关于van Herk算法的研究

摘要：本文主要针对van Herk算法的基本原理，以及van Herk 改进算法的描述与实验。第一部分主要是针对van Herk等人1992年在 Pattern Recognition Letters上发表的一篇提高医学图想清晰度算法的描述与实验。第二部分主要是针对Joseph Gil和Ron Kimmel等人发表的关于高校腐蚀、膨胀，开和闭算法实验的重复过程。

关键字：膨胀；腐蚀；van Herk改进算法

1 一种用于矩形和八角形内核的局部最小和最大滤波器的快速算法

针对局部最大和最小滤波器提出了一种新算法，**每个像素只需要6次比较而不依赖于内核大小**。该算法基于分离性和块递归序列的组合，这些序列是向前和向后评估的。

**1.1介绍**

这篇论文是Van Herk等人在1988年为了解决医学用的低对比度与低清晰度的医学图像而提出的一系列的图像增强算法，如最小平方反滤波算法。Bijhold等人（1991a）描述了自动门的边缘检测算法，Gilhuijs等人（1991）和Van Herk等人（1990）描述一种自动图像匹配方法，用于在线病人设置误差评估。后者的方法是Bijhold（1991b）等人和Meertens等人（1990）描述的门户形象评价交互程序的完全自动实现。

随着数学形态学的广泛使用（如Haralick（1987）、Serra（1982）和斯腾伯格（1986）），发现应用形态滤波器对一些图像问题处理时，很难用线性滤波器解决。Gilhuijs等人（1991）表明，灰色色调形态滤波器（例如，局部最小值和最大值）可以有效地从这类非常低对比度的X射线图像中提取解剖细节（如骨骼）。然而，对于在线应用，大内核尺寸的形态滤波器的计算时间方面变得非常长。**所以这篇论文研究的目的是找到局部最小和最大滤波的快速算法。由此，利用可分性和递归算法得到了速度上的改进。**

**1.2可分离的形态学操作**

使用以下符号：f，g和h代表离散的图像或过滤器内核（结构元素），x和y是图像索引，i和j是内核索引。图像索引运行从1到N，其中N是图像大小。内核索引从-(k-1)/2 to (k-1)/2运行，其中k是内核大小（假设奇数保持对称）。以下面的方式考虑边缘效应：在允许的折射率范围之外的图像值，例如，，对于局部最大滤波器取为-oo，对于局部最小滤波器取为+oo。Serra（1982）定义了在图像f上具有结构元素g的灰色形态学膨胀（Minkowski 除法）我们将它表示为，

g⊕f



当使用最小值而不是最大值并且否定结构元素时，产生腐蚀的关系。当结构元素g可以写为x和y时，在行和列操作中分离（1）是可能的。

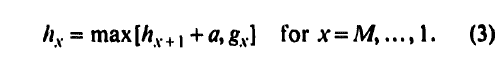
众所周知的可分离形态滤波器是由Borgefors（1986），Gonzales和Wintz（1977）和Sternberg（1986）等人描述的街区距离变换。用于矩形内核的局部最小和最大滤波器也属于这一类别（例如，Groen等人（1988））。 对于k\*k像素的核的分离的优点在于，**当分离时（k对于行和k对于列），操作的数目从每像素的k\*k个操作减少到用于直接求值的2k个操作**。 用于大内核大小的快速滤波的不同方法是重复应用小的 3 x 3内核大小的过滤器。 对于k\*k内核（k odd），必须应用（k-1）/ 2次3×3滤波器。当3×3滤波器是可分离的时，这种方法可以用3（k-1）个操作来实现。

**1.3递归形态滤波器**

分离后，剩余的一维滤波器有时可以递归地评估。 一个很好的例子是距离变换。 在一个维度中，距离变换的基本递归系列是：



其中a定义距离尺度（a <0）。 以这种方式形成的结构元件具有单个锯齿的形状，导致单向距离变换。为了获得具有三角形结构元素的全距离变换，必须通过等式（2）的结果向后执行相同的递归操作，即:



这个操作也使得滤波器对称。Borgefors（1986）给出了递归距离变换的详细描述。

使用该算法不可能获得平坦的结构元素，因为这要求a的值为0。在这种情况下，内核的大小不受限制。 对于距离变换，这意味着每个图像像素最终被设置为整个图像的最大值。

我们描述了先前的rect，rsive算法的修改，使得可以**发现一维图像上的局部最小和最大值，每个像素只有三个操作，与结构元素大小无关。**该算法可以用于在任意维度的矩形和八边形内核上创建局部最优滤波器。

**1.4新算法**

使用线性滤波器减法可以用于限制递归系列的范围，例如以获得高速均匀滤波器（例如，Groen等人（1988））。 因为均匀滤波器也是可分离的，所以它可以用每个像素只有4个运算来计算。 不幸的是，这种技术不适用于形态滤波器，因为没有取最大值的反向操作。 一旦输出变量被该变量的最大值和输入值替代，就没有简单快速的方法来撤消该操作。因此，我们必须找到一种替代方法来限制递归形态滤波器的传播。

我们解决这个问题如下。1输入数组被分成大小为k的子数组（组），其中k是结构化大小。如果需要，通过使用值填充-oo右侧的数组，将数组扩展为k的倍数。 a = 0的递归过程（2）应用于每个子阵列。2在存储在阵列g中的所得子阵列中，每个组的第i个像素包含输入阵列f的同一组的前i个像素的最大值。对原始输入数组再次执行相同的递归操作，但现在是向后。后一结果存储在3数组h中。这里，每个组的第（k + 1-i）像素包含输入数组的同一组的最后i个像素的最大值。在图1中，针对k = 5的结构元素列出了这两个阵列。从该图中可以看出，可以4通过组合向左移位k / 2的阵列g和阵列h向右移动k / 2。该操作的一些示例由图中的箭头示出。

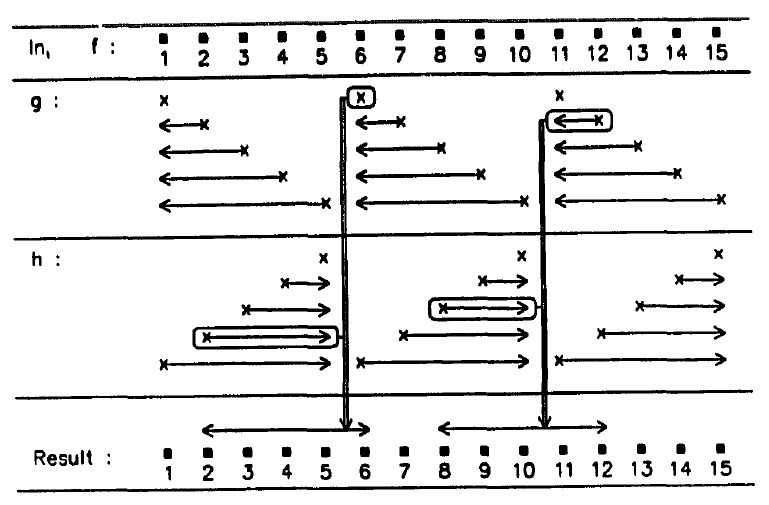
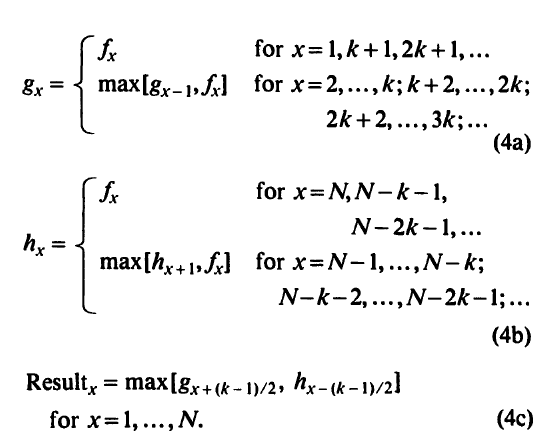


图1.执行局部最大和最小滤波器的递归方法。 在该示例中，内核大小k是5.阵列g和h是利用递归形态滤波器获得的临时结果。 符号“x->”表示存储在位置“x”中的值是由该箭头覆盖的所有值的最大值。 该图表明，通过将分别向左和向右移动的阵列g和h分别与该尺寸的一半相组合，获得最终结果。

递归过程（4a）和（4b）和操作（4c）描述了完整的算法：



**该方法的优点是操作（4a），（4b）和（4c）每个阵列元素仅需要一个单一比较。 因此，整个操作每个像素只需要3次操作，与结构元素大小无关。** 以这种方式，可以快速执行具有大结构元素的局部最大和最小操作。原则上，这种方法比构造大于3个像素的元素的非递归方法更快。然而，我们发现，建立阵列所需的额外时间会导致在值大于3的值处出现盈亏平衡点。

在二维情况下，通过连续地对图像的行和列应用操作（4）来实现滤波器。 因此，**对于矩形内核，2D中的局部最大或最小操作每个像素只需要6个操作**，与内核的大小无关。扩展到更高的维度是简单的。

该技术的缺点是需要两个临时缓冲器。由于此技术仅应用于当时的图像的一行（或列），所以额外的存储器要求是适度的。 另一个缺点是图像或缓冲区大小必须是结构化元素大小的倍数。 我们通过用大的负值填充缓冲区的末尾来实现这一点。 在实践中，我们用k / 2个元素在两侧进一步填充两个缓冲器。 这样，实现可以完全忽略边缘效应。

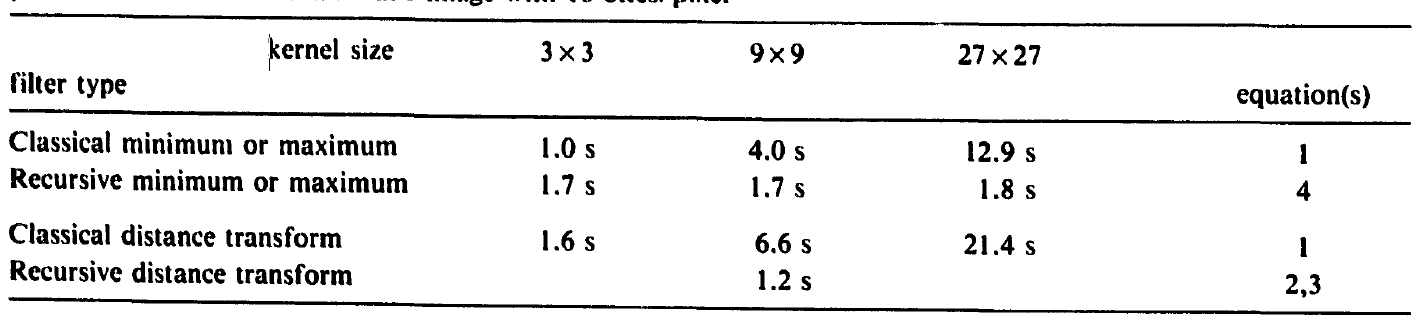
为了使局部最优滤波器的核心更圆形对称，滤波器在图像矩阵的两个对角线方向上附加地运行。 这使得执行时间加倍，但允许确定例如八边形扩张或八连接距离变换。

**1.5论文中的结果与讨论**

上述过滤器在IBM PC-AT计算机上实现。在表1中，列出了计算时间。这些例程以汇编语言编码以获得最佳性能并应用于256 \* 256像素的图像，每像素16位。

论文列出了用于局部最小和最大滤波器以及递归和非递归距离变换的经典实现的计算时间。局部最大值通过重复1,4和13次的3×3局部最大值来评估。通过应用3点水平最大值，随后是3点垂直最大滤波器，获得其3×3结构元素。 经典的距离变换是3×3锥体扩张，再次应用1,4和13次以获得所需的结构元素尺寸。平面形态滤波器的计算时间对结构元素尺寸具有小的依赖性，因为通过填充操作增加了阵列的长度。

表1

局部最小和距离变换的递归和经典实现的计算时间计算时间是针对具有 16位/像素 的256×256图像

该表显示，使用新的递归算法，即使在小型80286计算机上，也可以在1至2秒内执行大内核大小局部最大和最小值。 滤波器比相应的非递归操作快一个数量级。

即使局部最小和最大操作是灰色形态操作，所呈现的算法比用于二进制侵蚀和扩展的许多现有算法（例如，Van Vliet和Verwer（1987））更快。 假设灰度色调图像值为0和1描述的算法可以直接用于这些应用。

因为算法是递归的，所以它不能容易地在并行计算机上实现。然而，由于算法的简单性，用于这种算法的专用硬件不应当难以开发。 对于二进制形态运算，其中局部最大值等于逐位或运算，在硬件中的实现应当特别简单。

**2 高效的膨胀、腐蚀、开和闭算法**

**2.1、主要内容**

该文献主要是为一维膨胀和腐蚀滑动窗口过滤器提供一种高效的确定的算法。对于p元素滑动窗口，该算法计算一维滤波器使用1.5+o（1）比较每个样本点。该算法比之前已知的算法都好，由van Herk[ 25 ]和[ 12 ]Gil、Wrman（HGW算法）自主开发。同时，该文的结果构成了一种比Gevorkian等人的HGW算法变体提高的一种算法。对GAA变种算法的改进也是一种计算模型。GAA算法假设输入是独立同一的分布（IID假设），而论文的主要结果是确定的。该算法同样处理计算膨胀腐蚀问题，如果需要，比如计算不确定的边缘形态。我们转向开滤波器，定义最大最小滤波器的应用，并给它的计算给出高效的算法。尤其是，该算法仅仅比最大波波器计算稍微慢一点点。改进的算法很容易就能够用在二维，也可以应用在更高纬，每个窗口的数字比较，要保留常数。

**2.2、实验过程与结果**

**2.2.1 实验代码**

% This program performs morphological operations on binary image

% Part(1) : Entering the image for MATLAB

fprintf('\n This program performs morphological operations on binary image')

clc;

close all;

a = imread(uigetfile('lena.bmp'));

try

a=rgb2gray(a);

catch

end

% a = im2bw(a\_1,graythresh(a\_1));

siz = size(a);

figure

imshow(a)

title('Input Image after conversion to Binary')

fprintf('\n Click (1) to Perform Erosion')

fprintf('\n Click (2) to Perform Dilation')

fprintf('\n Click (3) to Perform Binary Opening')

fprintf('\n Click (4) to Perform Binary Closing')

choice = input('\n You select the Choice number : ');

switch (choice)

case 1

se = strel('disk',4);

a1 = imerode(a,se);

se = strel('disk',8);

a2 = imerode(a,se);

se = strel('disk',16);

a3 = imerode(a,se);

case 2

se = strel('disk',4);

a1 = imdilate(a,se);

se = strel('disk',8);

a2 = imdilate(a,se);

se = strel('disk',16);

a3 = imdilate(a,se);

case 3

se = strel('disk',4);

a1 = imopen(a,se);

se = strel('disk',8);

a2 = imopen(a,se);

se = strel('disk',16);

a3 = imopen(a,se);

case 4

se = strel('disk',4);

a1 = imclose(a,se);

se = strel('disk',8);

a2 = imclose(a,se);

se = strel('disk',16);

a3 = imclose(a,se);

otherwise

fprintf('\n Sorry! Wrong Choice')

end

figure

title('Input Image')

subplot(2,2,1)

imshow(a)

subplot(2,2,2)

imshow(a1)

title('Result Image-4\*4')

subplot(2,2,3)

imshow(a2)

title('Result Image-8\*8')

subplot(2,2,4)

imshow(a3)

title('Result Image-16\*16')

**2.2.2实验运行结果**

原图如图1，二值图像如图2。

腐蚀后的图像如图3。

膨胀后的图像如图4。

开运算的结果如图5。

闭运算的结果如图6。

图（1） 图（2）

图（3） 图（4）

图（5） 图（6）

**3 总结与心得体会**

Van Herk算法可以快速地计算局部最大值和最小值，对于核心尺寸较大的图像，更是能够大大提高运算效率。在传统的思路中，对一个矩阵进行最大最小值滤波时，最简单的思路是对一个点的邻域直接通过比较得到最大值，这样，计算的复杂度与窗口尺寸的大小有关系。van Herk算法就像快速求和算法一样，分别对行和列进行操作，这样最终得到最大值滤波结果。**对于每个点，平均只需要进行六次比较操作（行3次，列3次）即可得到最终值**。

而van herk的改进算法的计算过程则要经历两个阶段，一个是预处理阶段，一个是合并阶段，通过改进这两个步骤，从而减少样本间的比较次数，最终提高算法的执行效率。

另外，之前在阅读其它英文文献的时候，虽然也是英文的，但在阅读的时候，有相关的中文资料可以参考对比。而此次作业，找不到相关的中文资料，在阅读的过程中只能自己去理解去定义，很不适应，以后再这方面我会加强训练。做完此次作业，让我对膨胀、腐蚀、开和闭操作的基本概念，有了更深入的了解。在未来的学习时间里，我会继续努力，让自己在数字图像处理方面学习的更深入，更彻底。

**两个优化操作：**

**1行和列/k，末尾不进行补齐：每行少了k-N%k的像素，少了3\*（（k-N%k）\*N）；每列少了k-M%k的像素，少了3\*（（k-M%k）\*M）**

**2g和h两边不进行k/2的补齐：每行每列少了k次比较（k\*行数和k\*列数）**

**3g和h结尾进行mask的补齐，widrow（该算法的精髓：mask为单位进行min操作）**

**算法流程：**

**1对行操作**

**（1）j：0~row-mask，row-mask~wid，wid~row！！（难点：rear的标记）**

**（2）每行取值：0——k/2（g值）；k/2——row-k/2（g和h比较）；N-k/2——N（h值）（else处理）**

**2对列操作：矩阵转置后，把以上的N替换为M**

**写在后面：**

**1虽然算法上实现了优化，减少了很多像素，同时减少了比较次数，但是代码的实现上，把“统一”变成了“分类”，“分类”带来的恰好是比较次数的增加，最不值得的是增加了原本“统一”情况下处理每一个像素的比较次数，最为致命。结论就是：算法效率建立在实现“统一”的基础上。**

**2实验室代码的隐藏bug，小概率事件：**一旦被整除，多了一个滤波核操作，最后一个滤波核的操作会溢出（很危险）！！

**参考文献：**

[1]Marcel van Herk, A fast algorithm for local minimum and maximum filters on rectangular and octagonal kernels[J]. Pattern Recognition Letters , 1992, 13(7): 517-521.

[2]J. Y. Gil; R. Kimmel. Efficient dilation ,erosion, opening, and closing algorithms[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002, 24(12):1606-1617.

[3]Joseph Gil ; Michael Wcrman, Computing 2-D Min, Median, and Max Filters[J].IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE. VOL. IS, NO. 5 , MAY 1993

[4]David Z. Gevorkian ; Jaakko T. Astola ; Samvel M. Atourian, Improving Gil-Werman Algorithm for Running Min and Max Filters[J], IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 19, NO. 5, MAY 1997