

浙江大学实验报告

课程名称: 图象信息处理 指导老师: 宋明黎 成绩: _____
实验名称: Assignment-4 Simple geometric transformation

一、实验目的和要求

学习和掌握对图像的基本几何变换。

- Translation
- Rotation
- Scale
- Shear
- Mirror

专业: 求是科学班(计算机)

姓名: 蒋仕彪

学号: 3170102587

日期: 2018/11/22

二、实验内容和原理

1. Translation (平移变换)

Translation — Equation

$$\begin{cases} x' = x + x_0 \\ y' = y + y_0 \end{cases}$$

OR

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

Each pixel in the original image is translated x_0 and y_0 respectively.

2. Rotation (旋转变换)

Rotation — Equation

$$\begin{cases} x' = x \cos \theta - y \sin \theta \\ y' = x \sin \theta + y \cos \theta \end{cases}$$

OR

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

3. Scale (放大和缩小)

Scale — — Equation

$$\begin{cases} x' = cx \\ y' = dy \end{cases}$$

OR

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

4. Shear (错切变换)

Shear — — Equation

Shear on x axis

$$\begin{cases} a(x, y) = x + d_x y \\ b(x, y) = y \end{cases}$$

Shear on y axis

$$\begin{cases} a(x, y) = x \\ b(x, y) = y + d_y x \end{cases}$$

5. Mirror (镜像变换)

Mirror — — Equation

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

When $S_x = 1$, and $S_y = -1$, flip around the x axis

When $S_x = -1$, and $S_y = 1$, flip around the y axis

6. 插值

6.1 Nearest neighbor based interpolation

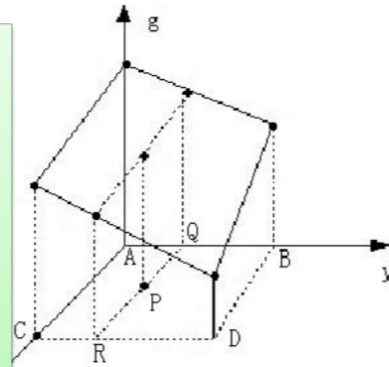
Nearest neighbor based interpolation——Computing process

$$(x', y') \xrightarrow[\text{assign value}]{\text{inverse transformation}} (x, y) \xrightarrow{\text{rounding operation}} (x_{int}, y_{int}) \\ \xrightarrow{\text{assign value}} I_{new}(x', y') = I_{old}(x_{int}, y_{int})$$

6.2 Linear interpolation

Linear interpolation——Equation (2D)

1. Define the bilinear equation $g(x, y) = ax + by + cxy + d$;
2. Substitute the coordinates and grayscale intensity of A, B, C and D into the equation, and an equation system can be obtained.
3. Solve the equation system to obtain a, b, c , and d .
4. Substitute the coordinate of P into the equation and obtain its grayscale intensity.

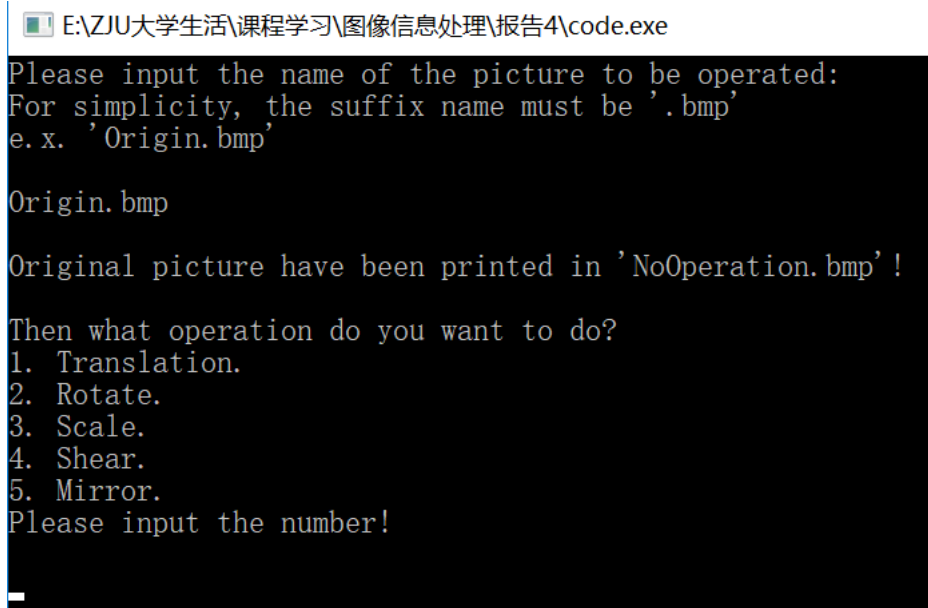


6.3 RBF interpolation

Radial Basis Function (RBF) based interpolation

$$G(x) = \sum_{i=1}^n w_i G(c_i),$$
$$\text{where } w_i = \frac{\varphi(|x - c_i|)}{\sum_{i=1}^n \varphi(|x - c_i|)}$$

三、成果展示



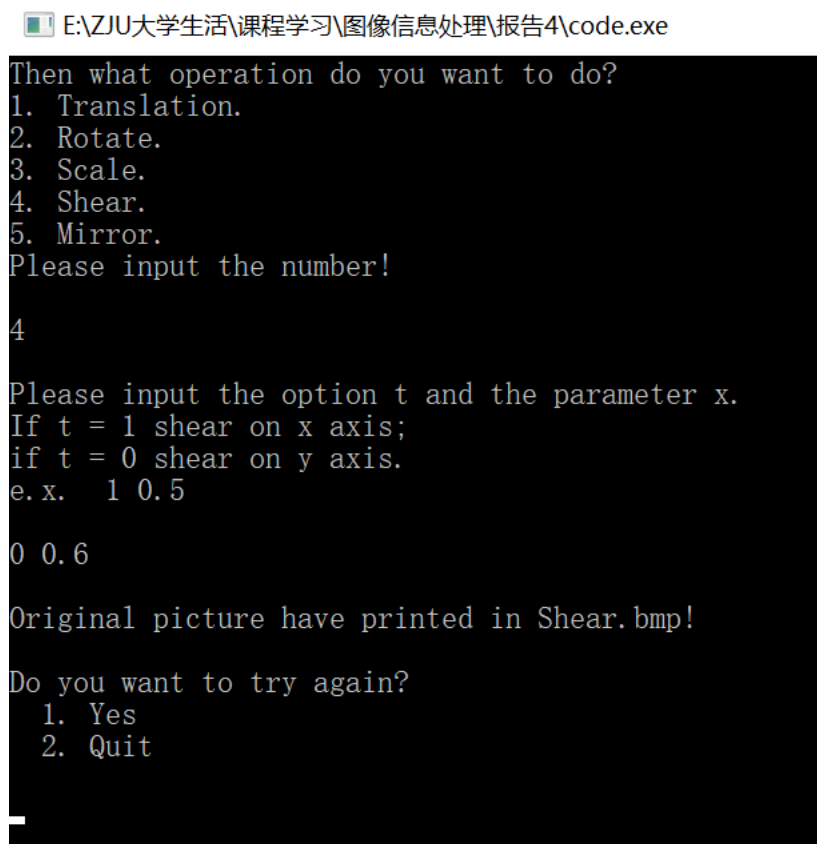
```
E:\ZJU大学生活\课程学习\图像信息处理\报告4\code.exe
Please input the name of the picture to be operated:
For simplicity, the suffix name must be '.bmp'
e. x. 'Origin.bmp'

Origin.bmp

Original picture have been printed in 'NoOperation.bmp'!

Then what operation do you want to do?
1. Translation.
2. Rotate.
3. Scale.
4. Shear.
5. Mirror.
Please input the number!
```

交互界面 1



```
E:\ZJU大学生活\课程学习\图像信息处理\报告4\code.exe
Then what operation do you want to do?
1. Translation.
2. Rotate.
3. Scale.
4. Shear.
5. Mirror.
Please input the number!

4

Please input the option t and the parameter x.
If t = 1 shear on x axis;
if t = 0 shear on y axis.
e. x. 1 0.5

0 0.6

Original picture have printed in Shear.bmp!

Do you want to try again?
1. Yes
2. Quit
```

交互界面 2



原图像



展示在画布上的图像



执行 Translation(100, 300) 后的图像



执行 Translation(100, 600) 后的图像



执行 `Rotate(0.785398)` 后的图像



执行 `Scale(1.8, 1.5)` 后的图像



执行 $\text{Scale}(0.6, 0.4)$ 后的图像



执行 $\text{Shear}(1, 0.5)$ 后的图像



执行 $\text{Shear}(0, 0.5)$ 后的图像



执行 $\text{Mirror}(0)$ 后的图像

四、源代码与分析

由于 Translation 操作和图像的相对位置有关，这涉及到一个“画布”的问题。

于是在读入原图像后，我重新创建了一个图像：大小为原图像长和宽的两倍，且将原图像放在新图像的中间位置。在新图像的两侧画上了边框。这样，新图像其他地方相当于就是画布。

注意到，平移时可能会超出画布。超出画布的部分我不予显示。

以前我都是在同样大小的图片中修改，但在这次 assignment 里，涉及到一个将 bmp 图像放大的问题。经过我的一番探索，**如果我们想扩大/缩小一张 bmp（相当于自定义它的像素矩阵的大小）**，除了修改对应的像素矩阵之外，只需把 biWidth, biHeight 和 biSizeImage（实际字节大小，即像素矩阵实际占用字节数）三个参数相对应地改一改即可。

为了形式化地进行操作，我先写了一个通解变换：给出变换矩阵，求其变换后的图像。默认画布最中间的位置是原点，竖直的是 x 轴，水平的是 y 轴。这里涉及到一个插值的问题，我采用的是 Nearest neighbor based interpolation。所以在传矩阵的时候，本质上只传逆矩阵就行了；然后我根据变换后图像的每个像素点，逆变换回去，找到距离最近的点。

这样，对于剩下五个操作，只要分别计算出逆矩阵，调用以上函数即可。

```
#include <stdio.h>
#include <assert.h>
#include <math.h>
#include <stdlib.h>
#include <algorithm>
#include <cstring>
#include <time.h>

using namespace std;

typedef unsigned char BYTE;
typedef unsigned short WORD;
typedef unsigned int DWORD;
typedef int LONG;

FILE *fin , *fout;

typedef struct tagBITMAPFILEHEADER{
    WORD type;
    DWORD bfSize;
    WORD bfReserved1;
    WORD bfReserved2;
    DWORD bfOffBits;
}head1;
//定义第一个头

typedef struct tagBITMAPINFOHEADER{
    DWORD biSize;
    LONG biWidth;
    LONG biHeight;
    WORD biPlanes;
    WORD biBitCount;
    DWORD biCompression;
    DWORD biSizeImage;
    LONG biXPelsPerMeter;
    LONG biYPelsPerMeter;
    DWORD biClrUsed;
    DWORD biClrImportant;
}head2;
//定义第二个头

typedef struct _RGB{
    BYTE R;
```

```

    BYTE G;
    BYTE B;
}RGB;

typedef struct _YUV{
    short Y;
    short U;
    short V;
}YUV;
//YUV 格式可能会有负数，就直接用 short 存了

typedef struct _HSV{
    short H;
    short S;
    short V;
}HSV;

YUV RGB_To_YUV(RGB cur){
    YUV ret;
    ret.Y = round(0.299 * cur.R + 0.587 * cur.G + 0.114 * cur.B);
    ret.U = round(-0.147 * cur.R - 0.289 * cur.G + 0.435 * cur.B);
    ret.V = round(0.615 * cur.R - 0.515 * cur.G - 0.100 * cur.B);
    return ret;
}

BYTE In(short cur){
    if (cur > 255) cur = 255;
    if (cur < 0) cur = 0;
    return (BYTE)cur;
}
//担心 YUV 转 RGB 时导致 RGB 范围出错，写一个框定范围的函数

RGB YUV_To_RGB(YUV cur){
    RGB ret;
    ret.R = In(round(cur.Y + 1.14 * cur.V));
    ret.G = In(round(cur.Y - 0.395 * cur.U - 0.581 * cur.V));
    ret.B = In(round(cur.Y + 2.033 * cur.U));
    return ret;
}

int line_byte, extra_byte, S, all;
head1 bmfh;
head2 bmih;
//原图

```

```

head2 canvas;
int newS, newall, border = 20;
//新图

void printStream(RGB *cur, BYTE *p, int W, int S, int extra_byte){
    for (int i = 0; i < S; i++){
        *p++ = cur->R;
        *p++ = cur->G;
        *p++ = cur->B;
        if ((i + 1) % W == 0)
            for (int k = 0; k < extra_byte; k++)
                *p++ = 0;
        cur++;
    }
}
//将宽度为 W，总大小为 S 的像素矩阵放入输出流 p 里。

void readStream(RGB *cur, BYTE *p, int W, int S, int extra_byte){
    for (int i = 0; i < S; i++){
        cur->R = *p++;
        cur->G = *p++;
        cur->B = *p++;
        if ((i + 1) % bmih.biWidth == 0)
            p = p + extra_byte;
        cur++;
    }
}
//从读入流里获取宽度为 W，总大小为 S 的像素矩阵

void putIntoMid(RGB *old, int w1, int h1, RGB *now, int w2, int h2){
    for (int i = 0; i < h1; i++){
        for (int j = 0; j < w1; j++){
            int p = i + (h2 - h1) / 2;
            int q = j + (w2 - w1) / 2;
            now[p * w2 + q] = old[i * w1 + j];
        }
        for (int i = 0; i < h2; i++){
            for (int j = 0; j < w2; j++){
                if (i < border || i + border >= h2 || j < border || j + border >= w2)
                    now[i * w2 + j] = (RGB){0, 0, 0};
            }
        }
    }
}
//把画布调大，将原图放置在画布的中央；在画布的边界处加一圈黑色的边框。

```

```

void generalTransform(RGB *old, int w1, int h1, RGB *now, int w2, int h2, double
rev[][3]){
    for (int i = 0; i < h2; i++)
        for (int j = 0; j < w2; j++)
            if (i < border || i + border >= h2 || j < border || j + border >= w2)
                now[i * w2 + j] = (RGB){0, 0, 0};
            else
                now[i * w2 + j] = (RGB){255, 255, 255};

    for (int i = border; i < h2 - border; i++)
        for (int j = border; j < w2 - border; j++){
            int ii = i - h2 / 2;
            int jj = j - w2 / 2;
            int pp = round(rev[0][0] * ii + rev[0][1] * jj + rev[0][2]);
            int qq = round(rev[1][0] * ii + rev[1][1] * jj + rev[1][2]);
            int p = pp + h2 / 2;
            int q = qq + w2 / 2;
            p -= (h2 - h1) / 2;
            q -= (w2 - w1) / 2;
            if (p < 0 || p >= h1 || q < 0 || q >= w1) continue;
            now[i * w2 + j] = old[p * w1 + q];
        }
}

```

//读入操作矩阵的逆矩阵，对图像进行相应地操作，采用 Nearest neighbor 插值。

```

void Translation(RGB *old, int w1, int h1, RGB *now, int w2, int h2, int deltax, int
deltay){
    double matrev[3][3]={
        {1, 0, -deltax},
        {0, 1, -deltay},
        {0, 0, 1}
    };
    generalTransform(old, w1, h1, now, w2, h2, matrev);
}

```

//将图像进行(+deltax, +deltay)的位移

```

void Rotate(RGB *old, int w1, int h1, RGB *now, int w2, int h2, double theta){
    double matrev[3][3]={
        {cos(-theta), -sin(-theta), 0},
        {sin(-theta), cos(-theta), 0},
        {0, 0, 1}
    };
    generalTransform(old, w1, h1, now, w2, h2, matrev);
}

```

```

}
//将图像旋转 cita 角

void Scale( RGB *old, int w1, int h1, RGB *now, int w2, int h2, double cx, double cy){
    double matrev[3][3]={
        {1.0 / cx, 0, 0},
        {0, 1.0 / cy, 0},
        {0, 0, 1}
    };
    generalTransform(old, w1, h1, now, w2, h2, matrev);
}
//将图像进行放缩, (x,y) 倍率分别为 (cx,cy)

void Shear( RGB *old, int w1, int h1, RGB *now, int w2, int h2, int flipx, double d){
    double matrev[3][3]={
        {1, flipx ? -d : 0, 0},
        {!flipx ? -d : 0, 1, 0},
        {0, 0, 1}
    };
    generalTransform(old, w1, h1, now, w2, h2, matrev);
}
//将图像进行错切变换

void Mirror( RGB *old, int w1, int h1, RGB *now, int w2, int h2, int flipx){
    double matrev[3][3]={
        {flipx ? 1 : -1, 0, 0},
        {0, flipx ? -1 : 1, 0},
        {0, 0, 1}
    };
    generalTransform(old, w1, h1, now, w2, h2, matrev);
}
//将图像进行镜像变换

void printPicture( RGB *p, head2 canvas, char *str){
    BYTE *oStream = (BYTE *) malloc(canvas.biSizeImage);
    printStream(p, oStream, canvas.biWidth, newS, extra_byte);
    fout = fopen(str, "wb");
    fwrite(&bmfh, 14, 1, fout);
    fwrite(&canvas, sizeof(head2), 1, fout);
    fwrite(oStream, 1, canvas.biSizeImage, fout);
}
//将像素矩阵 p 里的结果输出至 str 文件

```



```

void Sleep(int x){
    int cur = clock();
    for (int i = 1; ;i++)
        if (!(i & 31))
            if (clock() - cur >= x) return;
}

int main(){
    printf("Please input the name of the picture to be operated:\n");
    printf("For simplicity, the suffix name must be '.bmp'\n");
    printf("e.x. 'Origin.bmp'\n\n");

    char str[50];
    while (true){
        scanf("%s", str);
        fin = fopen(str, "rb");
        fread(&bmfh, 14, 1, fin);
        fread(&bmiH, sizeof(head2), 1, fin);
        if (bmiH.biBitCount != 24)
            printf("\nInput Error!\nPlease try it again!\n\n");
        else break;
    }
    canvas=bmiH;

    line_byte = (bmiH.biWidth * 3 + 3) / 4 * 4; //计算实际存储时每行的字节数
    extra_byte = line_byte - bmiH.biWidth * 3; //计算每行结尾空的字节数
    S = bmiH.biWidth * bmiH.biHeight; //计算像素总个数
    all = line_byte * bmiH.biHeight; //计算像素矩阵总的字节数

    BYTE *iStream = (BYTE *) malloc(all); //将原图读取到 iStream 里
    fread(iStream, 1, all, fin);

    RGB *Origin = (RGB*) malloc(S * sizeof(RGB));
    readStream(Origin, iStream, bmiH.biWidth, S, extra_byte);

    canvas.biWidth += canvas.biWidth / 4 * 4 + border * 2;
    canvas.biHeight += canvas.biHeight / 4 * 4 + border * 2;
    canvas.biSizeImage = canvas.biHeight * ((canvas.biWidth * 3 + 3) / 4 * 4) * 3;
    newS = canvas.biWidth * canvas.biHeight;

    RGB *Start = (RGB*) malloc(newS * sizeof(RGB));
    RGB *Scene = (RGB *) malloc(newS * sizeof(RGB));
    for (int i = 0; i < newS; i++)
        Start[i].R = Start[i].G = Start[i].B = 255;

```

```

int w1 = bmih.biWidth, h1 = bmih.biHeight;
int w2 = canvas.biWidth, h2 = canvas.biHeight;

putIntoMid(Origin, w1, h1, Start, w2, h2);
//调大画布，将原来的图放在画布的中央。
printPicture(Start, canvas, (char *)"NoOperation.bmp");
puts("");
printf("Original picture have been printed in 'NoOperation.bmp'!\n");
puts("");
Sleep(1000);

while (true){
    //system("cls");
    printf("Then what operation do you want to do?\n");
    printf("1. Translation.\n2. Rotate.\n3. Scale.\n4. Shear.\n5. Mirror.\n");
    printf("Please input the number!\n\n");

    int id;
    scanf("%d", &id);puts("");
    switch (id){
        case 1:{
            printf("Please input the delta vector x and y (in pixel).\n");
            printf("e.x. 100 300\n\n");
            int x, y;
            scanf("%d%d", &x, &y);
            Translation(Origin, w1, h1, Scene, w2, h2, x, y);
            printPicture(Scene, canvas, (char *)"Translation.bmp");
            printf("\nOriginal picture have printed in ""Translation.bmp""!\n");
            break;
        }
        case 2:{
            printf("Please input the theta x (In Radian).\n");
            printf("e.x. %.6f\n\n", acos(-1.0) / 4.0);
            double theta;
            scanf("%lf", &theta);
            Rotate(Origin, w1, h1, Scene, w2, h2, theta);
            printPicture(Scene, canvas, (char *)"Rotate.bmp");
            printf("\nOriginal picture have printed in ""Rotate.bmp""!\n");
            break;
        }
        case 3:{
            printf("Please input the Magnification dx and dy.\n");
            printf("If dx = dy = 1, then the picture do not change.\n");

```

```

        printf("e.x.  1.8 1.5\n\n");
        double dx, dy;
        scanf("%lf%lf", &dx, &dy);
        Scale(Origin, w1, h1, Scene, w2, h2, dx, dy);
        printPicture(Scene, canvas, (char *)"Scale.bmp");
        printf("\nOriginal picture have printed in ""Scale.bmp""!\n");
        break;
    }
    case 4:{
        printf("Please input the option t and the parameter x.\n");
        printf("If t = 1 shear on x axis; \nif t = 0 shear on y axis.\n");
        printf("e.x.  1 0.5\n\n");
        int opt; double x;
        scanf("%d%lf", &opt, &x);
        Shear(Origin, w1, h1, Scene, w2, h2, opt, x);
        printPicture(Scene, canvas, (char *)"Shear.bmp");
        printf("\nOriginal picture have printed in ""Shear.bmp""!\n");
        break;
    }
    case 5:{
        printf("Please input the option t.\n");
        printf("If t = 1 mirror on x axis; \nif t = 0 mirror on y axis.\n");
        printf("e.x.  0\n\n");
        int opt;
        scanf("%d", &opt);
        Mirror(Origin, w1, h1, Scene, w2, h2, opt);
        printPicture(Scene, canvas, (char *)"Mirror.bmp");
        printf("\nOriginal picture have printed in ""Mirror.bmp""!\n");
        break;
    }
    default:
        printf("Your input is wrong!\n");
    }
    Sleep(1000);
    printf("\nDo you want to try again?\n 1. Yes\n 2. Quit\n\n");
    int q; scanf("%d", &q);
    if (q == 2) break;
    system("cls");
}

return 0;
}

```