暨南大学本科实验报告专用纸

课程名称 数字图像处理 成绩评定

实验项目名称 图像的灰度变换 指导教师 刘晓翔

实验项目编号 02 实验项目类型 综合型 实验地点 三楼机房

学生姓名 赵俊文 学号 2022104002

学院 智能科学与工程学院 系 专业 人工智能

实验时间 2024 年 9 月 16 日 上 午~ 9 月 30 日 上 午

（一） 实验目的

①掌握灰度直方图的概念及其计算方法；②掌握线形点运算的概念及其计算方法；③掌握图像均衡化的计算过程。

（二） 实验内容和要求

利用 Visual C++6.0 软件开发工具编写程序，实现 256 灰度图像的直方图显示、线形点运算及图像均衡变换，执行结果应正确.

（三） 主要仪器设备

仪器： 计算机

实验环境： Windows XP + Visual C++6.0

（四） 实验步骤（附代码）与调试

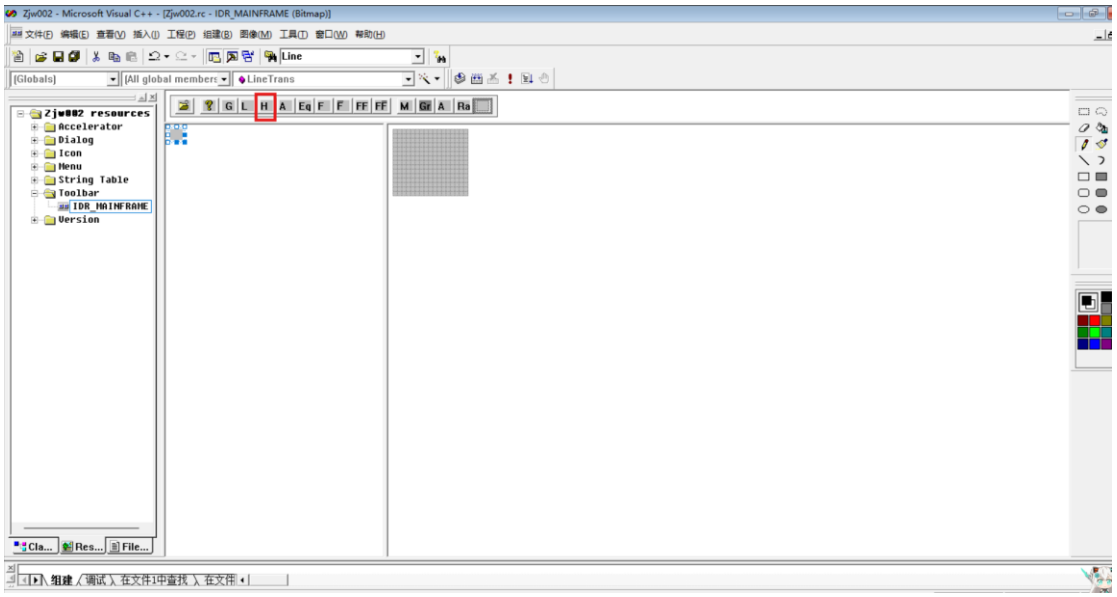
1. 实现直方图显示

a. 在 bmp 中编写判断是否为灰度图像的和计算直方图所用值的函数，在工具栏中添加相应按钮，函数代码如下

```
BOOL IsGray(){
    int r,g,b;
    if(lpBitsInfo->bmiHeader.biBitCount){
        r=lpBitsInfo->bmiColors[150].rgbRed;
        g=lpBitsInfo->bmiColors[150].rgbGreen;
```

```
        b=lpBitsInfo->bmiColors[150].rgbBlue;
        if(r==b&&r==g)
            return TRUE;

    }
    return FALSE;


}



DWORD H[256];
void HGray()
{
    int w = lpBitsInfo->bmiHeader.biWidth;
    int h = lpBitsInfo->bmiHeader.biHeight;
    int LineBytes = (w * lpBitsInfo->bmiHeader.biBitCount + 31)/32 * 4;
    BYTE* lpBits =
(BYTE*)&lpBitsInfo->bmiColors[lpBitsInfo->bmiHeader.biClrUsed];

    int i,j;
    BYTE* pixel;
    for(i = 0; i < 256; i++)
        H[i] = 0;

    for(i = 0;i < h; i ++){
        for(j = 0;j < w;j ++){
            pixel = lpBits + LineBytes * (h - 1 - i) + j;
            H[*pixel] ++;
        }
```

```
}
}
```



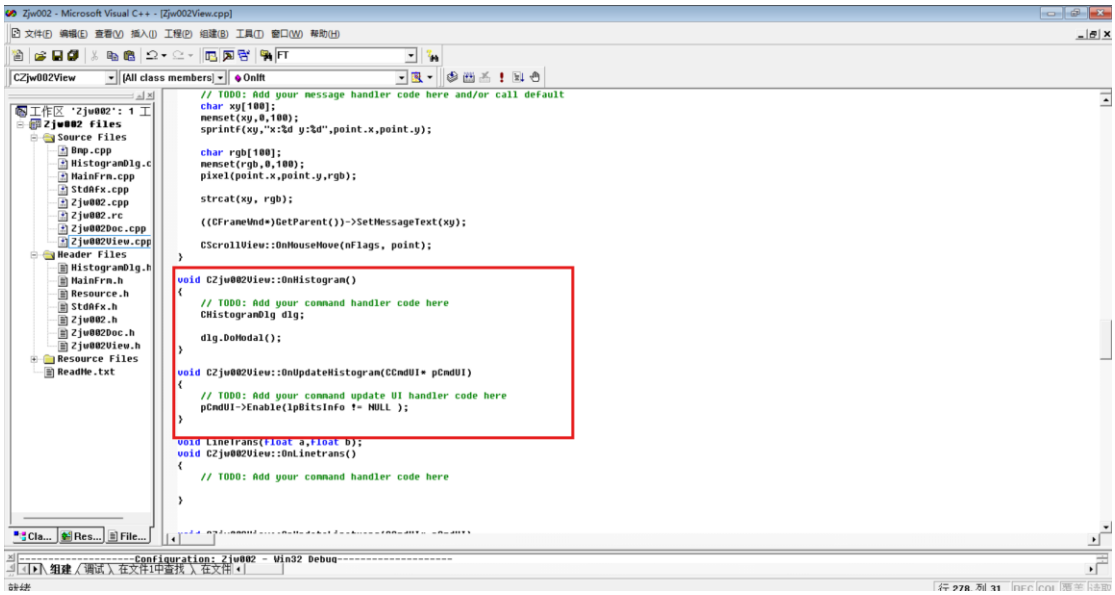
b.创建一个 HistogramDlg.cpp 文件,在其的 OnPaint 函数中编写代码用于绘制直方图,代码如下

```
void CHistogramDlg::OnPaint()
{
    CPaintDC dc(this); // device context for painting
    if(IsGray()){
        // TODO: Add your message handler code here
        dc.Rectangle(20,20,287,221);
        int i;
        DWORD graymax;
        graymax = 0;

        for (i = 0; i < 256; i ++)
        {
            if (H[i] > graymax)
                graymax = H[i];
        }
```

```
    for (i = 0; i < 256; i++)
    {
        dc.MoveTo(i + 20, 220);
        dc.LineTo(i + 20, 220 - (int)H[i] * 200 / graymax);
    }
}
else{
    int i;
    DWORD maxRed = 0, maxGreen = 0, maxBlue = 0;
    int offsetX = 0; // 每个直方图的横向偏移量
    dc.Rectangle(offsetX + 20, 20, offsetX + 277, 221);
    for (i = 0; i < 256; i++) {
        if (HR[i] > maxRed) maxRed = HR[i];
    }

    for (i = 0; i < 256; i++) {
        dc.MoveTo(i + offsetX + 20, 220);
        dc.LineTo(i + offsetX + 20, 220 - (int)(HR[i] * 200 / maxRed));
    }

    // 绘制绿色通道直方图
    offsetX += 300; // 更新偏移量
    dc.Rectangle(offsetX + 20, 20, offsetX + 277, 221);
    for (i = 0; i < 256; i++) {
        if (HG[i] > maxGreen) maxGreen = HG[i];
    }

    for (i = 0; i < 256; i++) {
        dc.MoveTo(i + offsetX + 20, 220);
        dc.LineTo(i + offsetX + 20, 220 - (int)(HG[i] * 200 / maxGreen));
```

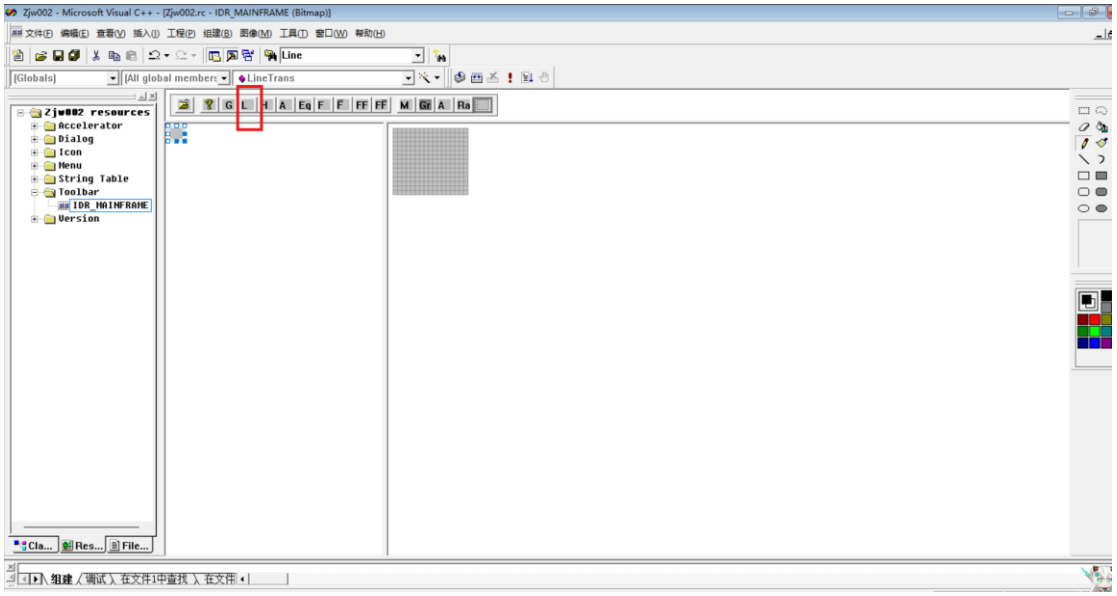
```
    }

    // 绘制蓝色通道直方图
    offsetX += 300; // 更新偏移量
    dc.Rectangle(offsetX + 20, 20, offsetX + 277, 221);
    for (i = 0; i < 256; i++) {
        if (HB[i] > maxBlue) maxBlue = HB[i];
    }
    for (i = 0; i < 256; i++) {
        dc.MoveTo(i + offsetX + 20, 220);
        dc.LineTo(i + offsetX + 20, 220 - (int)(HB[i] * 200 / maxBlue));
    }
}
// Do not call CDialog::OnPaint() for painting messages

}
```

c.在视图类中调用之.



2.线性点运算

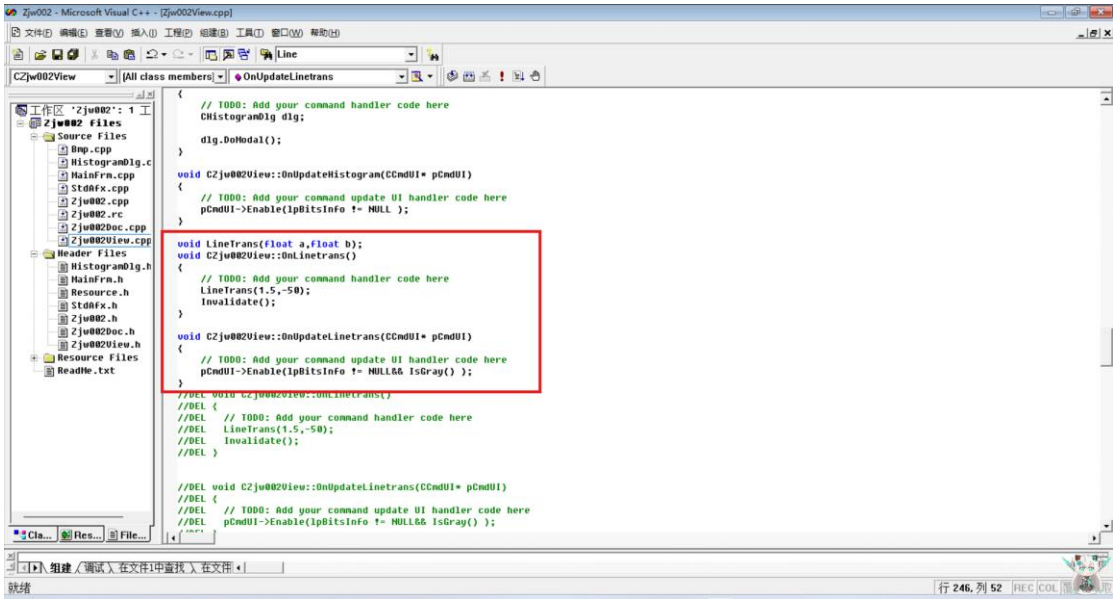
a.在工具栏中添加对应按钮并在类向导中给视图类添加响应函数



b.在 bmp 文件中编写 Linetrans 函数，并在视图类的相应函数中调用，函数代码如下

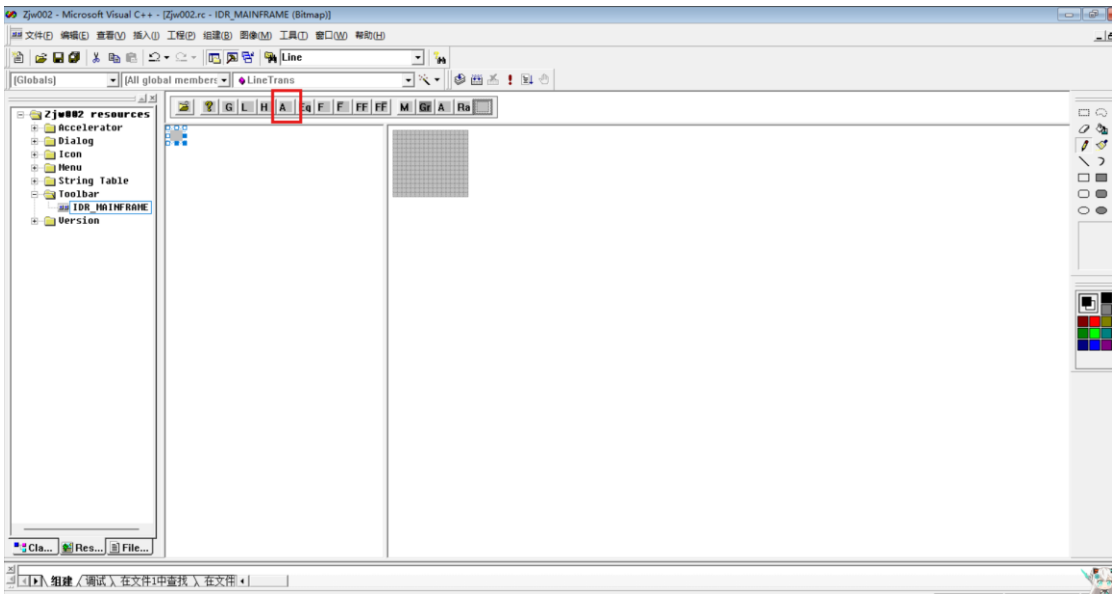
```
void LineTrans(float a,float b){
    int w = lpBitsInfo->bmiHeader.biWidth;
    int h = lpBitsInfo->bmiHeader.biHeight;
    int LineBytes = (w * lpBitsInfo->bmiHeader.biBitCount + 31)/32 * 4;
    BYTE* lpBits =
(BYTE*)&lpBitsInfo->bmiColors[lpBitsInfo->bmiHeader.biClrUsed];

    int i,j;
    BYTE* pixel;
    float temp;
    for(i = 0;i < h; i ++){
        for(j = 0;j < w;j ++){
            pixel = lpBits + LineBytes * (h - 1 - i) + j;
            temp = a * (*pixel) + b;
            if(temp > 255)
                *pixel = 255;
            else if(temp < 0)
```

```
            *pixel = 0;
        else
            *pixel = (BYTE)(temp + 0.5);
    }
}
}
```



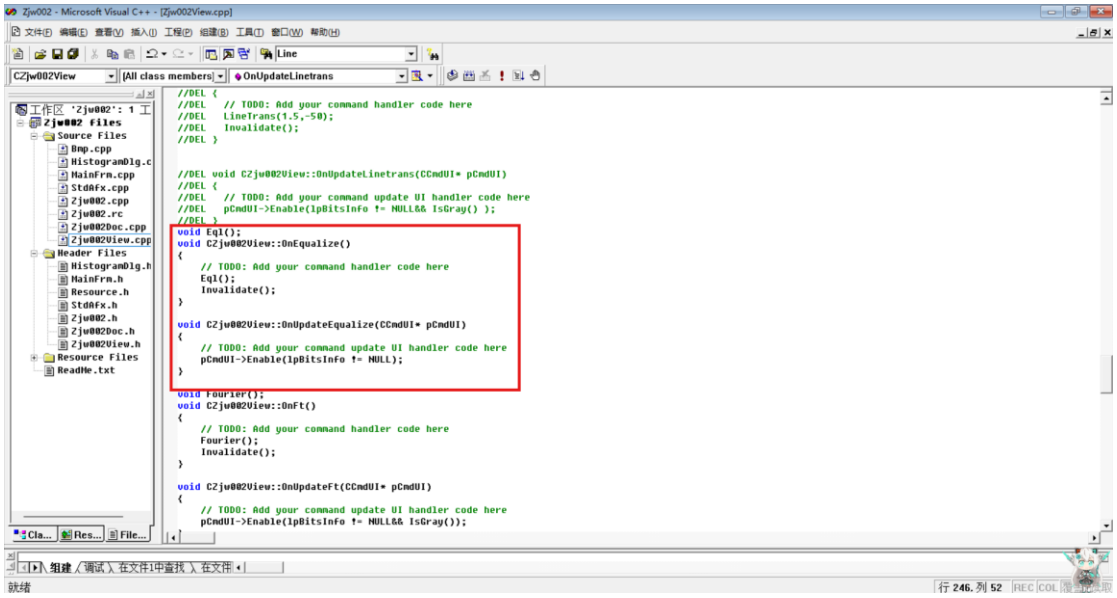
3. 图像均衡变换

a.在工具栏中添加对应按钮并在类向导中给视图类添加响应函数



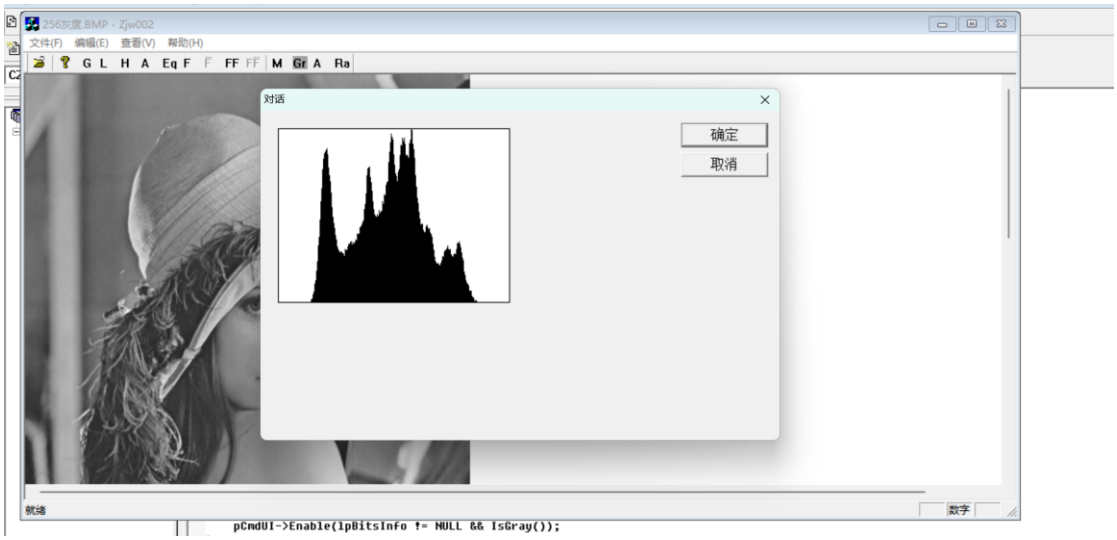
b.在 bmp 文件中编写函数，并在视图类的相应函数中调用，函数代码如下

```
void Equalize(){
    int w = lpBitsInfo->bmiHeader.biWidth;
    int h = lpBitsInfo->bmiHeader.biHeight;
    int LineBytes = (w * lpBitsInfo->bmiHeader.biBitCount + 31)/32 * 4;
    BYTE* lpBits =
(BYTE*)&lpBitsInfo->bmiColors[lpBitsInfo->bmiHeader.biClrUsed];

    int i,j;
    BYTE* pixel;
    DWORD temp;

    BYTE Map[256];
    Histogram();

    for(i = 0;i < 256; i++){
        temp = 0;
        for(j = 0;j <= i; j++){
            temp += H[j];
        }
        Map[i] = 255 * temp / (h * w);
    }

    for(i = 0;i < h; i ++){
        for(j = 0;j < w;j ++){
            pixel = lpBits + LineBytes * (h - 1 - i) + j;
            *pixel = Map[*pixel];
        }
    }
    }
```

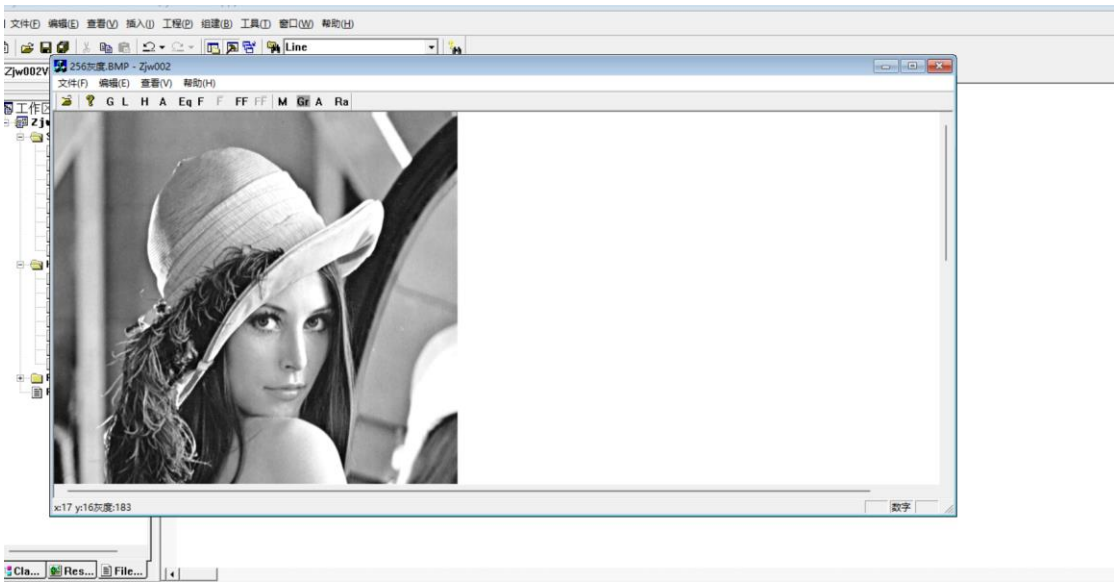


（五） 实验结果与分析

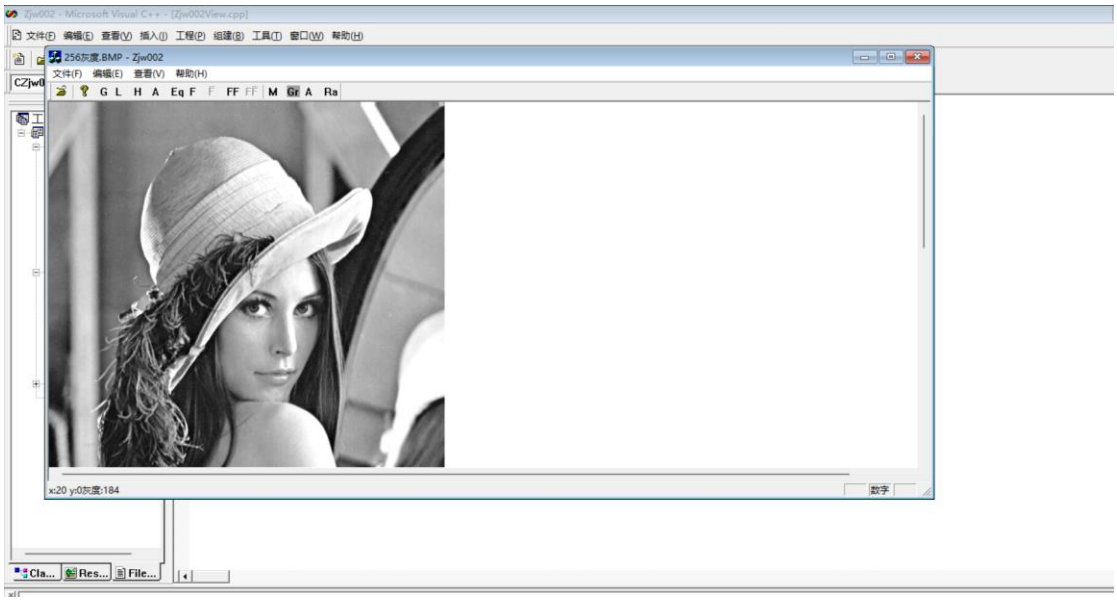
1.直方图绘制



2.线性点运算



3.图像均衡化变换



4.实验分析

通过本次实验，我掌握了图像的均衡化和线性点运算的计算方法，同时利用代码实现绘制图像的灰度直方图，以及对图像进行线性点运算处理和均衡化变换。