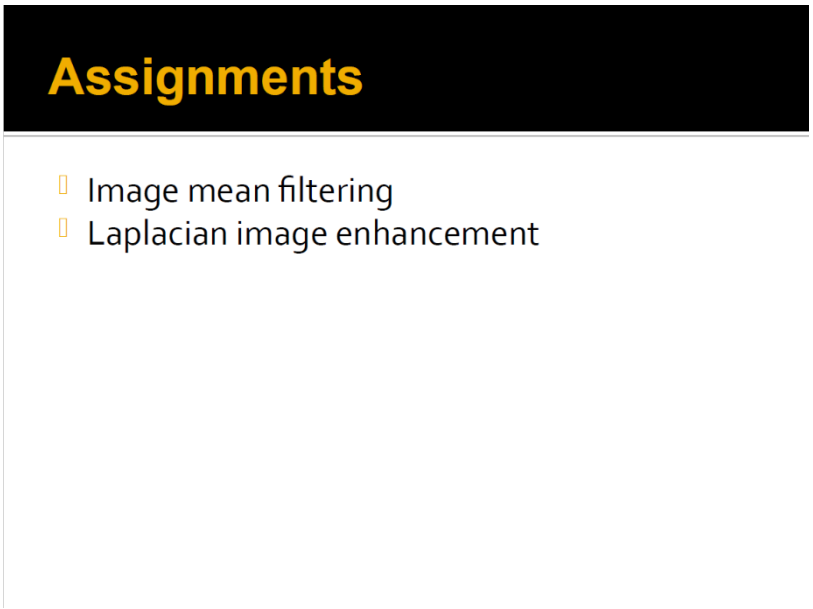
浙江大学实验报告

课程名称：图像信息处理 指导老师：宋明黎 成绩：

实验名称：图像均值滤波与拉普拉斯图像增强

一、实验目的和要求



二、实验内容和原理

1、图像均值滤波

Spatial filtering

Principle of filtering——Concept

Concept: Filter is a window sized by M×N, wherein the elements in the window take operation on the corresponding pixels from the original image in the window. And the results are saved as the pixels in a new image.

Alias: Filter, mask, kernel, template, window

The elements in the filter are coefficients instead of pixel values, which represent the weight applied on the pixels in the original image.

Spatial filtering

Linear smoothing filter——Concept

The output of the linear smoothing filter is the mean value of the pixels in the mask. It's also called mean filter.

$$\frac{1}{9}\times\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

2、拉普拉斯图像增强

Spatial filtering

Image enhancement based on second order differential——Laplacian operator

For a function $f(x,y)$, Laplacian operator is defined as:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Spatial filtering

The discrete Laplacian operator:

Along x axis:

$$\frac{\partial^2 f}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y)$$

Along y axis:

$$\frac{\partial^2 f}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y)$$

Hence, the discrete Laplacian operator is:

$$\nabla^2 f = \left[f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) \right] - 4f(x,y)$$

Spatial operator

Mask of Laplacian operator

$$\nabla^2 f = \left[f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) \right] - 4f(x,y)$$

0	1	0
1	-4	1
0	1	0

It is rotation invariant.

Spatial filtering

Extending the mask

The elements in the diagonal direction can also be taken into account：

$$\begin{aligned}\nabla^2 f = [&f(x-1,y-1)+f(x,y-1)+f(x+1,y-1)\\&+f(x-1,y)+f(x+1,y)\\&+f(x-1,y+1)+f(x,y+1)+f(x+1,y+1)]\\&-8f(x,y)\end{aligned}$$

Or

$$\nabla^2 f = \sum_{i=-1}^{1}\sum_{j=-1}^{1} f(x+i,y+j) - 9f(x,y)$$

Spatial filtering

Application of Laplacian operator

Image enhancement by Laplaician:

$$g(x,y) = \begin{cases} f(x,y) - \nabla^2 f(x,y) & \text{If the center element of the mask is negative} \\ f(x,y) + \nabla^2 f(x,y) & \text{If the center element of the mask is positive} \end{cases}$$

如果拉普拉斯掩模中心系数为负

如果拉普拉斯掩模中心系数为正

Fuse the original image and the Laplacian result can preserve the sharpening effect and restore the original visual information.

三、实验步骤与分析

1、读取 BMP 文件的基本信息

沿用上次作业的 ReadBmp 函数

2、图像均值滤波

先转灰度图，然后每三点作为一个单位（BGR 元素）一次循环进行均值滤波，即把点的值变为周围 9 点的均值，然后输出即可，和 HW3 除操作的算法外很类似。

```
void imf(char* b)
{
  //先转grey图像
  if (Clr == 0)
  {
    for (int i = 0; i < pH; i++)//行
    {
      if (bitCount == 24)
      {
        for (int j = 0; j < pW * 3; j = j + 3)// 列
        {
          int n = i * LineSize + j;
          P[n] = P[n + 1] = P[n + 2] = (BYTE)(0.299 * P[n] + 0.587 * P[n + 1] + 0.114 * P[n + 2]); //分别是r,g,b分量
        }
      }
    }
  }
  //均值滤波
  for (int i = 1; i < pH - 1; i++)//行
  {
    for (int j = 3; j < (pW - 1) * 3; j = j + 3)// 列
    {
      //int n = i*LineSize + j;
      P[i * LineSize + j] = P[i * LineSize + j + 1] = P[i * LineSize + j + 2] = (BYTE)((P[i * LineSize + j] + P[i * LineSize + j + 3] + P[i * LineSize + j - 3] + P
[(i - 1) * LineSize + j] + P[(i - 1) * LineSize + j + 3] + P[(i - 1) * LineSize + j - 3] + P[(i + 1) * LineSize + j - 3] + P[(i + 1) * LineSize + j + 3] + P[(i + 1
) * LineSize + j]) / 9);  //等于9个像素点的均值

    }
  }
  FILE* dFile = fopen(b, "wb");//创建目标文件

  fwrite(&fileHeader, sizeof(BITMAPFILEHEADER), 1, dFile);   //写入文件头

  fwrite(&infoHeader, sizeof(BITMAPINFOHEADER), 1, dFile);  //写入文件信息头
  if (QUAD)
  {
    fwrite(QUAD, sizeof(RGBQUAD) * Clr, 1, dFile);
  }
  //写入图像数据
  fwrite(P, pSize, 1, dFile);
  fclose(dFile);

  if (QUAD)
  {
    free(QUAD);
    QUAD = NULL;
  }
  if (P)
  {
    free(P);
    P = NULL;
  }
}
```

2、拉普拉斯图像增强

与上述代码类似，除运用的操作算法不同外其余部分一样。

```c
//先转grey图像
  if (Clr == 0)
  {
    for (int i = 0; i < pH; i++)//行
    {
      if (bitCount == 24)
      {
        for (int j = 0; j < pW * 3; j = j + 3)// 列
        {
          int n = i * LineSize + j;
          P[n] = P[n + 1] = P[n + 2] = (BYTE)(0.299 * P[n] + 0.587 * P[n + 1] + 0.114 * P[n + 2]); //分别是r,g,b分量

        }
      }

    }
  }
  //拉普拉斯
  int w; //和
  for (int i = 1; i < pH - 1; i++)//行
  {
    for (int j = 3; j < (pW - 1) * 3; j = j + 3)// 列
    {
      //int n = i*LineSize + j;
      w = P[i * LineSize + j + 3] + P[i * LineSize + j - 3] + P[(i - 1) * LineSize + j] + P[(i - 1) * LineSize + j + 3] + P[(i - 1) * LineSize + j - 3] + P[(i + 1
) * LineSize + j - 3] + P[(i + 1) * LineSize + j + 3] + P[(i + 1) * LineSize + j] - P[i * LineSize + j] * 8;
      w *= 0.1;
      //权重正负分类

      if (w >= 0) {
        //
        if (P[i * LineSize + j] + w > 255)
          P[i * LineSize + j] = P[i * LineSize + j + 1] = P[i * LineSize + j + 2] = (BYTE)(255);
        else if (P[i * LineSize + j] + w < 0)
          P[i * LineSize + j] = P[i * LineSize + j + 1] = P[i * LineSize + j + 2] = (BYTE)(0);
        else
          P[i * LineSize + j] = P[i * LineSize + j + 1] = P[i * LineSize + j + 2] = (BYTE)(P[i * LineSize + j] + w);
      }

      else {
        if (P[i * LineSize + j] - w > 255)
          P[i * LineSize + j] = P[i * LineSize + j + 1] = P[i * LineSize + j + 2] = (BYTE)(255);
        else if (P[i * LineSize + j] - w < 0)
          P[i * LineSize + j] = P[i * LineSize + j + 1] = P[i * LineSize + j + 2] = (BYTE)(0);
        else
          P[i * LineSize + j] = P[i * LineSize + j + 1] = P[i * LineSize + j + 2] = (BYTE)(P[i * LineSize + j] - w);
      }
    }
  }
```

四、实验环境及运行方法

编译环境：C 语言，C11 标准。ide 使用 visual studio。可直接打开 HW5.c 源文件进行编译。

测试方法：断点单步测试

五、实验结果展示



原图　　均值滤波　　拉普拉斯增强

六、心得体会

本次作业基本都可以复制前两次的代码，较为简单轻松。