深 圳 大 学 实 验 报 告

课程名称：数字图像处理

实验项目名称：图像的(空间增强)平滑处理实验(拓展)

学院：计算机与软件学院

专业：计算机科学与技术

指导教师：吴惠思 教授

报告人：林浩晟 学号：2022280310

实验时间：2025 年 4 月 21 日

实验报告提交时间：2025 年 4 月 22 日

教务部制

一、实验目的：

1. 掌握图像空间增强原理
2. 掌握图像平滑处理原理及实现方法
3. 掌握图像统计滤波原理方法
4. 掌握图像最大值(统计)滤波实现方法

二、实验原理：

实验要求：

1. 熟悉 C++语言编程
2. 熟练使用 C++语言实现图像文件的读取操作
3. 熟练使用 C++语言实现图像显示方法

实验内容：

1、 图像(3x3)模板处理函数
2、 实现图像均值滤波函数
3、 实现图像高斯(平滑)滤波函数
4、 实现图像最大值(统计)滤波函数
5、 实现图像随机生成噪声函数

1、图像(3x3 模板处理)函数

```
void ImageMaskProcessing(char *oImage, char *nImage, int wImage, int hImage,
                         int *Mask, int MaskWH, int MaskCoff)
{    int Coff;  int i, j, m, n;    int k;
     k = (MaskWH-1)/2;
     for (i=k; i<hImage-k; i++) {
          for (j=k; j<wImage-k; j++) {
               Coff = 0;
               for (m=-k; m<=k; m++) {
               for (n=-k; n<=k; n++) {
                    Coff += (BYTE) oImage[(i+m)*wImage+(j+n)] *
                            Mask[(m+k)*MaskWH+(n+k)];}}
               nImage[i*wImage+j] = (unsigned char) (Coff / MaskCoff);
               if (Coff < 0) nImage[i*wImage+j] = 0;
     }}}
```

2、图像均值滤波函数

```
void AverageMaskProcessing(char *oImage, char *nImage,
                           int wImage, int hImage)
{
     int Mask[9] = {1, 1, 1,
```

```
                    1, 1, 1,
                    1, 1, 1};
    ImageMaskProcessing(oImage, nImage, wImage, hImage, Mask, 3, 9);
}
```

3、图像高斯(平滑)滤波函数

```
void GuassAverageMaskProcessing(char *oImage, char *nImage,
                                int wImage, int hImage)
{
    int Mask[9] = {1, 2, 1,
                   2, 4, 2,
                   1, 2, 1};
    ImageMaskProcessing(oImage, nImage, wImage, hImage, Mask,3,16);
}
```

4、主函数中，变量说明及函数调用

```
char  NewImage[1024*1024];   //新的图像缓存
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{   :
    switch (message) {
        case WM_CREATE:
            hWind = hWnd;
            hWinDC = GetWindowDC(hWnd);
            getcwd(ImgDlgFileDir, MAX_PATH);
            break;
        case WM_COMMAND:
            wmId    = LOWORD(wParam);
            wmEvent = HIWORD(wParam);
            switch (wmId) {
        case IDM_AVERAGEFILTER:                 //平均滤波器
            AverageMaskProcessing(OrgImage, NewImage, IMAGEWIDTH, IMAGEHEIGHT);
            ShowImage(NewImage, IMAGEWIDTH, IMAGEHEIGHT, XPOS, YPOS+300); break;
        case IDM_AVERAGEGAUSS:                  //高斯滤波器
            GuassAverageMaskProcessing(OrgImage, NewImage, IMAGEWIDTH, IMAGEHEIGHT);
            ShowImage(NewImage, IMAGEWIDTH, IMAGEHEIGHT, XPOS, YPOS+300); break;
        case IDM_MAXFILTER:                     //最大值滤波
```

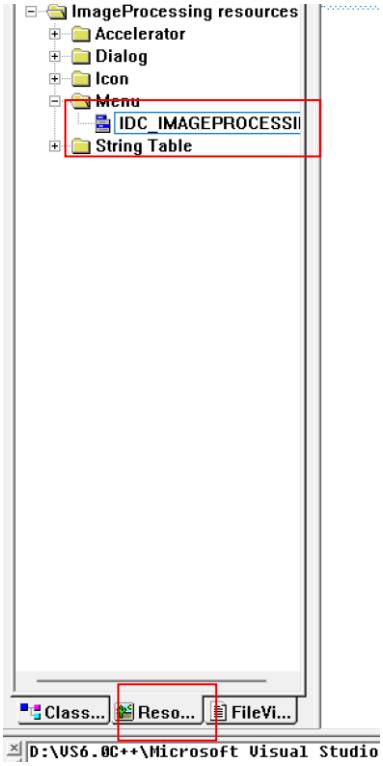
```
                MaxFilterProcessing(OrgImage, NewImage, IMAGEWIDTH, IMAGEHEIGHT);
                ShowImage(NewImage, IMAGEWIDTH, IMAGEHEIGHT, XPOS, YPOS+300); break;
            case IDM_ABOUT:
                :
```

三、实验用品：

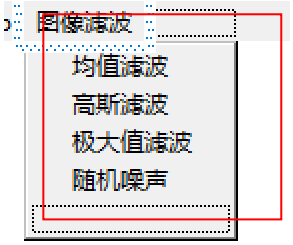
计算机、VS6.0C++

四、实验过程及内容：

1. 首先添加菜单项；



2. 随后按照教程加入选项，如下图所示；



3. 在 switch (wmId)语句中加入 case，其中 IDM_AVERAGEFILTER 为均值滤波，IDM_AVERAGEGAUSS 为高斯滤波，IDM_MAXFILTER 为极大值滤波，IDM_NOISE 为随机噪声；点击按钮即可进行对应的处理并且显示处理后的图像；定义与 OrgImage 同大小的 NewImage 参数和 LasImage，用于保存处理后的图像；

```
        ReadImage(ImgDlgFileName, OrgImage, IMAGEWIDTH, IMAGEHEIGHT);
        ShowImage(OrgImage, IMAGEWIDTH, IMAGEHEIGHT, XPOS, YPOS);
        break;
    case IDM_SHOWBMPIMAGE:
        OpenImageFileDlg("打开图像文件");
        ReadBmpImage(ImgDlgFileName, OrgImage);
        ShowBmpImage(OrgImage, ImageWidth, ImageHeight, XPOS, YPOS);
        break;
    case IDM_AVERAGEFILTER:            //平均滤波器
        AverageMaskProcessing(OrgImage, NewImage, IMAGEWIDTH, IMAGEHEIGHT);
        ShowImage(NewImage, IMAGEWIDTH, IMAGEHEIGHT, XPOS, YPOS+300);
        break;
    case IDM_AVERAGEGAUSS:             //高斯滤波器
        GuassAverageMaskProcessing(OrgImage, NewImage, IMAGEWIDTH, IMAGEHEIGHT);
        ShowImage(NewImage, IMAGEWIDTH, IMAGEHEIGHT, XPOS, YPOS+300);
        break;
    case IDM_MAXFILTER:            //最大值滤波
        MaxFilterProcessing(OrgImage, NewImage, IMAGEWIDTH, IMAGEHEIGHT);
        ShowImage(NewImage, IMAGEWIDTH, IMAGEHEIGHT, XPOS, YPOS+300);
        break;
    case IDM_NOISE:            //随机噪声
        AddWhiteNoise(OrgImage, LasImage, IMAGEWIDTH, IMAGEHEIGHT, 200);
        ShowImage(LasImage, IMAGEWIDTH, IMAGEHEIGHT, XPOS, YPOS+300);
        SaveRawImage("noise.raw",LasImage,IMAGEWIDTH, IMAGEHEIGHT);
        break;
    case IDM_ABOUT:
        DialogBox(hInst, (LPCTSTR)IDD_ABOUTBOX, hWnd, (DLGPROC)About);
        break;
    case IDM_EXIT:
```

```
char ImgDlgFileDir[MAX_PATH];
char OrgImage[1024*1024];
char NewImage[1024*1024];
char LasImage[1024*1024];
#define IMAGEWIDTH  256
#define IMAGEHEIGHT 256
```

4. 在函数开头定义各个函数，防止报错；

```
void AverageMaskProcessing(char* oImage, char* nImage,//均值滤波
    int wImage, int hImage);
void GuassAverageMaskProcessing(char* oImage, char* nImage,//高斯滤波
    int wImage, int hImage);
void MaxFilterProcessing(char* oImage, char* nImage,//极大值滤波
    int wImage, int hImage);
void AddWhiteNoise(char* oImage, char* nImage,  //随机噪声
                   int wImage, int hImage, int numPoints);
void SaveRawImage(char* filename,char* buffer, int width, int height);
HDC  hWinDC;
```

5. ImageMaskProcessing 函数对输入图像 oImage 进行卷积运算，并将结果存入 nImage，核心思想是用一个 Mask 矩阵对图像像素进行加权求和，然后归一化，为后续的均值滤波与高斯滤波做准备；详细步骤：首先计算掩模半径 k = (MaskWH - 1) / 2，确保卷积操作在图像中心有效执行，避免越界；再遍历图像，计算 i, j 位置的像素值加权和 Coff，通过 Mask 进行卷积计算，即像素值乘以对应权重后求和，归一化处理 Coff / MaskCoff，确保输出值在合理范围；同时确保只对 hImage - k 和 wImage - k 范围内的像素操作，避免越界访问；如果计算出的 Coff 小于 0，则设为 0。

```
void ImageMaskProcessing(char* oImage, char* nImage, int wImage, int hImage,
    int* Mask, int MaskWH, int MaskCoff)
{
    int Coff;   // 计算当前像素的卷积和
    int i, j, m, n; // i, j 为图像像素索引, m, n 用于遍历 Mask
    int k = (MaskWH - 1) / 2;  // 计算 Mask 的半径（用于卷积计算）
    // 遍历图像（跳过边缘，防止越界）
    for (i = k; i < hImage - k; i++) {
        for (j = k; j < wImage - k; j++) {
            Coff = 0;  // 计算当前像素的卷积累加值
            // 计算 Mask 卷积区域
            for (m = -k; m <= k; m++) {
                for (n = -k; n <= k; n++) {
                    // 对应像素点乘以 Mask 权重
                    Coff += (unsigned char)oImage[(i + m) * wImage + (j + n)] *
                            Mask[(m + k) * MaskWH + (n + k)];
                }
            }
            Coff /= MaskCoff;// 归一化处理

            if (Coff < 0) // 限制像素值在 0~255 之间
                nImage[i * wImage + j] = 0;
            else if (Coff > 255)
                nImage[i * wImage + j] = 255;
            else
                nImage[i * wImage + j] = (unsigned char)Coff;
        }
    }
}
```

6. AverageMaskProcessing 函数用于对图像进行 3*3 均值滤波；Mask 为 3×3 的均值滤波器，每个权重都为 1，归一化系数为 9；由于该函数计算的是邻域像素的均值，因此该处理会模糊图像，降低噪声，但也会导致细节损失；并且 ImageMaskProcessing 仅对 k ~ hImage-k 和 k ~ wImage-k 范围进行计算，因此图像四周的像素未被处理，会出现黑边。

```
void AverageMaskProcessing(char* oImage, char* nImage,
    int wImage, int hImage)
{
    int Mask[9] = { 1, 1, 1,
                    1, 1, 1,
                    1, 1, 1 };

    ImageMaskProcessing(oImage, nImage, wImage, hImage, Mask, 3, 9);
}
```

7. GuassAverageMaskProcessing 函数用于对图像实现 3*3 高斯滤波；Mask 采用 3×3 的高斯核，归一化系数为 16；相比普通均值滤波（所有权重相等），高斯滤波对中心像素赋予更高权重，使得平滑效果更自然，并且可以有效去除高频噪声，但仍能保留一定的边缘信息；

```
void GuassAverageMaskProcessing(char* oImage, char* nImage,
    int wImage, int hImage)
{
    int Mask[9] = { 1, 2, 1,
                    2, 4, 2,
                    1, 2, 1 };

    ImageMaskProcessing(oImage, nImage, wImage, hImage, Mask, 3, 16);
}
```

8. MaxFilterProcessing 函数实现了最大值滤波，用于去除噪声和增强图像亮度区域；最大值滤波在 3×3 的邻域窗口内，选取最大像素值作为当前像素值。详细步骤如

下：首先定义 3×3 过滤窗口，计算边界填充大小；随后遍历图像的每个像素，初始化 maxVal 用于储存最大值，遍历 3×3 邻域，获取所有邻域像素的最大值，使用边界镜像处理:①若 px 超出左边界 0,则 px = -px;②若 px 超出右边界 wImage-1，则 px = 2*wImage - px - 1；③py 方向同理；最后 maxVal 存储邻域最大值，并将其赋给 nImage[y * wImage + x]。

```
void MaxFilterProcessing(char* oImage, char* nImage,
    int wImage, int hImage)
{
    int kernelSize = 3;      // 过滤器大小 (3x3)
    int pad = kernelSize / 2; // 计算边界填充大小
    for (int y = 0; y < hImage; y++) { // 遍历图像每个像素
        for (int x = 0; x < wImage; x++) {
            char maxVal = 0; // 存储3×3邻域的最大值
            // 遍历3×3邻域
            for (int ky = -pad; ky <= pad; ky++) {
                for (int kx = -pad; kx <= pad; kx++) {
                    int px = x + kx;
                    int py = y + ky;
                    // 镜像边界处理（反射填充）
                    if (px < 0) px = -px;
                    if (py < 0) py = -py;
                    if (px >= wImage)  px = 2 * wImage - px - 1;
                    if (py >= hImage)  py = 2 * hImage - py - 1;

                    char pixel = oImage[py * wImage + px];// 获取邻域像素
                    maxVal = (pixel > maxVal) ? pixel : maxVal; // 更新最大值
                }
            }
            // 赋值最大值给图像
            nImage[y * wImage + x] = maxVal;
        }
    }
}
```

9. AddWhiteNoise 函数用于给图像添加指定数量的白色噪声点，从而随机生成噪声；首先将原图 oImage 的每一个像素值复制到输出图像 nImage 中；再用当前时间初始化随机数种子，以保证每次运行产生的随机结果都不同；最后在图像的随机位置设置像素值为 255（纯白色），模拟白噪声；

```
void AddWhiteNoise(char* oImage, char* nImage, int wImage, int hImage, int numPoints) {
    // 复制原图数据到输出图像
    int i;
    for (i = 0; i < wImage * hImage; i++) {
        nImage[i] = oImage[i];
    }

    // 初始化随机数种子
    srand((unsigned)time(NULL));

    // 添加随机白点
    for (i = 0; i < numPoints; i++) {
        int x = rand() % wImage;
        int y = rand() % hImage;
        nImage[y * wImage + x] = (char)255;
    }
}
```

10. SaveRawImage 函数用于将图像数据以 RAW 格式保存到文件中，以此保存随机生成噪声后的图片；首先以二进制写模式打开文件，再将整个图像缓冲区写入文件，每个像素一个字节（灰度图），最后关闭文件；

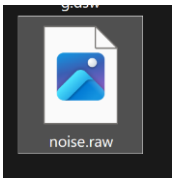
```
void SaveRawImage(char* filename,char* buffer, int width, int height)
{
    FILE* fp = fopen(filename, "wb"); // 以二进制写入方式打开

    size_t dataSize = width * height;
    size_t written = fwrite(buffer, sizeof(unsigned char), dataSize, fp);
    fclose(fp);
}
```

11. 实验结果

随机生成噪声

同时保存的图像也生成了：



均值滤波

高斯滤波


均值滤波
高斯滤波
极大值滤波
随机噪声
Hello World!

极大值滤波


均值滤波
高斯滤波
极大值滤波
随机噪声
Hello World!

五、实验现象及数据处理：

实验结果如下图：

随机生成噪声

均值滤波

高斯滤波


均值滤波
高斯滤波
极大值滤波
随机噪声
Hello World!

极大值滤波


均值滤波
高斯滤波
极大值滤波
随机噪声
Hello World!

六、实验结论：

实验成功完成。

实验体会：

通过本次图像处理实验，我深入理解了多种滤波技术的原理与应用。在实现图像均值滤波时，发现它能有效去除图像噪声，但会使图像变得模糊。高斯滤波则在平滑图像的同时，更好地保留了图像细节，体现了其在图像处理中的优越性。最大值滤波可以突出图像中的亮部区域，对于某些特定场景非常有用。而图像随机噪声生成函数的实现，让我更加直观地感受到噪声对图像质量的影响。整个实验过程中，我不仅掌握了不同滤波函数的编写技巧，还体会到了理论与实践相结合的重要性，为后续深入学习图像处理奠定了坚实基础。

思考题：
指导教师批阅意见： **成绩评定：** **指导教师签字：** **年　　月　　日**
备注：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。