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，我们将受到严肃处理。

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：

参赛队员 (签名) ：

队员 1： 刘奔

队员 2： 张浩楠

队员 3： 唐潇潇

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）：4537

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

题　目　位图放大保真模型

关 键 词　*Sobel*算子、*Thiele – Newton*插值法、分片连续曲面、增强处理

摘　要

本文针对位图的放大问题，以题中所给的位图为切入点，综合分析了位图各像素点的坐标及其对应的*RGB*分量，并通过文献的查阅，基于插值图像边缘部分的分辨率对整个图像放大的重要影响，确立了对边缘部分与非边缘部分采取不同插值算法的建模思路，建立了基于*Sobel*算子改进后的彩色图像边缘检测模、*Thiele - Newton*插值法图像边缘部分放大模型、图像放大的分片连续模型和图像“质检—去噪—后处理”模型，运用*Matlab*软件，C^{++}对图像数据进行处理、分析。最后，对整个模型存在的不足与优点进行讨论，提出对原模型的改进和推广。

针对问题一，首先，使用改进后的适用于彩色图像的*Sobel*算法对原图像，借助C^{++}程序对图像进行边缘检测，得到边缘像素点及其*RGB*值。然后，对边缘像素点进行精密的*Thiele - Newton*二元有理插值，实现边缘区域的放大算法。

针对问题二，通过对非边缘图像划分区域段，建立段内连续函数，连续段间的延拓将其分为分片连续的曲面。然后，将整个非边缘曲面表示为了二元的分片连续函数，通过像素*RGB*分量在新坐标系中的映射关系实现非边缘区域的放大算法。

针对问题三，首先，问题一与问题二中模型所产生两部分区域放大的组合已初步实现了整个图像的高保真放大，但基于对图像清晰度及背景平滑性的考虑，需要对放大后的图像进行进一步处理。使用彩色图像矢量中通滤波进行去噪处理，并利用反锐化掩模法对插值图像的细节进行进一步增强。

本文还对模型的误差进行了具体分析；对模型的优化提出了针对性的改进，分析了模型存在优势与不足。最后，我们又对模型进行了多个方向的推广，分析了其在三维图像放大处理与二维图像缩小处理上的应用前景。

参赛队号　4537
所选题目　B题

参赛密码 _____
（由组委会填写）

Abstract

In this article,we make the bitmap given from the question as the breakthrough point and point at the amplification of the bitmap.Analyzing each pixel coordinates of the bitmap and their corresponding RGB components comprehensively,and through literature review,we can know that the resolution of the image edges based on interpolation influences the whole image magnification greatly, then we establish an idea that we should take different interpolation algorithm to the edges and the edges' modeling,set up the improved color image edge detection based on Sobel operator,newton-Thiele interpolation image edges amplification model, piecewise continuous model of image magnification and image "quality inspection, denoising, post-processing"model,using the Matlab software,the C++ for image data processing and analysis. In the end, discussing the deficiencies and advantages of the entire model, proposing improvements on the original model and promotion.

Aiming at the first problem , first of all, using the improved Sobel algorithm for color image of the original image, using C++ program for image edge detection, we can get the edge pixels and its RGB values. Then, using $Newton-Thiele$ bivariate rational interpolation on the the edge pixels precisely,finishing the magnification of the edge.

In view of the question two,based on dividing the edge image area into segments,to set up within the period of continuous function, dividing it into piecewise continuous surface by consecutive continuation between each segment.Then,represent the whole the edge surface to binary piecewise continuous function,by pixel RGB components on the edge of the mapping relationship of the new coordinate system to realize the area magnification algorithm.

On question three,first of all,the model of the first and second question produced the combination of the two parts of amplifier is preliminary realized the whole image of high-fidelity amplifier,but based on the background of the image's resolution and smoothness,the enlarged image is needed for further processing.The use of color image vector filter to deal with the noise,and using the method of inverse sharpening mask to further enhance the interpolation of the image's details.

This article also gives a detailed analysis to the error of the model.puts forward the specific improvement of the optimization of model,analyzing the existing advantages and disadvantages of this model.Finally,we has promote this model on the multiple directions,analyzes its prospects of the magnification in three-dimensional image and the reduced processing application in two-dimensional image.

目录

§1 问题的提出 - 2 -
一、背景知识 - 2 -
1.位图 - 2 -
2. 灰度图像的表示 - 2 -
3. 彩色图像的表示 - 2 -
4. 图像放大 - 2 -
二、相关数据 - 2 -
1. 图像放大前的像素点坐标及其对应的 *RGB* 分量（见附录表[1]） - 2 -
2. 图像放大后的像素点坐标及其对应的 *RGB* 分量（见附录表[2]） - 2 -
三、要解决的问题 - 3 -
1. 问题的提出 - 3 -
2. 具体问题 - 3 -
§2 模型的假设 - 3 -
§3 名词解释与符号说明 - 4 -
一、名词解释 - 4 -
二、符号说明 - 4 -
§4 模型的建立与求解 - 5 -
一、问题一的分析与求解 - 5 -
1. 对问题的分析 - 5 -
2.问题的求解 - 5 -
二、问题二的分析与求解 - 9 -
1.对问题的分析 - 9 -
2.问题的求解 - 9 -
三、问题三的分析与求解 - 12 -
1.对问题的分析 - 12 -
2.问题的求解 - 13 -
§5 模型的误差分析 - 18 -
§6 模型的评价 - 18 -
一、模型的优点： - 18 -
二、模型的缺点： - 18 -
§7 模型的改进与推广 - 19 -
一、改进 - 19 -
1. 模型的初步改进 - 19 -
2. 模型的进一步改进 - 19 -
二、模型的推广 - 20 -
参考文献 - 20 -
附录 - 21 -
表[1] 原始图像的像素点坐标及其对应的 *RGB* 分量（部分） - 21 -
表[2] 放大图像的像素点坐标及其对应的 *RGB* 分量（部分） - 25 -
程序[1]：改进 *Sobel* 算子 C^{++} 源代码 - 29 -
程序[2]：中值滤波去噪代码 - 31 -

§1 问题的提出

一、背景知识

1.位图

位图，亦称栅格图，通过像素阵列来表示图像。这些像素可以进行不同排列和染色以构成图像，每个像素的颜色信息由 RGB 组合或者灰度值表示，在对位图图像进行编辑操作的时候，可操作的对象是每个像素。位图文件在放大后质量明显下降，容易产生较明显的模糊或马赛克等现象。

2. 灰度图像的表示

灰度图像是每个像素只有一个采样颜色的图像。表示灰度图像可以通过构建物体的二维光强度函数 $f(x,y)$，其中 x,y 是空间上的点坐标，任意点 (x,y) 处的数值 $f(x,y)$ 正比于图像在该点的灰度级 L，二变量实函数表示一幅灰度图像，是在空间的坐标和亮度上均已离散化的图像。表示一幅灰度图像可以把其化为一个矩阵，行和列表示图像中的一个点，相应矩阵中元素的值表示该点的灰度级。

3. 彩色图像的表示

彩色图像主要分为 RGB 和 $CMYK$ 两种类型。由于一般多用三原色（红色、绿色、蓝色）即 RGB 产生彩色图像，所以采用由 RGB 构成的彩色图像。表示彩色图像可以通过构建函数 $f_c(x,y)=\{f_r(x,y),f_g(x,y),f_b(x,y)\}$，其中 $f_c(x,y)$ 由 (r,g,b) 表示，r,g,b 分别为 f_r、f_g、f_b 三个灰度图像的灰度值。由于 RGB 是三个正交的分量，对彩色图像的表示和处理需要处理好单色灰度图像，进而对 RGB 三个矩阵分别处理。

4. 图像放大

图像放大是根据原始图像的像素点产生更多的像素点来改变图像的尺寸。当图像尺寸增大的时候，构成图像的像素也越来越大，由于不能获得更多细节，因此图像质量常会有所下降。图像放大的实质是进行图像数据的补充，即补充所需要的像素点。尽可能的保证图像放大后的质量，是图像放大的关键。

二、相关数据

1. 图像放大前的像素点坐标及其对应的 RGB 分量（见附录表[1]）

注：原始图像分辨率为 207×329，由于数据过于庞大，每组分量只选取前 30×30 的坐标点。

2. 图像放大后的像素点坐标及其对应的 RGB 分量（见附录表[2]）

注：放大图像分辨率为 731×1162，由于数据过于庞大，每组分量只选取前 30×30 的坐标点。

三、要解决的问题

1.问题的提出

现如今，位图在生活中的方方面面都有了十分广泛的应用，但其在经过放缩或其他变化后图形会有所改变，产生局部或整体模糊，并且线条会出现错位等模糊现象，非常影响图片质量。在有些领域中对图像的处理有着十分严格的要求，如医学系统、公安系统、航天系统等一些图像处理软件中,为适用特殊的场合和获得较好的视觉效果,常常需要一种有效的方法来改变已有图像的大小,并保证改变后的图像有较好的质量。

传统的放大或缩小变换不能保证图像内各物体之间边界的清晰。采用传统方法实现图片放大时,若放大倍数较大,则被放大图像的边缘会出现明显的锯齿现象,原本光滑的边界显得凹凸不平,同时画面显得十分模糊。这是因为一些传统的实现图象放大的算法,只是将图片中的各象素点通过简单的重复来实现数目的增加,也就是通过使图像的象素点变粗来实现图像的放大,这势必造成放大后的图像的产生锯齿状和图像模糊不清的现象。所以需要建立合理的数学模型，设计可以放大位图同时保证图像质量的算法，使得位图能被清晰的放大，且边界清晰，图片质量高。

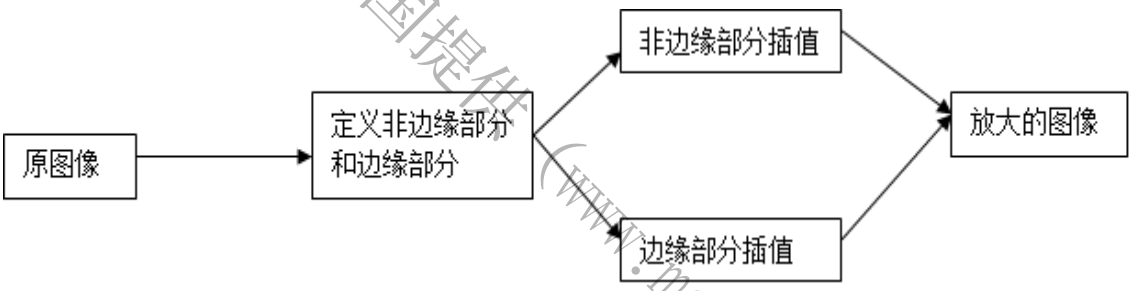


图 1-1 问题思路

2.具体问题

问题一：对彩色图像进行边缘区域检测并对其进行边缘插值。

问题二：对图像进行分片处理，确定局部连续区域（非边缘区域）分片为曲面，并对曲面进行插值。

问题三：对目标图像进行放大后的质量提升处理。

§2 模型的假设

1.假设目标图像水平清晰度较高，图像质量较高。

2.假设目标图像尺寸较小，像素点数量有限，可以进一步进行图像放大。

3.假设目标图像可能被噪声污染，存在一定噪点，需要进行去噪处理。

4.假设对目标图像的像素点进行插值得到的曲线或平面具有一定的光滑性。

§3 名词解释与符号说明

一、名词解释

1.灰度值：灰度值指黑白图像中像素点的颜色深度，范围一般从 0 到 255，黑色为 0，白色为 255。

2. *RGB* 值： *RGB* 是代表红、绿、蓝三个通道的颜色，是工业界的一种颜色标准。

3. *Sobel* 算子： *Sobel* 算子是一离散性差分算子，主要用于边缘检测。*Sobel* 算子有两个，一个检测水平边缘；另一个检测垂直边缘。*Sobel* 算子对于象素的位置的影响做了加权，可以降低边缘模糊程度，比 *prewitt* 算子效果更好。

4.阈值：阈值又称阈强度，是指释放一个行为所需要的最小刺激强度。阈值对确定图像的最亮和最暗区域有重要作用。在图像的二值化中常常使用阈值，二值化的结果严重依赖阈值的选择。

5.扭曲距离：扭曲距离(*Warped* Distance)是一种距离变化的方法，由 *Giovanni Ramponi* 提出的线性插值方法，该方法不仅能够降低插值误差而且计算简单方便，可以对数据实现整数倍和非整数倍的插值，可以替代传统的线性插值算法。

6.峰值信噪比：峰值信噪比（*PSNR*）经常用作图像压缩等领域中信号重建质量的测量方法，它常通过均方差（*MSE*）进行定义。

7.分片连续：分片连续指将平面上定义域分成若干个单连通区域，该函数在每一区域上是连续的。

8.滤波：滤波是将信号中特定波段频率滤除的操作，是抑制和防止干扰的一项重要措施。分经典滤波和现代滤波。

9.图像去噪：图像去噪是数字图像处理中的重要环节和步骤。去噪效果的好坏直接影响到后续的图像处理工作图像分片、边缘检测、图像插值。

二、符号说明

序号	符号	符号说明
1	RGB	工业界的一种颜色标准
2	L	灰度值
3	G	梯度值
4	$M \times N$	原始图像像素点
5	$M' \times N'$	处理后图像像素点
6	$f(x,y)$	原始图像
7	$f'(x,y)$	处理后图像

8	$g_{mask}(x,y)$	原图像细节图像
9	d	颜色值之差
10	$\frac{M'}{M},\frac{N'}{N}$	图像在 x,y 方向上的伸缩比
11	$C_{s\beta wi}$	平均色彩函数值
12	$D_{s\beta}$	垂直于 β 方向色彩函数值的突变程度

§4 模型的建立与求解

一、问题一的分析与求解

1. 对问题的分析

问题一：对彩色图像进行边缘区域检测并对其进行边缘插值。

将问题一拆分为两个部分：**第一，**改进 *Sobel* 算子，对目标彩色图像边缘区域进行检测；**第二**，对边缘区域像素点进行插值。**首先**，运用数学软件 *Matlab* 对检测目标图像的边缘区域，得到轮廓像素点的坐标及其对应的 *RGB* 分量。考虑到 *Sobel* 算子对灰度图像边缘检测效果较好，但是对彩色图像边缘检测会出现边缘模糊的现象，影响后续图像处理。**因此**，根据彩色图像特点，通过计算 *RGB* 分量梯度值，改进 *Sobel* 边缘检测方法，提升边缘检测效果。**其次**，在图像边缘区域采取自适应插值算法，运用较小的运算价，以便能够得到更好的放大效果。

原始图像为：

图 4-1 原始图像

2.问题的求解

◆ **模型Ⅰ基于 *Sobel* 算子改进后的彩色图像边缘检测模型**

（1）理论的准备

传统 *Sobel* 算子边缘检测原理：

传统 *Sobel* 算子边缘检测方法，是一种一阶微分算子。其利用图像空间中两个方向模板与图像进行邻域卷积完成的。两个方向模板一个检测垂直边缘，一个

检测水平边缘。

水平边缘 *Sobel* 算子：$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$　　垂直边缘 *Sobel* 算子：$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$

Sobel 算子通过分析目标图像 $f(x,y)$ 的每个像素，检测其相邻点灰度的加权差，与之接近的相邻点的权大，进行边缘检测。其检测原理为：

$$S_x = \{ f(x+1,y+1) - 2f(x+1,y) + f(x+1,y+1) \}$$

$$S_y = \{ f(x-1,y+1) - 2f(x,y+1) + f(x+1,y+1) \} - \{ f(x-1,y-1) + 2f(x,y) + f(x+1,\ y+1) \}$$

取适当的阈值 TH，做出判断，即 $R(i,j) >= TH$，则 (i,j) 为阶跃状边缘点，$R(i,j)$ 边缘图像。

传统 *Sobel* 算子边缘检测离散点图为：

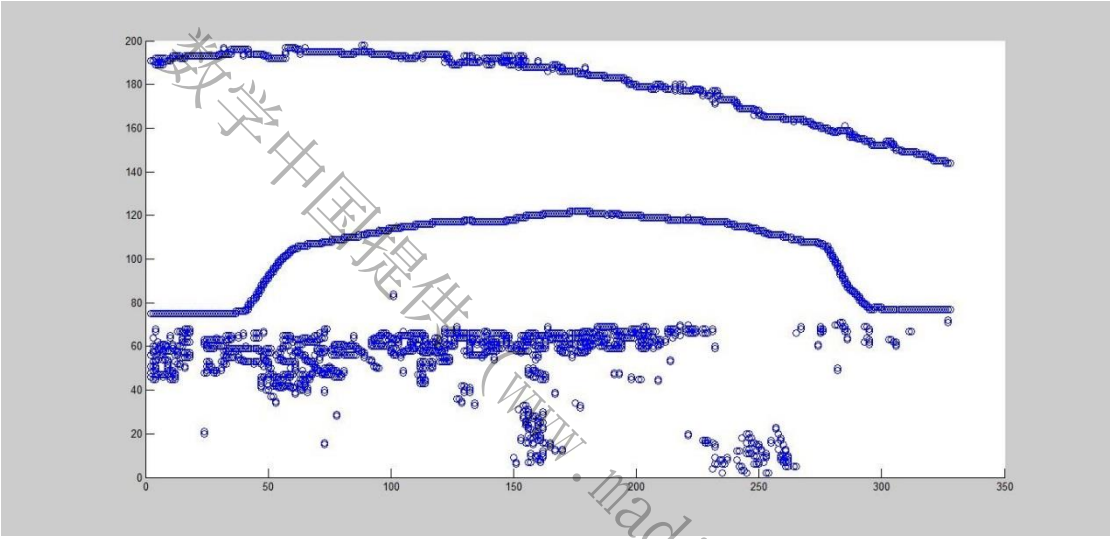


图 4-2 传统 *Sobel* 算子边缘检测离散点图

评价： *Sobel* 算子根据在边缘点出达到极值的原理进行边缘检测，可以产生较好的检测效果，但是对彩色图像进行检测会出现边缘模糊的现象。**因此**，要对传统 *Sobel* 算子进行改进。

（2）模型的建立

改进的 *Sobel* 算子的算法：

彩色图像的每个像素点包含 *RGB* 三个通道，且每个通道的分布不同，各个通道得到的边缘特征位置也不同。采用传统的 *Sobel* 算法处理彩色图像边缘，不能充分表示彩色图像的边缘信息，并影响图像的后续放大处理。为了提取清晰的图像边缘，对 *Sobel* 算子进行了色彩改进，建立了新的彩色边缘检测算法。

①将目标图像的每个像素点分解为 *RGB* 三个分量。

②用公式分别计算分量 *RGB* 在 *x* 方向上的梯度值。

公式如下：

$$G_{rx} = r(x-1,y+1) + 2r(x,y+1) + r(x+1,y+1) - r(x-1,y-1) + 2r(x,y-1) + r(x+1,y-1)$$

$$G_{gx} = g(x-1,y+1) + 2g(x,y+1) + g(x+1,y+1) - g(x-1,y-1) + 2g(x,y-1) + g(x+1,y-1)$$

$$G_{bx}=b(x-1,y+1)+2g(x,y+1)+b(x+1,y+1)-b(x-1,y-1)+2b(x,y-1)+b(x+1,y-1)$$

③用公式分别计算 *RGB* 在 *y* 方向上的梯度值。

公式如下：

$$G_{ry}=\left|r(x+1,y-1)+2r(x+1,y)+r(x+1,y+1)-r(x-1,y-1)+2r(x-1,y)+r(x-1,y+1)\right|$$

$$G_{gy}=\left|g(x+1,y-1)+2g(x+1,y)+g(x+1,y+1)-g(x-1,y-1)+2g(x-1,y)+g(x-1,y+1)\right|$$

$$G_{by}=\left|b(x+1,y-1)+2b(x+1,y)+b(x+1,y+1)-b(x-1,y-1)+2b(x-1,y)+b(x-1,y+1)\right|$$

④计算 *x* 方向的梯度值 G_x 和 *y* 方向的梯度值 G_y，取较大值为中心点的颜色值。

公式如下:

$$G_x=RGB(G_{rx},G_{gx},G_{bx})$$

$$G_y=RGB(G_{ry},G_{gy},G_{by})$$

⑤对图像中的每个像素点实行上诉四步处理，用改进的算法对图像进行处理，图像的边缘轮廓比较清晰，并较好地抑制了噪声。

改进的 *Sobel* 算子边缘检测离散图为：

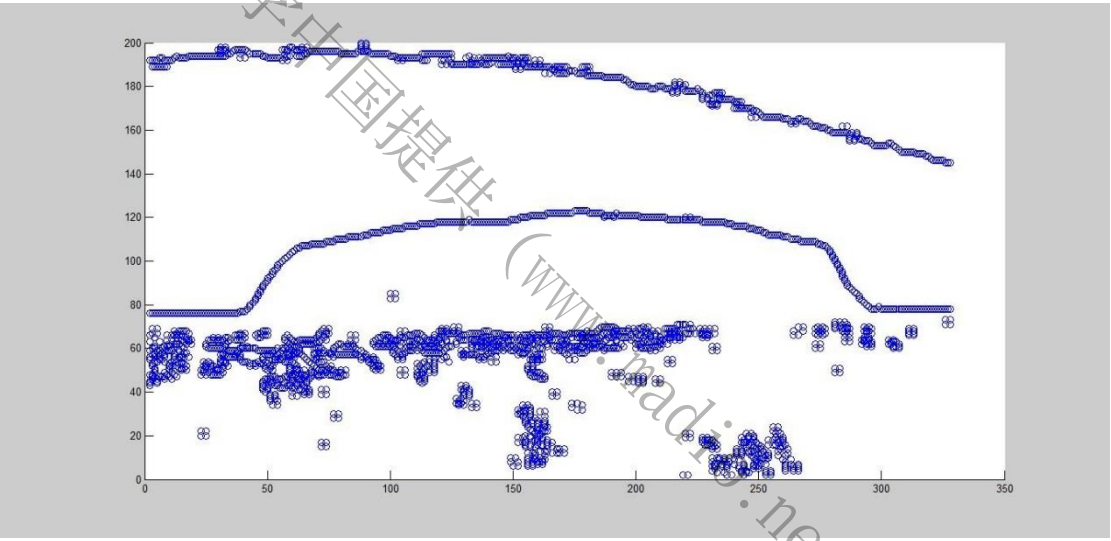


图 4-3 改进的 *Sobel* 算子边缘检测离散图

（3）模型代码

改进 *Sobel* 算子 C^{++} 源代码（详见附录程序[1]）

◆ 模型Ⅱ *Thiele_Newton* 插值法实现图像边缘部分放大

（1）理论的准备

Thiele-Newton 二元有理插值算法

Thiele-Newton 二元有理插值算法，将 *Newton* 插值多项式和 *Thiele* 型插值连分式结合起来，通过混合差分实现二元有理插值。

Thiele-Newton 二元有理插值函数如下：

$$R_{m,n}(x,y)=A_0(y)+(x-x_0)A_1(y)+\cdots+（x-x_0)\cdots(x-x_{m-1})A_m(y)$$

满足插值条件 $R_{m,n}(x_i,y_i)=F(x_i,y_i)$ 的构造方法。其中：

$$A_i(y)=a_{i0}+\frac{y-y_0}{a_{i1}}+\frac{y-y_1}{a_{i2}}+\cdots+\frac{y-y_{n-1}}{a_{in}},\quad i=0,1,\cdots,m$$

首先要决定系数 a_{ij}， $i=0,1,\cdots,m$， $j=0,1,\cdots,n$

定义混合差分如下：

$$\phi_{0,0}[xi;y]=f(x_i,y),\quad i=0,1,\cdots,m$$

$$\phi_{i,0}[x_0,\cdots,x_i;y_i]=\frac{\phi_{i-1,0}[x_0,\cdots,x_{i-2},x_i;y_i]-\phi_{i-1,0}[x_0,\cdots,x_{i-1};y_i]}{x_i-x_{i-1}}$$

$$\phi_{i,j}[x_0,\cdots,x_i;y_j]=\frac{y_i-y_{j-1}}{\phi_{i,j-1}[x_0,\cdots,x_i;y_0,\cdots,y_{j-2},y_j]-\phi_{i,j-1}[x_0,\cdots,x_i;y_0,\cdots,y_{j-1}]}$$

其中，总假定对 $\forall i$， $x_i\neq x_{i-1}$ 以及

$$\phi_{i,j-1}[x_0,\cdots,x_i;y_0,\cdots,y_{j-2},y_j]\neq\phi_{i,j-1}[x_0,\cdots,x_i;y_0,\cdots,y_{j-1}],\quad \forall j,i$$

于是令 $a_{ij}=\phi_{i,j}[x_0,\cdots,x_i;y_0,\cdots,y_j]$， $i=0,1,\cdots,m$， $j=0,1,\cdots,n$。

（2）模型的建立

Thiele - Newton 插值的具体步骤如下：

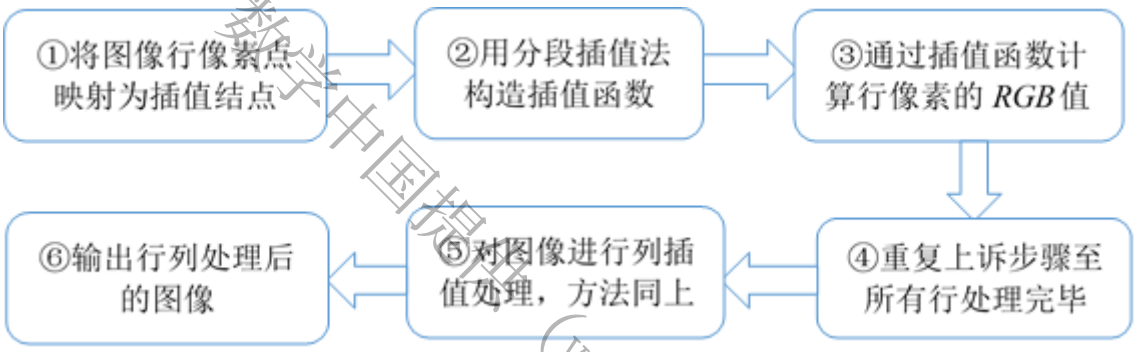


图 4-4 *Thiele - Newton* 插值具体步骤

Thiele - Newton 插值算法放大图像边缘区域

图像缩放实质上就是重采样的过程，其过程可以描述如下：

图像边缘部分为大小为 $M\times N$ 的图像，将其用 $f(x,y)$ 表示为一个采样点在整数点上的二位离散信号。处理后的图像 $f'(x,y)$ 大小为 $M'\times N'$，则在 x 方向的伸缩比为 $\frac{M'}{M}$，在 y 方向的伸缩比为 $\frac{N'}{N}$。

令 $R(i,j)$ 是 $f(x,y)$ 的第 i 行第 j 列像素点 $P(i,j)$ 的红色分量，则由映射关系知 $P(i,j)$ 对应于 $f'(x,y)$ 中的点 $P'(i',j')$，令：

$$i'=i\times\frac{M'}{M},\quad j'=j\times\frac{M'}{M},\quad x=\lfloor i'\rfloor,\quad y=\lfloor j'\rfloor$$

则 $R(i,j)$ 就是 $f'(x,y)$ 中的插值结点 (x,y) 红色分量的值。

利用同样方法，可以求出插值结点绿色分量 $G(i,j)$ 和蓝色分量 $B(i,j)$ 的值。

因此，原图像的像素点值 $P(i,j)$， $i=0,1,\cdots M-1$， $j=0,1,\cdots N-1$ 就是处理后图像的插值结点 $\left\|\left[i\times\frac{M'}{M}\right]\times\left[j\times\frac{N'}{N}\right]\right\|$， $i=0,1,\cdots M-1$， $j=0,1,\cdots N-1$ 的值。

为了减少计算时间和提高插值效果，对行和列采用分段插值的方法。同时

为了保证插值的连续性，前后两端插值端点首尾重叠。

通过上述算法可得到边缘部分的放大图像。

二、问题二的分析与求解

1.对问题的分析

问题二：对图像进行分片处理，确定局部连续区域（非边缘区域）分片为曲面，并对曲面进行插值。

经过模型Ⅰ和模型Ⅱ对图像边缘像的检测提取并进行插值放大处理后，我们需要对大量的非边缘图像部分进行放大处理。

使用较为普遍的算法如最近邻域法，双线性内插法，三次内插法等方法虽然能够快速生成较为视觉效果较为良好的目的图像，但仍然存在图像中物体边界区域模糊的问题，限制了其在实际生活场合以及专业图像处理场合的应用。基于此，我们采用一种图像的分片连续数学模型，先将图像分片为连续的曲面，再对曲面进行插值，将原始图像用二元分片连续函数表示，进而对非边缘部分进行放大处理。

2.问题的求解

◆ **模型Ⅲ 图像缩放的分片连续算法**

（1）理论的准备

分片连续，指将平面上定义域分成若干个单连通区域，该函数在每一区域上是连续的。

连续度是用来刻划区域的分片连续性的一个阈值，当相邻两像素的颜色值之差绝对值大于 d 时，就认为这两像素落在两个不同的连续段内。连续度的引入刻划了从一个连续段到另一个连续段时颜色的不连续过渡。

从某行扫描线的第一个像素出发，依次计算两相邻像素颜色值之差，一旦某差值大于 d，便认为是前一个连续段的结束，同时又是后一个连续段的开始。当穷尽该扫描行上所有像素时，该扫描行上用离散形式刻划的所有连续段便已确定。这些连续段的定义域是离散点集。

函数延拓最早来自复变函数的术语，后来也被用到泛函分析。将一个函数的定义域扩大的过程称为延拓。比如著名的黎曼（Re*imann*）ζ 函数：

$$\zeta(s) = 1 + 1/2^s + 1/3^s + \cdots + 1/n^s + \cdots$$

原本定义在实部大于 1 的复数上。但是通过延拓可以定义在任何不等于 1 的复数上。一般来说，延拓要求具有唯一性，只能按照唯一的方式来延拓原来的函数。

(2)模型的建立

分片连续算法缩放图像的**具体步骤**如下：

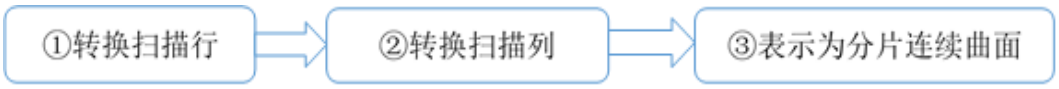


图 4-5 分片连续算法缩放图像具体步骤

分片连续算法

第一步，转换扫描行

计算每一扫描行两相邻像素颜色值之差并与连续度对比，将每一扫描行确定为离散形式的连续段。

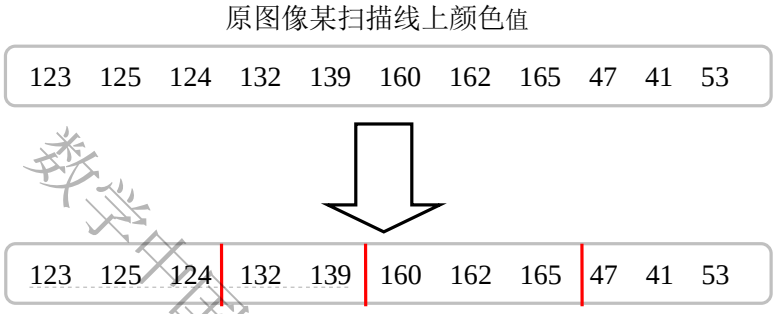


图 4-6 扫行示意图

将这些离散颜色集用连续函数表达。取满足 $g(x_i)=c$， $i=1,2,3$ 的函数形式为

$$f(x)=1/2^{*}\,g_1(x)(y-y_2)(y-y_3)-g_2(x)(y-y_{1)}(y-y_3)+1/2^{*}g_3(y-y_1)(y-y_2)$$

由于 x_i， $i=1,2,3$ 在扫描线上，故应有：

$$x_3=x_2+1=x_1+2$$

两个相邻连续段之间的函数表达：

设有两连续曲线段：

$$c=g_1(x)，\ x_1\le x\le x_t \text{ 及 } c=g_2(x)，\ x_t+1\le x\le x_s$$

其中 c 表示颜色。

线性插值可表示为：

$$f(x)=(1-s)f(x_k)+sf(x_k)$$

其中 $s=x-x_k$， $x_k\le x\le x_{k+1}$，将 s 替换成 s'，可得：

$$s'=s-Aks(s-1).，\ 0\le s'\le 1$$

其中 $A=\dfrac{|f(x_{k+1})-f(x_{k-1})|-|f(x_{k+2})-f(x_k)|}{L-1}.$，$k>0$ 用来控制扭曲程度。

常取 $k\le 1$。

为了将各连续段更合理地延拓至段边界，必须确定每一段的延拓范围，即确定两侧的延拓终点。将 $[x_t,x_{t+1}]$ 的中点 $(x_t+1)/2$ 经扭曲距离变换后得到的点 x_t+s'，取 $s=1/2$，再取 $g_1(x)$ 向 x_t 右侧延拓的终点为 x_t+s'，取 $g_2(x)$ 向 x_{t+1} 左

侧延拓的终点 $x_{t+1}+s'-1$。（当 $x_{t+1}=x_t+1$ 时即 x_t+s' ）

$$c=\begin{cases} g_1(x),\left(x_1\le x<x_t+s'\right); \\ g_2(x),\left(x_t+s'\le x\le x_s\right). \end{cases}$$

由此得到的 $f(x)$ 在上述 x_t+s' 处发生第一类间断。

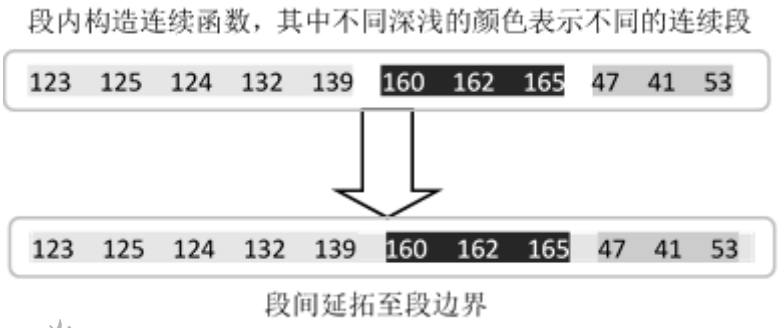


图 4-7 扫行间断示意图

第二步，转换扫描列

对于任意的 x，利用 $g_i(x)(i=1,2,\ldots,n)$，将 $f(x,y)$ 表示为关于 y 的分段连续的曲线。

上述 $g(x_i)=c$， $i=1,2,3$ 调整为 $f(x,y_i)=g_i(x),i=1,2,\ldots,k$ 。

$f(x)=1/2^*\,g_1(x)(y-y_2)(y-y_3)-g_2(x)(y-y_{1)}(y-y_3)+1/2^*g_3(y-y_1)(y-y_2)$

调整为

$f(x)=1/2^*\,g_1(x)(y-y_2)(y-y_3)-g_2(x)(y-y_{1)}(y-y_3)+1/2^*g_3(y-y_1)(y-y_2)$。

第三步将源图像表示成分片连续的曲面

通过以上两步，可以看到，对平面定义域内的任一点 (x,y) 均有确定的 $f(x,y)$ 与之对应，表明原始离散图像已用二元分片连续函数表示，同时将定义域分成若干个单连通区域，而该函数在每一区域上是连续的。有原始离散图像的二元分片连续函数后，可以进行重采样，满足任意倍数的放缩要求。

第四步曲面放大

设 $f(x,y)$ 是一个 $(n+1)\times(m+1)$ 的数字图像，$f'(x',y')$ 是 $f(x,y)$ 的一部分，其大小为 $(n'+1)\times(m'+1)$ 。现在要求将 $f'(x',y')$ 放大为 $(n+1)\times(m+1)$ 大小的图像 $J(x,y)$，如图所示。

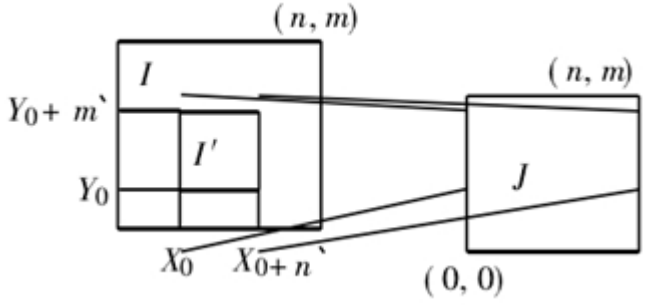


图 4-8 $J(x,y)$ 的图像

可以看出， $J(m,n)$ 相对于 $f'(m',n')$ 而言，在 i 和 j 方向上的压缩比为

$$scalex=(n'+1)/(n+1).$$
$$scaley=(m'+1)/(m+1).$$

令 $R(i,j)$ 是 $J(m,n)$ 的第 i 行第 j 列像素 $P(i,j)$ 的红色分量。由映射关系可以知道， $P(i,j)$ 对应于 $f'(m',n')$ 中的位置 (x',y') 为：

$$x'=scalex\times i$$
$$y'=scaley\times j$$

令 $x=[x']$， $y=[y']$， $coordx=x'-x$， $coordy=y'-y$

则 $f(m',n')$ 中坐标为 (x',y') 的点即为 $f'(m',n')$ 的第 (x,y) 像素内 $(coor,dcoor)$ 坐标处的点。所以 $R(i,j)$ 的值就等于 $f'(m',n')$ 的红色分量曲面上第 (x,y) 块 S_{xy} 在 $(coordx,coordy)$ 坐标处的值。

利用同样的方法，可以求出绿色分量 $G(i,j)$ 和蓝色分量 $B(i,j)$ 的值。

因此，原图像的像素点值 $P(i,j)$， $i=0,\ldots,n+1$； $j=0,\ldots,m+1$ 就是处理后图像的插值结点 $\left[\left[i\times\frac{n'+1}{n+1}\right],\left[j\times\frac{m'+1}{m+1}\right]\right]$， $i=0,1,\ldots$， $n+1$； $j=0,1,\ldots,m+1$ 的值。

三、问题三的分析与求解

1.对问题的分析

问题三：对目标图像进行放大后的质量提升处理。

经过对目标图像两部分有针对性地进行不同的插值放大算法后，我们得到了目标图像初步放大后的结果。但为了保证放大后图像的视觉质量，我们需要对放大后的图像进行如下操作：

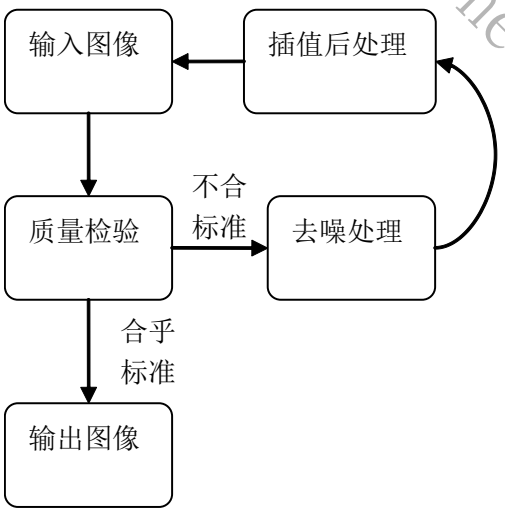


图 4-9 问题三操作图

2.问题的求解

⑴理论的准备

图像质量的评价

对于图像的质量评价包括主观评价以及客观评价,所谓主观评价就是观察者根据图像的视觉效果对图像进行打分,目前有基于主观人眼视觉特性的图像质量评价方法和基于退化模型的评价方法,但是这些方法对插值图像评价不太适合,针对插值图像需要客观评价,即要计算一些数据的值来判断图像的质量好坏。

我们利用计算原图像和插值后的图像之间的均方误差 MSE，图像平均值 AVG，峰值信噪比 $PSNR$ 以及特征值距离差 $EigD$：

$$MSE=\frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\left[f(i,j)-f'(i,j)\right]^2,\quad AVG=\sum_{i=1}^{M}\sum_{j=1}^{N}\frac{f(i,j)}{M*N}$$

$$PSNR=10*\log\left(\frac{255^2\times M\times N}{\sum_{i=1}^{M}\sum_{j=1}^{N}\left(f(i,j)-f'(i,j)\right)^2}\right),$$

$$EigD=Sqrt\left(\sum eig(f')^2\right)-Sqrt\left(\sum eig(f^2)\right)$$

其中，$f(i,j)$是原图像，$f'(i,j)$为插值图像。M,N 是图像的尺寸大小。$Sqrt\left(\sum eig(f')^2\right)$为原始图像的特征值距离，$Sqrt\left(\sum eig(f^2)\right)$为插值图像的特征值距离。

矢量中值滤波原理

中值滤波由 *Turky* 在 1971 年提出，最初用于时间序列分析，后来被用于图像处理，并在去噪复原中取得了较好的效果。标准中值滤波算法的基本思想是将滤波窗口内的最大值和最小值均视为噪声，用滤波窗口内的中值代替窗口中心像素点的灰度，在一定程度上抑制了噪声。

矢量中值滤波的原理，设 n 各矢量的集合为：

$$\sum_{i=1}^{n}\left\|x_i-x_m\right\|\le\sum_{i=1}^{n}\left\|x_i-x_j\right\|,\ j=1,2,\cdots,n$$

其中，$x_m\in\{x_1,x_2,\cdots,x_n\}$。

由集合可见，矢量中值滤波就是在滤波窗口中寻找一个与其它像素矢量距离最近的像素，然后以此像素替代窗口的中心像素，其计算量与矢量个数的平方成正比。

反锐化掩模法

反锐化掩模(*Unsharpmasking*)先通过对图像进行低通滤波得到图像的平滑版本,再将原来的图像与它的平滑版本相减得到图像的细节,最后将细节叠加在原图像上,从而突出图像的细节,提升图像对比度。

$$g_{mask}(x,y)=f(x,y)-\bar{f}(x,y)$$
$$g(x,y)=f(x,y)+k*g_{mask}(x,y)$$

其中，$f(x,y)$为输入的图像，$\bar{f}(x,y)$为原图像的模糊图像，$g_{mask}(x,y)$为原图像的细节图像，$g(x,y)$为变换后的图像。k可以是常数，也可以是一个矩阵。当k为一个矩阵时，上式可以写作：

$$g(x,y)=f(x,y)+k(x,y)*g_{mask}(x,y)$$

即$k(x,y)$可以使一个常数矩阵，也可以是一个与坐标点位置相关的权值矩阵。当$k(x,y)$为常数时，得到的图像即为传统的反锐化掩模增强图像。

（2）模型的建立

◆ **模型Ⅳ图像“质检—去噪—后处理”模型**

为了能更加精确地对模型进行说明，这里假设图片在放大过程中收到了噪声污染，这里我们分别用高斯噪声与椒盐噪声对图片进行处理，例如：

椒盐噪声 **高斯噪声**

图 4-10 椒盐噪声与高斯噪声对比图

通过*Matlab*对图像*PSNR*值进行计算与基于人体视觉的判断，图像不符合质量检验的标准，因而需要对图像进行后期处理。

①彩色图像矢量中值滤波去噪（代码详见附录程序[2]）

第一步，计算图像像素的方向区域距离测度和此时的β角

设彩色像素$C(i,j)=\big(R(i,j),G(i,j),B(i,j)\big)$，其中$R(i,j)$，$G(i,j)$，$B(i,j)$分别是像素$C(i,j)$在*RGB*彩色空间中的红、绿和蓝颜色分量。以$C(i,j)$为中心取$(2s+1)(2s-1)$的窗口，其中$s$为大于等于 1 的整数。$L$在窗口内过$C(i,j)$做一条与垂直方向所成夹角为$\beta(0\le\beta\le\pi)$的线段并将窗口划分为$W_1$和$W_2$两个子窗口，如下图所示：

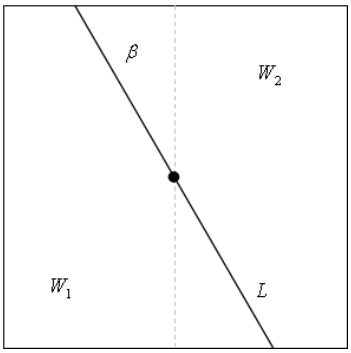


图 4-11 窗口像素示意图

第二步，使用子窗口内的像素进行矢量中值滤波

设W_1和W_2内包含N个像素，则W_1和W_2内像素的平均色彩函数值分别为：

$$\begin{aligned}C_{s\beta W_1} &= \frac{1}{N}\sum_{k=1}^{N} C_{s\beta W_{1k}} \\ &= \frac{1}{N}\sum_{k=1}^{N}\left(R_{s\beta W_{1k}}, G_{s\beta W_{1k}}, B_{s\beta W_{1k}}\right) \\ &= \left(\frac{1}{N}\sum_{k=1}^{N} R_{s\beta W_{1k}}, \frac{1}{N}\sum_{k=1}^{N} G_{s\beta W_{1k}}, \frac{1}{N}\sum_{k=1}^{N} B_{s\beta W_{1k}}\right) \\ &= \left(R_{s\beta W_1}, G_{s\beta W_1}, B_{s\beta W_1}\right)\end{aligned}$$

同理，$C_{s\beta W_2} = \frac{1}{N}\sum_{l=1}^{N} C_{s\beta W_{2l}} = \left(R_{s\beta W_2}, G_{s\beta W_2}, B_{s\beta W_2}\right)$。其中，$C_{s\beta W_{1k}}$和$C_{s\beta W_{2l}}$分别是$W_1$和$W_2$内的第$k$个和第$l$个矢量像素，$R_{s\beta W_i}$，$G_{s\beta W_i}$，$B_{s\beta W_i}$分别是$W_i\ (i=1,2)$内$N$个矢量像素$RGB$颜色分量的平均值。

定义W_1和W_2间的平均色彩距离为：

$$D_{s\beta} = \sqrt{(R_{s\beta W_1} - R_{s\beta W_2})^2 + (G_{s\beta W_1} - G_{s\beta W_2})^2 + (B_{s\beta W_1} - B_{s\beta W_2})^2}$$

当β在0和π间变化时，$D_{s\beta}$必有一个最大值，定义方向区域距离测度为：

$$D_s = \max_{\beta} D_{s\beta},\quad 0 \le \beta \le \pi$$

$D_{s\beta}$表示$C(i,j)$点垂直于β方向色彩函数值的突变程度，如果$D_{s\beta}$值较小，说明$C(i,j)$点位于平滑区，如果$D_{s\beta}$值较大，说明$C(i,j)$点可能位于边缘区。无论$C(i,j)$点位于平滑区还是边缘区，都不必使用窗口内的全部象素，只需使用此时子窗口内的象素，即参与滤波运算的象素数量减少了一半，因此所用时间也大大减少。但求取方向区域距离测度要花费一定的时间，为保证算法的快速性，可选择β角增幅为45°。

经过上述处理得到放大后的图像为：

图 4-12 滤波处理图

②基于区域分割的反锐化掩模方法图像增强

基于区域分割的反锐化掩模方法。这里根据局部方差将图像进行划分（这里分出三个区域：低频的区域、中频的边界区域、高频的噪声区域），然后对不同的区域选取不同的增强系数 $k(i,j)$。

局部方差定义为一个给定的窗口内的所有像素的方差，比如一个3×3窗口的局部方差为：

$$v_i(n,m)=\frac{1}{9}\sum_{i=n-1}^{n+1}\sum_{j=m-1}^{m+1}\left[x(i,j)-\bar{x}(n,m)\right]^2$$

根据 $v_i(\mathrm{n},\mathrm{m}$ 的大小，可以将图像分成三个区域，即：$\mathrm{v_i(n,m)}<\tau_1$，$\tau_1\le v_i(\mathrm{n,m})<\tau_2$ 和 $v_i(\mathrm{n,m})\ge\tau_2$。其中 τ_1、τ_2 $(\tau_1<\tau_2)$ 为自己设定的门限值。$\mathrm{v_i(n,m)}<\tau_1$ 的区域为低频区域，一般为图像的背景等，不希望将其增强。$\tau_1\le v_i(\mathrm{n,m})<\tau_2$ 的区域为中频区域，一般为边缘区域，希望能得到很大的增强。$v_i(n,m)\ge\tau_2$ 的区域为高频部分，希望得到较大的增强。

因此根据不同的区域设定增强系数 $K'(\mathrm{x,y})$：

$$K'(\mathrm{x,y})=\begin{cases}1 & \mathrm{v_i}(n,m)<\tau_1\\ \alpha_{dh} & \tau_1\le v_i(n,m)<\tau_2\\ \alpha_{dl}(\alpha_{dl}<\alpha_{dh}) & v_i(n,m)\ge\tau_2\end{cases}$$

对于高频细节信号，使用下列式求解：

$$g_{mask}(x,y)=(1+\lambda)f(x,y)-B(x,y)$$

$$B(x,y)=f(x,y)H(x,y)$$

其中：

$$H(\mathrm{m},\mathrm{n})=\frac{1}{40}\begin{bmatrix}2&2&2&2&2\\2&1&1&1&2\\2&1&0&1&2\\2&1&1&1&2\\2&2&2&2&2\end{bmatrix}$$

由以上的求解，可以得到基于分割的反锐化掩膜的增强图像：

$$g(\mathrm{x},\mathrm{y})=\mathrm{f}(\mathrm{x},\mathrm{y})+\mathrm{k}(\mathrm{x},\mathrm{y})*\mathrm{g}_{\mathrm{mask}}(\mathrm{x},\mathrm{y})$$

以下是基于区域分割的反锐化掩膜的算法框图：

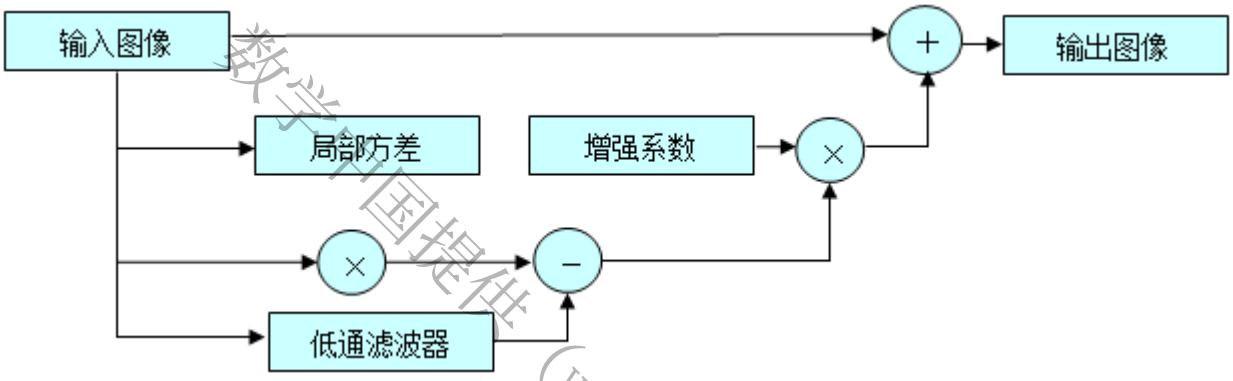


图 4-13 区域分割的反锐化掩膜的算法框图

经反锐化掩膜方法增强后图像为：

图 4-14 增强后的图像

以下是对不同滤波处理后图像的评价标准：

PSNR 对比图

	中值滤波处理后	反锐化掩模法再次处理后
Salt and Pepper Noise	24.5669	35.1547
Gaussian Noise	22.8771	28.8974

MSE 对比图

	中值滤波处理后	反锐化掩模法再次处理后
Salt and Pepper Noise	0.0043	0.0022
Gaussian Noise	0.0023	0.0019

§5 模型的误差分析

1、考虑两部分放大后图像组合的影响：为了精确放大目标图像，提高图像在人眼当中的质量，我们采取对图像的分离，然后再将分别插值扩大的部分组合，这种方法可能会造成像素点各颜色分量的重叠，使放大后的图像缺乏一定的连贯性或部分扭曲。

2、考虑图像本身质量的影响：图像本身的质量会对边缘提取，图像分片造成一些次要影响，例如造成边界线模糊不清，和产生重复的像素点导致插值的保真程度不高等。

§6 模型的评价

一、模型的优点：

1.利用数学软件 *Matlab* 对数据进行处理并做出各种图表，简便，直观，快捷，准确。

2.本文建立的模型与实际联系紧密，充分考虑基于人眼对于边缘部分的敏感程度将目标图像分解为边缘区域与非边缘区域，并用不同的方式进行插值以便达到图像放大而不失真的结果。

3.在对非边缘区域进行插值时，为了避免产生区域间的边界模糊对图像处理带来的问题，使用了一种分片连续算法达到更专业、精确的处理效果。

4.对边缘部分的运用了 *Thiele - Newton* 插值算法，能够精细地描述边缘图像的渐变性与收敛性。

5.考虑到噪点对图像的影响，运用中值滤波处理彩色图像噪点，并用改进后的区域反锐化掩模法对放大后的图像进行增强，并使用信噪比峰值等参量来描述图像的质量。

二、模型的缺点：

1.模型中为了追求图像放大后的保真度，使用了多种精细的算法对图像进行处理，使计算过于繁琐、庞大，不能做到快速处理图像。

2.模型中对图像边缘部分和非边缘部分采用了完全不同的插值算法，并完全

独立地进行插值，可能导致边缘部分与非边缘部分脱节，不能很好地表示出放大后的图像。

§7 模型的改进与推广

一、改进

1.模型的初步改进

八方向*Sobel*算子的改进

传统的*Sobel*图像边缘检测方法，是在图像空间利用两个方向，垂直边缘和水平边缘的检测完成的。不足是传统*Sobel*算子得到的边缘信息并不完整。于是，在传统*Sobel*边缘检测方法的两个模板的基础上，新增加了六个方向的模板，即45°，135°，180°，225°，270°，315°。

如图：

图7-1 八方向*Sobel*算子示意图

这样可以更加有效地检测图像多个方向的边缘，使边缘信息更加完整。

2.模型的进一步改进

基于*Bézier*插值曲面的图像放大改进

在对非边缘区域进行分片处理时，可以考虑采用*Bézier*曲面对分片的曲面进行插值，*Bézier*曲面是由一组控制顶点生成的曲面。它可以用于在一组数据点之间进行插值，以便获得平滑过渡。本模型可以考虑在分片的各区域使用*Bézier*曲面插值。

*Bézier*曲面的定义如下：

在空间给定$(n+1)\times(m+1)$个点$P_{i,j}\left(i=0,1,\cdots,n;\mathrm{j}=0,1,\cdots,\mathrm{m}\right)$，称下列张量积形式的参数曲面为$n\times m$次的*Bézier*曲面：

$$P(u,v)=\sum_{i=0}^{n}\sum_{j=0}^{m}P_{i,j}B_i^u(u)B_j^m(v),\ \ (u,v\in[0,1])$$

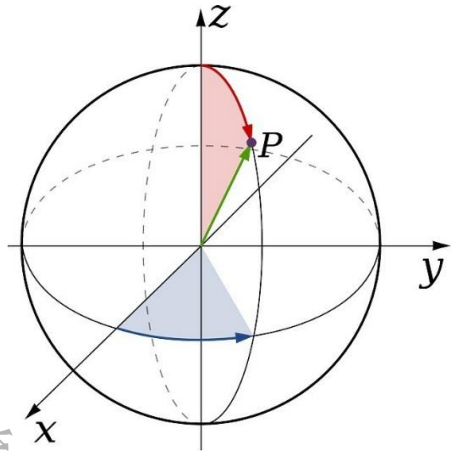
$$B_i^m(t)=C_m^i t^i(1-t)^{m-i}$$

其中$P_{i,j}$是$P(u,v)$的控制顶点，由两组多边形$P_{i,0}P_{i,1}\ldots,P_{i,m}\left(i=0,1,\ldots,n\right)$和$P_{0,j}P_{1,j}\ldots,P_{n,j}\left(j=0,1,\ldots,m\right)$组成的网称为$P(u,v)$的控制网格，记为$\{P_{i,j}\}$可以认为，控制网格$\{P_{i,j}\}$是$P(u,v)$的大致形状的勾画，$P(u,v)$是对$\{P_{i,j}\}$的逼近。

二、模型的推广

1、三维图像的放大模型

可以考虑将模型推广至三维图像的体积放大，在 x, y 轴之上增加 z 轴，把二维图像 扩张为 ，将离散的像素点分割成数个平面，每个平面分别进行二维图像扩大插值算法，以实现三维图像在空间中的不失真放大。



7-2 模型的三维推广

2、二维图像的缩小算法

基于放大算法的逆过程，将多个像素点合并为一个像素点，实现图像在缩小过程中的保真性。

参考文献

[1] 姚敏.数字图像处理［M］.北京：机械工业出版社，2008，226-229.

[2] Xin fu Li，Jiao m in Liu. Edge detection on arc im-age of low volt-age apparatus［A］.IEEE Proceed-ings of the Second International Conference on Ma-chine Learning and Cybernetics［C］.Xi'an，2003：2921-2924.

[3] 袁春兰，熊宗龙，周雪花，等.基于 Sobel 算子的图像边缘检测研究［J］.激光与红外，2009，39（1）：60-62.

[4] Du rand C X, Fagu y D. Rat ional zoom of bit maps using B-sp line int er polat ion in comput er ized 2-D an imat ion. Computer Graph ics Forum,1990,9(1):27-37.

[5] Jie qing Tan and Yi Fang,Newton-Thiele ’ s rationalinter-polantsss.NumerialAlgo-rithms24(2000)141-157.

[6] 唐荣锡,汪嘉业,彭群生.计算机图形学教程.北京:科学出版社,1994.

附录

表[1] 原始图像的像素点坐标及其对应的 *RGB* 分量（部分）

1. R 值分量

2	2	2	2	2	2	3	3	2	2	3	3	3	3	3	3	4	4	3	3	3	3	2	2	3	3	3	4	4	4
2	2	2	3	3	3	3	3	3	3	3	4	4	4	4	3	4	4	4	3	3	3	3	3	3	4	4	4	4	5
3	3	3	3	3	3	3	4	4	4	4	5	5	5	5	4	2	2	2	2	2	2	2	2	2	2	2	3	3	3
4	4	4	4	4	4	4	4	4	5	5	5	5	5	5	3	2	2	3	3	3	3	4	4	3	3	3	3	4	4
2	2	3	3	3	3	3	3	3	3	3	4	4	4	3	3	3	2	3	3	3	3	4	4	3	3	3	3	3	4
3	3	3	3	4	4	4	4	3	3	4	4	4	4	4	3	4	4	4	4	4	4	4	4	3	3	4	4	4	4
3	3	3	3	3	3	3	3	3	3	4	4	4	4	4	4	5	4	4	4	4	3	3	3	3	3	3	3	4	4
3	3	3	3	4	4	4	4	4	4	5	5	5	5	5	4	5	5	4	4	4	4	3	3	3	3	4	4	4	4
4	4	4	4	3	3	3	3	5	5	4	4	4	5	5	6	4	2	2	2	2	2	3	3	3	3	3	3	3	3
5	5	5	4	4	4	4	4	5	5	4	4	4	5	5	4	3	3	3	3	3	3	3	4	4	4	4	3	4	4
4	4	3	3	3	3	3	3	3	3	3	3	3	3	3	4	4	3	3	3	3	3	4	4	4	4	4	3	4	4
4	4	4	4	4	4	4	4	3	3	3	3	3	4	4	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4
3	3	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4	2	2	2	2	3	3	3	3	2	2	2	2	3
3	3	4	4	4	4	4	4	4	4	4	5	5	5	4	4	3	3	3	3	3	3	3	3	3	3	3	3	3	3
4	4	4	5	5	5	5	5	4	5	5	6	6	6	5	5	4	4	4	4	4	4	4	4	4	4	4	4	4	4
3	3	3	3	4	4	4	4	3	3	4	5	5	4	4	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4
3	3	3	3	3	3	3	3	5	4	3	3	3	3	3	4	4	4	5	4	5	5	6	5	5	4	5	4	5	3
3	3	3	3	2	2	2	2	4	4	3	3	3	3	3	4	3	3	3	3	4	4	4	4	4	4	4	4	4	4
3	3	3	3	3	3	3	3	5	5	4	4	4	4	4	4	3	3	4	4	4	4	5	5	5	5	5	5	5	4
4	4	4	4	4	4	4	4	5	5	5	5	5	5	5	5	5	5	5	6	6	6	6	7	5	5	5	5	5	5
4	4	4	5	5	5	5	5	5	5	5	5	5	5	5	5	4	4	4	5	5	5	5	5	4	4	4	4	4	4
5	5	5	5	6	6	6	6	5	5	6	6	6	6	6	4	4	4	4	4	5	5	5	5	5	5	5	5	5	5
3	3	4	4	4	4	5	5	4	4	5	5	5	5	5	4	4	3	4	4	4	4	5	5	4	4	4	4	5	5
4	4	4	4	5	5	5	5	4	5	6	6	6	6	6	5	5	5	5	5	5	6	6	6	4	4	5	5	5	5
4	3	3	4	4	4	5	5	5	5	5	5	5	5	6	6	6	6	6	6	6	7	7	7	6	6	6	7	7	
4	4	4	4	4	5	5	6	5	5	5	5	5	5	6	6	6	6	6	6	7	7	7	7	8	7	6	6	7	7
3	3	3	3	3	3	4	4	4	4	4	4	4	4	4	5	5	5	5	5	5	5	5	5	7	6	5	4	5	5
4	4	4	4	4	4	5	5	5	4	4	4	4	5	5	5	5	5	5	6	6	6	6	6	7	7	6	5	5	5
5	5	5	5	5	5	5	6	5	5	5	5	5	5	6	6	5	5	5	5	5	6	6	6	7	7	6	5	5	5
6	6	6	6	6	6	6	6	6	6	6	5	6	6	6	7	6	6	6	6	6	6	6	7	7	7	7	6	6	

2. G 值分量

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	2	2	1	1	2	2	2	2	2	2	3	3	2	2	2	2	1	1	2	2	2	3	3	3
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	2	2	2	2	2	2	2	2	3	3	3	3	2	3	3	3	2	2	2	2	2	2	3	3	3	3	4
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	3	3	3	3	4	4	4	4	3	3	3	3	3	3	3	3	3	3	3	3	4	4	4
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	3	3	3	3	3	3	3	3	4	4	4	4	4	4	4	3	3	4	4	4	4	5	5	4	4	4	4	5	5
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	3	4	4	4	4	4	4	4	4	4	5	5	5	4	4	4	5	6	6	6	6	7	7	6	6	6	6	6	7
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	4	4	4	5	5	5	5	4	4	5	5	5	5	5	6	7	7	7	7	7	7	7	7	6	6	7	7	7	7
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	6	6	6	6	6	6	6	6	6	7	7	7	7	7	7	8	8	8	8	8	7	7	7	7	7	7	7	8	8
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	6	6	6	7	7	7	7	7	7	8	8	8	8	8	8	9	9	8	8	8	8	7	7	7	7	8	8	8	8
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	1	1	1	1	1	1	2	2	2	2	2	2	2	2
8	8	8	8	7	7	7	7	9	9	8	8	8	9	9	0	8	9	9	9	9	9	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
9	9	9	8	8	8	8	8	9	9	8	8	8	9	9	1	0	0	0	0	0	0	0	1	1	1	1	0	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	2	2	2	2	2	3	3	3	3	3	2	3	3
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2	2	3	3	3	3	3	4	4	4	4	5	5	5	5	4	4	4	4	4	5
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
2	2	3	3	3	3	3	3	3	3	3	4	4	4	3	3	5	5	5	5	5	5	5	5	5	5	5	5	5	5
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	4	4	4	4	4	3	4	4	5	5	5	4	4	6	6	6	6	6	6	6	6	6	6	6	6	6	6
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	6	6	6	6	5	5	6	7	7	6	6	6	7	8	7	8	7	8	7	8	7	8	7	8	7	8
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	3	2	3	2	3	3	3	2	3	2	3	2	2
7	7	7	7	7	7	7	7	9	8	7	7	7	7	7	8	8	0	9	0	9	1	0	1	9	0	9	0	9	9
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3
8	8	8	8	7	7	7	7	9	9	8	8	8	8	8	9	9	9	9	9	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	3	3	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3
8	8	8	8	8	8	8	8	0	0	9	9	9	9	9	9	9	9	0	0	0	0	1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
9	9	9	9	9	9	9	9	0	0	0	0	0	0	0	1	1	1	1	2	2	2	2	3	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	2	3	3	4	4	4	4	4	3	3	3	3	3	3
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
1	1	1	1	2	2	2	2	1	1	2	2	2	2	2	3	3	3	3	3	4	4	4	4	4	4	4	4	4	4
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
2	2	3	3	3	3	4	4	3	3	4	4	4	4	4	3	3	4	5	5	5	5	6	6	5	5	5	5	6	6
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	4	4	4	4	3	4	5	5	5	5	5	4	6	6	6	6	6	7	7	7	5	5	6	6	6	6
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
5	4	4	5	5	5	6	6	6	6	6	6	6	6	7	7	7	7	7	7	7	8	8	8	7	7	7	8	8	8
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
5	5	5	5	5	6	6	7	6	6	6	6	6	6	7	7	7	7	7	7	8	8	8	8	9	8	7	7	8	8
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	4	4	3	3	3	3	3
6	6	6	6	6	6	7	7	7	7	7	7	7	7	7	8	9	9	9	9	9	9	9	9	1	0	9	8	9	9
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4	3	3	3
7	7	7	7	7	7	8	8	8	7	7	7	7	8	8	9	9	9	9	0	0	0	0	0	1	1	0	9	9	9
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	3	3	3	3	3	3	4	3	3	3	3	3	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
9	9	9	9	9	9	9	0	9	9	9	9	9	9	0	0	0	1	1	1	1	2	2	2	3	3	2	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	4	4	4	4	4	4	4	4	4	4	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
0	0	0	0	0	0	0	0	0	0	9	0	0	0	1	2	2	2	2	2	2	2	2	3	3	3	3	2	2	

3.B 值分量

2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
2	2	2	2	2	2	3	3	2	2	3	3	3	3	3	4	4	3	3	3	3	2	2	3	3	3	4	4	4	4
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
2	2	2	3	3	3	3	3	3	3	3	4	4	4	4	3	4	4	4	3	3	3	3	3	3	4	4	4	4	5
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
3	3	3	3	3	3	3	4	4	4	4	5	5	5	5	4	4	4	4	4	4	4	4	4	4	4	5	5	5	
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
4	4	4	4	4	4	4	4	4	5	5	5	5	5	5	5	4	4	5	5	5	5	6	6	5	5	5	5	6	6
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
6	6	7	7	7	7	7	7	7	7	7	8	8	8	7	7	7	7	8	8	8	9	9	8	8	8	8	8	9	
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
7	7	7	7	8	8	8	8	7	7	8	8	8	8	8	8	9	9	9	9	9	9	9	9	8	8	9	9	9	9
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	5	5	5	5	5	4	4	4	4	4	4	5	5
8	8	8	8	8	8	8	8	8	8	9	9	9	9	9	9	0	0	0	0	0	9	9	9	9	9	9	9	0	0
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
4	4	4	4	4	4	4	4	4	4	5	5	5	5	5	5	5	5	5	5	5	4	4	4	4	5	5	5	5	
8	8	8	8	9	9	9	9	9	9	0	0	0	0	0	0	1	1	0	0	0	0	9	9	9	9	0	0	0	0
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	4	4	4	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
0	0	0	0	9	9	9	9	1	1	0	0	0	1	1	2	0	0	0	0	0	0	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
1	1	1	0	0	0	0	0	1	1	0	0	0	1	1	2	1	1	1	1	1	1	2	2	2	2	1	2	2	
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	3	3	3	3	3	2	3	3
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
2	2	2	2	2	2	2	2	1	1	1	1	1	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3

2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	2	2	2	2	1	1	1	1	2
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
0	0	1	1	1	1	1	1	1	1	1	2	2	2	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
1	1	1	2	2	2	2	2	1	2	2	3	3	3	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
2	2	2	2	3	3	3	3	2	2	3	4	4	3	3	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
3	3	3	3	3	3	3	3	5	4	3	3	3	3	3	4	2	3	3	3	3	4	4	4	3	3	3	3	3	2
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
4	4	4	4	3	3	3	3	5	5	4	4	4	4	4	5	2	2	2	2	3	3	3	3	3	3	3	3	3	3
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
4	4	4	4	4	4	4	4	5	5	5	5	5	5	5	5	2	2	3	3	3	3	4	4	4	4	4	4	4	3
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	4	4	4	4	5	5	5	5	5	4	4	4	4	4	4
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
3	3	3	4	4	4	4	4	4	4	4	4	4	4	4	5	3	3	4	4	4	4	4	3	3	3	3	3	3	3
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
4	4	4	4	5	5	5	5	4	4	5	5	5	5	5	3	3	3	3	3	4	4	4	4	4	4	4	4	4	4
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
2	2	3	3	3	3	4	4	3	3	4	4	4	4	4	3	3	2	3	3	3	3	4	4	3	3	3	3	4	4
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
3	3	3	3	4	4	4	4	3	4	5	5	5	5	5	4	4	4	4	4	4	5	5	5	3	3	4	4	4	4
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
3	2	2	3	3	3	4	4	4	4	4	4	4	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
3	3	3	3	4	4	5	4	4	4	4	4	4	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
4	4	4	4	4	4	5	5	5	5	5	5	5	5	5	5	4	4	4	4	4	4	4	4	5	5	4	3	4	4
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	4	4	4	4	5	5	5	5	5	5	5	4	4	4	4
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
4	4	4	4	4	4	4	5	4	4	4	4	4	4	5	5	5	3	3	3	3	4	4	4	5	5	4	3	3	3
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	4	5	5	5	5	4	4	4	4	4	4	4	4	5	5	5	5	4	4

表[2] 放大图像的像素点坐标及其对应的 *RGB* 分量（部分）

1.R 值分量

2	2	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
9	9	9	9	9	0	1	1	1	1	1	0	0	0	0	0	0	1	1	1	0	0	1	1	2	2	1	1	1	2
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
2	2	2	2	3	3	4	4	4	4	4	3	3	3	3	3	3	4	4	4	3	3	4	4	6	5	5	5	5	6
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	5	5	5	5	5	5	5	5	4	4	5	5	6	5	5	5	5	6	6	6	6	6	6	7
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
6	6	6	5	6	6	6	6	7	7	7	6	6	6	6	6	6	7	7	7	6	6	7	7	7	7	7	8	8	8
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	4	4	4
8	7	7	7	7	7	7	7	8	8	8	8	7	7	7	7	7	8	8	8	8	7	8	8	8	9	9	0	0	0
3	3	4	4	4	3	3	3	3	3	3	4	4	4	3	3	3	3	3	4	4	4	3	3	4	4	4	4	4	4
7	8	0	0	0	9	9	9	8	9	9	0	0	0	9	9	9	9	9	0	0	0	9	8	0	0	1	2	2	2
3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
8	0	1	1	1	0	0	0	1	1	1	2	2	2	2	1	1	0	0	1	2	2	1	0	2	2	2	3	3	3
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
0	1	2	3	2	2	1	2	3	3	3	3	3	3	3	3	3	2	2	3	3	3	3	2	4	4	4	4	4	4
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
2	2	3	3	3	3	3	3	3	3	3	2	2	3	4	4	5	4	4	4	5	5	4	4	6	6	5	5	5	5
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
3	3	4	4	4	3	4	4	4	4	4	3	3	4	5	5	6	5	5	5	6	6	6	5	7	6	6	6	6	6
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
4	5	5	5	5	5	5	6	6	6	6	6	6	6	7	7	7	6	6	6	6	7	7	6	7	7	7	7	7	7
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	5	4	4	4	4	4	4	4	4	4	4	5	5	5
7	7	8	7	7	7	8	8	8	8	9	9	9	9	9	9	0	9	8	8	8	9	9	9	8	9	9	0	0	0
4	4	4	4	4	4	4	5	4	4	4	4	4	4	4	4	5	5	4	4	4	5	5	5	4	4	5	5	5	5
8	8	9	8	8	8	9	0	8	8	9	9	9	9	9	8	1	0	9	9	9	0	0	0	8	9	0	1	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
1	2	2	2	2	1	1	1	2	2	2	2	2	3	3	4	2	2	3	3	4	4	4	4	4	4	5	6	5	5
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
3	3	4	3	3	2	3	2	4	3	4	4	5	4	5	4	5	4	5	5	6	5	6	5	6	5	7	7	8	6
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	6	5	6	5	6	5
6	5	7	5	7	4	6	4	7	6	8	7	9	7	9	7	9	7	9	7	9	7	9	7	0	8	1	9	1	9
5	5	5	5	5	5	5	5	5	5	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
7	8	9	9	9	8	9	9	8	9	0	1	1	1	0	0	1	1	1	1	1	1	1	1	2	2	2	3	2	
5	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
9	0	1	2	2	2	2	3	0	1	2	3	3	3	2	2	4	4	3	3	3	3	3	3	4	4	4	5	5	5
6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
1	2	3	3	3	4	4	5	2	3	3	4	4	4	4	4	5	5	5	5	5	5	5	5	5	5	6	6	6	6
6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
4	5	6	6	6	5	6	6	5	6	6	6	6	7	7	7	7	7	7	8	8	8	8	8	8	8	8	9	9	
6	6	6	6	6	6	6	6	6	6	6	6	6	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
8	9	9	9	8	8	8	8	9	9	9	9	9	0	0	0	0	0	0	1	1	1	2	2	2	1	1	1	2	2
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
0	0	0	0	1	1	2	2	3	2	1	2	3	4	3	1	2	2	3	4	4	4	3	3	4	4	5	5	6	7
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
4	4	4	4	4	4	5	5	5	5	6	6	6	6	6	6	7	7	7	8	7	7	7	6	7	7	7	8	8	8
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	8	8	8	8	7	7	7	7	8	8	8	8	8	8
6	6	5	5	6	6	7	7	6	8	9	8	7	6	8	9	0	0	0	0	9	9	9	9	0	0	0	0	0	0
7	7	7	7	7	7	7	7	7	7	8	8	7	7	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
8	7	7	7	7	8	8	8	8	9	0	0	9	9	0	1	2	2	1	1	1	1	1	2	2	2	2	2	2	3
8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
0	0	0	0	0	0	0	1	2	1	1	1	3	3	3	2	4	3	3	4	4	4	5	5	4	4	4	5	5	6
8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8

3	2	3	1	3	2	3	2	5	3	3	3	6	6	6	5	5	5	6	6	7	7	8	8	5	5	6	6	7	8
8	8	8	8	8	8	8	8	8	8	8	8	8	8	9	9	8	8	9	9	9	9	9	9	9	9	9	9	9	9
7	5	7	4	6	5	7	5	8	6	8	6	9	8	0	0	9	9	0	1	1	1	1	0	0	0	0	0	1	1
8	8	8	8	8	8	8	8	8	8	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
9	8	9	8	9	8	9	8	9	9	2	1	1	0	2	3	1	2	5	3	5	3	4	1	6	4	6	4	6	4
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
4	2	3	1	3	1	3	1	2	1	5	4	6	4	6	6	7	8	9	7	7	5	7	6	7	6	8	7	8	7

2.G值分量

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	4	4	4	4	4	4	4	4	4	4	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	1	2	2	2	2	2	2	2	2	3	3	3	3	2	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
1	1	0	0	0	0	1	1	0	0	0	0	0	0	1	1	2	2	2	2	3	3	3	3	3	4	4	4	3	3
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
1	2	1	2	1	2	1	2	1	1	0	1	0	2	1	2	2	2	3	3	3	3	3	4	3	4	5	5	4	3
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
3	3	2	2	2	3	3	4	3	2	2	2	2	3	3	3	4	4	4	5	6	5	5	6	5	5	5	5	5	5
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
4	4	3	3	3	4	4	5	3	3	3	3	3	3	4	4	4	4	5	5	6	6	6	6	7	7	6	6	5	6
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
4	4	4	4	4	4	5	5	4	4	4	4	4	5	5	5	5	5	6	7	7	6	7	7	8	8	7	7	8	8
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
4	4	4	4	4	4	4	4	6	6	5	5	6	6	6	6	6	7	7	7	8	8	8	9	9	8	8	8	8	8
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	5	5	5	4	4	5	5
6	6	6	6	6	6	6	6	8	8	8	8	8	8	8	9	9	9	9	9	9	9	0	0	0	9	9	0	0	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5
8	8	8	9	9	9	9	8	9	8	8	8	8	9	9	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	1	0	2	2	2	2	2	2	1	2	2	2	2	2	3
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
9	1	0	2	1	2	0	1	0	1	0	1	0	1	1	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
3	3	2	3	3	4	3	3	3	4	2	3	2	3	3	4	2	2	4	4	4	4	4	4	4	3	3	4	4	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
4	4	3	4	4	4	4	3	4	4	4	4	4	4	4	4	3	4	4	4	5	5	4	4	4	4	5	5	5	5
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	4	4	4	5	5	5	4	4	5	5	6	6	5	5	4	4	4	5	6	6	6	6	6	6	7	7	8	7	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
6	5	5	5	6	6	6	5	5	5	6	6	6	6	5	5	5	5	7	7	7	7	7	7	6	7	8	9	9	8
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	6	5	5
8	7	7	7	8	8	7	7	6	7	7	8	8	7	7	6	7	7	8	8	9	9	9	9	8	9	9	0	9	9

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	6	6	6	6	5	6	6	6	6	5
9	8	8	8	9	9	8	8	8	8	8	9	9	8	8	8	8	8	9	9	0	0	0	0	9	0	0	1	0	9
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	5	5	5	5	5	5	5	6	6	6	6	6	6	6	6	5	5	6	6	6	6	6	6	6	6	6	6	6	6
9	9	8	9	9	9	9	8	0	0	0	0	0	0	0	0	9	9	0	1	1	1	1	1	2	2	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	5	5	5	6	6	5	5	6	6	6	6	6	6	6	6	5	6	6	6	6	6	6	6	6	6	6	6	6	6
0	9	9	9	0	0	9	9	1	1	1	1	1	1	1	1	9	0	1	1	1	1	2	2	3	3	2	2	2	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
2	1	1	2	2	3	2	2	1	1	1	2	3	3	2	2	2	3	2	2	1	3	3	4	4	4	3	4	3	4
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
2	2	2	2	3	3	3	2	2	2	2	2	3	3	4	3	3	3	3	3	3	4	4	4	5	5	4	5	5	5
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
4	3	3	3	4	4	4	3	4	4	4	5	5	5	5	4	4	4	5	5	5	5	5	5	6	6	5	6	7	7
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
5	5	4	4	5	5	5	5	6	6	6	6	7	7	6	5	5	6	6	7	7	7	6	6	7	7	7	7	8	9
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	7
7	6	6	6	6	7	6	6	7	7	6	7	7	7	6	5	6	6	7	8	8	7	7	7	8	8	7	8	9	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	7	6	7	6	7	6	7	6	7	7	7
8	7	7	7	8	8	7	7	9	8	8	9	8	9	7	7	8	9	8	0	9	0	9	0	9	0	9	0	0	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	6	6	6	6	6	6	6	7	7	6	6	6	6	6	6	7	6	6	6	6	7	7	7	7	7	7	7	7	7
9	9	8	8	9	9	8	7	0	0	9	9	9	9	8	7	0	9	9	9	9	0	0	0	1	0	0	0	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	6	6	6	6	6	6	6	7	7	7	7	7	6	6	6	7	7	7	7	7	7	7	7	7	7	7	7	7	7
0	9	9	9	9	9	9	8	1	0	0	0	0	9	8	8	1	1	0	1	0	1	1	2	1	2	0	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
0	1	0	0	0	0	0	0	2	2	1	1	1	1	1	1	1	2	2	2	1	1	2	2	2	3	3	4	4	4
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
2	1	1	1	1	1	2	2	3	3	3	3	2	2	2	2	2	3	3	3	3	2	3	3	2	2	3	3	3	4
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
3	3	3	3	3	3	4	4	4	4	4	5	4	5	5	5	5	6	6	6	5	5	5	6	5	5	5	5	5	6

3.B 值分量

2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	4	4	4	4	5	5	5	5	5	5	5	5	5
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	4	5	4	5	4	5	4	5	3	5	3	5	4	5	4	4	2	5	3	5	3	5	3	5	4	5	5	5	3
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	3	4	2	4	3	5	4	5	2	4	2	4	3	5	3	4	2	4	3	5	2	4	2	4	2	4	2	5	3
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5

4	4	3	3	3	4	4	5	3	3	3	3	3	3	4	4	2	2	3	3	3	3	2	2	3	3	3	3	3	4
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
2	2	2	2	2	2	3	3	2	2	2	2	2	3	3	3	3	3	3	4	4	3	3	3	4	4	4	4	5	5
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
2	2	2	2	2	2	2	2	4	4	3	3	4	4	4	4	4	4	4	4	4	4	4	4	5	5	5	5	5	5
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
3	3	3	3	3	3	3	3	5	5	5	5	5	5	5	5	4	4	4	4	4	4	4	4	4	4	4	4	5	5
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	4	4	4	4	4	4	3	4	4	4	4	4	5	5
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
4	3	5	4	5	4	5	3	5	3	5	3	5	3	5	4	4	4	4	4	4	4	4	4	5	5	5	5	5	5
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	2	4	2	5	3	5	2	5	3	4	2	4	2	5	3	2	2	3	3	3	3	3	3	3	3	2	2	3	3
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
3	3	2	3	3	3	3	2	3	3	3	3	3	3	3	3	3	3	3	3	3	4	4	3	3	3	3	4	4	4
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
2	1	1	1	2	2	2	1	1	2	2	3	3	2	2	1	1	1	2	2	2	2	2	2	2	2	3	3	4	3
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
3	2	2	2	3	3	3	2	2	2	3	3	3	3	2	2	2	2	3	3	3	3	3	3	2	3	4	5	5	4
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
4	3	3	3	4	4	3	3	2	3	3	4	4	3	3	2	2	2	3	3	3	3	3	3	2	3	3	4	4	4
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	4	4	4	5	5	4	4	4	4	4	5	5	4	4	3	3	3	4	4	4	4	4	4	3	4	4	5	5	4
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
4	4	3	4	4	4	4	3	5	5	5	5	5	5	5	5	4	2	2	3	3	3	2	2	3	3	3	3	3	3
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	4	4	4	5	5	4	2	5	4	5	4	5	4	4	4	2	0	3	1	3	1	3	1	4	2	4	2	4	2
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	4	5	5	5	5	5	5	5	5	5	5
4	3	3	4	4	5	4	2	3	1	3	2	5	3	2	2	2	0	2	9	1	0	3	1	4	1	3	1	3	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
4	4	4	4	5	5	3	2	2	2	2	2	3	3	1	0	0	0	0	0	0	1	1	1	2	2	1	2	2	2
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5

4	3	3	3	4	4	4	3	1	1	1	2	2	2	2	1	1	1	2	2	2	2	2	2	3	3	2	3	4	4
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	4	4	5	5	5	2	3	3	3	3	4	4	3	2	2	3	3	4	4	4	3	3	4	4	4	4	5	5
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	4	4	4	4	5	4	2	3	3	2	3	3	3	2	1	2	2	3	4	4	3	3	3	4	4	3	4	5	5
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	3	5	4	3	3	5	4	4	3	4	3	3	1	4	3	4	4	5	4	5	4	5	4	5	4	5	5
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	4	3	5	4	3	2	5	5	4	2	4	2	3	0	5	2	4	2	4	3	5	3	5	3	5	3	5	4
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	4	4	4	4	4	4	1	5	3	3	3	3	2	1	8	4	1	3	1	3	1	4	2	4	2	3	1	4	2
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	4	3	3	3	3	3	1	5	3	2	2	2	2	2	0	2	1	3	1	2	0	3	1	3	2	4	3	5	3
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	2	4	2	4	2	3	3	4	4	4	2	3	1	1	1	1	2	2	2	2	1	2	2	1	1	2	2	2	3
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
4	2	4	2	4	2	3	3	3	3	3	2	3	2	2	2	2	3	3	3	2	2	2	3	2	2	2	2	2	3
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
3	3	3	3	3	4	5	5	5	4	4	3	3	3	3	3	3	4	4	4	3	3	4	4	5	5	4	4	4	5

程序[1]：改进*Sobel*算子C^{++}源代码

```
// Sobel.cpp : 定义控制台应用程序的入口点。
//

#include "stdafx.h"
#include "cv.h"
#include "highgui.h"

int _tmain(int argc, _TCHAR* argv[])
{
// TODO: Add your command handler code here
//定义的变量
IplImage* pImage= NULL; // 声明 IplImage 变量
IplImage* pImgSobelgray= NULL;// 声明 IplImage 变量，用于灰度图像 Sobel 变换
IplImage* pImg8u= NULL;// 声明 IplImage 变量，用于图像格式转换
IplImage* pImg8uSmooth= NULL;// 声明 IplImage 变量，用于存储平滑后的图像
IplImage* pImgColor= NULL;// 声明 IplImage 变量，用于 Sobel 变换
```

```
IplImage* pImgSobelcolor= NULL;// 声明 IplImage 变量，用于彩色图像 Sobel 变换
IplImage* pImgPlanes[3] = { 0, 0, 0 };

IplImage* pImage = cvLoadImage ( "barbara.png", CV_LOAD_IMAGE_GRAYSCALE );
cvNamedWindow ( "Original Image " , 1 );
cvShowImage ( " Original Image " , img );
//将已读入系统的图像复制一份
//pImage=cvCloneImage( img );

//建立和原始图像一样图像内存区，图像元素的位深度设为 IPL_DEPTH_8U
//即无符号 8 位整型
pImg8u = cvCreateImage(cvGetSize(pImage),IPL_DEPTH_8U, 1);
pImg8uSmooth = cvCreateImage(cvGetSize(pImage),IPL_DEPTH_8U, 1);

//对灰度图像进行 Sobel 变换
//将彩色图像转换为灰度图像
cvCvtColor(pImage, pImg8u, CV_BGR2GRAY);

//对图像进行高斯滤波
cvSmooth( pImg8u, pImg8uSmooth,CV_GAUSSIAN,3,0,0);

//建立一新图像内存区，图像元素的位深度设为 IPL_DEPTH_16S 有符号 16 位整型
//因为 cvSobel 函数要求目标图像必须是 16-bit 图像
pImgSobelgray = cvCreateImage(cvGetSize(pImage),IPL_DEPTH_16S, 1);

//计算一阶 x 方向的图像差分，可根据需要设置参数
cvSobel(pImg8uSmooth, pImgSobelgray,0,1,3);

//将图像格式再转换回来，用于显示
    cvConvertScaleAbs(pImgSobelgray,pImg8u,1,0 ) ;
//创建窗口，显示图像
cvvNamedWindow( "Sobel gray Image", 1 );
    cvvShowImage( "Sobel gray Image", pImg8u  );
//对彩色图像进行 Sobel 变换
//建立 3 个图像内存区，分别存储图像 3 个通道，图像元素的位深度设为 IPL_DEPTH_8U
int i;
for( i = 0; i < 3; i++ )
pImgPlanes[i] = cvCreateImage( cvSize(pImage ->width, pImage ->height), 8, 1 );
//建立一新图像内存区，图像元素的位深度设为 IPL_DEPTH_16S 有符号 16 位整型
pImgSobelcolor = cvCreateImage( cvSize(pImage ->width, pImage ->height),
IPL_DEPTH_16S, 1 );
//要求输出图像是 16 位有符号的
pImgColor = cvCreateImage( cvSize(pImage ->width, pImage ->height), 8, 3 );
//将彩色图像分成 3 个单通道图像
```

```
cvCvtPixToPlane(pImage, pImgPlanes[0], pImgPlanes[1], pImgPlanes[2], 0 );
for( i = 0; i < 3; i++ )
{
//分别对每通道图像进行 Sobel 变换
cvSobel( pImgPlanes[i], pImgSobelcolor,0,1,3 );
//转化为 8 位的图像
cvConvertScaleAbs(pImgSobelcolor, pImgPlanes[i], 1, 0 );
}
//将各通道图像进行合并
cvCvtPlaneToPix( pImgPlanes[0], pImgPlanes[1], pImgPlanes[2], 0, pImgColor);
//创建窗口，显示图像
cvvNamedWindow( "Sobel color Image", 1 );
cvvShowImage( "Sobel color Image", pImgColor);
//等待按键
cvWaitKey(0);
//销毁窗口
cvDestroyWindow( " Sobel gray Image " );
cvDestroyWindow( " Sobel color Image " );
//将程序开始定义的变量释放
cvReleaseImage( & pImage);
cvReleaseImage( & pImgSobelgray);
cvReleaseImage( & pImgSobelcolor);
cvReleaseImage( & pImg8u);
cvReleaseImage( & pImg8uSmooth);
return 0;
}
```

程序[2]：中值滤波去噪代码

```
I=imread('写入图片存放的位置，后缀.图像格式');
I1=rgb2gray(I);
I2=medfilt2(I1,[m,n]);
%%%I2 就是中值滤波后的图像，medfilt2 是 matlab 中中值滤波函数，直接调用即可，m 和 n
是选取的平滑窗口，一般为 3*3，可以进行调整
要分离的话，可以这样做：
M=imread('D:\ebook\lena.bmp');  %读取 MATLAB 中的名为 cameraman 的图像
subplot(2,2,1)
imshow(M)  %显示原始图像
title('original')
P1=imnoise(M,'gaussian',0.02);  %加入高斯躁声
subplot(2,2,2)
imshow(P1)  %加入高斯躁声后显示图像
title('gaussian  noise');
g1=medfilt2(P1(:,:,1));%%红
```

```
g2=medfilt2(P1(:,:,2));%%绿
g3=medfilt2(P1(:,:,3));%%蓝
g(:,:,1)=g1;
g(:,:,2)=g2;
g(:,:,3)=g3;
subplot(2,2,3)
imshow(g)
title('medfilter  gaussian')
```