参赛队号#1800

第七届“认证杯”数学中国

数学建模网络挑战赛

承　诺　书

我们仔细阅读了第七届“认证杯”数学中国数学建模网络挑战赛的竞赛规则。

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上咨询等）与队外的任何人（包括指导教师）研究、讨论与赛题有关的问题。

我们知道，抄袭别人的成果是违反竞赛规则的，如果引用别人的成果或其他公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

我们郑重承诺，严格遵守竞赛规则，以保证竞赛的公正、公平性。如有违反竞赛规则的行为，我们接受相应处理结果。

我们允许数学中国网站(www.madio.net)公布论文，以供网友之间学习交流，数学中国网站以非商业目的的论文交流不需要提前取得我们的同意。

我们的参赛队号为：1800

参赛队员　(签名) ：章晨、何英杰、郑鹏

队员 1：章晨

队员 2：何英杰

队员 3：郑鹏

参赛队教练员 (签名)：

参赛队伍组别：本科组

第七届“认证杯”数学中国

数学建模网络挑战赛
编 号 专 用 页

参赛队伍的参赛队号：（请各个参赛队提前填写好）：

1800

竞赛统一编号（由竞赛组委会送至评委团前编号）：

竞赛评阅编号（由竞赛评委团评阅前进行编号）：

2014 年第七届“认证杯”数学中国
数学建模网络挑战赛第一阶段论文

题　　目　位图图像放大算法

关 键 词　双边滤波　canny 算子 边缘提取　偏微分方程

摘　　要：

对低分辨率位图图像放大的研究具有重要的意义。传统的插值放大技术有着效率高、简单快捷的优点，但是也存在着放大后图像质量差的致命缺点，其原因在于原图像存在噪点干扰信号、图像边缘无法保持和未知像素的插值方法不当。针对以上存在的问题，我们在查阅相关文献的基础上，提出了基于微分方程的放大算法。

- **模型一：图像去噪预处理模型。**要将图像进行完美放大，我们得首先保证图像信号不受噪点的影响。我们在查阅文献的基础上，详细分析了传统插值导致失真的原因，并建立优化的双边滤波自适应去噪方法。这种方法最大的优点在于考虑了边缘的灰度突变情况，先处理边缘区域的去噪情况，然后用处理后的值代替原始值，最后在处理后的平滑区域用高斯滤波平滑非边缘区域。此外，本模型在值域滤波核函数的期望值和方差上做了自适应的处理，大大提高了精度。
- **模型二：边缘提取模型**。在对图像进行去噪之后，为了避免放大之后的失真情况，我们需要将图像的边缘界限比较完整准确的提取出来。通过查阅相关文献，我们在 Canny 算子的基础上，建立了边缘提取模型。Canny 算法相对于其他算法而言优越性在于他提出的三大提取准则以及提取过程中的两大阀值。最后，我们还通过对比不同算法提取的边界效果图，从实际上验证了本模型的的优越性。
- **模型三：基于偏微分的图像放大模型。**在对图像进行边缘提取后，我们可以严格的区分了边缘区域和平滑区域。然后我们根据物理学中的热传导效应，建立基于偏微分方程的放大算法。在边缘区域，运用放大算法可以保持边缘的矢量化放大，不仅避免了加入边缘的突变形成锯齿化，也能够保持放大后的边缘修复。在平滑区域，该放大算法很好地将未知像素用已知像素表示出来。从结果比较来看，模型结果区别于传统插值的生硬联系，使得成像在保持边缘上的表现最好，不会出现任何模糊现象，而且和放大倍数无关。

对于算法放大的图像，我们建立从主观和客观上评价图像质量的模型，分别展示了本文算法的优越性和可操控性。在论文最后，我们对模型的优缺点进行了分析，并针对去噪算法的角点检测准确性提出了改进的方向。

参赛队号： 1800

所选题目： B　　**题**

参赛密码 _____

（由组委会填写）

英文摘要

The studying of enlarging low-resolution bitmap image has significant importance. Traditional interpolation amplifier technology boasts the advantages of high efficiency, simplicity and convenience, but there are also fatal flaw that amplified image are of poor quality. This is because of the existence of the interference noise signal in the original image, image edges that cannot be maintained and improper interpolation method of unknown pixels. To solve the above problems, we have, on the basis of relevant researches, proposed enlargement algorithm based on differential equations.

- **Model I: image de-noising preprocessing model.** To enlarge the image perfectly, we have to first ensure that the image signals are not affected by noise. So we apply the bilateral filtering adaptive de-noising method to the image. The biggest advantage of this method lies in taking into consideration of the mutation of the gray edge. Firstly, deal with the de-noising edge region, and then replace the original value with the value of the modified, and then use Gaussian filtering to smoothen the non-edge region of the processed smooth area. In addition, the model greatly improve the accuracy, by doing adaptive processing on the expected value and variance range of filter kernels.

- **Model II: edge extraction model.** After the image de-noising, in order to avoid distortion after amplification, we need a more complete method of accurately extracting image edge boundaries. Compared to other extracting algorithms, Canny algorithm has the advantages of extracting the three criteria mentioned and the extraction process of the two thresholds. We compare different algorithms to extract the boundary renderings, thus verifying the superiority of Canny operator practically.

- **Model III: partial derivatives based image enlarging model.** Finally, in the processed image, we can distinguish strictly smooth region and the edge region. We apply the use of an enlarging algorithm in the smoothing region, not only to avoid the formation of a mutant added jagged edge, but also to maintain the repair of the enlarged edge. For enlarging algorithm for smooth region, we choose to use the zoom algorithm based on partial differential equations, which uses the physical effects of thermal conductivity and well present unknown pixel with known pixel. The above method is different from the traditional stiff contact, making results more convincing and getting better image quality.

For the finally enlarged image, we establish the evaluation model of image quality, from the subjective aspects and objective aspects respectively. And we also show the superiority and practicality of the proposed algorithm.

Keywords: ***bilateral filtering, canny operator, edge extraction, partial differential equations***

目录

1. 问题的概述

1.1. 问题背景

图像是人们通过成像手段所获得的被人们视觉系统察觉的各种数据的统称。据统计，人类约有 75%的信息是通过视觉系统获取的。人们为了记录和表达现实生活中看到的、触摸到的具体事物和其他信息，经常采用图像作为主要的表达方式[1]。随着计算机和网络技术的快速发展，图像大部分都是离散化后以数字形式存储在计算机中，我们称为数字图像。

目前，视觉信息在社会中的作用越来越巨大。对图像放大算法的研究也变得更加有意义。数字图像放大作为数字图像处理领域中的一个重要课题，有着重要的实用价值，例如对卫星图片放大，分析气候变化、寻找矿藏；对 CT 图像放大，方便医生诊断；对侦察照片放大，便于判读，发现有价值的目标；对数码相机拍摄的照片放大，突出人物或部分场景等等。

1.2. 问题提出

虽然图像放大的研究意义重大，但是由于在实际中受制于各种条件，我们很难获得高分辨率的优质图像。例如航天应用中的遥感探测图像，超声图像和红外图像等等，无法达到理想的分辨率，这时就需要人工提高其分辨率，使得图像在更大的范围内比原来有更加明显的特征。而另一个影响图像质量的重要因素是通信传输方面的问题，现如今通过互联网等来源可以下载到数百万计的数字图像，但是由于受到通信容量的制约，这些图像的分辨率无法保持较高的水准，人们在接收到图像后，会发现其显示效果相对于高分辨率显示设备而言是很不匹配的。低分辨率图像很多时候不能满足人们研究和应用的需求，这在目前是一个经常遇到并且日益突出的问题。为了适用于特殊场合和满足特殊需求并获得较好的视觉效果，需要一种有效的方法对图像进行放大，使放大后的图像达到理想的分辨率[2]。

图像放大算法的研究已经经历了几十年的发展。但是随着需求的不断提高和细化，目前仍然没有在各个方面都令人满意的算法。已有的经典线性插值算法始终无法解决走样现象，只能满足基本的需求，且走样现象会随着放大倍数的增加越发明显。最近几年提出的各种自适应算法因为普适性和稳定性都不高，再加上算法比较复杂，没有成为被广泛认可和应用的算法。因此对位图图像放大算法的研究是很有意义的。

1.3. 相关研究

随着图像处理技术的发展，出现了很多图像放大的方法，对于基于单幅图像的放大方法，从处理方式上可以分成两大类：传统的插值方法和自适应方法。

线性插值方法具有计算简单的特点，且易于利用软、硬件实现，获得了广泛的商业应用。但是由于线性方法不能产生新的高频成分，导致放大后的图像其边缘细节区域会产生锯齿现象或者平滑现象，又由于人眼对图像的边缘部分特别敏感，插值后的图像其边缘部分对一幅图像的质量有非常重要的影响，所以经过线性插值的图像没有良好的视觉效果。

传统插值方法实质上都是低通滤波器，它们对整幅图像按相同的方法处理，这类方法会使图像产生不同程度的细节退化，在增强图像平滑效果的同时丢失了高频信息，而

有时这些细节信息恰恰是非常重要的，因此如何在保证平滑效果的同时尽可能地保留细节信息是很有研究价值的问题。自适应图像放大方法便是基于这一目的而发展起来的。自适应插值法的基本思想是先对原始图像做结构上的特性分析，然后针对不同的区域采取不同的方法进行插值，其中包括自适应插值方法以及结合小波、分形、偏微分方程、神经网络等新理论的图像放大方法。

2. 问题的分析

2.1. 总体分析

数字图像放大本身是一个欠约束的问题。如果我们想要得到把原来的图像的高和宽各放大一倍，那么像素数会增加三倍。对于放大后的图像，我们需要通过四分之一的已知像素确定另外四分之三的未知像素的值。这是一个欠约束问题。要解决这样的问题，我们需要增加一些假设条件。在合理的假设条件下，我们将问题分解为以下几个步骤：

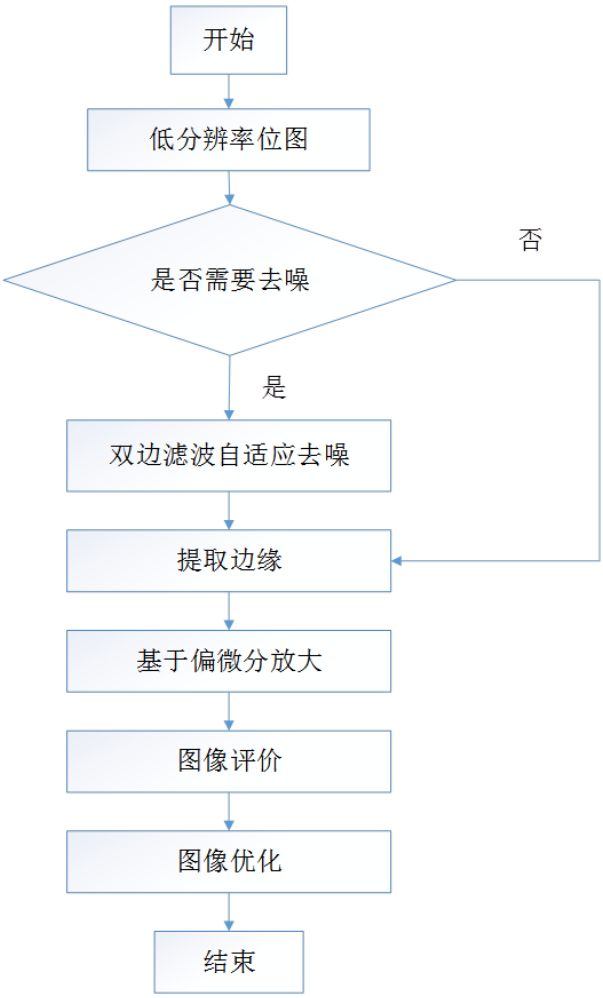


图 1 问题求解思路分析

第一步，分析了位图放大模糊或者马赛克现象出现的原因；并对低分辨率位图进行预处理，通过建立双边滤波模型对图像进行去噪处理。

第三步，采用 Canny 边缘算子对去噪后的图像进行边缘检测。

第四步，建立基于偏微分方程的图像放大模型。
第五步，对放大算法进行评估。
第六步，模型的优缺点分析和改进。

2.2. 具体分析

2.2.1. 对图像去噪的分析

在自适应算法比较复杂的处理步骤中，图像去噪预处理和基础的插值放大处理往往是必不可少的关键步骤。因为原始图像的质量参差不齐，图像中的噪声会对边缘识别产生干扰。这对基本的插值放大算法不会造成影响，但是基于边缘信息的图像放大算法需要识别边缘，噪声的干扰就无法避免。所以去噪在自适应放大算法中是不可缺少的步骤。基本的插值算法在大部分自适应放大算法中也是关键的步骤。这一步骤一般作为前期处理，把图像放大到目标尺寸。然后采用自适应算法对插值放大后的结果进行再处理。所以插值放大效果的好坏，直接影响到最终自适应结果的质量。我们在查阅文献的基础上，建立一种自适应的双边滤波算法。该算法同时考虑噪声去除和边缘保持两个方面，通过识别图像的边缘区域以及边缘梯度的大小自动调整滤波核函数的参数，实现滤波去噪效果。

2.2.2. 对边缘检测的分析

在图像上，所谓的边缘点就是在边缘点两边，图像的灰度值有明显的跳变即图像灰度值在边缘点附近并不是连续的。图像的边缘常常包含着丰富的信息，如何找到图像的边缘是一个重要的问题。通过查阅相关文献，我们发现图像的边可以由一些参数表示：方向、对比度、宽度等，不同的边缘有着不同的参数。一个物体能够被识别的区分于其他物体的主要特征就是边缘轮廓。更进一步，我们将边缘用以下特征定义：

（1）灰度突变；
（2）是不同区域的边界；
（3）具有方向性；

跟据边缘的这三个特征，可以判断所关心的区域其特征是否存在差异来判断是否存在边缘的可能性。如果特征没有差异，则认为是平滑区；如果特征有差异，则判断为边缘点。

2.2.3. 对偏微分放大的分析

在对图形的去噪预处理目的在于保证边缘提取的准确性，防止边缘的提取受噪声点影响而产生较大的错误。当去噪过程完成之后，我们选取了性能优越的 canny 算子[3]提取边缘，将图像进行分块。

进行了边缘提取之后，在查阅文献的基础上，我们根据物理学中的热传导效应，建立基于偏微分方程的放大算法。首先基于各向同性和各向异性扩散放大进行研究。这两种方法主要是利用扩散原理，将图像的灰度值视为平面物体的温度，根据图像放大的要求将放大图像的灰度值看作是一些定点（原图像的像素值）作为边界条件。然后比较它们在保持放大图像边缘方面的作用和贡献，使得我们可以用广义上连续的二维函数对图像进行建模，从而可以对图像进行求导求积分等操作，这就使问题的描述在形式上变得简单，使得公式不再依赖于网格，即各向同性的。在这个算法中，我们在已经提取出来的边缘包含的区域内分别放大，然后保持边缘的矢量化放大，很好的消除马赛克和锯齿化现象。

3. 模型的假设

1. 图像的颜色相近区域的像素是平滑的；
2. 图像中边界区域的灰度是突变或不连续的；
3. 图像边缘的特征点都是紧密排列并且有序的；
4. 原图像受噪声的干扰较小，且未被污染；
5. 图像上所有的像素点都是可获取的。

4. 名词解释与符号说明

4.1. 名词解释

(1) 阈值：在自动控制系统中能产生一个校正动作的最小输入值，刺激引起应激组织反应的最低值 。

(2) 边缘：是图像中邻近点色彩发生剧烈变化的像素点的集合，常常包含着丰富的信息不仅能够传递图像大部分信息，而且能勾勒出物体的基本轮廓。

(3) Canny 算子：先对处理的图像选择一定的高斯滤波器进行平滑滤波，然后采用一种称之为“非极值抑制”的技术，对平滑后的图像处理得到所需要的边缘图像。

4.2. 符号说明

序号	符号	符号说明
1	$\mathbf{I}_x$	像素在 x 方向上的梯度值
2	$\mathbf{I}_y$	像素在 y 方向上的梯度值
3	$\mathbf{J}$	像素梯度的张量积
4	An	像素点的局部结构各向异性系数
5	G_x	水平梯度
6	G_y	垂直梯度
7	θ	方位角
8	SNR	信噪比
9	$Location$	定位性能
10	$x_{sc}(f)$	零交叉点平均距离
11	MSE	均方误差
12	$PSNR$	峰-峰信噪比
13	$PENSR$	边缘峰值信噪比

5. 模型的建立与求解

5.1. 模型一：位图的预处理

5.1.1. 位图放大模糊的原因

传统的线性插值方法计算简单，易于软硬件实现，因而获得了广泛的商业应用。但是由于线性插值方法不能产生新的高频成分，导致放大后的图像在边缘细节区域会产生阶梯失真或者模糊，而人眼对图像的边缘部分特别敏感，因而通过线性插值算法得到的图像不能带给观察者良好的视觉效果。

接下来通过对图像边缘特征和线性插值过程分析，找到传统插值放大方法造成图像模糊的原因，为保持边缘的插值算法奠定了基础。

(1) 图像边缘特征分析

对于灰度变化平缓的图像区域，采用传统经典线性移不变插值技术，通常已能达到较好的视觉效果[4]。但是，由于它们不能很好地处理图像中剧烈跳变的局部特征，如：边缘、纹理等细节，导致放大图像的边缘细节模糊。图像的边缘常常包含着丰富的信息，如何找到图像的边缘是一个重要的问题。首先我们来简单的分析图像的边缘。图像的边可以由一些参数表示：方向、对比度、宽度等，不同的边缘有着不同的参数。图 2 是含有边的图像的灰度值的三维表示图，从图 2 可以得到以下两个结论：

（1）从边缘区域穿到非边缘区域，灰度值会出现显著的落差。

（2）在边缘区域内，像素点灰度值变化缓慢。

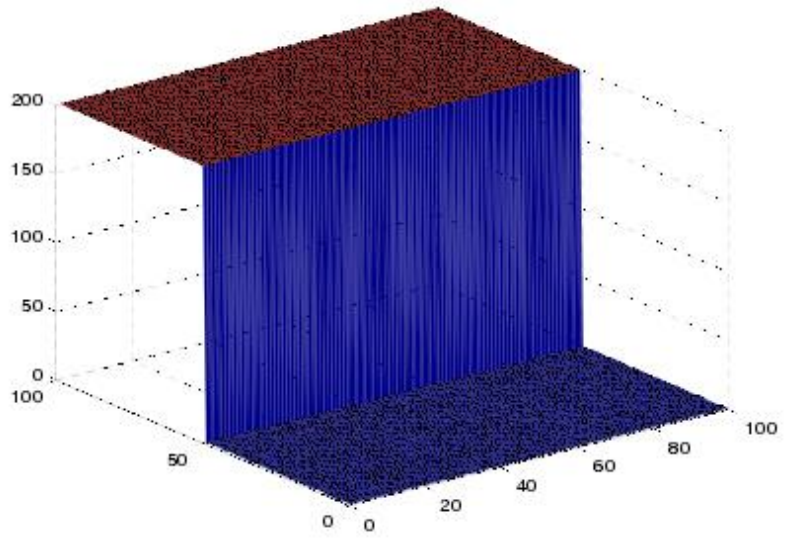


图 2 包含一条边的图像像素值的 3D 视觉图

(2) 图像插值模糊原因分析

在图 3 中列出了在图像边缘附近进行插值的过程，以一维的方式对图像插值模糊原因进行直观说明。

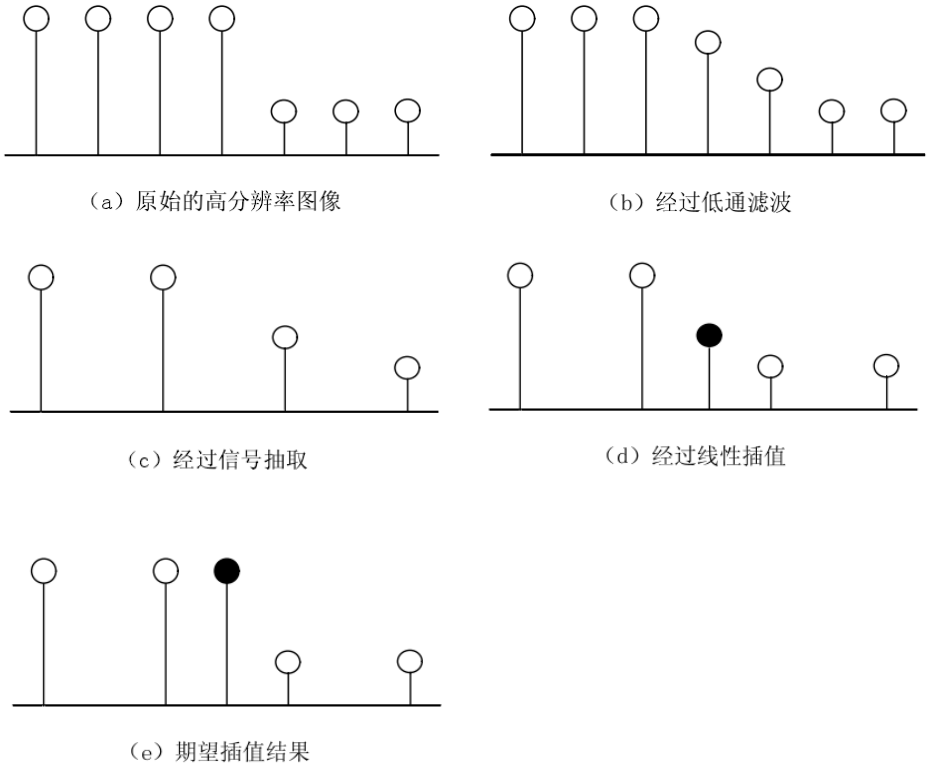


图 3 图像边缘附近插值过程

对图 3 中的各个子图的解释如下：

（a）：原始的高分辨率图像，图像左边的四个点灰度值相同，右边的三个点的灰度值相同，并且左边的四个像素的灰度值要大于右边的三个像素的灰度值。

（b）：对原图像进行了低通滤波，也就是让低频信号通过，减少高频信号与截止信号的通过。可以看到位于边缘处的灰度值发生了改变，大小介于周围两个灰度值之间。对图像进行低通滤波后的感官效果就是边缘比较平滑。

（c）：对图像进行抽样，信号抽取是一个通过减小采样频率来去掉多余数据的过程。由于抽样不满足香农定理时，抽样信号会发生混叠。所以在抽样前，必须对被采样信号做低通滤波来压缩频带，然后再抽取，以避免出现混叠现象。这也是必须经过图（b）处理的原因。

（d）：对图像进行线性插值，其中的黑色像素点是生成的新的边缘像素点。

（e）：我们期望得到的插值后的图像，其中黑色像素点是我们期望的边缘像素点。从图像中，可以清楚的看出，线性插值后的图像边缘处的像素值介于周围两个像素值之间，与我们期望的值差距较大，分析其原因如下：

（1）图像的细节信息通常体现在图像的边缘区域上，对应着高频信号，图像的非细节信息通常体现在图像的平滑区域，对应着低频信号。

（2）根据边的几何性质，边缘区域中的像素点的灰度值不会出现太大的落差，但是在边缘与非边缘区域的交接部分存在较大不同。

（3）线性插值方法采用统一的插值核函数，没有区分图像的边缘和非边缘部分，又由于插值核函数具有低通特性，抑制了图像中的高频细节，不能很好的处理图像中细节，导致放大图像的边缘阶梯失真和模糊。

图像边缘细节包含图像的重要信息，消除插值图像边缘细节模糊是提高图像整体视觉效果的关键。图像边缘产生模糊的原因是由于经典插值算法的平滑功能，使边缘像素

和邻近像素的值大小接近。研究发现如果扩大边缘处和邻近像素的差值，边缘处会变得清晰。依据边的几何性质，如果对图像边缘部分的像素点和图像非边缘部分的像素点采用不同的插值方法，即不采用统一的插值核函数，就会取得较好的插值效果。

5.1.2. 自适应双边滤波去噪模型

在自适应算法比较复杂的处理步骤中，图像去噪预处理和基础的插值放大处理往往是必不可少的关键步骤。因为原始图像的质量参差不齐，图像中的噪声会对边缘识别产生干扰。这对基本的插值放大算法不会造成影响，但是基于边缘信息的图像放大算法需要识别边缘，噪声的干扰就无法避免。所以去噪在自适应放大算法中是不可缺少的步骤。基本的插值算法在大部分自适应放大算法中也是关键的步骤。这一步骤一般作为前期处理，把图像放大到目标尺寸。然后采用自适应算法对插值放大后的结果进行再处理。所以插值放大效果的好坏，直接影响到最终自适应结果的质量。同时为了提高算法的整体计算速度，插值算法的速度优化也是需要考虑进去的。

在查阅文献的基础上，建立一种自适应的双边滤波算法[5]。滤波过程如图 4 该算法同时考虑噪声去除和边缘保持两个方面，通过识别图像的边缘区域以及边缘梯度的大小自动调整滤波核函数的参数，实现滤波效果。

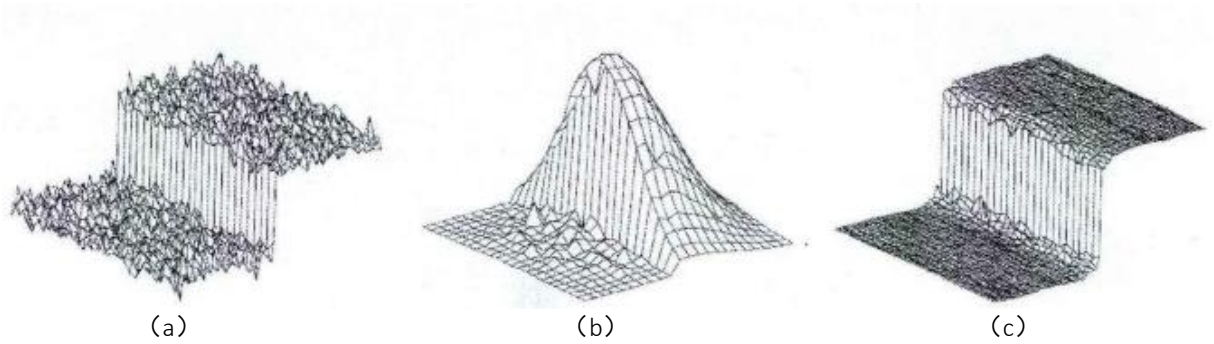


图 4 双边滤波处理过程

(1) 图像区域的划分

数字图像是二维的信号，通过颜色值的波动反应信息。如果按照颜色值波动情况来划分，图像可以分成三类区域：

- 边缘区域(颜色值波动较大且方向明确)；
- 纹理区域(颜色值变化幅度小且震荡反复出现)；
- 平滑区域(颜色值不变或者局部区域内变化可以忽略不计)。

这三类信号在视觉上的重要性依次递减，特别是边缘，往往能反映整张图像的绝大部分信息。

由于去噪模型的目的是为了更好地做边缘提取对原图做的去噪预处理工作。我们这里不需要精确的边缘，只要大概的识别边缘区域即可。需要说明的是上面曾提到的纹理区域的问题，纹理区域属于高频信号部分，在低通滤波处理中无法很好地保留。图像的纹理单元(即可以表示纹理元素的最小形状)有大有小，纹理单元大的话我们把纹理视作边缘，否则把纹理视作噪声并归入平滑区域[6]。

考虑到噪声的干扰，使用梯度信息或者亮度差来识别图像的边缘区域无法获得很好的效果，本文借助结构张量的概念识别边缘区域[7]。

设图像中某像素为$\mathbf{I}$，$\mathbf{I}_x$和$\mathbf{I}_y$分别表示该像素在 x 和 y 方向上的梯度值，则结构张量为该像素梯度的张量积：

$$\mathbf{J}=\begin{bmatrix}\mathbf{I}_x^2 & \mathbf{I}_x\mathbf{I}_y\\ \mathbf{I}_x\mathbf{I}_y & \mathbf{I}_y^2\end{bmatrix} \tag{1}$$

求解矩阵的特征值λ_1和λ_1，且$\lambda_1 \geq \lambda_2$。特征值λ_1所对应的特征向量的方向就是梯度方向，最小特征值λ_1所对应的特征向量的方向就是图像局部结构的方向。$An=(\lambda_1-\lambda_2)^{\alpha}$的结果表述了局部结构的各向异性或称相关性。

以上属性可以反映很多结构信息，如果λ_1和λ_1的差别比较大说明该像素点在边缘上，如果 An 比较小而λ_1和λ_1都比较大，则说明该像素点为角点。

一般噪声点处的表现类似于角点，该方法能比较容易地识别边缘像素与噪声像素，所以具有一定的抗噪声性。本文通过计算结构张量的特征值，得到像素点的局部结构各向异性系数 An，对该系数设定阈值划分图像区域。中间有两个值需要确定，An 中的α值以及阈值T_a。对于这两个值的设定，本文做了大量试验统计工作。选取十张经典的图像，加上不同的噪声政行测试，包括椒盐噪声、高斯噪声、泊松噪声、乘性噪声、均值噪声等。实验发现，当α取值为 1 时 An 的抗噪声干扰性最强。取值小于 1 会对噪声敏感，而取值大于 1 会降低识别边缘的能力。阈值T_a的选取考虑两个因素，边缘的识别能力和划分出的边缘区域的比例。算法争取识别出尽可能多的边缘，同时不想边缘区域过大，否则会影响后续步骤的计算效率和处理效果。经过对多种类别的多张图像的测试，该阈值设置为 0.005 效果比较好。阈值划分是对单个像素进行判断的，为了保证区域的连贯性消除孤立点，需要后续的调整操作。

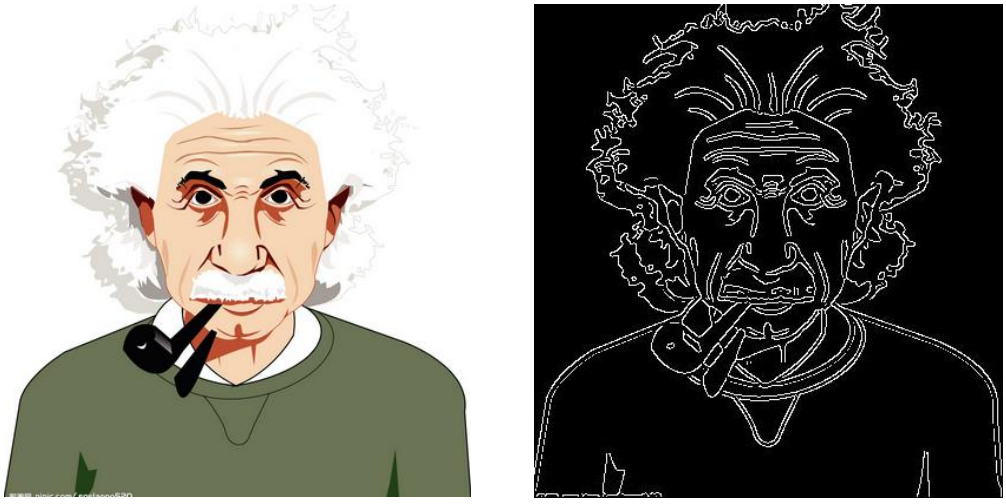


图 5 边缘区域划分示例

图 5 给出了边缘区域划分的结果示例，对原图加了高斯噪声，划分结果中白色区域为边缘区域。白色区域基本覆盖了图像的边缘信息，且抗干扰性比较好。边缘区域大概图像的四分之一，其他区域为平滑区域，这些区域信息含量不高，不是重点处理的对象，可以采用更加简单且平滑性更好的滤波算法去噪，可以同时提高去噪效果和计算速度。

(2) 自适应滤波处理

对图像划分之后，找到了去噪的重点和难点区域，这些区域包含了图像的绝大部分信息，处理时自然要特别慎重。剩余的区域比较简单，我们先进行讨论。

根据上面的讨论，单纯从去除噪声的角度考虑，双边滤波没有高斯滤波效果好。如果一个区域确定没有边缘信息，那么双边滤波的保边缘特性无法发挥效果，反而高斯滤波成为更好的选择。本文算法就采取这种策略，在已经识别的平滑区域采用高斯滤波去噪。这样做有两个方面的优点：首先计算量上，高斯滤波比双边滤波高效很多，预先计算高斯核系数得到一个蒙板即可，更或者采用分方向的两次以为滤波处理更高效;其次，高斯滤波核的参数调节变得不重要，没有避免边缘模糊的考虑，可以让滤波核的区间大一些。

在边缘区域，本文采用改进的双边滤波算法，改进的对象是值域滤波核系数。该核函数是一个高斯函数，期望值和方差两个参数[8]。

$$h(x)=k^{-1}(x)\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}f(\xi)c(\xi,x)s(f(\xi),f(x))d\xi \tag{2}$$

其中，

$$c(\xi,x)=e^{-\frac{1}{2}(\frac{\xi-x}{\delta_d})^2}$$

$$s(f(\xi),f(x))=e^{-\frac{1}{2}(\frac{f(\xi)-f(x)}{\delta_r})^2}$$

双边滤波中设定均值期望值为目标像素的原始值，方差可以根据图像的情况设定。参数的确定是模型求解的关键。

如果目标像素是噪声点，值域滤波核的均值期望值不应该取该像素值，上面也讨论了这样的情况。本文采用沿着局部结构方向(边缘方向)取相邻像素计算均值的算法，而局部结构方向可以在结构张量计算特征值时得到。考虑到噪声点处方向会受到噪声的干扰，需要借助该像素周围像素的方向抗噪声干扰。

具体算法描述：

第一步：得到该像素点和上下左右四个相邻像素的局部结构方向，对这个五个求平均得到目标局部结构方向;

第二步：以该像素点为中心沿着这个方向两边各找到两个像素点作为求均值的待选点。

第三步：逐个判断四个点的局部结构方向与目标方向是否接近(夹角小于 30 度)，如果接近则选取该点计算均值。这样判断是为了在边缘方向改变时依然能够获得准确的均值。

在值域滤波核的高斯核函数中，方差σ_r的选取同样是重点。σ_r是根据边缘像素的梯度大小自适应调整。对不同类型的图像和图像不同级别的边缘区域处理更有针对性。其取值策略如下：

1) 方差与该点的梯度正相关：

$$\sigma_r=k\sqrt{\lambda_1}$$

2) 方差与该点的梯度负相关：

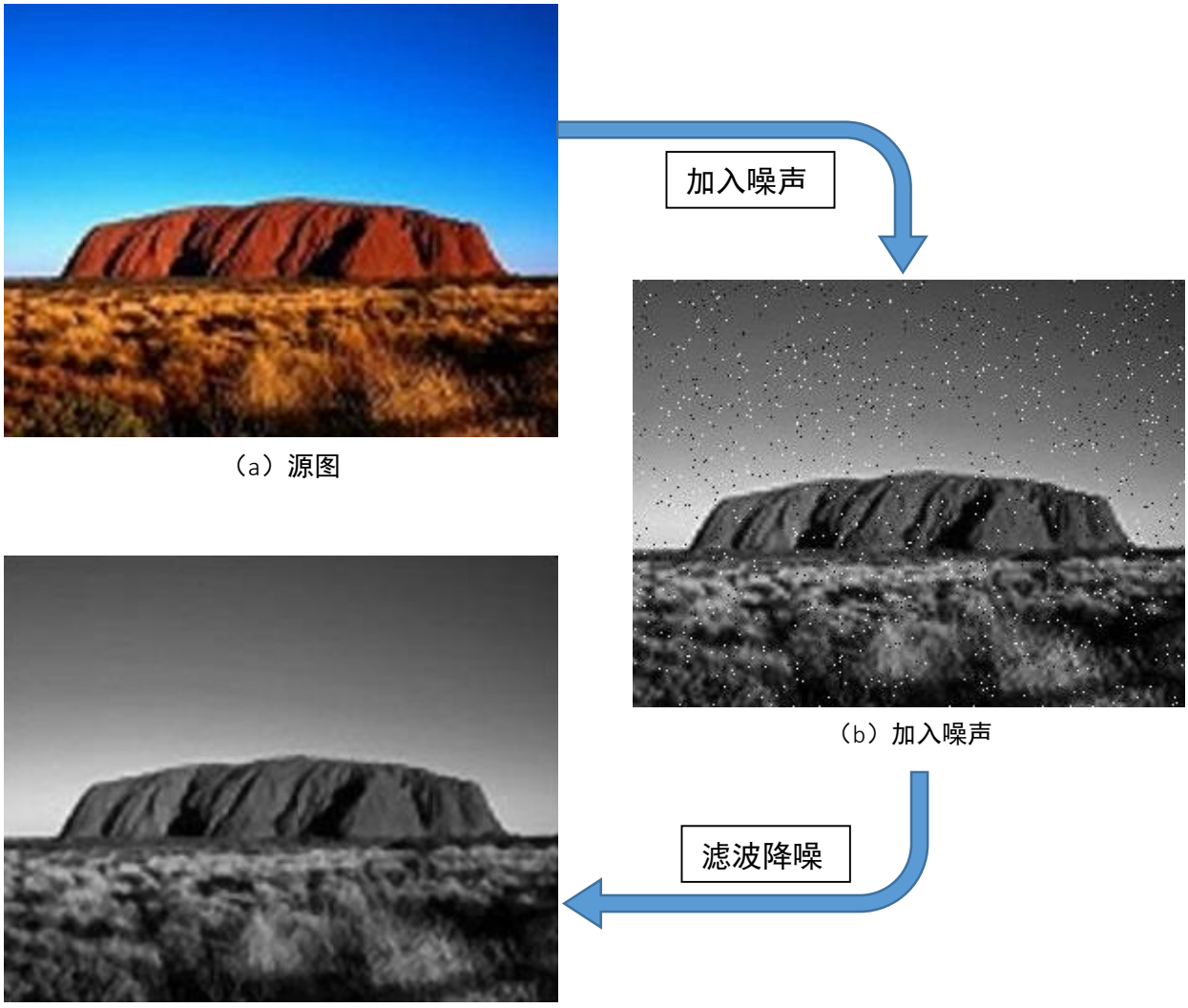
$$\sigma_r=\sigma_0-k\sqrt{\lambda_1}$$

其中，系数 k 的取值为 0.5。像素点处的梯度大小与结构张量特征值 λ_1 正相关， 即通过该特征值反应梯度的大小。

因为图像划分成两类区域且各个区域没有规则的边界，两类区域采用不同的去噪方法，可能会导致在边界处不平滑的问题。这个问题可以通过调整两类区域处理的先后顺序避免。划分好区域之后，先处理边缘区域，用处理后的值替代原始值，然后再用高斯滤波平滑非边缘区域。非边缘区域内靠近边界处的像素在平滑处理时受到边缘区域内像素的影响，使得边界处过渡更加平滑。

(3) 噪声滤除结果

去噪的视觉效果如图 6 所示， 本文提出的滤波算法和标准双边滤波算法相比，在椒盐噪声的去噪效果上更加突出。椒盐噪声是孤立的噪声点，且噪声值与周围像素值的差别比较大。本文算法在值域滤波核函数的期望值和方差上做了自适应的处理，特别是在获取均值时根据周围特征加权，相比于标准双边滤波的直接取中心像素值的方法，在抗噪声干扰特别是孤立点噪声干扰上具有很大的优势。



（a）源图

（b）加入噪声

（c）双边滤波后的图像

图 6 加噪和去噪效果示例

5.2. 模型二：基于 canny 算法的边缘提取

5.2.1. 图像边缘特性的研究

(1) 边缘点的定义：

一个物体能够被识别的区分于其他物体的主要特征就是边缘轮廓，在图像上，所谓的边缘点就是在边缘点两边，图像的灰度值有明显的跳变即图像灰度值在边缘点附近并不是连续的。

一个像素如果满足下列条件，就认为是边缘点：

1） 像素边缘强度大于沿梯度方向的两个相邻像素的边缘强度；
2） 与该像素梯度方向上相邻两点的方向差小于 45 度；
3） 以该像素为中心的 3×3 邻域中的边缘强度极大值小于某个阈值。

(2) 边缘提取的原因：

在对图像进行放大时，放大算法一般使用在平滑区域，例如传统的基于插值的放大算法中，对未知的像素按周围四像素点的灰度值进行插值，而由于边缘点周围的灰度值突变，会导致插值后的像素产生异常，形成锯齿化。所以，在进行放大图形前，进行边缘提取是很有必要的。

(3) 边缘与平滑区域的判断：

边缘以下特征提取：

1） 灰度突变；
2） 是不同区域的边界；
3） 具有方向性；

跟据边缘的这三个特征，可以判断所关心的区域其特征是否存在差异来判断是否存在边缘的可能性。如果特征没有差异，则认为是平滑区；如果特征有差异，则判断为边缘点。

5.2.2. 边缘检测方法简介以及比较

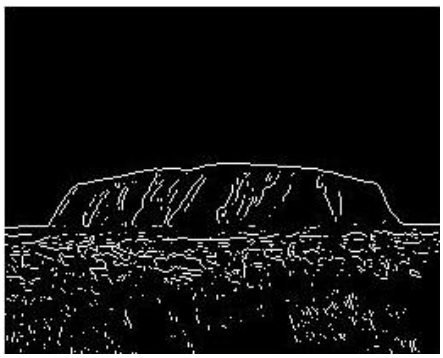
(1) 边缘提取算法介绍

边缘提取首先要检测出图像局部特性的不连续性，然后再将这些不连续的边缘像素连成完备的边界。边缘的特性是沿边缘走向的像素变化平缓，而垂直于边缘方向的像素变化剧烈；所以，从这个意义上说，提取边缘的算法就是检测出符合边缘特性的边缘像素的数学算子。常见的边缘提取算法有 Sobel 算法、Prewitt 算法、Robert 算法、Laplace 算法、LOG 算法和 Canny 算法[9,10,11]。

这些算法都是通过计算像素点的灰度变化率和变化方向来判别该点是否是边缘点，灰度变化率用数学上梯度这一概念来描述，灰度变化方向用方向角描述，将数学概念用在数字图像处理中，就要将数学概念离散化，这些算法都是使用一个卷积核与图像中每个像素的邻域进行卷积，从而得到中心点灰度，达到增强边缘点的目的，再结合阈值选定，就可以区分边缘点和非边缘点。

(2) 结果比较

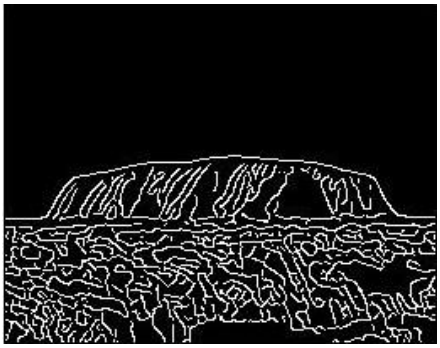
利用 matlab 边缘提取工具箱对几种算子得出的结果进行比较，结果如下图所示：



prewitt 算子

robet 算子

sobel 算子



canny 算子

图 7 几种算子结果比较

如图 7 所示，四种算子提取出来的边缘检测结果效果不同，我们可以明显的看出，canny 算子结果较其他结果有效。这是因为 canny 算子它使用两个阈值来检测强边界和弱边界，只有相连时才进行显示。所以，该方法与其他方法相比。受噪声影响的机会更少，并且更有可能检测到尖的弱边界。下面，我们将着重的介绍一下基于 canny 算子的边缘检测算法。

5.2.3. Canny 算法准则：

Canny 算法是一种比较新的边缘检测算子，具有很好的边缘检测性能，得到了越来越广泛的应用，这主要是因为它提出了迄今为止定义最为严格的边缘检测的三个标准，另外就是其相对简单的算法使得整个过程可以在较短的时间内实现。

（1）低出错率：好的信噪比既不丢失重要的边缘，也不应有虚假的边缘。好的信噪比即将非边缘点判为边缘点的概率要低，信噪比的数学表达式为：

$$SNR = \frac{\left| \int_{-\infty}^{\infty} E(-x) f(x) d_x \right|}{n_0 \sqrt{\int_{-\infty}^{\infty} f^2(x) d_x}} \tag{3}$$

其中，$f(x)$是边界为$[-\omega, +\omega]$的滤波器脉冲响应，$E(x)$代表边缘，n_0是高斯噪声的均方根。若信噪比大，则边缘提取质量好。

（2）定位准确：实际边缘与检测到的边缘位置之间的偏差最小。好的定位性能是指检测出的边缘点要尽可能在实际边缘的中心。定位性能的数学表达式为：

$$Location=\frac{\left|\int_{-\infty}^{\infty}E(-x)f'(x)d_x\right|}{n_0\sqrt{\int_{-\infty}^{\infty}f'^2(x)d_x}} \tag{4}$$

如果满足此准则，那么边缘定位精度就高。求取梯度分量的乘积的最大值，是设计最佳算子的基础。除此之外，还应满足多重响应约束条件。

（3）单边缘响应：即单个边缘产生的多个响应的概率要低，并且虚假的边缘响应应得到最大抑制。这就要求在 f 对噪声的响应中，两个相邻最大值间的距离为 $x_{max}(f)$，f' 的零交叉点平均距离为 $x_{sc}(f)$。两者间的关系为：

$$x_{max}(f)=2x_{sc}(f) \tag{5}$$

其中，

$$x_{sc}(f)=\pi\left[\frac{\int_{-\infty}^{\infty}E(-x)f'(x)d_x}{n_0\sqrt{\int_{-\infty}^{\infty}f'^2(x)d_x}}\right]^2$$

若满足此准则，就能保证单边缘只有一个响应。

5.2.4. Canny 边缘检测方法

Canny 算子的基本思想是：先对处理的图像选择一定的高斯滤波器进行平滑滤波，然后采用一种称之为“非极值抑制”的技术，对平滑后的图像处理得到所需要的边缘图像。

图 8 边缘检测算法流程图

Canny 算法过程

Canny 算法共分为如下 6 个步骤。

Step 1：在定位和检测前先滤除噪声

用高斯滤波器来对图像滤波，以去除图像中的噪声。选用高斯滤波器，计算出一个合适的掩模，用标准卷积实现高斯平滑。通常情况下，很多应用中采用图 9 所示的高斯掩模。

1/555

2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

图 9 方差为 1.4 的高斯函数离散近似

Step 2：利用梯度搜寻边缘

对滤波后的图像中的每个像素，计算其梯度的大小。使用 Sobel 梯度算子计算每一像素点的梯度估计值。Sobel 算子有两个的卷积核，如下列公式所示，一个计算水平方向的梯度分量，另一个计算垂直方向的梯度分量。

$$G_x=\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix},\quad G_y=\begin{bmatrix}1 & 2 & 1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{bmatrix}$$

Step 3：计算方向角

若水平方向梯度分量和垂直方向梯度分量已知，用下式计算方向角：

$$\theta = ar\tan(G_y / G_x) \tag{6}$$

Step 4：方向角规范化

对于图像中每个像素，它只有 4 个可能的方向与邻点相连：0 度（水平方向）、45 度（正对角线）、90 度（垂直方向）和 135 度（负对角线）。这样方向角被规范到以下 4 个角度：

表格 1 方向角度规范化

规范角度	角度范围
0°	$0^\circ \sim 22.5^\circ, 157.5^\circ \sim 180^\circ$
45°	$22.5^\circ \sim 67.5^\circ$
90°	$67.5^\circ \sim 112.5^\circ$
135°	$112.5^\circ \sim 157.5^\circ$

Step 5：非最大化抑制

幅值图像阵列中某一点的值越大，其对应的图像梯度也越大，但这还不足以确定边

缘。为了确定边缘，必须细化图像中的屋脊带，这样才会生成细化的边缘。遍历边缘点，若该点在方向角方向上是梯度值最大，则保留，否则将该点从边缘点集合中去除。运算结果将得到一幅细线图。

Step 6：双阈值分割

对非最大化抑制幅值进行阈值化后的结果是一个图像的边缘阵列。阈值化后得到的边缘阵列仍有假边缘存在，如果用简单的阈值来处理，选择合适的阈值是很困难的，常需要经过反复试验。有效的方法是选择两个阈值，一个高阈值 TH 一个低阈值 TL，TH 通常与 TL 的比值为 3：2。先从边缘点集合中去除梯度值小于高阈值的像素点，得边缘点集合 F，再处理梯度介于高低阈值之间的像素点集合 M。若 M 中一点在 F 中有邻点，则将该点加入 F。最终得到的集合 F 就是边缘点集合。

5.3. 模型三：基于偏微分方程的图像放大模型

在上面我们已经做到图形的简单预处理: 去噪，目的在于保证边缘提取的准确性，防止边缘的提取受噪声点影响而产生较大的错误。当去噪过程完成之后，我们选取了性能优越的 canny 算子提取边缘，大致的将图像分块。边缘提取的必要性在于对平滑地区运用放大算法时不会因为边缘区域的灰度值突变而产生锯齿化或者马赛克现象。进行了边缘提取之后，我们根据物理学中的热传导效应，建立基于偏微分方程的放大算法。在这个算法中，我们在已经提取出来的边缘包括的区域内分别放大，然后保持边缘的矢量化放大，最终，我们可以发现这种放大算法保持了边缘，减少产生了边缘的锯齿化现象。

5.3.1. 模型简介

传统的插值方法在对图像进行放大时，不可避免地在图像中产生了比较明显的人工痕迹，主要表现为图像的边缘锯齿化和边缘模糊化，严重影响图像的视觉质量。为了提高图像放大的质量，我们必须对插值放大后的图像进行后处理。

偏微分方程作为一种后处理的工具，对经典的插值方法得到的图像进行处理，在提高图像放大质量方面起到了重要的作用。

本节就两种新的图像放大模型：基于各向同性和各向异性扩散[11]放大进行研究。这两种方法主要是利用扩散原理，将图像的灰度值视为平面物体的温度，根据图像放大的要求将放大图像的灰度值看作是一些定点（原图像的像素值）作为边界条件。这里考虑的是扩散平衡后的稳定状态，所以与时间 t 无关。本文就这两种扩散方法及保持边缘的放大算法进行研究，比较它们在保持放大图像边缘方面的作用和贡献。使用偏微分方程进行图像处理有很多优点，它使得我们可以用广义上连续的二维函数对图像进行建模，从而可以对图像进行求导求积分等操作，这就使问题的描述在形式上变得简单，使得公式不再依赖于网格，即各向同性的。使用偏微分方程的突出优点就是可以使图像处理和分析的速度、准确性和稳定性都有很大的提高。

5.3.2. 基于各向同性扩散的图像放大模型

灰度图像数据是二维平面上的灰度值，设原始图像离散后为 v ，一般可用二元函数表示： $v(i,j),(i=1,2,...,m;j=1,2,...,n)$ ，其中 m，n 分别表示图像的行数和列数。设 u 为 v 的原始重构图像。对于图像来说，灰度的连续性是一个基本的性质，图像中相近的像素点往往取相近的值，这些数据整体的分布具有一定的规律，即是高值倾向于分布在其他高值附近，低值倾向与分布在其他低值附近，而且灰度值的改变或是渐变

的，或是突变的，一般并不完全呈线性关系。基于偏微分方程所表示量的物理意义，可以把图像数据 $v(x,y)$ 看作是平面物体的温度分布，当图像放大后，放大后的图像数据同样可以看作是平面物体的温度分布。例如，当图像横向放大 k_1 倍，纵向放大 k_2 倍后，记放大图像所占有的平面区域为：$\Omega=\{(x,y)|1\le x\le k_1(m-1)+1,1\le y\le k_2(n-1)+1\}$。根据图像放大的要求，假设放大后图像在邻域 $O(k_1(i-1)+1,k_2(j-1)+1),(i=1,2,...,m,j=1,2,...,n)$ 上的像素值与原图像在 (i,j) 点的像素值保持一致。其中，$\mathrm{diam}(o(k_1(i-1)+1,k_2(j-1)+1))<1(i=1,2,...,m,j=1,2,...,n)$，即 $u|_{o(k_1(i-1)+1,k_2(j-1)+1)}=v(i,j)(i=1,2,...,m,j=1,2,...,n)$，其他点上的灰度值可以利用这些区域上的像素值，通过扩散方程来确定。

一般而言，我们考虑的是扩散平衡后的稳定状态，所以我们考虑的方程与时间 t 无关。由平面物体扩散方程的数学模型知，其温度满足：

$$\frac{\partial^2 u(x,y)}{\partial x^2}+\frac{\partial^2 u(x,y)}{\partial y^2}=\Delta u(x,y)=0,(x,y)\in\Omega \tag{7}$$

我们假定物体在边界上满足绝热条件即：

$$\frac{\partial u(x,y)}{\partial n}=0,(x,y)\in\partial\Omega \tag{8}$$

又记 $S(i,j)=O(k_1(i-1)+1,k_2(j-1)+1)$。令 $S=\bigcup S(i,j),i=1,2,...,m;j=1,2,...,n$，根据图像放大的要求，我们设在 S 上保持恒温不变（就是原始像素值），即

$$u|_{S(i,j)}=v(i,j)(i=1,2,...,m,j=1,2,...,n) \tag{9}$$

对于放大图像在 $\Omega\backslash S$ 区域上的取值，若取初值为 0，即：$u(x,y)=0,(x,y)\in\Omega\backslash S$，经过实验后发现，由于扩散均衡过度，图像边缘保持的效果还是不十分理想。我们知道双三次插值具有一定的边缘保持作用，因此可以利用扩散模型对经双三次插值放大后的图像进行处理，就能获得分片光滑的对比度较强的图像。假设原图像 v 经过双三次插值放大后的图像为 g ，那么可以构造以下方程：

$$u(x,y)=g(x,y),(x,y)\in\Omega\backslash S \tag{10}$$

由（7）式到（10）式可以看作是对原始图像进行放大 $k_1\times k_2$ 倍的重构。

根据扩散原理，S 区域上的像素值是图像放大的动力，由 S 区域上的值作为扩散源，反复利用 S 上确定的像素值经过多次迭代，使其邻近区域的像素值与其接近，从而实现了图像的放大。

我们注意到放大图像在 S 区域上的像素与原图像保持一致，即满足 Drichlet 边界条件，定解问题（7）式到（10）式的解存在且唯一。由于此模型很大程度上依赖于 S 区域上的温度，所以必须充分利用 S 上的温度值，进行多次迭代，因此我们只应用简单的中心差分格式来进行求解，这样更有利于模型的实现。我们把 $u(x_i,y_j)$ 记作放大后图像在像素点 (x_i,y_j) 的灰度值，其中 $x_i=ih,y_j=jh,i=1,2,...,k_1(m-1)+1;j=1,2,...,k_2(n-1)+1$，其中 h 为空间步长，通常取为 1，我们有以下差分格式：

$$\begin{cases}(u_x)_{i,j}=\dfrac{u_{i+1,j}-u_{i-1,j}}{2}\\(u_y)_{i,j}=\dfrac{u_{i,j+1}-u_{i,j-1}}{2}\\(u_{xx})_{i,j}=u_{i+1,j}+u_{i-1,j}-2u_{i,j}\\(u_{yy})_{i,j}=u_{i,j+1}+u_{i,j-1}-2u_{i,j}\end{cases} \tag{11}$$

所以，$\Delta u_{(i,j)} = (u_{xx})_{i,j} + (u_{yy})_{i,j} = u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}$，利用 Gauss-Seild 迭代格式，可以得到以下迭代格式：

$$u_{i,j} = \frac{1}{h_1^2}(u_{i+1,j} + u_{i-1,j}) + \frac{1}{h_1^2}(u_{i,j+1} + u_{i,j-1}) \tag{12}$$

由 Neumann 边界条件（7），初始条件式（9）以及固定区域的灰度值条件，我们可以设计以下的计算方法：

$$\begin{cases} u(x,y) = g(x,y), (x,y) \neq (k_1(i-1)+1, k_2(j-1)+1) \\ u(x,y) = v(\frac{x-1}{k_1}+1, \frac{y-1}{k_2}+1), (x,y) = (k_1(i-1)+1, k_2(j-1)+1) \end{cases} \tag{13}$$

即：

$$\begin{cases} u(x,y) = \dfrac{u_{x+1,y} + u_{x-1,y} + u_{x,y+1} + u_{x,y-1}}{4}, (x,y) \neq (k_1(i-1)+1, k_2(j-1)+1) \\ u(x,y) = v(\dfrac{x-1}{k_1}+1, \dfrac{y-1}{k_2}+1), (x,y) = (k_1(i-1)+1, k_2(j-1)+1) \end{cases} \tag{14}$$

5.3.3. 基于各向异性扩散的图像放大模型

在各向异性扩散模型中，各基本量与各向同性扩散模型的规定一样，设 u 为 g 的重构图像，其灰度值为 $u(i,j), (i = 1,2,...,m; j = 1,2,...,n)$，并且条件（7）、（8）、（9）在各向异性放大模型中仍旧成立。在这里考虑的仍是扩散平衡后的稳定状态，所以仍与时间 t 无关。我们考虑给出如下稳态的各向异性扩散模型：

$$\begin{cases} div(C|\nabla u|\bullet\nabla u) = 0, (x,y) \in \Omega \\ \dfrac{\partial u(x,y)}{\partial n} = 0, (x,y) \in \partial\Omega \\ u|_{S(i,j)} = v(i,j)(i = 1,2,...,m, j = 1,2,...,n) \\ u(x,y) = g(x,y), (x,y) \in \Omega \setminus S \end{cases} \tag{15}$$

在图像处理，比如噪声的去除中，Perona 和 Malik [13]建议选择 C(x)：

$$C(x) = \frac{1}{1+(x/k)^2} \quad 或 \quad C(x) = \exp(-(x/k)^2) \tag{16}$$

我们采用的差分格式和各向同性扩散放大模型（10）一样，用其对（13）式进行离散。整个过程中需要确定参数 k 的取值，它是一个梯度模的阈值，当梯度的模值超过 k 时，在边缘处会使图像越发陡峭，所以会使图像有锐利的边缘，k 可以选择较大的值。一般而言，k 的选择可以参照 Canny 的直方图估计法，先求出整个图像的累积直方图，k 就选取整个像素的百分之八十到百分之九十处的值。

5.3.4. 基于预估-校正的图像放大

在上一个部分，我们研究了采用各向同性扩散的偏微分方程模型进行图像放大，这是偏微分方程模型在图像放大领域的应用。但由于各向同性扩散偏微分方程模型本身的固有性质，使得放大后的图像边缘出现模糊、细节特征丢失。在第三节中，采用各向异性扩散的偏微分方程进行图像放大，使得放大后图像的细节特征得以保持，但随着迭代求解次数的增加，放大后图像部分重要的信息偏离原图像，导致图像模糊。

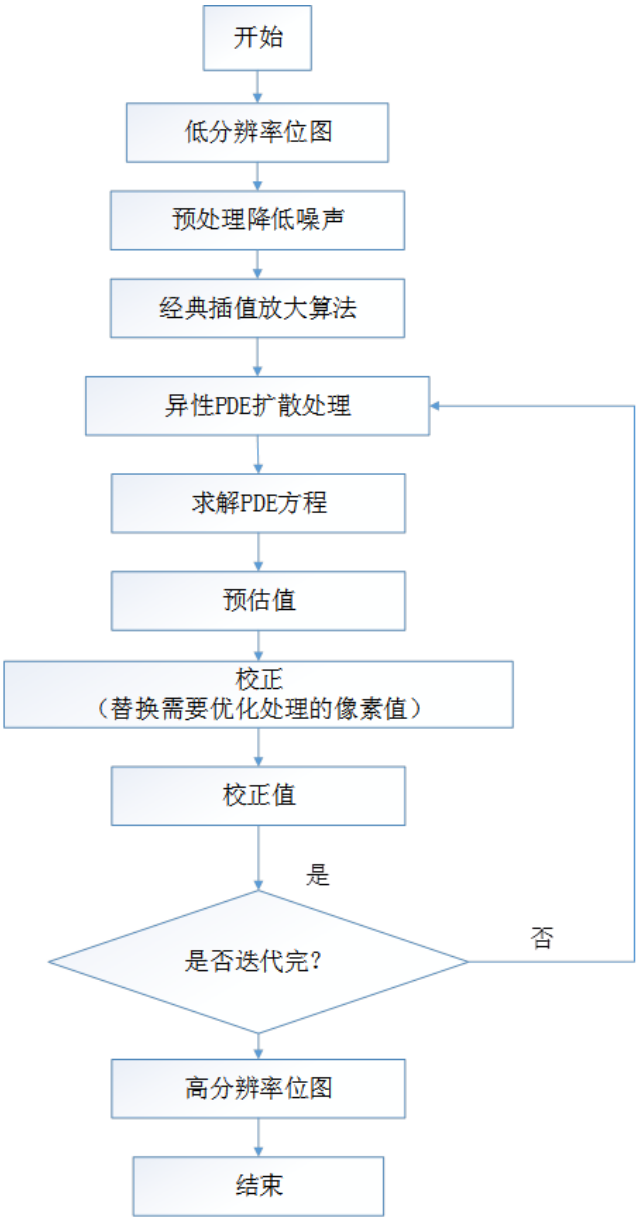


图 10 算法总流程图

针对各向同性扩散和各向异性扩散存在的问题，本文提出了一种基于偏微分方程的图像放大的新方法。在该方法中，我们引入预估 - 校正思想。空间连续性是空间属性的基本性质，空间中相邻的样本点往往取相近的值，这样才符合人的视觉效果。这些数据整体也不是随机分布的，而是高值倾向其他高值附近，低值倾向其他低值附近，但值的改变可以是渐进的，也可以是突然性的，一般并不完全呈线性关系。根据这一特性，我们把偏微分方程理论用在图像放大中。基于偏微分方程所表示量的物理意义，把图像的灰度值视为平面物体的温度，以原图像的某种插值结果作为各向异性扩散偏微分方程的初值，利用数值方法对方程进行求解。将上述所得结果作为预估值，将预估结果中相应于原图像位置的像素值以原图像替换，所得结果作为校正值。校正值将作为下一步预估计算的初值。用该方法对图像进行放大，不但能使放大后的图像保持锐利的边缘、保持图像的细节，而且算法具有长时间计算的稳定性，即对该算法进行多次迭代时，放大后的图像仍然能保留原图像的部分信息而且图像不会模糊。算法流程图如图 10 所示。

(1) 预估方法

本文利用异性扩散模型进行图像放大，即预估算法采用异性扩散方程。考虑给出如下稳态的各向异性扩散模型：

$$\begin{cases} div(C\,|\nabla u|\bullet\nabla u)=0,(x,y)\in\Omega \\ \dfrac{\partial u(x,y)}{\partial n}=0,(x,y)\in\partial\Omega \\ u\,|_{S(i,j)}=v(i,j)(i=1,2,...,m,\ j=1,2,...,n) \\ u(x,y)=g(x,y),(x,y)\in\Omega\setminus S \end{cases} \tag{17}$$

我们采用的差分格式是：

$$\begin{cases} (u_x)_{i,j}=\dfrac{u_{i+1,j}-u_{i-1,j}}{2} \\ (u_y)_{i,j}=\dfrac{u_{i,j+1}-u_{i,j-1}}{2} \\ (u_{xx})_{i,j}=u_{i+1,j}+u_{i-1,j}-2u_{i,j} \\ (u_{yy})_{i,j}=u_{i,j+1}+u_{i,j-1}-2u_{i,j} \\ (u_{xy})_{i,j}=\dfrac{u_{i+1,j+1}+u_{i-1,j+1}-u_{i+1,j-1}+u_{i-1,y-1}}{4} \end{cases} \tag{18}$$

(2) 校正方法

设原始图像离散后为 v，其灰度值为 $v(i,j),(i=1,2,...,m;\ j=1,2,...,n)$，其中 m,n 分别表示图像的行数和列数。基于偏微分方程所表示量的物理意义，我们可以把这种图像数据 $v(x,y)$ 看作是平面物体的温度分布，当图像横向放大 k_1 倍，纵向放大 k_2 倍时，放大后的图像数据可以看作是横向放大 k_1 倍，纵向放大 k_2 倍后平面物体的温度分布，即 $u(i,j),(i=1,2,...,k_1(m-1)+1;\ j=1,2,...,k_2(n-1)+1)$，记放大图像所占有的平面区域为 $\Omega=\{(x,y)|1\le x\le k_1(m-1)+1,1\le y\le k_2(n-1)+1\}$。根据图像放大的要求，我们假设放大后图像在邻域 $O(k_1(i-1)+1,k_2(j-1)+1),(i=1,2,...,m;\ j=1,2,...,n)$ 上的像素值与原图像在 (i,j) 点的像素值保持一致。其中，$\mathrm{diam}(o(k_1(i-1)+1,k_2(j-1)+1))<1(i=1,2,...,m,\ j=1,2,...,n)$，即 $u\,|_{O(k_1(i-1)+1,k_2(j-1)+1)}=v(i,j)(i=1,2,...,m,\ j=1,2,...,n)$，其他点上的灰度值即由偏微分方程扩散后的值来确定。

利用前述预估思想，记 $u^n_{x,y}$ 为预估值，则校正值可表述为：

$$u^n_{x,y}=\begin{cases} u^n_{x,y},(x,y)\ne(k_1(i-1)+1,k_2(j-1)+1) \\ v(\dfrac{x-1}{k_1}+1,\dfrac{y-1}{k_2}+1),(x,y)=(k_1(i-1)+1,k_2(j-1)+1) \end{cases} \tag{19}$$

校正值将作为下一步预估计算的初值。

5.3.5. 模型的求解结果

填充算法插值结果比较如图 12 和图 13 所示，对图像进行三倍放大的结果，其中图 12 采用常规线性插值的方法获得。通过对比可以发现，线性插值包括最近邻算法会导致走样和边缘变宽的现象。局部放大后可以观察到走样还是比较严重的，图像在边缘处也有一定程度的模糊平滑，而且随着放大倍数的增加变得越来越严重自适应插值算法在边

缘保持上表现的更好。本文的算法在保持边缘上的表现最好，不会出现任何模糊现象，而且和放大倍数无关。图 13 给出了本文算法在更大放大倍数(5 倍放大)情况下的结果，从结果中可以看出只要边缘提取的完整，填充插值的结果就会比较满意。不论是边缘清晰且无噪声的图像还是边缘复杂的照片级的图像，本算法都获得了不错的放大效果。

图 11 待放大源图

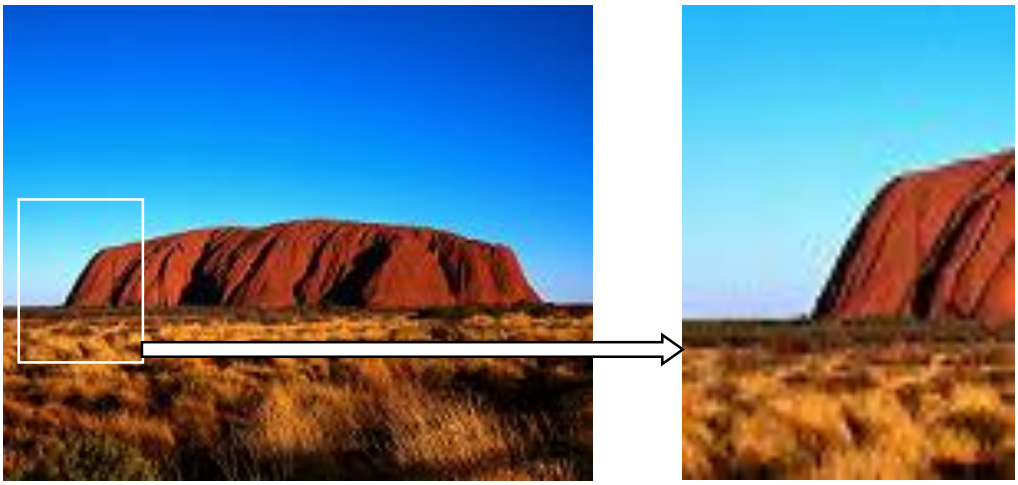


图 12 线性插值放大效果

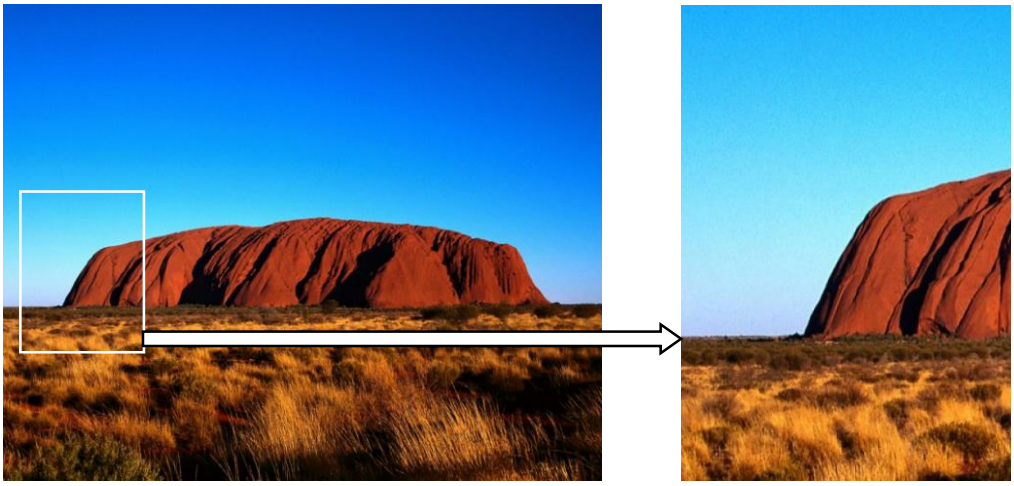


图 13 本模型算法放大效果

6. 模型的分析与改进

1．在模型噪算法在值域滤波核的均值获取方法在处理这类嗓声上也相对有效，在块状的角点附近，因为均值综合了两边颜色颜色值，减弱了值域滤波核系数的权重，使得总体遽波更倾向于高斯诚波模糊角点的效果。但是，当压缩率比较高时，块状区域变大，去噪效果不太明显，这里是未来需要改进的地方。可以在计算结构张量时加上角点检测的步骤，然后根据检测到的角点的密度和走向判断是否存在倾斜的边缘，然后对这种边缘进一步处理。

2．双边滤波具有两面性，在尽量保持了边缘之后，去噪的效果并不尽如人意，或者说仅从去噪上考虑，没有单纯的高斯滤波效果好。特别是当噪声不是一个单独的像素时，双边滤波会把噪声当做边缘对待，因为相邻的噪声像素的权重会非常大，其他像素对目标像素几乎没有影响。可以作如下改进：

$$S(f(\xi),f(x))=e^{-\frac{1}{2}(\frac{f(\xi)-f(x)-\varsigma(x)}{\delta_r})^2}$$

在值域滤波核的计算上加上一个控制系数 ς，通过该控制系数的设定控制结果图像的平滑或者尖锐程度。

7. 模型的评价

7.1. 模型的优点

(1) 我们在本文的算法中考虑到了图像的去噪问题，提高边缘提取的准确性。
(2) 在考虑放大算法的过程中，我们考虑了保持边缘的完整性和平滑性，不会产生明显的锯齿化。
(3) 利用基于偏微分方程的放大算法，区别于传统差值方法的在于保证了未知像素与已知像素点的关联性，得到的结果更易令人接受。

7.2. 模型的缺点

(1) 在本文的去噪方法中，对于不同种类的噪点去除效果不一致，对单一的噪点类型图形去噪效果明显较好，导致对多噪点图像的去除效果较差。
(2) Canny 算子提取边缘法虽然性能优越，但是在处理的速度上较为缓慢。
(3) 我们本文的算法在处理放大倍数较小的图像放大时较为优越，而图像放大倍数较大时，效果不明显。

参考文献

[1] Abdul J. Jerri. The Shannon sampling theorem: Its various extensions and applications: Authorial review[J].Proceedings of the IEEE，1977，65(11):1565-1596.
[2] Raanan Fattal. Image upsampling via imposed edge statistics[J]. ACM Trans. Graph.， 2007，26(3).
[3] Hu Min， Zhang Yousheng. Image zooming based on Thicle’s rational

interpolation[J]. Journal of
[4] Computer-Aided Design & Computer Graphics, 2003, 15(8):1004-1007.
[5] 龚奕刚.图像放大算法的研究[D].[硕士学位论文], 无锡:江南大学, 2008.
[6] 勒中鑫.数字图像信息处理[M], 北京: 国防工业出版社, 2003.
[7] 朱秀昌，刘峰，胡栋.数字图像处理与图像通信[M], 北京: 北京邮电大学出版社, 2002.
[8] Rafael C. Gonzalez, Richard E. Woods 著, 阮秋琦, 阮宇智译. 数字图像处理(第二版) [M], 北京:电子工业出版社, 2005.
[9] Carlo. Tomasi, Roberto Manduchi. Bilateral filtering for gray and color images[C]. In Sixth Inlernalional Conference on Computer Vision, 1998, 839 -846.
[10] Johannes Kopf, Dani Lischinski. Depixelizing pixel art[J]. ACM Trans. Graph., 2011, 30(4):99:1-99:8.
[11] Jorge Jimenez, Diego Gutierrez, Jason Yang, Alexander Reshetov, Pete Demorcuille, Tobias Berghoff, Cedric Perthuis, Henry Yu, Morgan McGuire, Timothy Lottes, Hugh Malan, Emil Persson, Dmitry Andreev, Tiago Sousa. Filtering approaches for real-time anti-aliasing[C]. In ACM SIGGRAPH 2011 Courses, 2011, 6:1-6:329.
[12] Jianchao Yang, John Wright, Thomas Huang, Yi Ma. Image super-resolution as sparse representation of raw image patches[C]. In IEEE Conference on Computer Vision and Pattern Recognition, 2008, 1-8.
[13] Jian Sun. Zongben Xu, Heung-Yeung Shum. Image super-resolution using gradient profile prior[C]. IEEE Confcrcncc on Computer Vision and Pattern Recognition. 2008, 1-8.
[14] Ncel Joshi, Richard Szeliski, David J. Kriegman. Psf estimation using sharp edge prediction[C]. In IEEE Conference on Computer Vision and Pattern Recognition, 2008, 1-8.
[15] [Qi Shan, Zhaorong Li, Jiaya Jia, Chi-Keung Tang. Fast image/video up sampling[J]. ACM Trans. Graph.2008, 27(5):153:1-153:7.
[16] 敬照亮，MATLAB 教程与应用[M]，北京：北京交通大学出版社，2011

附件

程序一 提取边缘

```
function [eout , thresh , gv_45 , gh_135] = edge(varargin)
 [a , method , thresh , sigma , thinning , H , kx , ky] = parse_inputs(varargin{:});
if ~any(strcmp(method , {'sobel' , 'roberts' , 'prewitt'})) && (nargout>2)
    error(message('images:edge:tooManyOutputs'))
end
if ~isa(a , 'double') && ~isa(a , 'single')
    a = im2single(a);
end
[m , n] = size(a);
```

```
e = false(m，n);
if strcmp(method，'canny')

    PercentOfPixelsNotEdges = .7;
    ThresholdRatio = .4;
    [dx，  dy] = smoothGradient(a，  sigma);
    magGrad = hypot(dx，  dy);
    magmax = max(magGrad(:));
    if magmax > 0
        magGrad = magGrad / magmax;
    end

    [lowThresh，  highThresh] = selectThresholds(thresh，  magGrad，
PercentOfPixelsNotEdges，  ThresholdRatio，  mfilename);

    e = thinAndThreshold(e，  dx，  dy，  magGrad，  lowThresh，  highThresh);
    thresh = [lowThresh highThresh];
  elseif strcmp(method，'canny_old')

    GaussianDieOff = .0001;
    PercentOfPixelsNotEdges = .7;
    ThresholdRatio = .4;
      pw = 1:30;
    ssq = sigma^2;
    width = find(exp(-(pw.*pw)/(2*ssq))>GaussianDieOff，1，'last');
    if isempty(width)
        width = 1;
    end
      t = (-width:width);
    gau = exp(-(t.*t)/(2*ssq))/(2*pi*ssq);
    [x，y]=meshgrid(-width:width，-width:width);
    dgau2D=-x.*exp(-(x.*x+y.*y)/(2*ssq))/(pi*ssq);
    aSmooth=imfilter(a，gau，'conv'，'replicate');
    aSmooth=imfilter(aSmooth，gau'，'conv'，'replicate');
    ax = imfilter(aSmooth，  dgau2D，  'conv'，'replicate');
    ay = imfilter(aSmooth，  dgau2D'，  'conv'，'replicate');
      mag = sqrt((ax.*ax) + (ay.*ay));
    magmax = max(mag(:));
```

```
if magmax>0
    mag = mag / magmax;     ze
end
   resholds
if isempty(thresh)
    counts=imhist(mag ,  64);
    highThresh = find(cumsum(counts) > PercentOfPixelsNotEdges*m*n , ...
        1 , 'first') / 64;
    lowThresh = ThresholdRatio*highThresh;
    thresh = [lowThresh highThresh];
elseif length(thresh)==1
    highThresh = thresh;
    if thresh>=1
        error(message('images:edge:thresholdMustBeLessThanOne'))
    end
    lowThresh = ThresholdRatio*thresh;
    thresh = [lowThresh highThresh];
elseif length(thresh)==2
    lowThresh = thresh(1);
    highThresh = thresh(2);
    if (lowThresh >= highThresh) || (highThresh >= 1)
        error(message('images:edge:thresholdOutOfRange'))
    end
end

idxStrong = [];
for dir = 1:4
    idxLocalMax = cannyFindLocalMaxima(dir , ax , ay , mag);
    idxWeak = idxLocalMax(mag(idxLocalMax) > lowThresh);
    e(idxWeak)=1;
    idxStrong = [idxStrong; idxWeak(mag(idxWeak) > highThresh)];
end
   if ~isempty(idxStrong)
    rstrong = rem(idxStrong-1 ,  m)+1;
    cstrong = floor((idxStrong-1)/m)+1;
    e = bwselect(e ,  cstrong ,  rstrong ,  8);
    e = bwmorph(e ,  'thin' ,  1);
end
```

```
elseif any(strcmp(method , {'log' , 'zerocross'}))
 rr = 2:m-1; cc=2:n-1;
        if isempty(H) ,
      fsize = ceil(sigma*3) * 2 + 1;
      op = fspecial('log' , fsize , sigma);
 else
      op = H;
 end
    op = op - sum(op(:))/numel(op);
 b = imfilter(a , op , 'replicate');
    if isempty(thresh)
     thresh = .75*mean2(abs(b));
 end

 [rx , cx] = find( b(rr , cc) < 0 & b(rr , cc+1) > 0 ...
      & abs( b(rr , cc)-b(rr , cc+1) ) > thresh );
 e((rx+1) + cx*m) = 1;
 [rx , cx] = find( b(rr , cc-1) > 0 & b(rr , cc) < 0 ...
      & abs( b(rr , cc-1)-b(rr , cc) ) > thresh );
 e((rx+1) + cx*m) = 1;
 [rx , cx] = find( b(rr , cc) < 0 & b(rr+1 , cc) > 0 ...
      & abs( b(rr , cc)-b(rr+1 , cc) ) > thresh);
 e((rx+1) + cx*m) = 1;
 [rx , cx] = find( b(rr-1 , cc) > 0 & b(rr , cc) < 0 ...
      & abs( b(rr-1 , cc)-b(rr , cc) ) > thresh);
 e((rx+1) + cx*m) = 1;

 [rz , cz] = find( b(rr , cc)==0 );
 if ~isempty(rz)

      zero = (rz+1) + cz*m;
      zz = (b(zero-1) < 0 & b(zero+1) > 0 ...
           & abs( b(zero-1)-b(zero+1) ) > 2*thresh);       '
      e(zero(zz)) = 1;
      zz = (b(zero-1) > 0 & b(zero+1) < 0 ...
           & abs( b(zero-1)-b(zero+1) ) > 2*thresh);       '
      e(zero(zz)) = 1;
      zz = (b(zero-m) < 0 & b(zero+m) > 0 ...
```

```
            & abs( b(zero-m)-b(zero+m) ) > 2*thresh);
        e(zero(zz)) = 1;
        zz = (b(zero-m) > 0 & b(zero+m) < 0 ...
            & abs( b(zero-m)-b(zero+m) ) > 2*thresh);
        e(zero(zz)) = 1;
    end
else)
    if strcmp(method , 'sobel')
        op = fspecial('sobel')/8;
        x_mask = op';
        y_mask = op;
                scale = 4;
        offset = [0 0 0 0];
            elseif strcmp(method , 'prewitt')
        op = fspecial('prewitt')/6;
        x_mask = op';
        y_mask = op;
                scale = 4;
        offset = [0 0 0 0];
            elseif strcmp(method ,  'roberts')
        x_mask = [1 0; 0 -1]/2;
        y_mask = [0 1;-1   0]/2;
                scale = 6;
        offset = [-1 1 1 -1];
            else
        error(message('images:edge:invalidEdgeDetectionMethod' ,  method))
    end
    bx = imfilter(a , x_mask , 'replicate');
    by = imfilter(a , y_mask , 'replicate');
    if (nargout > 2)
        gv_45   = bx;
        gh_135 = by;
    end
    b = kx*bx.*bx + ky*by.*by;

    if isempty(thresh) ,
            cutoff = scale*mean2(b);
        thresh = sqrt(cutoff);
```

```
    else
        cutoff = (thresh).^2;
    end
      if thinning
        e = computeEdgesWithThinning(b，bx，by，kx，ky，offset，cutoff);
    else
        e = b > cutoff;
    end
  end
if nargout==0，
    imshow(e);
else
    eout = e;
end
function e = computeEdgesWithThinning(b，bx，by，kx，ky，offset，cutoff)
bx = abs(bx);
by = abs(by);
e = computeedge(b，bx，by，kx，ky，int8(offset)，100*eps，cutoff);
function idxLocalMax = cannyFindLocalMaxima(direction，ix，iy，mag)
 [m，n] = size(mag);
switch direction
    case 1
        idx = find((iy<=0 & ix>-iy)   | (iy>=0 & ix<-iy));
    case 2
        idx = find((ix>0 & -iy>=ix)   | (ix<0 & -iy<=ix));
    case 3
        idx = find((ix<=0 & ix>iy) | (ix>=0 & ix<iy));
    case 4
        idx = find((iy<0 & ix<=iy) | (iy>0 & ix>=iy));
end
if ~isempty(idx)
    v = mod(idx，m);
    extIdx = (v==1 | v==0 | idx<=m | (idx>(n-1)*m));
    idx(extIdx) = [];
end
ixv = ix(idx);
iyv = iy(idx);
gradmag = mag(idx);
```

```
switch direction
    case 1
        d = abs(iyv./ixv);
        gradmag1 = mag(idx+m).*(1-d) + mag(idx+m-1).*d;
        gradmag2 = mag(idx-m).*(1-d) + mag(idx-m+1).*d;
    case 2
        d = abs(ixv./iyv);
        gradmag1 = mag(idx-1).*(1-d) + mag(idx+m-1).*d;
        gradmag2 = mag(idx+1).*(1-d) + mag(idx-m+1).*d;
    case 3
        d = abs(ixv./iyv);
        gradmag1 = mag(idx-1).*(1-d) + mag(idx-m-1).*d;
        gradmag2 = mag(idx+1).*(1-d) + mag(idx+m+1).*d;
    case 4
        d = abs(iyv./ixv);
        gradmag1 = mag(idx-m).*(1-d) + mag(idx-m-1).*d;
        gradmag2 = mag(idx+m).*(1-d) + mag(idx+m+1).*d;
end
idxLocalMax = idx(gradmag>=gradmag1 & gradmag>=gradmag2);
function [I , Method , Thresh , Sigma , Thinning , H , kx , ky] =
parse_inputs(varargin)

narginchk(1 , 5)
I = varargin{1};
validateattributes(I , {'numeric' , 'logical'} , {'nonsparse' , '2d'} , mfilename , 'I' , 1);
Method      = 'sobel';
Direction = 'both';
Thinning    = true;
methods      = {'canny' , 'canny_old' , 'prewitt' , 'sobel' , 'marr-hildreth' , 'log' ,
'roberts' , 'zerocross'};
directions = {'both' , 'horizontal' , 'vertical'};
options      = {'thinning' , 'nothinning'};
nonstr = [];
for i = 2:nargin
    if ischar(varargin{i})
        str = lower(varargin{i});
        j = find(strcmp(str , methods));
        k = find(strcmp(str , directions));
```

```
            l = find(strcmp(str , options));
            if ~isempty(j)
                Method = methods{j(1)};
                if strcmp(Method , 'marr-hildreth')
                    error(message('images:removed:syntax' , 'EDGE(I , ''marr-
hildreth'' , ...)' , 'EDGE(I , ''log'' , ...)'))                end
            elseif ~isempty(k)
                Direction = directions{k(1)};
            elseif ~isempty(l)
                if strcmp(options{l(1)} , 'thinning')
                    Thinning = true;
                else
                    Thinning = false;
                end
            else
                error(message('images:edge:invalidInputString' ,  varargin{ i }))
            end
        else
            nonstr = [nonstr i];
        end
    end

    [Thresh , Sigma , H , kx , ky] = images.internal.parseNonStringInputsEdge(varargin ,
    Method , Direction , nonstr);

    function [GX ,  GY] = smoothGradient(I ,  sigma)

    filterLength = 8*ceil(sigma);
    n = (filterLength - 1)/2;
    x = -n:n;

    c = 1/(sqrt(2*pi)*sigma);
    gaussKernel = c * exp(-(x.^2)/(2*sigma^2));

    gaussKernel = gaussKernel/sum(gaussKernel);

    derivGaussKernel = gradient(gaussKernel);
```

```
negVals = derivGaussKernel < 0;
posVals = derivGaussKernel > 0;
derivGaussKernel(posVals) =
derivGaussKernel(posVals)/sum(derivGaussKernel(posVals));
derivGaussKernel(negVals) =
derivGaussKernel(negVals)/abs(sum(derivGaussKernel(negVals)));

GX = imfilter(I , gaussKernel' , 'conv' , 'replicate');
GX = imfilter(GX , derivGaussKernel , 'conv' , 'replicate');

GY = imfilter(I , gaussKernel , 'conv' , 'replicate');
GY  = imfilter(GY , derivGaussKernel' , 'conv' , 'replicate');

function [lowThresh , highThresh] = selectThresholds(thresh , magGrad ,
PercentOfPixelsNotEdges , ThresholdRatio , ~)
[m , n] = size(magGrad);

if isempty(thresh)
    counts=imhist(magGrad , 64);
    highThresh = find(cumsum(counts) > PercentOfPixelsNotEdges*m*n , ...
        1 , 'first') / 64;
    lowThresh = ThresholdRatio*highThresh;
elseif length(thresh)==1
    highThresh = thresh;
    if thresh>=1
        error(message('images:edge:thresholdMustBeLessThanOne'))
    end
    lowThresh = ThresholdRatio*thresh;
elseif length(thresh)==2
    lowThresh = thresh(1);
    highThresh = thresh(2);
    if (lowThresh >= highThresh) || (highThresh >= 1)
        error(message('images:edge:thresholdOutOfRange'))
    end
end

function H = thinAndThreshold(E , dx , dy , magGrad , lowThresh , highThresh)
```

```
idxStrong = [];
for dir = 1:4
    idxLocalMax = cannyFindLocalMaxima(dir，dx，dy，magGrad);
    idxWeak = idxLocalMax(magGrad(idxLocalMax) > lowThresh);
    E(idxWeak)=1;
    idxStrong = [idxStrong; idxWeak(magGrad(idxWeak) > highThresh)]; W>
end
[m，n] = size(E);
if ~isempty(idxStrong)      rstrong = rem(idxStrong-1，  m)+1;
    cstrong = floor((idxStrong-1)/m)+1;
    H = bwselect(E，  cstrong，  rstrong，  8);
else
    H = zeros(m，  n);
end
```

程序二 去除噪声

```
function b = imnoise(varargin)
[a，  code，  classIn，  classChanged，  p3，  p4] = ParseInputs(varargin{:});
clear varargin;
b = images.internal.algimnoise(a，  code，  classIn，  classChanged，  p3，  p4);
function [a，  code，  classIn，  classChanged，  p3，  p4，  msg] =
ParseInputs(varargin)
p3              = [];
p4              = [];
msg = '';
narginchk(1，4);
a = varargin{1};
validateattributes(a，  {'uint8'，'uint16'，'double'，'int16'，'single'}，  {}，
mfilename，  ...
                   'I'，  1);
classIn = class(a);
classChanged = 0;
if ~isa(a，  'double')
  a = im2double(a);
  classChanged = 1;
else
  a = max(min(a，1)，0);
end
```

```
if nargin > 1
   if ~ischar(varargin{2})
      error(message('images:imnoise:invalidNoiseType'))
   end
    allStrings = {'gaussian' ,  'salt & pepper' ,  'speckle' , ...
                        'poisson' , 'localvar'};
   idx = find(strncmpi(varargin{2} ,  allStrings ,  numel(varargin{2})));
   switch length(idx)
    case 0
      error(message('images:imnoise:unknownNoiseType' ,  varargin{ 2 }))
    case 1
      code = allStrings{idx};
    otherwise
      error(message('images:imnoise:ambiguousNoiseType' ,  varargin{ 2 }))
   end
else
   code = 'gaussian';   % default noise type
end
 switch code
 case 'poisson'
   if nargin > 2
      error(message('images:imnoise:tooManyPoissonInputs'))
   end
      if isa(a ,  'int16')
      error(message('images:imnoise:badClassForPoisson'));
   end
    case 'gaussian'
   p3 = 0;        % default mean
   p4 = 0.01;   % default variance
      if nargin > 2
      p3 = varargin{3};
      if ~isRealScalar(p3)
         error(message('images:imnoise:invalidMean'))
      end
   end
      if nargin > 3
      p4 = varargin{4};
      if ~isNonnegativeRealScalar(p4)
```

```
        error(message('images:imnoise:invalidVariance' ,   'gaussian'))
      end
    end
     case 'salt & pepper'
    p3 = 0.05;     % default density
      if nargin > 2
      p3 = varargin{3};
      if ~isNonnegativeRealScalar(p3) || (p3 > 1)
        error(message('images:imnoise:invalidNoiseDensity'))
      end

      if nargin > 3
        error(message('images:imnoise:tooManySaltAndPepperInputs'))
      end
    end

  case 'speckle'
    p3 = 0.05;
    if nargin > 2
      p3 = varargin{3};
      if ~isNonnegativeRealScalar(p3)
        error(message('images:imnoise:invalidVariance' ,   'speckle'))
      end
    end
    if nargin > 3
      error(message('images:imnoise:tooManySpeckleInputs'))
    end
  case 'localvar'
    if nargin < 3
      error(message('images:imnoise:toofewLocalVarInputs'))
    elseif nargin == 3
      code = 'localvar_1';
      p3 = varargin{3};
      if ~isNonnegativeReal(p3) || ~isequal(size(p3) , size(a))
        error(message('images:imnoise:invalidLocalVarianceValueAndSize'))
      end
    elseif nargin == 4
      code = 'localvar_2';
```

```
        p3 = varargin{3};
        p4 = varargin{4};
            if ~isNonnegativeRealVector(p3) || (any(p3) > 1)
          error(message('images:imnoise:invalidImageIntensity'))
        end
            if ~isNonnegativeRealVector(p4)
          error(message('images:imnoise:invalidLocalVariance'))
        end
            if ~isequal(size(p3) , size(p4))
          error(message('images:imnoise:invalidSize'))
        end
      else
        error(message('images:imnoise:tooManyLocalVarInputs'))
      end
      end
function t = isReal(P)
  isFinite    = all(isfinite(P(:)));
  t = isreal(P) && isFinite && ~isempty(P);
function t = isNonnegativeReal(P)
  t = isReal(P) && all(P(:)>=0);
function t = isRealScalar(P)
  t = isReal(P) && (numel(P)==1);
function t = isNonnegativeRealScalar(P)
  t = isReal(P) && all(P(:)>=0) && (numel(P)==1);
function t = isVector(P)
  t = ((numel(P) >= 2) && ((size(P , 1) == 1) || (size(P , 2) == 1)));
function t = isNonnegativeRealVector(P)
  t = isReal(P) && all(P(:)>=0) && isVector(P);
```