文章编号:1001 - 9081(2006)06 - 1386 - 03

基于弹性算法的矢量场可视化方法的研究

吉晓娟,顾耀林
(江南大学 信息工程学院,江苏 无锡 214122)
(jixiaojuan1234@163.com)

摘 要:分析了二维非稳定场的可视化方法(Unsteady flow Line Integral Convolution,UFLIC),并针对其产生每帧图像费时较多的缺陷,进行了改进。采用弹性算法来控制种子的释放,通过重用、复制相关迹线来减少迹线的积分计算,达到减少生成每帧图像所需的时间的效果。实验证明在生成的图像质量相当的前提下,种子控制算法比 UFLIC 算法更快。

关键词:非稳定场;二维非稳定场的可视化方法;弹性算法;迹线;种子
中图分类号: TP391 **文献标识码**:A

Study of visualization based on flexible algorithm vector fields

JI Xiao-juan, GU Yao-lin
(*College of Information and Engineering, Southern Yangtze University, Wuxi Jiangsu* 214122, *China*)

Abstract: One weakness of Unsteady flow Line Integral Convolution(UFLIC) was time-consuming to generate frames, the improved method was proposed. The effect that reduced the time of advect frames was achieved by flexible algorithm and reuse and copy relative pathlines. The results show that our algorithm is fast than UFLIC based on the similar quality of frames.

Key words: unsteady flow field; Unsteady flow Line Integral Convolution(UFLIC); flexible algorithm; pathline; seed

0 引言

非稳定矢量场可视化是科学计算可视化中最具挑战性的课题之一。它以直观的图形图像显示场的运动,透过抽象数据有效洞察其内涵本质和变化规律,广泛应用于计算流体力学、航空动力学、大气物理和气象分析等领域。

近年来很多人对非稳定矢量场可视化做了大量的研究。文献[2]提出了 IBFV 方法,此方法还被推广到三维非稳定流场[3]。文献[4]提出了 LEA 方法。文献[5]提出了 UFAC 方法。通过这些方法产成的每帧图像空间一致性不佳。文献[1]提出的 UFLIC 算法产成的图像空间一致性和时间一致性都优良,但产生每帧图像费时较多。本文对 UFLIC 算法进行了改进,采用弹性算法来控制种子的释放以减少所需生成的迹线数量,通过重用可用的相关迹线来减少迹线的积分计算,从而减少产生每帧图像所需的时间。

1 算法的分析与改进

1.1 UFLIC

文献[1]提出了基于 LIC[6] 方法的 UFLIC 算法,通过观察随时间变化粒子运动的轨迹即迹线来显示二维非稳定流场。UFLIC 的算法示意图如图 1。UFLIC 算法包括两大重要的部分:纹理值发散过程和连续纹理输入法。如图 2 所示,在每个时间步,从每个像素中心释放一个种子生成迹线,则该种子在它的生命周期(通常为时间步的整数倍)内向种子通过的像素发散它的纹理值,这就是纹理值发散过程。每个像素作为纹理值的接收者,保存通过它的种子发散给它的纹理值。每帧图像通过卷积纹理值和像素索引得到空间一致性优良的图像。连续纹理输入法即把上次发散过程得到的纹理图像经过噪声抖动高通滤波处理后作为下次发散过程的输入纹理,得到时间一致性优良的图像。白噪声图像仅作为首次发散过程的输入纹理。

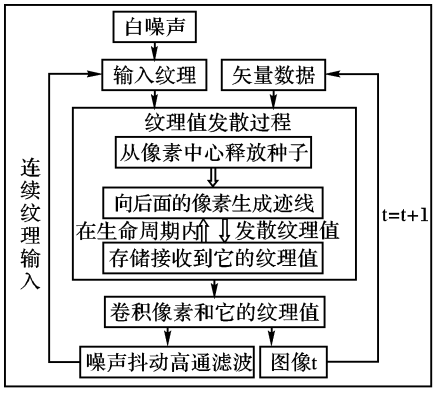


图 1 UFLIC 示意图

在 UFLIC 算法中,每个时间步开始那一刻即从像素中心释放一个种子,然后每个种子开始迹线积分,大约 90% 的时间花费在发散过程的迹线积分上。每个种子生成的迹线不管它有没有遇到边界点或关键点,只要该种子的生命周期结束则迹线的生成过程结束。假设一个矢量场的分辨率为 $x * y$,种子的生命周期为 N 个时间步,则经过 $n(n \geq N-1)$ 个时间步后,将大约有 $N * x * y$ 个种子存在。然而每个时间步只需 $x * y$ 个种子足够观察流的变化。所以要控制释放的种子数量。

1.2 迹线的生成

一般将一个粒子的迹线方程定义为如下的形式:

$$\frac{d(\rho(t))}{dt} = v(\rho(t),t) \quad (1)$$

收稿日期:2005 - 12 - 19

作者简介:吉晓娟(1980 -),女,江苏射阳人,硕士研究生,主要研究方向:计算机图形学与虚拟现实; 顾耀林(1948 -),男,江苏无锡人,教授,主要研究方向:计算机图形学与虚拟现实.

式(1)中$\rho(t)$表示粒子的位置,$v(\rho(t),t)$表示时间t时刻粒子的矢量,ρ为t的函数。通过一步一步的积分求解式(1)可以得到粒子路径:

$$\rho(t+\Delta t)=\rho(t)+\int^{t+\Delta t} v(\rho(t),t)\,dt$$

下面是三种典型的积分方法:

1) 欧拉法

$$\rho(t+s)=\rho(t)+s\times v(\rho(t),t)$$

2) 中点法

$$\rho(t+s)=\rho(t)+s\times v(\rho(t)+v(\rho(t),t)\times s/2,t+s/2)$$

3) 四阶龙格库塔法

$$\Delta\rho_0=s\times v(\rho(t),t)$$
$$\Delta\rho_1=s\times v(\rho(t)+\Delta\rho_0/2,t+s/2)$$
$$\Delta\rho_2=s\times v(\rho(t)+\Delta\rho_1/2,t+s/2)$$
$$\Delta\rho_3=s\times v(\rho(t)+\Delta\rho_2/2,t+s/2)$$
$$\rho(t+s)=\rho(t)+\Delta\rho_0/6+\Delta\rho_1/3+\Delta\rho_2/3+\Delta\rho_3/6$$

$s=\Delta t$为积分步长

由文献[7]知,欧拉法不精确且稳定性不好。中点法实质即是二阶龙格库塔法。四阶龙格库塔法可以用至少两倍长的步长就能达到与中点法一样的精度。三种方法中最优的即为四阶龙格库塔法。如果两次步长之间的流的变化是线性的,则四阶龙格库塔法就退化成二阶。为了用尽可能少的计算量达到解的预定精确度,我们采用误差允许的范围内的自适应步长控制的四阶龙格库塔法。由于积分器所采用的步长较长且不相等,则返回的点之间的距离较大且不均匀,需要在迹线上连续的两个点之间进行插值。我们使用的积分器是高阶的,相应的也采用高阶的插值方法,用厄密多项式进行插值。假设相临的两点为x_n和x_{n+1},则插值公式[7]如下:

$$p(u)=au^3+bu^2+cu+d \tag{2}$$

式(2)的参量$u\in[0,1]$

$$u=\frac{s-s_{(n)}}{s_{(n+1)}-s_{(n)}} \tag{3}$$

式(3)的s为积分步长:

$$a=2p(0)-2p(1)+p'(0)+p'(1)$$
$$b=-3p(0)+3p(1)-2-p'(0)-p'(1)$$
$$c=p'(0)$$
$$d=p(0)$$

p和p'表示的意义如下:

$$p(0)=x_n$$
$$p(1)=x_{n+1}$$
$$p'(0)=(s_{(n+1)}-s_{(n)})v(x_n)$$
$$p'(1)=(s_{(n+1)}-s_{(n)})v(x_{(n+1)}) \tag{4}$$

最后生成的迹线的样本点分布均匀。发散过程中的种子点就是从迹线上的样本点中选取出来的。

1.3　弹性算法及迹线的重用、复制

弹性算法包括两方面:空间弹性,即种子只要在像素里释放就可以了,不一定必须在像素中心释放;时间弹性,即某些种子可以不必在每个发散过程开始那一刻发散,可以在发散过程开始后的一个时间片里释放。我们用相应的像素名称命名从中释放的种子。

空间弹性控制种子的释放体现在连续的发散过程之间。如图2所示,在发散过程k中,在k时刻从像素A中心释放一个A种子,在它的生命周期T(设$T=4\triangle t$)里生成一条迹线并向迹线上的像素发散纹理值,该迹线在$k+1,k+2,k+3$时刻分别通过像素B,C,D。如果在发散过程$k+1,k+2,k+3$中的$k+1,k+2,k+3$时刻分别从像素B,C,D释放一个种子,则得到与发散过程k相同的轨迹,则B,C,D种子可以直接摘取A种子生成的迹线的共同部分。而UFLIC总是从像素中心释放种子生成迹线,忽略连续发散过程中的这种关系,弹性算法通过重用已经保存的有用的迹线来加快发散过程。保存发散过程K的BE段,删除AB段,扩充EF段,就可以得到在$k+1$时刻B种子经过一个生命周期生成的完整的迹线BF。

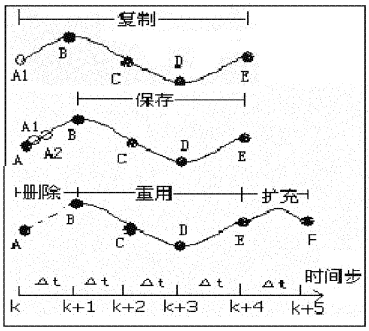


图2　迹线的重用、复制

时间弹性控制种子的释放体现在同一发散过程内。如图2所示,种子$A1$,$A2$是种子A在发散过程开始一小个时间片里经过的像素释放的种子,它们可以不需进行积分计算直接复制种子A的迹线。种子B、C、D也可以做相应的处理。

根据种子点获得迹线的方式给种子点分类。如图2,称A为父种子,也称其为根种子。种子B、C、D、E叫子种子,$A1$,$A2$叫兄弟种子。B种子作为A种子的子种子将成为下次发散过程的父种子,B种子也叫继承种子。种子C、D、E在下面的时间步处理方式同种子B一样。弹性算法通过给每个像素设置两个信号量CS和NS以控制每次发散过程不超过一个种子从该像素释放出来和控制迹线的保存或删除。信号量的含义、状态、操作定义如下:

CS表示某个像素在当前发散过程的状态和决定释放的种子类型。

NS表示某个像素在下次发散过程的状态和决定保存或删除迹线。

开状态:可以从像素释放一个父种子或兄弟种子。

半开状态:可以从像素释放一个兄弟种子。

关状态:不能释放种子。

开操作:置信号量开状态。

半开操作:置信号量半开状态。

关操作:置信号量关状态。

每次发散过程一开始,用NS矩阵刷新CS矩阵,并对NS矩阵进行开操作,NS矩阵在发散过程中动态更新,这是一个循环的过程。一旦从一个像素释放了一个种子,对该像素进行关操作以阻止在本次发散过程中从该像素再释放种子。一次发散过程从释放继承种子开始,接着按从上到下,从左到右的顺序从CS为开状态的像素里发散根种子。每释放一次父种子要相应释放一些兄弟种子。我们定义一个置信号量函数操作父种子的信号量。对于一个父种子无论它的迹线是生成的还是重用的都对它的CS进行关操作,半开操作它的NS。在流方向上释放一些兄弟种子,并关操作CS。只要子种子所在像素NS为开状态,就删除迹线,因为该子种子将成为下次发散过程的父种子;否则,保存迹线,并对该像素的NS进行关操作。弹性算法描述如下:

```
开操作 NS 矩阵;
for 每次发散过程
  用 NS 矩阵刷新 CS 矩阵;
  开操作 NS 矩阵;
  for 上次发散过程保存的每条迹线 l
    释放一个继承种子 i 并扩充迹线 l;
    调用置信号量操作函数(i,l)
    if 子种子所在象素的 NS 为关状态
      then 删除 l;
    else 保存 l 并关操作子种子所在象素的 NS;
    end if
  next 下条迹线
  for 每个象素
    if 象素的 CS 为开状态
    then 从该象素释放一个种子 r 并生成迹线 l;
      调用置信号量操作函数(r,l);
      if 子种子所在象素的 NS 为关状态
        then 删除 l;
      else 保存 l 并关操作子种子所在象素的 NS
      end if
    end if
  next 下个象素
next 下次发散过程
```

置信号量操作函数(种子 s,迹线 l)描述如下:

```
关操作种子 s 象素的 CS 信号量;
半开操作种子 s 象素的 NS 信号量;
for 对于种子 s 的每个兄弟种子
  if 兄弟种子的象素的 CS 为开状态
    then 沿释放兄弟种子并关操作 CS;
  end if
next 兄弟种子
```

2 算法的比较与实验结果

从图像的生成时间和质量对弹性算法和 UFLIC 算法进行比较。实验使用的数据源是由法国 Pau 大学的 Wilfrid Lefer 教授提供的。数据源 1 有 56 个时间步,256×256 个矢量数据,设种子的生命周期为 4 个时间步,则生成 53 帧图像,连续播放这些图像就形成了动画。数据源 2 有 72 个时间步,320×256 个矢量数据,种子的生命周期为 4 个时间步,则生成 69 帧图像。实验中两种方法仅纹理值发散过程的代码不同其他代码一样。在配置为 Intel® Celeron® CPU 2.10 GHz, Intel® 82845G/GL/GE/PE/GV Graphics Controller, 1GB 内存的工作站上进行实验。

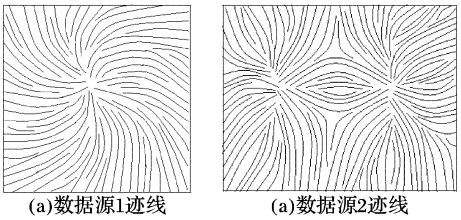
表 1 数据源 1 计算时间比较(单位:s)

方法	数据加载	发散过程	卷积	滤波	合计	加速比
UFLIC	28	3990	20	19	4057	3.2
弹性算法	27	1196	20	20	1263	3.2

表 2 数据源 2 计算时间比较(单位:s)

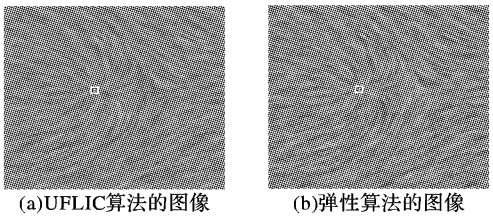
方法	数据加载	发散过程	卷积	滤波	合计	加速比
UFLIC	37	5081	27	26	5171	3.1
弹性算法	36	1539	28	27	1630	3.1

实验中各部分所花的时间如表 1、表 2 所示,两种方法在数据加载,卷积,滤波方面所花的时间相当,但种子控制算法在发散过程所花的时间比 UFLIC 算法的少得多,加速了 3 倍。图 3 是图 4、图 5 所示弹性算法图像没有卷积滤波前数据源生成迹线的。图 4、图 5 是从两种算法生成的动画中抓取的相同的时间步生成的图像,(a)取自 UFLIC 算法生成的动画,(b)取自弹性算法生成的动画,可以看出图像区别不大。实验证明了种子控制算法在不降低图像质量的前提下明显的缩短了计算时间。



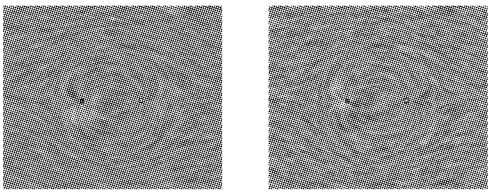
(a)数据源1迹线 (a)数据源2迹线

图 3 迹线图



(a)UFLIC算法的图像 (b)弹性算法的图像

图 4 数据源 1 两种算法生成的图像比较



(a)UFLIC算法的图像 (b)弹性算法的图像

图 5 数据源 2 两种算法生成的图像比较

3 结语

本文对 UFLIC 算法进行了改进,提出了弹性算法,通过控制种子的释放,重用、复制相关迹线,在不降低生成的图像的质量的前提下明显减少了生成图像的时间。此算法还可以进一步推广到三维非稳定矢量场。

参考文献:

[1] SHEN H-W, KAO DL. A New Line Integral Convolution Algorithm for Visualizing Time-Varying Flow Fields[J]. IEEE Trans. Visualization and Computer Graphics, 1998,4(2):98 - 108.

[2] WIJK JJV. Image Based Flow Visualization[A]. Proc. ACM SIGGRAPH'02[C]. July 2002.745 - 754.

[3] TELEA A, WIJJ JJV. 3D IBFV: Hardware-Accelerated 3D Flow Visualization[J]. Proc. IEEE Visualization'03, 2003.233 - 240.

[4] JOBARD B, ERLEBACHERS G, HUSSAINT MY. Lagrangian-Eulerian Advection of Noise and Dye Textures for Unsteady Flow Visualization [J]. IEEE Trans. Visualization and Computer Graphics, 2002, 8(3):211 - 222.

[5] WEISKOPF D, ERLEBQCHER G, ERTL T. A Texture-Based Framework for Spacetime-Coherent Visualization of Time-Dependent Vector Fields[A]. Proc. IEEE Visualization'03[C].2003:107 - 114.

[6] CABRAL B, LEEDOM L. Imaging Vector Fields Using Line Integral Convolution[A]. Proc. ACM SIGGRAPH'93[C]. Aug.1993: 263 - 270.

[7] PRESS WH, TEUKOLSKY SA, VETTERLING WT. C 数值算法[M].北京:电子工业出版社,2004.

[8] HEGE HC, STALLING D. LIC: Acceleration, Animation, and Zoom [A]. Texture Synthesis with Line Integral Convolution[C]. Course No. 8, Proc. ACM SIGGRAPH'97 Conf. Aug.1997.17 - 49.