              if(x1 >= 0 && x1 < width-1){
                  if(y1 >= 0 && y1 < height-1){
                      sc[i][j].Y = lip(yuv, x1, y1);
                  }
              }
        }
    }

////////////////////////////////////////////////////////////////////////////
/////////////
```

四、实验环境及运行方法

本实验运行在MACOS系统下VM Ware软件下Windows 10虚拟机的Dev C++中。源代码可编译成功。

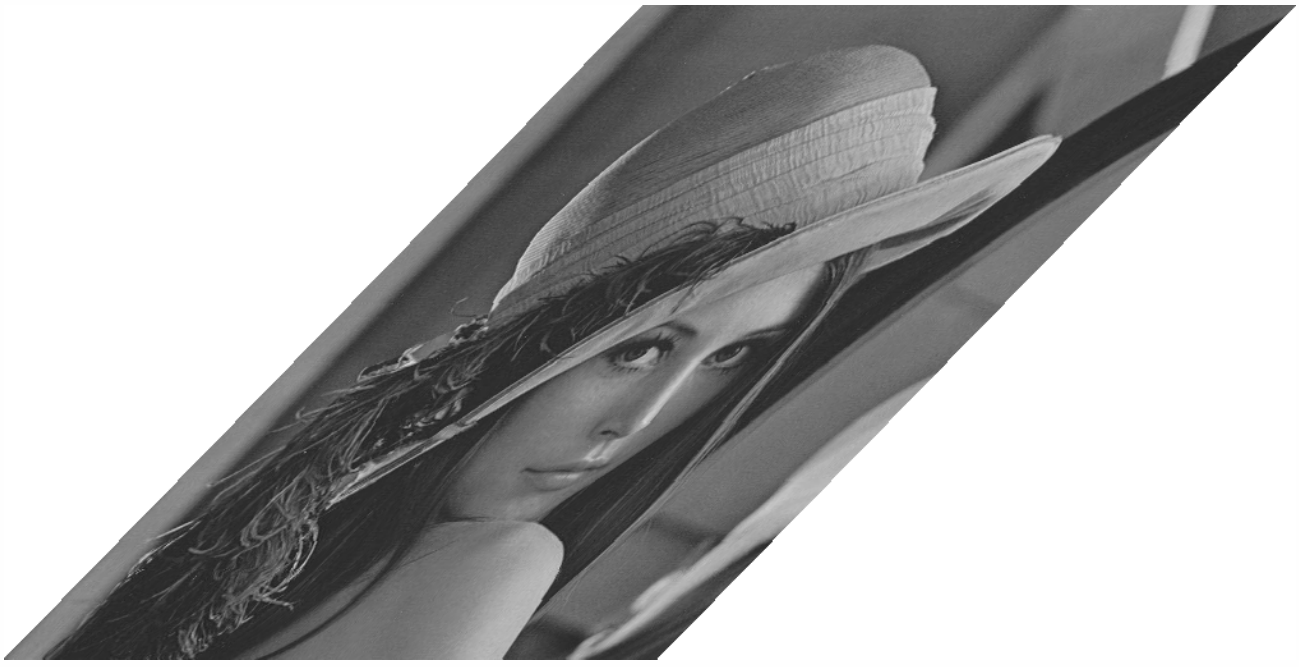
五、实验结果展示

1、原始图片

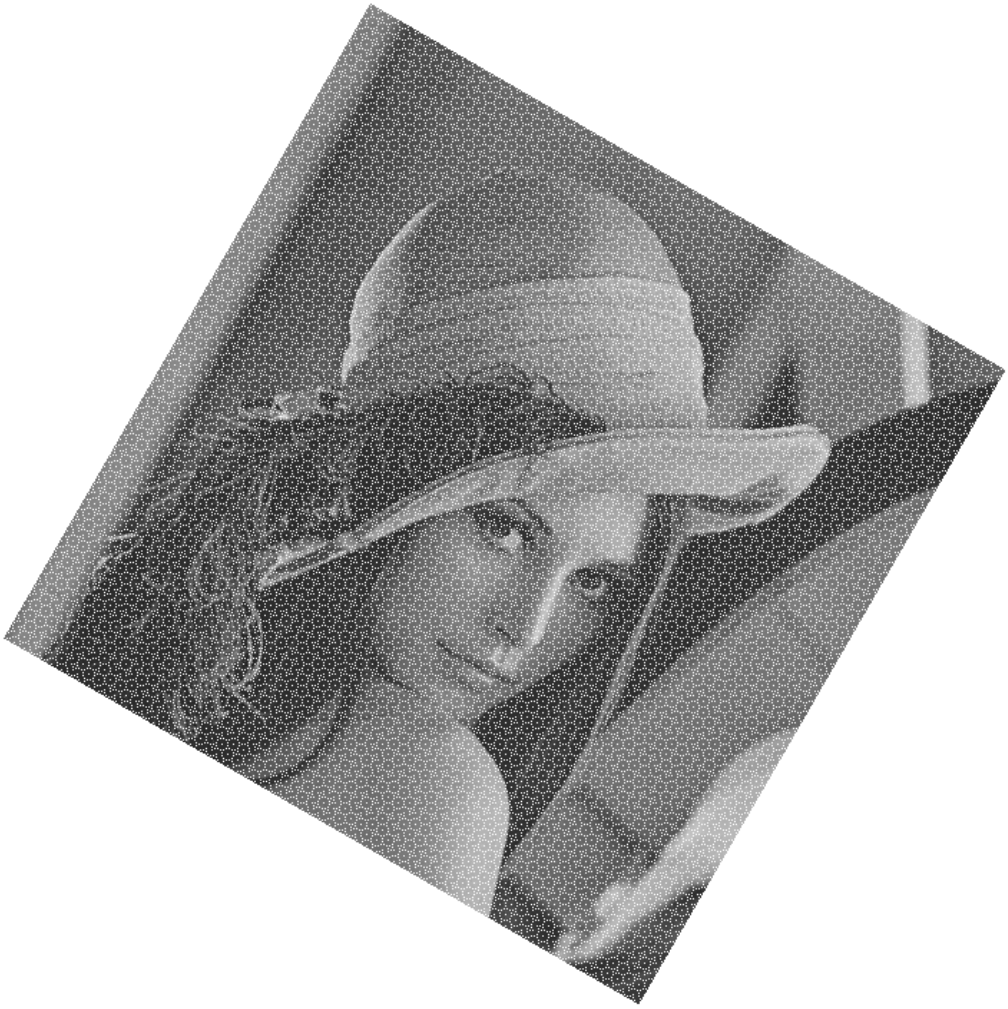
2、Translation

3、Mirror

4、Shear



5、Rotation



- 图像经过旋转变化，但是未插值。

- 图像经过旋转变化，并将空缺的“洞”进行*最近邻插值*插值处理

- 图像经过旋转变化，并将空缺的“洞”进行***线性插值***处理

6、Scale

六、心得体会

1、讨论

（1）插值的影响：对于变换后出现空洞的现象，我们需要对其进行插值，插值的方法在实验中试验了两种。对于最近邻插值来说，其可以满足填补空洞的要求，但是插值的效果并不是很好。会存在变模糊了的缺点。线性插值得到的结果优于最近邻插值，图像相对来说更清晰。但是在scale中，如果变换幅度很大的话，插值的效果还是会下降，会出现竖条纹等。

2、注意点

（1）画布的选择。本次实验在多个部分都有对图像的画布大小进行更改。比如在translation、shear、scale中，可以根据实际得到的图像大小设定尺寸。在旋转操作中，我直接采用了设定最大可能出现的画布大小。在实际操作中，可以根据效果以及需求来更改画布的大小。

（2）旋转操作的位移。图像的旋转操作是图像绕着左上角进行旋转，旋转后的图像必定会超出现有的图像范围，要将其移至画布的中心，还需要对其进行一次位移的操作。位移的常数很好推导，我也将其写在代码中了。

（3）类型转换。实验中涉及到很多次的类型转换，要注意`math.h`库中的`sin`、`cos`等函数都是`double`型的，而数组的索引却要求是整数型的，可以通过强制类型转换等方法，改变其类型。

（4）旋转角度是弧度制的，并且是`double`型的。

（5）在实验中，我还发现一个奇怪的现象：如果画布的长、宽是一个奇数的话，那么该图像便无法正常显示，只要改为偶数，图像便可正常显示。

（6）在调整画布大小之后，还要注意修改输出文件的头文件内的
bfSize、biImageSize、biWidth 、biHeight的值。

3、心得

本次实验我们对图像进行了简单变换，实验还是很有意思的。在旋转的操作中由于画布大小和位移中遇到了一些阻碍，其他的操作还是比较简单的。