PDEs在图形去噪中的研究

3110102577 戴维　3110102522 林敏杰　3110102421 诸葛军暇

2013年7月25日

目录

Chapter 1

研究问题

1.1 历史背景及其意义

图像处理是指将图像信号转换成数字信号并利用计算机对其进行处理的过程。图像处理最早出现于20世纪50年代，当时的电子计算机已经发展到一定水平，人们开始利用计算机来处理图形和图像信息。数字图像处理作为一门学科大约形成于20世纪60年代初期。早期的图像处理的目的是改善图像的质量，它以人为对象，以改善人的视觉效果为目的。图像处理中，输入的是质量低的图像，输出的是改善质量后的图像，常用的图像处理方法有图像增强、复原、编码、压缩等。

早期由于图像处理领域涉及的数学理论较浅，尽管图像处理与分析与计算机科学有很强的联系，但在相当长的一段时间里一些在特定条件下的算法的正确性没有得到很好的证明，图像处理研究的进展不大。近年来由于该领域研究者数学功底的增强，同时，由于该领域的巨大市场需求吸引了越来越多的数学工作者的加入。使该领域得到了前所未有的发展。图像增强、图像恢复和图像分割是图像处理与分析中的主要问题，对图像进行平滑和边缘检测等处理是常用的方法；然而，图像的平滑和边缘的保持是一对矛盾的关系；图像的低通滤波在降噪的同时模糊的图像的边界。而人对图像的高频部分（边缘细节）是很敏感的，图像的大部分信息存在于边缘和轮廓部分。传统的滤波和边缘检测方法难以处理这类问题。由于基于PDEs的图像处理方法在平滑噪声的同时可以使边界得到保持，因此在图像处理中得到广泛的运用。

近年来，图像处理和计算机视觉中应用偏微分方程受到国内外学者的极大关注。国际上一些图像处理和计算机视觉领域的顶级学术期刊(如：IEEE Trans.on PAMI、IEEE Trans.onimage Processing与International Journal of Computer Vision Journal of VisualCommunication and Image Representation等)分别专门发表过以PDE应用为主题的特刊，国际学术会议CVPR，ICCV，ICIP等也为此召开了特别国际会议。目前美国加州大学洛杉矶分校、法国Sophia Antipolis等机构在该领域研究处于领先地位。近年来国内也非常重视对该领域的研究，中科院自动化所等单位也为此召开了专题研讨会。

1.2 研究现状及发展动态

PDEs在图像处理领域的应用研究最早可以追溯到D.Gabor和A.K.Jain的工作。然而该领域研究的真正兴起源于Koenderink和Witkin各自独立的研究。他们严格引入尺度空间的概念，他们的这种原创性工作是PDEs在图像处理领域运用的基础。

Perona和Malik在各向异性扩散方面的论文是图像正则化领域最有影响的研究成果（下文简称P-M方法）。他们建议用一种保边界的扩散来代替基于热传导等式的各向同性扩散的高斯光滑滤波。在此基础上S.Osher和L.Rudin等提出了冲击滤波器(Shock Filters)，L.Rudin提出的全变分(Total Variation)下降法(下文简称TV方法)，Price等提出的反应-扩散等式，都成为当前PDEs在图像恢复领域成功应用的典范与当前研究的热点。

在图像处理和计算机视觉中应用的PDEs大多数研究如何基于曲率的速率流来改变曲线、曲面或图像的位置。在该领域Osher和Sethian[9]提出的水平集数值方法具有深远的影响，该方法的基本思想就是用高维的超曲面的水平集表征变形的曲线、曲面或图像。该技术不但提供了PDEs的非常精确的数值解法，而且解决了非常棘手的拓扑问题。

此外D.Mumford与J.Shah提出的M-S图像分割方法，Kass提出的snake活动轮廓模型等都是PDEs在图像分割与目标提取领域开创性的成果。

Chapter 2

实验内容及解决方案

2.1 实验目标及研究内容

本试验的目标是运用PDE中的热扩散方程方法，P-M方法和TV方法对Lena图进行消噪，取得较好的消噪效果，并对结果进行分析以及比较。

研究的内容包括：将矩阵转化为稀疏矩阵的rowidex,colidex,entries的三元组，三元组转化为矩阵，对三元组的加法、乘法，对三元组的Gauss-sediel迭代方法，热扩散方程的图形去噪模型以及去噪的数值实现，PM 方法图形去噪模型以及去噪数值实现，TV方法的图形去噪模型以及去噪数值实现。

完成以上目标以及内容，最关键的问题是：

- 热扩散方程：对稀疏矩阵的三元组操作，在大矩阵的情况下，matlab无法运行，因此只能用稀疏矩阵的三元组进行操作；有限差分法矩阵的构造，通过公式以及边值条件推导出矩阵，并且用matlab将矩阵变成稀疏矩阵的三元组形式；
- PM方法：公式推导以及编程上的技巧；
- TV方法：直接运用梯度，公式推导以及编程上的技巧。

2.2 实验解决方案

2.2.1 热扩散方程方法

首先，做出将矩阵与稀疏矩阵三元组的转化，三元组加法、乘法的操作，对三元组Gauss-Sediel迭代法，为下面操作准备。进行热方程方法对图形去噪：通过有限差分法将热扩散方程化简，得到方程组形式，结合边界条件构造出矩阵并且转化为稀疏矩阵的三元组，并进行三元组Gauss-Sediel迭代,得到去噪后的图。

矩阵与稀疏矩阵间的操作

将矩阵转化为三元组rowidx,colidx,entries。

rowidx[$m+1$] 存放第i行中第一个非零元素的colidx中指标，但是若对角线为零，且对角线前无非零元，则存对角线元素的colidx中指标。设矩阵共m行，则rowidx($m+1$)=N，N为非零元个数以及加上对角线元素（当对角线元素为0）。

colidx[N] 按行存放该行非零元素列指标以及对角元素（当对角元素为0）的列指标。

entries[N] 存放Colidx中元素的值。

Matlab编程的关键技巧在于先对该行对角线元素，再对其他非零元操作。将三元组转化为矩阵，由上易知方法。（代码见附录1）

三元组的加法、乘法

（代码见附录2）

三元组Gauss-Sediel迭代法

$$AX = b \tag{2.1}$$

$$\begin{aligned} A &= (a_{ij})_{n*n} \quad i,j = 1,\cdots,n && (2.2)\\ X &= (x_1, x_2, \cdots, x_n)^{\mathrm{T}} && (2.3)\\ b &= (b_1, b_2, \cdots, b_n)^{\mathrm{T}} && (2.4) \end{aligned}$$

$$x_i^{n+1} = \frac{1}{a_{ii}}(-\sum_{j<i} a_{ij}x_j^{n+1} - \sum_{j>i} a_{ij}x_j^n + b_i) \tag{2.5}$$

Matlab编程的关键是按行恢复，进行操作。（代码见附录3）

热扩散方程方法

$$\begin{cases} \dfrac{\partial I}{\partial t} = \dfrac{\partial^2 I}{\partial x^2} + \dfrac{\partial^2 I}{\partial y^2} \\ I(t=0) = I_0 \\ \dfrac{\partial I}{\partial n} = 0 \end{cases} \tag{2.6}$$

运用有限差分法，得：

$$\begin{cases} I_{i,j}^n = I(x_i, y_j, t_n) \\ x_i = i * h \\ y_j = j * h \\ t_n = n * \Delta t \end{cases} \tag{2.7}$$

$$\frac{I_{i,j}^{n+1} - I_{i,j}^n}{\Delta t} = \frac{I_{i+1,j}^{n+1} - 2I_{i,j}^{n+1} + I_{i-1,j}^{n+1}}{h^2} + \frac{I_{i,j+1}^{n+1} - 2I_{i,j}^{n+1} + I_{i,j-1}^{n+1}}{h^2} \tag{2.8}$$

令$\frac{h^2}{\Delta t}=r$

$$I_{i,j}^{n}=-rI_{i-1,j}^{n+1}-rI_{i,j-1}^{n+1}+(1+4r)I_{i,j}^{n+1}-rI_{i+1,j}^{n+1}-rI_{i,j+1}^{n+1} \tag{2.9}$$

算法步骤：

Step1 读入带噪图形I（Lena图）；

Step2 根据上式以及边界条件，构造出五对角矩阵并转化为稀疏矩阵的三元组形式，记矩阵为A，并存为r,c,e；

Step3 b为Lena图的矩阵的数值以及由边界条件为0的值的列向量；

Step4 当次数n <最大迭代次数，重复执行4.1；

Step4.1 由AX=b，X就是通过热方程方法处理后的数值AX=b，用对三元组的Gauss-Sediel迭代法求出X；

Step5 结束迭代，最后一次迭代结果即为所求去噪图形。

（代码见附录4）

2.2.2 P-M方法

P-M方法图形去噪模型

非线性的各向异扩散滤波方法，使图像在不同区域得到不同程度的平滑（及使用不同的扩散系数），从而是图像的边缘特征得到较好的保护。

正对热扩散方程的特点，Perona进一步完善了去噪方程必须满足的三个准则：

1. 因果性：当尺度由小变大时，不会产生新的细节特征；
2. 即时定位：在任意尺度下，图像中的区域边界要明显，并与对应尺度下图像中的物体边界相一致；
3. 分段平滑：在所有尺度上，区域内部要优先于区域之间进行平滑。

根据以上规定，Perona和Malik提出各向异性扩散模型。在常系数热传导方程产生的线性尺度空间中，热传导系数为1，它没有考虑图像的空间位置。其实，热传导系数要根据图像的空间位置来确定，Perona给出的热传导系数可变的异性扩散方程如下：

$$\frac{\partial I(x,y,t)}{\partial t}=div(c(x,y,t)\bigtriangledown I(x,y,t)) \tag{2.10}$$

方程出事条件为图像$I(x,y,0)$，$c(x,y,t)$为1时，方程变为给徐昂同性的热传导方程，假设在每个瞬时时间尺度上，已知图像区域的边界，则在图像区域的内部进行平滑，而在穿过区域边界的位置上，不执行平滑过程。

方程(2.10)基本思想是利用一种扩散系数$c(x,y,t)$来控制空间个点的扩散量，使扩散主要发生在图像个区域内部，因此随着扩散的进行，低对比度的区

域内部变得越来越平滑，而高对比度的区域（边缘）则得到平滑和锐化的平衡。

Perona等人提出$c(x,y,t)$是这样选择的：他是关于图像函数的梯度幅度值的函数，即

$$c(x,y,t) = g(|| \bigtriangledown I(x,y,t)||) \tag{2.11}$$

通常采用两种式$g_1(x) = \exp(-(x/K)^2)$和$g_2(x) = \frac{1}{1+(x/K)^2}$

当$x \to \infty$时，$g(x) \to 0$，$K = 40$为梯度阈值，当梯度值小于K 时，起的作用跟低通滤波器一样，的那个梯度值接近K 时，扩散过程停止。当梯度值大于K时，扩散过程还能增强轮廓。因此在一定程度上实现对边缘的保持和增强。

P-M方法去噪数值实现

对于P-M模型可以写成如下形式：

$$\partial_t I = \partial_x[g(| \bigtriangledown I|)\partial_x I] + \partial_y[g(| \bigtriangledown I|)\partial_y I] \tag{2.12}$$

对式()进行差分，方向和方向是等步长的，即$\Delta x = \Delta y = h$，时间步长$\Delta t = \tau$，网格坐标为$(ih, jh, n\tau), h = 1/N, 0 < i \leq N, 0 < j \leq N$，$I^n_{i,j}$是$I(ih, jh, n\tau)$的近似值，$g_{i,j} = g(| \bigtriangledown I|)_{i,j}, \lambda = \tau/h^2$。

$$\frac{\partial^2 I}{\partial x^2} = (\frac{I_{i+1,j} - I_{i,j}}{h} - \frac{I_{i,j} - I_{i-1,j}}{h})/h = \frac{I_{i+1,j} - I_{i,j}}{h^2} + \frac{I_{i-1,j} - I_{i,j}}{h^2} \tag{2.13}$$

$$\frac{\partial^2 u}{\partial y^2} = (\frac{I_{i,j+1} - I_{i,j}}{h} - \frac{I_{i,j} - I_{i,j-1}}{h})/h = \frac{I_{i,j+1} - I_{i,j}}{h^2} + \frac{I_{i,j-1} - I_{i,j}}{h^2} \tag{2.14}$$

$$\partial_x[g(| \bigtriangledown I|\partial_x I)] + \partial_y[g(| \bigtriangledown I|\partial_y I)] =$$

$$\frac{\tau}{h^2}[g(||(\bigtriangledown I)_{i+1/2,j}||)(I_{i-1,j} - I_{i,j}) + g(||(\bigtriangledown I)_{i-1/2,j}||)(I_{i+1,j}j - I_{i,j}) +$$

$$g(||(\bigtriangledown I)_{i,j+1/2}||)(I_{i,j-1} - I_{i,j}) + g(||(\bigtriangledown I)_{i,j-1/2}||)(I_{i,j+1} - I_{i,j})] \tag{2.15}$$

令

$$\bigtriangledown_N I_{i,j} \equiv I_{i-1,j} - I_{i,j} \tag{2.16}$$

$$\bigtriangledown_S I_{i,j} \equiv I_{i+1,j} - I_{i,j} \tag{2.17}$$

$$\bigtriangledown_E I_{i,j} \equiv I_{i,j+1} - I_{i,j} \tag{2.18}$$

$$\bigtriangledown_W I_{i,j} \equiv I_{i,j-1} - I_{i,j} \tag{2.19}$$

$$c^t_{N(i,j)} = g(||(\bigtriangledown I)^t_{i+1/2,j}||) \tag{2.20}$$

$$c^t_{S(i,j)} = g(||(\bigtriangledown I)^t_{i-1/2,j}||) \tag{2.21}$$

$$c^t_{E(i,j)} = g(||(\bigtriangledown I)^t_{i,j+1/2}||) \tag{2.22}$$

$$c^t_{W(i,j)} = g(||(\bigtriangledown I)^t_{i,j-1/2}||) \tag{2.23}$$

则上式可化为：

$$I_{i,j}^{t+1} = I_{i,j}^{t} + \lambda[c_N \cdot \bigtriangledown_N I + c_S \cdot \bigtriangledown_S I + c_E \cdot \bigtriangledown_E I + c_W \cdot \bigtriangledown_W I]_{i,j}^{t} \tag{2.24}$$

为了求解的更快，不采用稀疏矩阵三元组这样操作的方法，根据推导出的公式来进行编程求解。

算法步骤：

Step1 读入带噪图形I;

Step2 初始化参数：$t = 0, \Delta t = 0.25, h = 1, I^0 = I_0, K = 40$;

Step3 当$n <$最大迭代次数时

Step3.1 $t = t + 1$,根据公式(2.24)计算

Step3.2 由公式计算$\bigtriangledown_N I_{i,j}, \bigtriangledown_S I_{i,j}, \bigtriangledown_E I_{i,j}, \bigtriangledown_W I_{i,j}, c_{N(i,j)}^t, c_{E(i,j)}^t, c_{S(i,j)}^t, c_{W(i,j)}^t$。

Step4 结束迭代，最后一次迭代结果即为所求去噪图像。

（代码见附录5）

2.2.3 TV方法

TV图像去噪模型

由公式推导，得：

$$-\bigtriangledown \cdot (\frac{\bigtriangledown I}{|\bigtriangledown I|}) + \lambda(I - I_0) = 0 \tag{2.25}$$

从方程可以看出，扩散系数为$\frac{1}{|\bigtriangledown I|}$。在图像边缘处，$|\bigtriangledown I|$较大，扩散系数较小，因此沿边缘方向的扩散较弱，从而保留了边缘；在平滑区域，$|\bigtriangledown I|$较小，扩散系数较大，因此在图像平滑区域的扩散能力较强，从而去除了噪声。

TV去噪的数值实现

用$I_{i,j}$表示图像I在像素点$x_i = ih, y_j = jh$的灰度值，其中$i, j = 0, 1, \cdots, N, Nh = L$，$L$为图像的长度，$h$是空间步长，$I(x_t, y_t, t_n)$表示第n次迭代的值，并记做$I_{i,j}^n$，其中$t_n = n \cdot \Delta t$，$\Delta t$为时间步长。本实验采用最陡下降法实现TV去噪，其扩散项为：

$$\bigtriangledown \cdot (\frac{\bigtriangledown I}{|\bigtriangledown I|}) = \frac{I_y^2 I_{xx} - 2I_x I_y I_{xy} + I_x^2 f_{yy}}{I_x^2 + I_y^2} \tag{2.26}$$

用差商代替偏导数，可得：

$$\begin{cases} (I_x)_{i,j}^n = I_{i+1,j}^n - I_{i-1,j}^n \\ (I_y)_{i,j}^n = I_{i,j+1}^n - I_{i,j-1}^n \\ (I_{xx})_{i,j}^n = I_{i+1,j}^n - 2I_{i,j}^n + I_{i-1,j}^n \\ (I_{yy})_{i,j}^n = I_{i,j+1}^n - 2I_{i,j}^n + I_{i,j-1}^n \\ (I_{xy})_{i,j}^n = I_{i+1,j+1}^n - I_{i-1,j+1}^n - I_{i+1,j-1}^n + I_{i-1,j-1}^n \end{cases} \tag{2.27}$$

则求解上述方程的的离散迭代格式为：

$$I_{i,j}^{n+1} = I_{i,j}^{n} - \Delta t\lambda(I_{i,j}^{n} - I_{i,j}^{0}) + \Delta t(\bigtriangledown \cdot (\frac{\bigtriangledown I_{i,j}^{n}}{|\bigtriangledown I_{i,j}^{n}|})) \tag{2.28}$$

其中，n为迭代次数；$i, j = 0, 1, \cdots, N$。边界条件满足$I_{0,j}^{n} = I_{1,j}^{n}, I_{N,j}^{n} = I_{N-1,j}^{n}, I_{i,0}^{n} = I_{1,N}^{n} = I_{i,N-1}^{n}$。

算法步骤：

Step1 读入带噪图像I_0；

Step2 初始化参数：$\lambda = 0, n = 0, \Delta t = 0.25, h = 1, I^0 = I_0, divp = \bigtriangledown \cdot (\frac{\bigtriangledown I_{i,j}^{n}}{|\bigtriangledown I_{i,j}^{n}|}) = 0$；

Step3 当n¡最大迭代次数时，重复执行**Step3.1**至**Step3.2**：

Step3.1 $n = n + 1$，根据公式(2.28)计算$I_{i,j}^{n}$；

Step3.2 根据公式计算扩散项$divp = \bigtriangledown \cdot (\frac{\bigtriangledown I_{i,j}^{n}}{|\bigtriangledown I_{i,j}^{n}|})$；

Step4 结束迭代，最后一次迭代结果即为所求去噪图形。

（代码见附录6）

Chapter 3

实验的结果和结论

3.1 热扩散方程方法结果

各参数选取为：$t = 1, h = 1$。
经过Matlab处理后，去噪结果如下：

(a) 原图

(b) 热扩散迭代5次

(c) 热扩散迭代50次

(d) 热扩散迭代100次

可以看出，热扩散方程方法对Lena图像去噪的效果比较好，即使对含噪图

处理迭代了100 次，Lena还是非常好地保持了脸部，帽檐以及手臂等部分的边缘还没有被平滑掉。当然也可以看到，当迭代50次以后，去噪效果就不太明显。可得出结论：热扩散方程方法特征为整体模糊。

3.2 P-M方法结果

各个参数的选取为：时间步长$t = 0.25$,边缘值$K = 40$，扩散系数函数$g_2(x) = \frac{1}{1+(x/K)^2}$。

经过Matlab处理后，去噪结果如下：

(e) 原图

(f) P-M方法迭代5次

(g) P-M方法迭代50次

(h) P-M方法迭代100次

可以看出，P-M方法对Lena图像去噪的效果相当好，尽管使用P-M方法对含噪的Lena图像迭代了50次，Lena还是良好地保持了脸部，帽檐以及手臂等部分的边缘还没有被平滑掉。

同样也可以发现Lena的帽子上的羽毛纹理，嘴唇等一些细节出现了模糊。这是因为P-M扩散方程能够良好捕捉图像梯度比较大的变化，无法捕捉由于细小变化所产生的纹理和细小的边缘，于是出现了这些图像信息的丢失。在噪点减小的同时又出现了一些比较大的噪点，这是由于某些噪点因为梯度值表达而被误判为边缘，这样，它们没有被去除，反而在某种程度上得到加强。

当使用P-M方法对含噪Lena图像迭代了100次后，图像中Lena的眉毛，羽毛纹理等凝聚成块状，与周围区域的过度很不自然，这就是“阶梯效应”。即图

像经过P-M扩散后，产生很多灰度值相同的区域。P-M方法特征为：相同区域模糊化相似。

3.3 TV方法结果

各参数选取为：

$$\begin{aligned} \lambda &= 1 \\ \Delta t &= 0.25 \\ h &= 1 \\ divp &= \bigtriangledown \cdot (\frac{\bigtriangledown I_{i,j}^n}{|\bigtriangledown I_{i,j}^n|}) = 0 \end{aligned}$$

经过matlab处理后，去噪结果如下：

(i) 原图

(j) TV方法迭代5次

(k) TV方法迭代50次

(l) TV方法迭代100次

可以看出，TV方法对Lena图像去噪的效果很好，即使TV方法对含噪的Lena图像迭代了100 次，Lena还是良好地保持了脸部，帽檐以及手臂等部分的边缘还没有被平滑掉。并且TV方法并没有像P-M方法，在100次后，图像大部分都被平滑掉。并且，TV方法在迭代100次时，图像去噪十分明显，并且图中Lena的脸、帽子等仍然很清晰。

为了客观地评价去噪效果，才去峰值信噪比($PSNR$)作为评价指标。$PSNR$越大，去噪能力越强。设f表示原始的无噪声图像，$\hat{f}$表示去噪后的图像，图像大小为$M*N$，$PSNR$定义为：

$$PSNR = 10log_{10}(255^2/MSE)$$

其中均方误差$MSE = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(f[i,j]-\hat{f}[i,j])^2$。选取每个方法迭代5次的图进行比较：

(m) 原图

(n) 热扩散方法 PSNR=23.1375

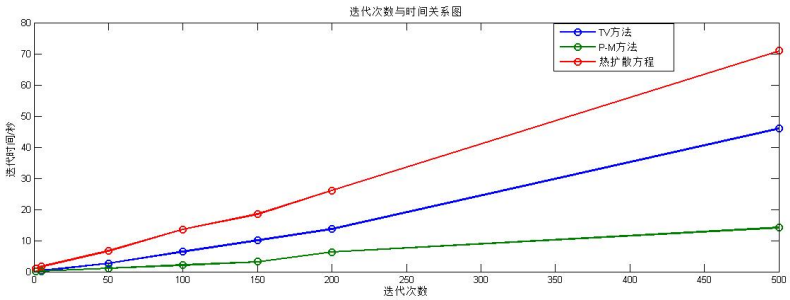
(o) P-M方法 PSNR=22.7196

(p) TV方法 PSNR=37.5631

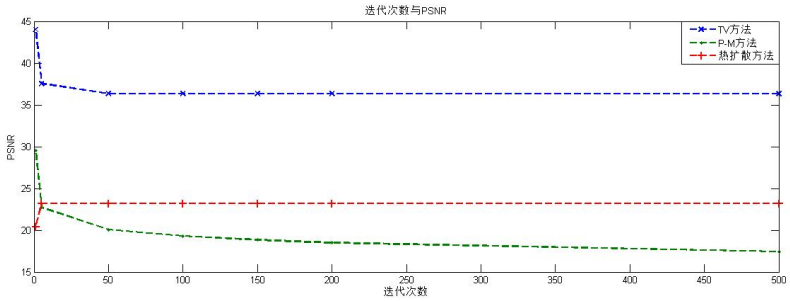
可以看出在迭代次数较少的情况下，TV方法的PSNR最高为37.5631，可以说明热扩散方程方法、PM方法、TV方法中，TV方法的效果更加好。

如下图q为3种方法在迭代次数为1,5,50,100,150,200,500次时与迭代时间的关系图，可以看出，热扩散方程方法耗时最长，P-M方法耗时最短。

如下图r为3种方法在迭代次数为1,5,50,100,150,200,500次时与PSNR的关系图，TV方法的PSNR最高，去噪处理效果最好，P-M方法在前5 次迭代中，去噪效果比热扩散方法好，但是在5次迭代以后，热扩散方法比P-M方法去噪效果好。



(q) 迭代次数与时间关系图



(r) 迭代次数与PSNR关系图

参考文献

[1] 周晓,朱才志.偏微分方程在图像处理中的应用.安徽教育学院学报,2007,25(3).

[2] 胡浩,王明照.自适应模糊加权均值滤波器[J].系统工程与电子技术,2002,24(2):15-17.

[3] 靳明,宋建中.一种自适应的图像双边滤波方法[J].光电工程,2004,31(7):65-68.

[4] 王发牛,韦穗.基于偏微分方程的噪声消除方法[J].计算机技术与发展,2002,12(4):81-82.

[5] Gabor D.Information theory in electron microscopy [J].Lab Investig, 1965, 14; 801-807.

[6] Jain A K.Partial differential equations and finite-difference methods in image processing,part 1: Image representation [J].J Optim Theory Appl, 1997, 23:65-91.

[7] Koenderink J J. The structure of the image [J]. Biol Cybernet, 1984, 50: 363-370.

[8] Rudin L I,Osher S,Fatemi E.Nonlinear total variation based noise removal algorithms.Physica D,1992,60:259-268.

[9] Chan T F,Golub G,Mulet P. A nonlinear primal-dual method for total variation-based image restoration. SIAM J.Sci.Comp.1999,20:1964-1977.

Appendices

附录1

```
function [rowidx colidx entrices] = spattern(A)
[m,n] = size(A);
rowidx = zeros(1,m+1);
colidx = [];
entrices = [];
numb = 0;
for i = 1:m
    numb = numb+1;
    rowidx(i) = numb;
    colidx(numb) = i;
    entrices(numb) = A(i,i);
    for j = 1:n
        if(A(i,j) ~= 0 & j ~= i)
            numb = numb + 1;
            colidx(numb) = j;
            entrices(numb) = A(i,j);
        end
    end
end
rowidx(m+1) = numb;

function [matr] = sparsen(rowidx,coo,entrices)
[m,n] = size(entrices);
[l,k] = size(rowidx);
matr = zeros(k-1,k-1);
a = 1;
for i = 1:n
    c = entrices(1,i);
    b = coo(1,i);
    for j = k-1:-1:1
        if (i >= rowidx(1,j))
            a = j;
            break
        end
    end
    matr(a,b)=c;
end
```

附录2

```
function [Crowidx,Ccolidx,Centrices] = sprpls(Arowidx,Acolidx,...
Aentrices,Browidx,Bcolidx,Bentrices)
[~,k] = size(Arowidx);
[~,r] = size(Aentrices);
[~,s] = size(Bentrices);
Crowidx = zeros(1,k);
Crowidx(1) = 1;
Ccolidx = [];
Centrices = [];
num = 1;
for i = 1:k-1
    if(i ~= k-1)
        a = Arowidx(i+1) - Arowidx(i);
        b = Browidx(i+1) - Browidx(i);
    else
        a = Arowidx(i+1) - Arowidx(i)+1;
        b = Browidx(i+1) - Browidx(i)+1;
    end
    Aj = 1;
    Bj = 1;
    while(Aj <= a && Bj <= b)
        if(Acolidx(Arowidx(i)-1+Aj) < Bcolidx(Browidx(i)-1+Bj))
            Ccolidx(num) = Acolidx(Arowidx(i)-1+Aj);
            Centrices(num) = Aentrices(Arowidx(i)-1+Aj);
            num = num + 1;
            Aj = Aj + 1;
        elseif(Acolidx(Arowidx(i)-1+Aj) > Bcolidx(Browidx(i)-1+Bj))
            Ccolidx(num) = Bcolidx(Browidx(i)-1+Bj);
            Centrices(num) = Bentrices(Browidx(i)-1+Bj);
            num = num + 1;
            Bj = Bj + 1;
        else
            Ccolidx(num) = Acolidx(Arowidx(i)-1+Aj);
            Centrices(num) = Aentrices(Arowidx(i)-1+Aj) + ...
            Bentrices(Browidx(i)-1+Bj);
            num = num + 1;
            Aj = Aj + 1;
            Bj = Bj + 1;
        end
    end
    if(Aj > a)
        for z = Bj:b
            Ccolidx(num) = Bcolidx(Browidx(i)-1+z);
            Centrices(num) = Bentrices(Browidx(i)-1+z);
            num = num + 1;
        end
    else
        for z = Aj:a
            Ccolidx(num) = Acolidx(Arowidx(i)-1+z);
            Centrices(num) = Aentrices(Arowidx(i)-1+z);
            num = num + 1;
        end
    end
    Crowidx(i+1) = num;
```

```
end
Crowidx(k) = num - 1;

function [Crowidx,Ccolidx,Centrices] = sprmulti(Arowidx,Acolidx,...
Aentrices,Browidx,Bcolidx,Bentrices)
[~,k] = size(Arowidx);
[~,l] = size(Bentrices);
Crowidx = zeros(1,k);
Crowidx(1) = 1;
Centrices = [];
Ccolidx = [];
num = 1;
ent = 0;
for i = 1:k-1
    if(i ~= k-1)
        j = Arowidx(i+1) - Arowidx(i);
    else
        j = Arowidx(i+1) - Arowidx(i) + 1;
    end
    for n = 1:k-1
        for p = 1:l
            if(Bcolidx(p) == n)
                for q = k-1:-1:1
                    if(Browidx(q) <= p)
                        break;
                    end
                end
                for t = 1:j
                    if(q == Acolidx(Arowidx(i)-1+t))
                        ent = ent + Bentrices(p) * ...
                        Aentrices(Arowidx(i)-1+t);
                    end
                end
            end
        end
        if(ent ~= 0 || i == n)
            Centrices(num) = ent;
            Ccolidx(num) = n;
            num = num + 1;
        end
        ent = 0;
    end
    Crowidx(i+1) = num;
end
Crowidx(end) = num - 1;
```

附录3

```
function X = gauss(rowidx,celidx,entries,b)
m = length(rowidx);
rowidx(m) = rowidx(m) + 1;
X = zeros(m-1,1);
sum = 0;
for j = 1:1
  for i = 1:m-1
     for k = rowidx(i):rowidx(i+1)-1
         if celidx(k) == i
              a = entries(k);
              if a == 0
                  a = 0.0008;
              end
         else
             sum = sum + X(celidx(k)) * entries(k);
         end
     end
     X(i) = (b(i)-sum)/a;
     sum = 0;
  end
end
```

附录4

```
function [] = lenagraph(t)
lena = imread('LenaNoisedSq.jpg');
l = 434;
b = zeros(l*l,1);
for i = 2:l-1
    for j = 2:l-1
        b(i+(j-1)*l) = lena(i,j);
    end
end
[r,c,e] = matr(434,1,1);
X = gauss(r,c,e,b,t);
X = reshape(X,434,434);
imshow(X,[]);

function [rowidx colidx entrices] = matr(n,t,h)
r = t/(h)^2;
rowidx = zeros(1,n^2+1);
rowidx(1) = 1;
colidx = [];
entrices = [];
temp = 0;
for j = 1:n
    colidx(2*j-1) = j;
    entrices(2*j-1) = r;
    colidx(2*j) = j + n;
    entrices(2*j) = -r;
    rowidx(j+1) = 2 * j + 1;
    temp = temp + 2;
end
for i = 2:n-1
    colidx(temp+1) = (i-1) * n + 1;
    entrices(temp+1) = r;
    colidx(temp+2) = (i-1) * n + 2;
    entrices(temp+2) = -r;
    temp = temp + 2;
    rowidx((i-1)*n+2) = temp + 1;
    for k = 2:n-1
        colidx(temp+1) = (i-2) * n + k;
        entrices(temp+1) = -r;
        colidx(temp+2) = (i-1) * n + k - 1;
        entrices(temp+2) = -r;
        colidx(temp+3) = (i-1) * n + k;
        entrices(temp+3) = 1 + 4 * r;
        colidx(temp+4) = (i-1) * n + k + 1;
        entrices(temp+4) = -r;
        colidx(temp+5) = i * n + k;
        entrices(temp+5) = -r;
        temp = temp + 5;
        rowidx((i-1)*n+k+1) = temp + 1;
    end
    colidx(temp+1) = i * n - 1;
    entrices(temp+1) = -r;
    colidx(temp+2) = i * n;
    entrices(temp+2) = r;
```

```
55      temp = temp + 2;
56      rowidx(i*n+1) = temp + 1;
57  end
58  for j = 1:n
59      colidx(temp+1) = n^2 - 2 * n + j;
60      entrices(temp+1) = r;
61      colidx(temp+2) = n^2 - n + j;
62      entrices(temp+2) = -r;
63      rowidx(n^2-n+j+1) = rowidx(n^2-n+1) + 2 * j;
64      temp = temp + 2;
65  end
66  rowidx(end) = temp;
```

附录5

```
function It = smooth_diffusion(I,niter,K,Io,dt)
if ~exist('K') K = 40; end
if ~exist('Io') Io = I; end
if ~exist('dt') dt = 0.25; end
[row,col,nchannel] = size(I);
k = K;
a = 1;
for i = 1:niter
    Gn = [I(1,:,:);I(1:row-1,:,:)]-I;   %N-O
    Gs = [I(2:row,:,:);I(row,:,:)]-I;   %S-O
    Ge = [I(:,2:col,:) I(:,col,:)]-I;   %E-O
    Gw = [I(:,1,:) I(:,1:col-1,:)]-I;   %W-O
    Cn = 1./(1+(Gn/k).^2).*a;
    Cs = 1./(1+(Gs/k).^2).*a;
    Ce = 1./(1+(Ge/k).^2).*a;
    Cw = 1./(1+(Gw/k).^2).*a;
    I = I + dt*(Cn.*Gn+Cs.*Gs+Ce.*Ge+Cw.*Gw);
end
imshow(uint8(I));drawnow;
It=I;
end
```

附录6

```
function J = tv(I,iter,dt,ep,lam,I0,C)
I = double(I(1:434,1:434));
if ~exist('ep')
    ep=1;
end
if ~exist('dt')
    dt=ep/4;
end
if ~exist('lam')
    lam=1;
end
if ~exist('I0')
    I0=I;
end
if ~exist('C')
    C=0;
end
[ny,nx]=size(I); ep2=ep^2;

for i=1:iter,
    I_x = (I(:,[2:nx nx])-I(:,[1 1:nx-1]))/2;
    I_y = (I([2:ny ny],:)-I([1 1:ny-1],:))/2;
    I_xx = I(:,[2:nx nx])+I(:,[1 1:nx-1])-2*I;
    I_yy = I([2:ny ny],:)+I([1 1:ny-1],:)-2*I;
    Dp = I([2:ny ny],[2:nx nx])+I([1 1:ny-1],[1 1:nx-1]);
    Dm = I([1 1:ny-1],[2:nx nx])+I([2:ny ny],[1 1:nx-1]);
    I_xy = (Dp-Dm)/4;
    Num = I_xx.*(ep2+I_y.^2)-2*I_x.*I_y.*I_xy+I_yy.*(ep2+I_x.^2);
    Den = (ep2+I_x.^2+I_y.^2).^(3/2);
    I_t = Num./Den + lam.*(I0-I+C);
    I=I+dt*I_t;
end
J=I;
figure(1); imshow(uint8(J));
```