**实验报告**

专业：计算机科学与技术

姓名：颜晗

学号：3200105515

日期：2021/11/14

课程名称： 图像信息处理 指导老师： 宋明黎 成绩：

实验名称： 图像几何操作

1. **实验目的和要求**

了解并掌握对图像进行几何操作的原理和方法，针对放缩、旋转，还需要掌握对图像进行插值操作的方法。

使用C语言实现它们。

**二、实验内容和原理**

实验内容：对某一图像分别进行平移、镜像、放缩、旋转、错切操作，并使用插值使图像操作后依旧可正常查看。

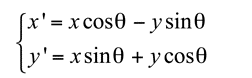
实验原理：对像素在图像中的相对坐标进行操作改变其位置，即可改变图像的观感，即对图像进行几何操作。

**三、实验步骤与分析**

1.平移：根据平移的像素数重新定义新的画布大小，然后将原图像的像素重新映射到新画布中，即可得到平移后的图像，主要代码如下：

1. for(i=0;i<height2;i++){
2. for(j=0;j<width2\_;j++){
3. bmp\_T->imageData[i\*width2\_+j]=0;
4. }
5. }//初始化画布，背景为黑色
6. long Sx;
7. long Sy;
8. if(LR<0){Sx=0;}
9. else  {Sx=LR;}
10. if(UD<0){Sy=0;}
11. else  {Sy=UD;}
12. //分别计算x，y方向上的起始坐标
13. for(i=Sy,m=0;m<height;i++,m++){
14. for(j=Sx,n=0;n<width;n++,j++){
15. bmp\_T->imageData[i\*width2\_+j]=bmpImg->imageData[m\*width\_+n];
16. }//坐标映射
17. }

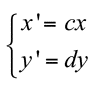
2.旋转：利用下图的公式即可将原坐标转换为新图坐标，注意坐标轴与实际文件中索引位置的转换；同时由于坐标转换时新图中像素点之间会存在背景黑色，无法覆盖，我们需要进行插值，可以观察到黑色背景像素均匀且交错分布，插值思路如下：一个黑色点左右两边都不是黑色，基本可以确认这是背景像素点，选择左侧的像素值为其像素值。



主要代码如下（初始化代码与其他代码因重复而忽略）：

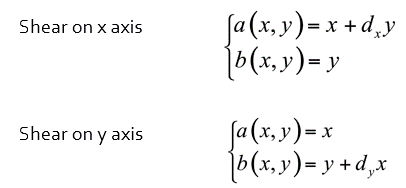
1. for(i=0;i<height;i++){
2. for(j=0;j<width\_;j++){
3. int x1=j-dx1,x2,y1=i-dy1,y2;
4. x2=(int)(x1\*cos(theta))-(int)(y1\*sin(theta))+dx2;
5. y2=(int)(x1\*sin(theta))+(int)(y1\*cos(theta))+dy2;
6. bmp\_R->imageData[y2\*width2\_+x2]=bmpImg->imageData[i\*width\_+j];
7. }*//将原图的像素点映射至新图中。*
8. }
9. for(i=0;i<height2;i++){
10. for(j=1;j<width2\_;j++){
11. if(bmp\_R->imageData[i\*width2\_+j-1]&&bmp\_R->imageData[i\*width2\_+j+1]&&!bmp\_R->imageData[i\*width2\_+j])
12. bmp\_R->imageData[i\*width2\_+j]=bmp\_R->imageData[i\*width2\_+j-1];
13. }
14. } *//在旋转时图像有缺损像素之间无法完全覆盖背景，即需要插值补足全图，此处使用最近邻插值*

3.放缩：对于缩小操作，计算选取原图的像素即可；而对于放大操作，还需要对新图中的新增的位置插值（依旧偷懒使用最近邻插值），可以利用整型结构小数部分自动舍弃的特点，新图中坐标除以放缩比例，自动舍弃后相邻的像素点便有相同原图像素值来源，即完成了插值。



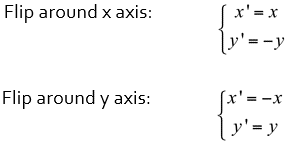
1. for(i=0;i<height2;i++){
2. for(j=0;j<width2\_;j++){
3. bmp\_Sc->imageData[i\*width2\_+j]=bmpImg->imageData[(int)(i/Ytimes)\*width\_+(int)(j/Xtimes)];
4. }*//利用整型自动舍弃小数位同时完成映射和插值操作。*
5. }

4.错切：沿x轴（y轴）错切即从第一行（列）开始，随着行数（列数）增大，行（列）不断向一个方向偏移，最终效果即从一个正方形变成了平行四边形。由于是按错切方向分情况扩大画布的，需要注意坐标转换时的起始坐标。



主要源码如下：

1. if(direction){*//沿x错切*
2. int Sx;*//初始坐标+偏移量即可得到所需坐标*
3. if(d<0) Sx=(-d)\*height2;
4. else Sx=0;
5. for(i=0;i<height;i++){
6. for(j=0;j<width;j++){
7. bmp\_Sh->imageData[i\*width2\_+Sx+j+(int)(d\*i)]=bmpImg->imageData[i\*width\_+j];
8. }
9. }
10. }
11. else {*//沿y错切*
12. int Sy;*//同上*
13. if(d<0) Sy=(-d)\*width2;
14. else Sy=0;
15. for(i=0;i<height;i++){
16. for(j=0;j<width\_;j++){
17. bmp\_Sh->imageData[(Sy+i+(int)(d\*j))\*width2\_+j]=bmpImg->imageData[i\*width\_+j];
18. }
19. }
20. }
21. 镜像：即沿x轴或y轴进行轴对称，将坐标对称的像素值交换即可。主要源码如下：



* 1. if(direction){*//关于x轴对称*
  2. for(i=0;i<height;i++){
  3. for(j=0;j<width\_;j++){
  4. bmp\_M->imageData[i\*width\_+j]=bmpImg->imageData[(height-i-1)\*width\_+j];
  5. }
  6. }
  7. }
  8. else {*//关于y轴对称*
  9. for(i=0;i<height;i++){
  10. for(j=0;j<width\_;j++){
  11. bmp\_M->imageData[i\*width\_+j]=bmpImg->imageData[i\*width\_+width\_-j];
  12. }
  13. }
  14. }

**四、实验环境及运行方法**

本人使用dev-c++ 5.11版C语言工程运行代码，如果电脑上安装有dev-c++,单击文件“bmpwork.dev”即可打开工程文件，点击编译运行的图标或按f11即可。

使用vscode运行的方法：打开程序所处文件夹（包含 bmp.h 、bmp.c以及main.c文件），在终端输入“gcc bmp.c main.c -o main”,敲击enter键后即可编译生成main.exe，再直接在终端输入“.\main”或在文件夹中打开main.exe即可运行程序得到结果。

**注意：**在运行程序前，保证待操作的bmp文件在对应的路径（sample文件夹）中。

**五、实验结果展示**

原图

|  |
| --- |
|  |

平移

|  |  |
| --- | --- |
|  |  |
|  |  |

镜像

|  |  |
| --- | --- |
|  |  |

放缩（展示效果所限，此处仅放置等比例放缩图像）

|  |  |
| --- | --- |
| 放大 | 缩小 |
|  |  |
|  |  |

拉近距离，放大像素，可以看出放大时最近邻插值法的效果并不太好。

|  |  |
| --- | --- |
| 放大 | 原图 |
|  |  |

旋转

|  |  |
| --- | --- |
|  |  |
|  |  |

错切

|  |  |
| --- | --- |
|  |  |
|  |  |

**六、心得体会**

本次实验同样是大量的数学操作，但是更加复杂，主要是关联的变量增加了，虽然重复度很高但是相互之间也有一些细微不同，不仔细的话会直接导致出现几个字母引发的惨案。虽然最后主要代码十分简单，但是写的时候还是花了不少时间，插值也用了最简单的算法。不过之后看到自己代码运行的效果还是很欣慰的，对图像的学习又进了一步。