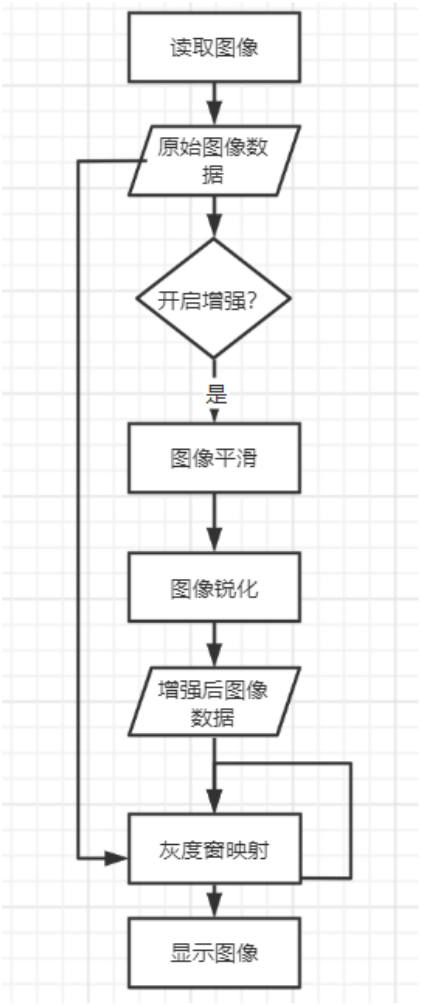
# 图像细节增强实验报告

## 处理流程

处理流程分 4 部分，依次为读取图像、图像平滑、图像锐化、灰度映射。图像平滑用于降低图像中的噪声，避免锐化之后噪声过于明显。图像锐化用于实现图像细节的增强。图像平滑和图像锐化可由用户选择是否开启，若开启，则这两步在读取原始图像后立刻实现，并将增强后的图像数据存储。灰度映射由用户指定窗宽和窗位，并将原始图像、增强后图像的灰度映射到 $0 \sim 255$ 之间。

上述处理过程用流程图描述如下：




处理流程

## 处理算法

1. 图像平滑：
   采用高斯滤波器，卷积核 $h$ 为 $3*3$ 矩阵。定义 $h_{0,0}$ 为卷积核左上角，则 $h(i,j)=\sqrt{2\pi}\sigma A e^{-2\pi^2\sigma^2((i-1)^2+(j-1)^2)}$，这里取 $\sigma=0.001, A=1$。将 $h$ 归一化，再对原图像做 $valid$ 卷积，便可得到平滑后的图像。
   代码实现如下：

```
void GaussFilter() {
    const static double pi = acos(-1.0);
    const static double sigma = 0.001;
    const static double A = 1;
    const static int coreSize = 3;
    static double core[3][3] = {};
    double totWeight = 0;
    for (int i = 0; i < coreSize; i++)
        for (int j = 0; j < coreSize; j++) {
            core[i][j] = A * sqrt(2 * pi) * sigma
                * exp(- 2.0 * pi * pi * sigma * sigma * ((i - coreSize / 2) * (i - coreSize
            totWeight += core[i][j];
        }
    for (int i = 0; i < coreSize; i++)
        for (int j = 0; j < coreSize; j++)
            core[i][j] /= totWeight;

    T* tmp = new T[nHeight * nWidth];
    for (int i = 0; i < nHeight * nWidth; i++) tmp[i] = pPixels[i];

    for (int i = 0; i <= nHeight - coreSize; i++)
        for (int j = 0; j <= nWidth - coreSize; j++) {
            double w = 0;
            for (int p = 0; p < coreSize; p++)
                for (int q = 0; q < coreSize; q++)
                    w += core[p][q] * tmp[(i + p) * nWidth + j + q];
            pPixels[(i + coreSize / 2) * nWidth + j + coreSize / 2] = w;
        }
    delete[] tmp;
}
```

2. 图像锐化：
   采用拉普拉斯锐化，记 $\hat{f}_{i,j}=f_{i,j}-\alpha\nabla^2 f_{i,j}$，其中 $f$ 为原始图像，$\hat{f}$ 为增强后图像。由于 $f$ 离散，因此用二阶差分代替二阶导数，从而有 $\nabla^2 f_{i,j}=(f_{i+1,j}-f_{i,j})-(f_{i,j}-f_{i-1,j})+(f_{i,j+1}-f_{i,j})-(f_{i,j}-f_{i,j-1})$。为保证二阶差分存在，限定 $1<i<nHeight, 1<j<nWidth$。
   定义 $3*3$ 卷积核 $h$ 如下：

$$h=\begin{pmatrix}0&1&0\\1&-4&1\\0&1&0\end{pmatrix}$$

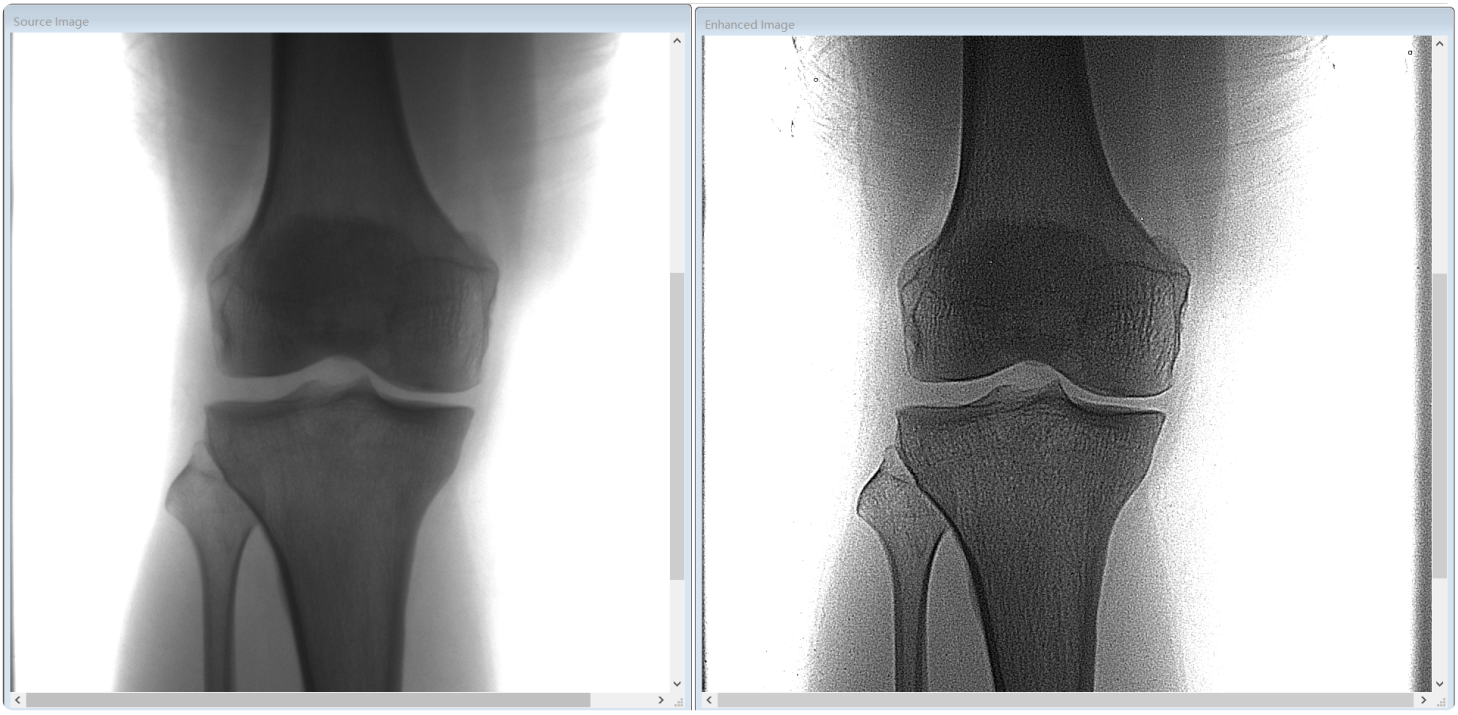
那么 $\nabla^2 f$ 可由原图像对 $h$ 做 $valid$ 卷积得到。
考虑到在对角线方向也存在边缘的变化，因此在对角线方向也应该做拉普拉斯锐化，于是对 $h$ 进行改进，得：

$$h=\begin{pmatrix}1&1&1\\1&-8&1\\1&1&1\end{pmatrix}$$

为实现图像锐化，应对高频分量进行增强，故取 $\alpha>0$。在本实验中，取 $\alpha=5.5$ 左右处理结果较好。

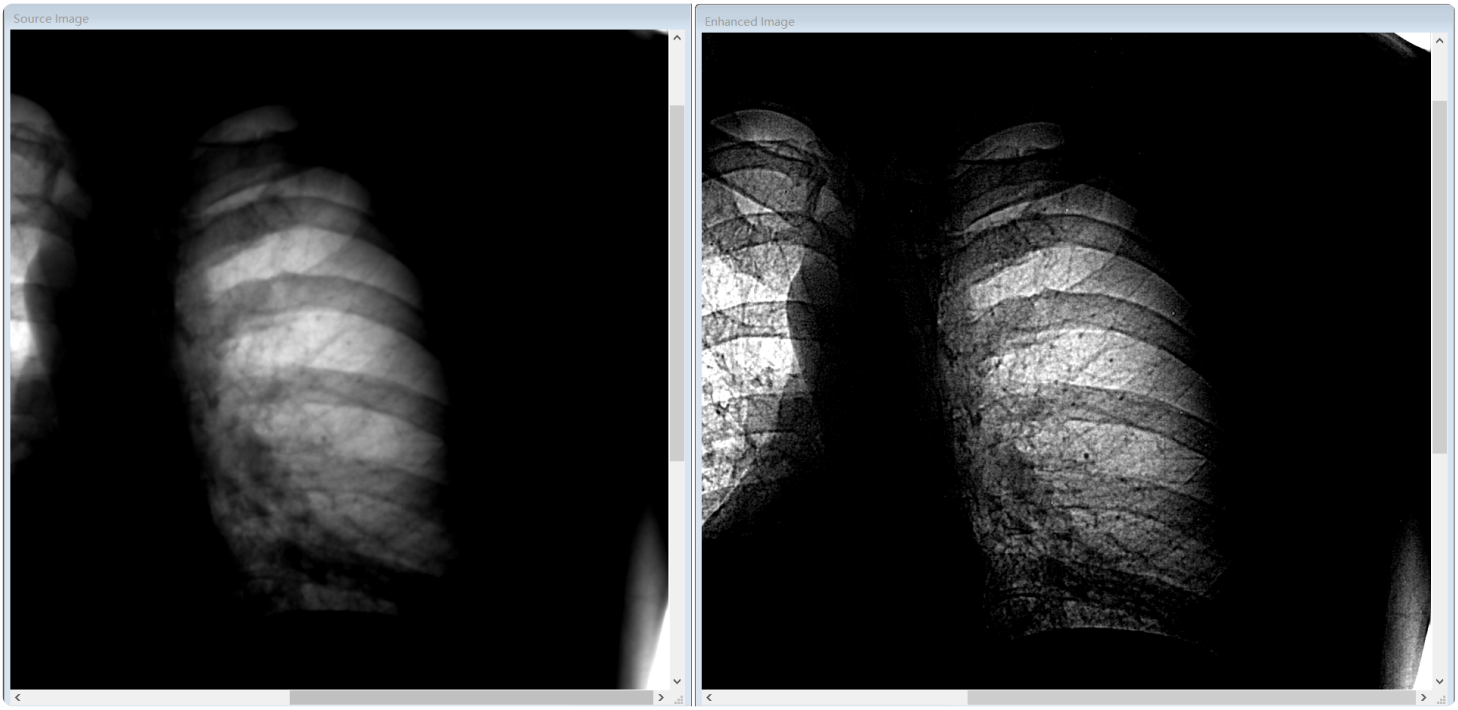
```
void Laplacian() {
    const static int coreSize = 3;
    const static int core[3][3] = {
        {1, 1, 1},
        {1, -8, 1},
        {1, 1, 1},
    };
    const static double a = 5.5;

    T* tmp = new T[nHeight * nWidth];
    for (int i = 0; i < nHeight * nWidth; i++) tmp[i] = pPixels[i];

    for (int i = 0; i <= nHeight - coreSize; i++)
        for (int j = 0; j <= nWidth - coreSize; j++) {
           int w = 0;
            for (int p = 0; p < coreSize; p++)
                for (int q = 0; q < coreSize; q++)
                    w += core[p][q] * tmp[(i + p) * nWidth + j + q];
            if (w * a <=  pPixels[(i + coreSize / 2) * nWidth + j + coreSize / 2]) pPixels[(
            else pPixels[(i + coreSize / 2) * nWidth + j + coreSize / 2] = 0;
        }
    delete[] tmp;
}
```

# 处理结果




knee[250, 500]


lung[2048,4096]




lung[3000,2000]

# 分析评价

从实验结果来看，图像质量得到明显提高，噪声也没有明显的放大，上述处理处理过程中处理算法和参数的选择较为合理。做的不足的是，在选取参数时，并没有通过频谱进行分析，而是根据处理结果进行手动调整。此外，使用拉普拉斯锐化会导致伪影的出现，锐化后图像中出现的条纹可能不是真实存在的，对医学图像而言这是有害的。