基于分组行进算法的图像修补方法

王志鹏 张桂戌
(华东师范大学计算机科学技术系,上海 200062)

摘 要 图像修补是图像恢复研究中的一个重要内容,它的目的是根据图像的现有信息来自动恢复丢失的信息。虽然图像修补的基本思想十分简单,但是许多的图像修补算法都十分复杂,而且难于实现。快速行进算法(FMM)与水平集法(Level Set)相结合进行曲线进化是一种高效的曲线进化算法,该算法的时间复杂度是 $O(N\text{lb}N)$。Kim 提出了另一种水平集的曲线进化算法——分组行进算法(GMM),该算法的时间复杂度是 $O(N)$。受其启发,为了更快地进行图像修补,提出了一种基于 GMM 算法的图像修补的新算法,并研究了对 GMM 算法的细节改进。为了验证算法的快速性,还给出了使用 Bertalmio 提出的算法、Telea 提出的算法以及新算法对同一幅图片进行修补的实验结果。通过比较发现,该新算法在大幅度提高修补速度的同时,仍能保持较好的修补效果。

关键词 图像复原 图像修补 分组行进算法 快速行进算
中图法分类号: TP391.41 **文献标识码**: A **文章编号**: 1006-8961(2007)05-0799-06

Digital Image Inpainting Based on Group Marching Method

WANG Zhi-peng, ZHANG Gui-xu
(*Department of Computer Science & Technology, East China Normal University, Shanghai* 200062)

Abstract Image inpainting is an important research topic in the area of image restoration. Its objective is to restore the lost information according to around image information. Although the inpainting basics are straightforward, most inpainting techniques published in the literature are complex to understand and implement. Fast marching method (FMM) is an efficient algorithm for level set applications whose total computation cost is $O(N\text{lb}N))$. Kim presented a more efficient algorithm called group marching method(GMM) with the complexity of $O(N)$. Motivated by his work, we propose a new technique for image inpainting based on GMM. Examples of experiment using Bertalmio's algorithm, Telea's algorithm, and our algorithm are illustrated. The result shows that the technique we proposed is faster than the other inpainting methods while preserving almost the same inpainting result.

Keywords image restoration, image inpainting, group marching method(GMM), fast marching method(FMM)

1 引 言

图像修补是图像恢复研究中的一个重要内容,它的目的是根据图像的现有信息来自动恢复丢失的信息。目前,图像修补技术(inpainting)是计算机图形学和计算机视觉研究中的一个热点,而且在文物保护、影视特技制作、多余物体去除(如静态或视频图像中去除部分人物、文字、小标题等)等方面都有重要的应用价值。

目前,许多图像修补算法的步骤都可归结为:

(1) 人工选择一块需要进行修补的区域;

(2) 利用待修补区域边缘的图像信息对修补区域的边缘进行修补;

(3) 缩小待修补区域 如果还有未完成修补的区域,则返回执行步骤(2)。

为了保持图像边缘处的边界连续性,图像修补应当使图像的等值线(isophote)尽量光滑地扩散到

基金项目:国家重点基础研究发展"973"计划前期研究专项资助项目(2006CB708305)
收稿日期:2006-01-04;改回日期:2006-04-21
第一作者简介:王志鹏(1982 ~),男。2004 年获得华东师范大学计算机科学技术系学士学位,现为华东师范大学计算机科学技术系研究生。主要研究领域为数字图像处理。E-mail: wallancewang@gmail.com

待修补区域的内部像素中去。

Bertalmio 等人采用偏微分方程进行图像修复[1],取得了较好的效果,但用户需指定需要修复的区域。文献[1]算法先将彩色图像分为红(R)、绿(G)、蓝(B)3 个独立的通道,然后对每个通道将待修补区域边界的等值线外部的信息沿轮廓法向扩散到中间待修补的像素上。该算法先使用 2 维的拉普拉斯方法[2]估计局部颜色的光滑度,再利用这个光滑度沿着等值线扩散,但该过程需要考虑各向异性的扩散,才能保证边缘处的边界联系。许威威等提出了基于全变分(total variational)模型的图像修补算法[3]。整体变分方法是采用欧拉-拉格朗日方程和各向异性扩散的方法,在扩散过程中,由于考虑了轮廓的几何信息,因此可以处理较大的区域,但边界处往往很模糊。总的来说,由于这些算法本身存在缺点,如基于偏微分方程的图像修补算法和基于整体变分的图像修补算法都需要反复执行多种诸如各向异性的扩散(anisotropic diffusion)的复杂运算,这就导致了算法执行速度比较慢,因而限制了它们的实际应用,而且,这些算法很少考虑极端情况或者涉及如何进行离散化等具体的实现细节。

FMM(fast marching method)是一种跟踪运动界面演化的技术[4]。Telea 提出了一种快速有效的基于 FMM 的图像修补技术[5]。由于其采用了 FMM 的水平集法曲线进化算法,因此该算法大大提高了图像的修补速度。其中,FMM 算法的时间复杂度是 $O(N\text{lb}N)$。受其启发,本文提出了一种基于 Kim 提出的 GMM(group marching method)算法[6]的图像修补技术。GMM 算法时间复杂度是 $O(N)$。

本文算法主要有如下优点:

(1) 实现简单;

(2) 相比其他图像修补算法,执行速度更快;

(3) 可达到与其他图像修补算法几乎完全相同的修补效果。

2 图像修补的基本思想及实现

2.1 图像修补的基本思想

计算机视觉中有一个结论——人是根据其生活中积累的经验来感知世界的,即人们经常根据其经验对周围环境中被遮挡物体的形状做出一个最佳猜测。图像修补的基本思想就是使用待修补区域周围的图像信息来对待修补区域进行修补。

假设 Ω 表示待修补区域,$\partial\Omega_0$ 表示初始的待修补区域的边缘,$\partial\Omega_i$ 表示第 i 个修补区域的边缘,$\partial\Omega$ 表示正在修补区域的边缘,则基于水平集的图像修补算法可以表示如下:

$\partial\Omega = \partial\Omega_0$

while($\partial\Omega$ 不为空){

修补$\partial\Omega$ 上的所有点;

移动$\partial\Omega$ 到 Ω 内部的下一个待修补区域边缘 $\partial\Omega_i$;

}

下面,介绍从待修补区域边缘$\partial\Omega$ 一步一步进化到待修补区域 Ω 内部的以下两种水平集曲线进化算法:快速行进算法和分组行进算法。

2.2 快速行进算法

在数值分析领域,Osher 和 Sethian 提出了一种用水平集法来模拟动态曲线和曲面的方法[7,8],该方法依赖于曲率的速度进化,但是由于这种方法对所有的水平集都要进行计算,而不单是对与传播前沿相关的零水平集进行计算,所以它存在一个很大的缺点就是运算量很大,对一个 $N\times N$ 大小的图像进行运算,每一步的运算复杂度为 $O(N^2)$。如果传播前沿的法向速度总是保持或全正或全负,那么就可以把上面的问题转化为寻求每个水平集的到达时间的问题,这样就把问题转化为可大大提高运算速度的 FMM。

FMM 把问题变为求解方程

$$\nabla T\cdot v(x) = \Phi(x) \tag{1}$$

这是一个表示曲线随时间进化的 Eikonal 方程,还可以表示为如下形式:假设 $T(x,y)$ 为到达点 (x,y) 的时间,其满足方程:

$$|\nabla T|F = 1 \tag{2}$$

从式(2)可以看出,到达时间的梯度曲面与传播前沿的速度成反比。接下来即可利用数值方法求解以下 Eikonal 方程:$\nabla T\cdot v(x) = \Phi(x)$。两边取平方得

$$(\nabla T\cdot v(x))^2 = (\Phi(x))^2 \tag{3}$$

再利用差分格式求解,这里只考虑 2 维的情况[5]:

$$\max(D_{-x}T, -D_{+x}T,0)^2 + \max(D_{-y}T, -D_{+y}T,0)^2 = 1 \tag{4}$$

其中,$D_{-x}T$ 和 $D_{+x}T$ 离散化的有限差分格式为 $D_{-x}T_{i,j} = T_{i,j} - T_{i-1,j}$,$D_{+x}T_{i,j} = T_{i+1,j} - T_{i,j}$),同理可得$D_{-y}T$和 $D_{+y}T$。

FMM 方法是通过完全二叉树方法对窄带区域(narrow band)(即待修补区域边缘像素集合)中的所有点的到达时间进行排序找出到达时间最小的点的技术,假设该窄带中像素的个数为 N,则由于每一步都要进行此操作,因此其运算复杂度为 $O(N\text{lb}N)$。

使用 FMM 的水平集法进行曲线进化的情况如图 1 所示。

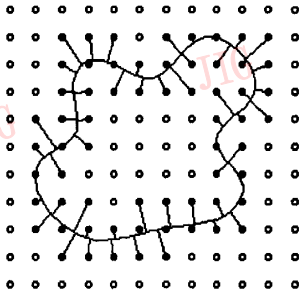

图 1　使用 FMM 的水平集法的曲线进化

Fig. 1　Level set for curve evolution using FMM

2.3　分组行进算法

为了减少其运算量,文献[6]提出了一种运算复杂度为 $O(N)$ 的 GMM 算法。此时 Eikonal 类型方程就成为

$$|\nabla T(x_{\text{from}},x)|^2 = \frac{1}{v^2(x)} \tag{5}$$

其中,$T(x_{\text{from}},x)$ 表示曲线从 x_{from} 行进到 x 所需的时间,$v(x)$ 表示波峰 x 处的速度。

GMM 方法是先找出到达时间满足式(9)的一组点同时进行行进的,而不是对窄带中所有点的到达时间进行排序,然后找出最小到达时间的点来进行行进。需要注意的是,在选择一组点时必须保证因果性[6]。

对窄带中的第 m 个像素 $p_m=(i\Delta x, j\Delta y)$,假设其速度为 v_m,x 轴和 y 轴方向上的速度分量分别为 v_m^x、v_m^y。为了简化问题,令 $\nabla x=\nabla y=h$,则从 p_m 到 y 轴的距离 s 可以通过下式求出(图 2):

$$s = \frac{h|v_m|}{v_m^x + v_m^y} \tag{6}$$

从 p_m 到 y 轴的时间可以由下式求出:

$$\Delta T_m = \frac{s\Phi_m}{|v_m|} = \frac{h\Phi_m}{v_m^x + v_m^y} \tag{7}$$

假设差值节点值都已知,则已求得的 T_m 需要满足下面不等式:

$$T_m \geqslant \min(T_{i+1,j}, T_{i,j+1}) + \Delta T_m \tag{8}$$

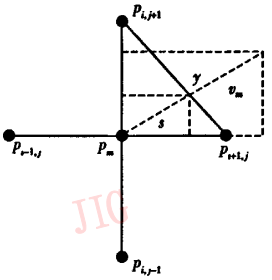

图 2　5 点格式插值求解到达时间示意

Fig. 2　Resolution to arrival time using five-point format

根据文献[6]的方法,从窄带 Ω_{NB}(下角 NB 代表 narrow band)中选取的一组像素 Ω_{group} 如下[5]:

$$\Omega_{\text{group}} = \{p_m \in \Omega_{\text{NB}} : T_m \leqslant T_{\Omega_{\text{NB}}}^{\min} + \Delta T_{\Omega_{\text{NB}}}\} \tag{9}$$

其中,$T_{\Omega_{\text{NB}}}^{\min} = \min\limits_{p_m \in \Omega_{\text{NB}}} T_m$,$\Delta T_{\Omega_{\text{NB}}} = \min\limits_{p_m \in \Omega_{\text{NB}}} \Delta T_m$。关于文献[6]算法细节将在下一个小节结合本文算法进行讨论。

2.4　基于 GMM 的图像修补算法

FMM 算法需要对窄带中所有点到进化曲线的距离(或称为到达时间)进行排序,在曲线行进时每次从窄带中选出一个与进化曲线距离最小的一个点加入到已知像素集合中。与之不同的是,GMM 算法的基本思想是每次选择一组到进化曲线的距离小于某一规定值的点来进行行进,由于该算法不需要对窄带中所有点到进化曲线的距离进行排序,从而节省了曲线进化的时间。

本文提出的基于 GMM 的图像修补技术的基本思路是在曲线进化的同时对需要修补的图像区域进行修补。下面给出算法的详细步骤。

规定:Ω_{accepted} 表示已知像素点的集合;Ω_{group} 表示选中的一组行进的像素集合;Ω_{NB} 表示窄带,即待修补区域的边缘像素集合;Ω_{in} 表示待修补区域的内部像素点集合。

对于每个像素,保存以下 3 个属性:T 表示到达时间;G 表示灰度值(彩色图像采用 R,G,B3 个通道独立处理);F 表示像素位置,取值为以下 3 个常量之一:

- F_{accepted}:表示该像素属于集合 Ω_{accepted};
- F_{NB}:表示该像素属于集合 Ω_{NB};
- F_{in}:表示该像素属于集合 Ω_{in}。

则基于 GMM 的图像修补算法执行步骤如下:

初始化步骤为

(1) 将 Ω_{accepted} 中所有像素的到达时间 T 初始化为 -1,F 初始化为 F_{accepted};

(2) 将 Ω_{NB} 中所有像素的到达时间 T 初始化为 0,F 初始化为 F_{NB};

(3) 将 Ω_{in} 中所有像素的到达时间 T 初始化为 ∞(在程序中为 1.0e6),F 初始化为 F_{in}。

修补过程步骤为

(1) 根据式(9)寻找要行进的一组点 Ω_{group};

(2) 按照反向顺序对 Ω_{group} 中的所有不在 $\Omega_{accepted}$ 中的邻域点重新计算到达时间;

(3) 按照正向顺序对 Ω_{group} 中的所有不在 $\Omega_{accepted}$ 中的邻域点重新计算到达时间,如果点在 Ω_{in} 中,则修补这个点,然后把它从 Ω_{in} 中移去,加入到 Ω_{NB} 中;

(4) 把 Ω_{group} 中的点标记为 $\Omega_{accepted}$ 中的点;

(5) 如果 Ω_{NB} 不为空,则返回执行步骤(1)。

为了保证算法的稳定性,必须以正向和反向两种顺序计算到达时间。在 GMM 方法中,由于在对 Ω_{group} 中的点进行更新时,要计算全部点,因此相对来讲比较麻烦。算法实现中,本文研究了它的改进方法,即首先利用前面的插值法来求解点的到达时间,每选出一个像素,则重新计算该像素不在 $\Omega_{accepted}$ 中的邻域点的到达时间 T,然后将这个像素放入一个栈 S_{group} 中。反向更新时,从栈 S_{group} 中依次出栈,每出栈一个像素,再重新计算该像素不在 $\Omega_{accepted}$ 中的邻域点的到达时间 T。

3 修补一个像素

前面讨论了基于 GMM 的图像修补技术,下面讨论在步骤(3)中提到的修补一个像素的问题。本文采用文献[5]中提出的一种加权平均数学模型。

规定 $B_\varepsilon(p)$ 为待修补像素 p 周围一个大小为 ε 的邻域,则 $B_\varepsilon(p)$ 中的像素 q 对像素 p 的贡献 $I_q(p)$ 可用下式表示[5]:

$$I_q(p) = I(q) + \nabla I(q)(p - q) \tag{10}$$

其中,$I(q)$ 表示像素 q 的灰度值,$\nabla I(q)$ 表示像素 q 的梯度值。

这样,p 的灰度值就可以用 $B_\varepsilon(p)$ 中所有已知像素对像素 p 的贡献进行加权平均求得。其计算公式如下[5]:

$$I(p) = \frac{\sum_{q \in B_\varepsilon(p)} w(p,q)[I(q) + \nabla I(q)(p - q)]}{\sum_{q \in B_\varepsilon(p)} w(p,q)} \tag{11}$$

在程序中,$B_\varepsilon(p)$ 采用像素 p 的 8-邻域,而权值 $w(p,q)$ 则使用下式求得[5]:

$$w(p,q) = dir(p,q) \times dst(p,q) \times lev(p,q) \tag{12}$$

其中,

$$dir(p,q) = \frac{p - q}{\|p - q\|} \cdot N(p)$$

$$dst(p,q) = \frac{d_0^2}{\|p - q\|^2}$$

$$lev(p,q) = \frac{T_0}{1 + |T(p) - T(q)|}$$

$N(p) = \nabla T$ 是像素 p 的梯度方向,d_0 是参考距离常量,T_0 表示参考到达时间常量。在程序中,d_0、T_0 均取 1,其表示的是相邻两个像素之间的距离和到达时间。$\|p - q\|$ 表示像素 p 与 q 的距离值。$dir(p,q)$ 表示像素 p 的梯度方向上的像素 q 对像素 p 的贡献,距离越小,贡献越大;$dst(p,q)$ 表示像素 q 和像素 p 之间距离对像素 p 的贡献,距离越小,贡献越大;$lev(p,q)$ 考虑的是轮廓线(contour)周围的像素 q 对像素 p 的贡献,距离越小,贡献越大。

4 实验与结果分析

本文所有实验都是在微机 Pentium IV 2.79G,RAM 512M 上实现的。

为了检验本文算法的修补效果,采用 C++ Builder 6.0 实现了本文算法。实验时,首先给出一张分辨率为 600×400 的原始图片(图 3)和损坏后的图片(图 4);然后,分别采用文献[1]算法、文献[5]算法以及本文算法对图 4 进行修补,修补结果分别如图 5、图 6、图 7 所示。

表 1 列出了使用不同修补算法对图 4 进行修补所需的时间。文献[1]算法由于考虑了各向异性扩

图 3 分辨率为 600×400 的原始图片

Fig. 3 A 600×400 original image

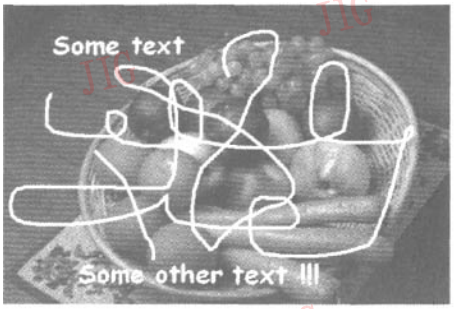

图 4　损坏图片

Fig. 4　Damaged image

图 5　使用文献[1]算法修补后的图片(迭代次数 1 000)

Fig. 5　Image inpainted with Bertalmio's algorithm(iteration:1 000)

图 6　使用文献[5]的算法修补后的图片

Fig. 6　Image inpainted with Telea's algorithm

散以保证边缘处的边界联系,需要进行复杂的数学计算,因而所需修补时间最长,约 5min。文献[5]算法由于采用了时间复杂度为 $O(N\mathrm{lb}N)$ 的 FMM 水平集曲线进化算法,因而大幅度提高了图像修补的速度,所需修补时间仅为文献[1]算法的 1/60。同时,由于本文算法采用了时间复杂度为 $O(N)$ 的 GMM 水平集曲线进化方法,因此进一步缩短了图像的修补时间。

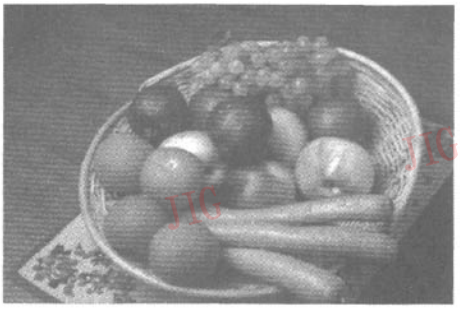

图 7　使用本文算法修补后的图片

Fig. 7　Image inpainted with our algorithm

表 1　使用不同算法所需修补时间

Tab. 1　Inpaint time of different algorithms

修补算法	文献[1]算法(迭代次数 1 000)	文献[5]算法	本文算法
修补时间(s)	300	5 ±	3 ±

比较原始图片(图 3)与修补后的图片(图 5、图 6、图 7)可以发现,文献[1]算法、文献[5]算法以及本文算法都能够很好去除损坏图片上多余的文本和曲线,可取得较好的修补效果。

比较图 5、图 6、图 7 可以发现,文献[1]算法由于考虑了各向异性扩散,以保证边缘处的边界联系,因而获得了最好的修补效果。这体现在图 5 的边缘的边界区域比图 6、图 7 清晰。文献[5]算法采用 FMM 的水平集曲线进化算法,虽然大幅度提高了图像修补的速度,但是,由于加权平均修补算法的局限性,致使该算法用于修补较大面积损坏区域时,往往很难保持边缘的边界的连续性,其修补效果如图 6 所示。由于本文算法采用了 GMM 的水平集曲线进化算法,因此进一步提高了图像的修补速度,其修补效果如图 7 所示。但是,为了保证算法的稳定性,GMM 必须按照顺序和逆序重新计算邻域点的到达时间,因而本文算在处理包含较小的窄带图片时,相对于文献[5]算法的优势不是十分明显。同时,由于本文算法采用了文献[5]提出的使用待修补像素邻域点的灰度值的加权平均来进行修补,因此同样存在因局部化修补而带来的边界模糊问题。

5　结　论

本文提出了一种基于 GMM 的图像修补技术,

实验结果表明,该技术在大幅度提高图像修补速度的同时,仍能保持较好的修补效果。从上面的实验结果及分析可见,水平集方法和 FMM 结合起来不仅可以处理一些具有复杂拓扑性的图像,并且具有较高的运算速度。同时,引入文献[6]中提出的 GMM 算法还可以更提高其效率,文章研究了对 GMM 算法还实现细节的改进。但是,由于本文采用了文献[5]提出的使用单个像素邻域点的灰度值的加权平均来对需要修补的区域进行修补,致使本文算法用于修补较大区域图像时,往往会产生模糊。接下去的研究,其主要目标是如何消除局部化带来的影响,以进一步提高修补的效果。

参考文献(References)

1 Bertalmio M, Sapiro G, Caselles V, *et al*. Image inpainting[A]. In: Proceedings of SIGGRAPH 2000[C], New York, NY, USA, 2000: 417 ~ 424.

2 Gomes J, V elho L. Image Processing for Computer Graphics[M]. New York: Springer-Verlag, 1997.

3 XU Wei-wei, PAN Zhi-geng, ZHANG Ming-min. Image inpainting based on total variational model[J]. Journal of Image and Graphics, 2002, **7A**(4): 351 ~ 355. [许威威,潘志庚,张明敏. 一种基于全变分的图像修补算法[J]. 中国图象图形学报, 2002, **7A**(4): 351 ~ 355.

4 Sethian J. A fast marching level set method for monotonically advancing fronts[J]. Proceedings of the National Academy Science, 1996, **93**(4): 1591 ~ 1595.

5 Telea A. An Image inpainting technique based on fast marching method[J]. Journal of Graphics Tools, 2004, **9**(1): 25 ~ 36.

6 Kim S. An $O(N)$ level set method for Eikonal equation[J]. SIAM Journal on Scientific Computing, 2001, **22**(6): 2178 ~ 2193.

7 Sethian J. A Level Set Methods[M]. Cambridge, UK: Cambridge University Press, 1996.

8 Osher S, Sethian J A. Front propagating with curvature-dependent speed: algorithm based on Hamilton-Jacobi formulation[J]. Journal of Computational Physics, 1988, **79**: 12 ~ 49.