密集导向流线算法研究

张 冬

(福建信息职业技术学院,福建 福州 350003)

摘 要: 稠密矢量场可视化有着广泛的应用。除了像箭线法、粒子追踪法这些标准方法外,基于纹理的可视化方法几乎能够展示矢量场中全部细节。我们提出了密集导向的流线算法,使用这个算法可以通过模拟卷积过程显示密集矢量场的方向、位置、局部流动速度。通过实验密集导向的流线算法比其他的 LIC 和 FastLIC 算法,生成的输出纹理速度更快。

关键词: 密集导向流线;矢量场;可视化; LIC

中图分类号: TP391.41 **文献标识码:** A **文章编号:** 1674-2109(2014)05-0057-04

一个可视化算法, 我们最关心两个方面的内容: 1、最终结果图像提供的信息的多少;2、可视化过程中的计算的成本有多大。基于这两点,我们对 LIC 方法进行了改进, 为了尽可能多的显示矢量场中的细节, 就应该选择一个稠密的纹理,此外,应该丢弃那些次要的特征;另一方面稠密的纹理在计算的时候又涉及了大量的像素点。为了减少像 LIC 中巨大的计算成本, 我们有必要对沿流线上的像素点的颜色进行研究,由这些高度相关的流线自己将整个的场线展示出来。此外,垂直于流线方向的像素点之间的相关性要尽量小。遵循这些原则,就有可能一次性的分配沿流线的像素点的颜色。还有,颜色的分布能够用来表示流体的方向, 而不需要像 FastLIC 中那样必须用动画作为流动的方向。

1 密集导向流线算法描述

图 1 和图 2 所示的是我们算法的伪代码。首先,设置好流线的长度(得到流线的长度);这个长度值作为计算每条流线的参考长度值(这个长度值可以将一条流线完全计算出来)。除了流线的长度值,算法中还需要沿流线的两个连续像素点的最大的色阶(得到最大色阶)和流线的数量。

第一部分只有图像的一部分(由用户预置)被填充,而第二部分提供了为剩余所有的像素点通过流线计算填充输出纹理。在第一部分中,为了流线的计算,我们用一个二维 Sobol 序列确定开始像素的一个数据集, 而在第二部分在依次填充相关像素时没有使用 Sobol 序列。因为只有这样的细微差别,所以我们将只针对第一部分讲述。

第一部分中, 计算的次数和流线的数量是一样的, 一个二维 Sobol 序列被用来选择流线的起始像素点(用 Sobol 提取像素点 P)。如果像素点没有被访问到和如果这点所在的流线没有被计算,一段程序就从 So-L 到 So+L 对流线进行计算, 这里的 So 是由 Sobol 选择的像素点。特别的,使用准随机 Sobol 序列可以将起始的像素点随机的展开,这是为了避免可能由标准的随机序列产生的聚合。另一方面, Sobol 序列和标准的随机序列一样可以在没有丢失统一属性的前提下被提前停止。可见,Sobol 序列同时具有了规则网格和标准随机序列两者的优点的。如果经过一点的流线不能够被计算出来,这可能是由于起始像素点位于图像的拐角处而且这个点的矢量场的方向与临近像素点所在的场的方向是相反而引起的,那么就给这个点赋

收稿日期: 2014-03-01

作者简介: 张冬(1973-),男,汉族,高级工程师,主要研究方向:计算机应用。

予一个随机的值。虽然这样可能产生人工的产物,但这些人工的产物可以用于检测出最终的错误。

为了表示出流体的速度,针对流线的每个点都计算出其上的矢量场的大小。其中最大的值用来限定沿流线方向相邻的两个点的灰度值的增量。一个一维 Sobol 序列用来选择一个灰度值,把这个灰度值分配给 So - L 处的像素点。如果 So - L + 1 的像素点的矢量大小等于最大值,那么它的灰度值等于 So - L 处的灰度值加上色阶;否则,增量被调整为最大值。

```
/* texture_out is the output image */
get streamline length;
get the max color step col_step;
get the number of streamlines;

/* first part (mandatory) */
for(i=0; i < number of streamlines; i++)
{
     extract a pixel p(coord_x,coord_y) by Sobol;
(b)  if(p not yet considered)
     {
          if the streamline in p can be computed
          {
               compute streamline;
               compute the max value of the vector field module
                along the streamline max_mod;
(a)            extract a color(col) between BEGIN_COL and
                END_COL from a single-dimension Sobol sequence;
               for each pixel s in the streamline starting from
                the first pixel
               {
                 set s as considered;
                 texture_out[s] = col;
                 /* loc_mod is the local value
                    of the vector field module */
(c)              col += ceil(col_step*loc_mod/max_mod);
               }
          }
          else
          {
               texture_out[p] = 127;  /* arbitrary choice */
          }
          set p as considered;
     }
}
```

图 1 密集导向流线算法的伪代码(第一步)

Fig.1 The pseudo-code of thick oriented stream-line algorithm(step1)

```
/* texture_out is the output image */
/* length is the streamline length */
/* col_step is the max color step */

/* second part (optional) */
for each pixel p in the image
{
(b)  if(p not yet considered)
     {
          if the streamline in p can be computed
          {
               compute streamline;
               compute the max value of the vector field module
                along the streamline max_mod;
(a)            extract a color(col) between BEGIN_COL and
                END_COL from a single-dimension Sobol sequence;
               for each pixel s in the streamline starting from
                the first pixel
               {
                 set s as considered;
                 texture_out[s] = col;
                 /* loc_mod is the local value
                    of the vector field module */
(c)              col += ceil(col_step*loc_mod/max_mod);
               }
          }
          else
          {
               texture_out[p] = 127;  /* arbitrary choice */
          }
          set p as considered;
     }
}
```

图 2 密集导向流线算法的伪代码(第二步)

Fig.2 The peseudo-code of thick oriented stream-line algorithm(step2)

通过一维 Sobol 序列得到的初始灰度值是为了使生成的纹理独立于操作系统(Unix、Windows 等)。特别是在动画的情况下,相同流线的相同起始点有相同的灰度值,保证了更好的视觉效果。初始灰度值位于开始颜色和终止颜色之间(实验证明最佳取值范围是 30—120)可以避免纯黑色灰度和避免灰度从 255 到 0 的跳跃。但是,只是起始点的值有限制,而流线上的一个普通点的灰度值可以设定为 0 到 255 的任意值;实验证明,这样一个“跳跃”对最终的质量和“可读性”不会造成太大影响。每当一个点被赋灰度值,这个点就会被标记为“已处理”,这样在后面的计算中,它不会被用作起始像素点。

值得一提的是,这个算法被分为两部分,原因是:1、第一部分可以用来作为稀疏纹理方法单独执行。2、当大量的像素点在第一步被计算后,再使用 Sobol 序列是无效的了,这时,适合采用扫描线方法。

2 密集导向流线算法的特点

(1)不需要输入纹理,需要的时候会随机生成。如图,见检查点(a)。

(2)在整个计算过程中,每个像素点只被计算一次。如图,见检查点(b)。不会产生重复的计算量。

(3)矢量场的所有的主要属性都会展示出来,如方向、位置、大小。算法中是通过灰度值沿流线的增量变化来表现流体速度的。如果矢量场局部的速度值比较高,则表示为此处有大的灰度值增量。见图中的检查点(c)。LIC 算法可以通过对图像着色来描绘矢量大小,但是这种后处理的方法增加了计算时间。

(4)由于矢量场中包含大量的流线,流线的交叉是是难免的,那么对临界点的选择是非常关键的。我们选择的 Sobol 序列,可以防止所谓的宏观结构,准确的找到这些临界点,使密集导向流线算法在不同的操作系统上,得到相同的输出纹理。

(5)可以通过对起始像素点数量的选择来控制计算的成本。如图,在具体算法中,我们可以对流线的数量进行更改。

(6) 密集导向流线算法由于使用了稠密的纹理,这样比 OLIC 算法可以更细致的可视出矢量场。

3　实验比较

利用 VC++和 OpenGL,在 cpu 为 3.2GHz,内存 4G 的 PC 机上,实现了密集导向流线算法。表 1 所示为,用密集导向流线算法、原始 LIC、FastLIC 分别对四种矢量场和不同的流线长度进行可视化的最终运算速度的比较。其中,2L+1 为积分长度。

表 1　密集导向流线算法和原始 LIC、FastLIC 性能比较

Tab.1　Comparison of thick oriented stream-line algorithm VS. LIC and fast LIC

矢量场	2L+1	LIC(s)	FastLIC(s)	密集导向流线算法(s)
Vortex 512×512	11	2.867	1.110	0.883
	21	6.316	1.433	0.983
	41	16.299	3.567	1.183
Attractor 256×256	11	0.717	0.210	0.200
	21	1.583	0.333	0.233
	41	4.033	0.833	0.317
Two points 256×256	11	0.717	0.250	0.233
	21	1.600	0.350	0.250
	41	4.050	0.833	0.283
LC image 512×512	11	2.567	1.117	0.950
	21	4.833	1.833	1.100
	41	9.266	3.050	1.450

如上表所示,密集导向流线算法比原始 LIC 算法运算速度平均提高了 30 倍,比 FastLIC 算法平均提高了 3 倍。

下图是对 Vortex 矢量场用密集导向流线算法(左)与 LIC 算法(右)进行可视化的结果对比。

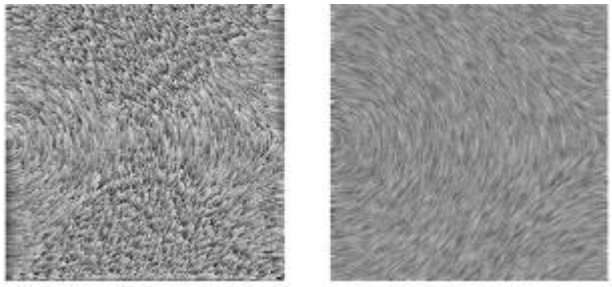

图 **3 Vortex** 矢量场中,密集导向流线算法(左)与 **LIC** 算法(右)的结果对比

Fig.3　Comparison of thick oriented stream-line(L) VS. LIC(R) in vortex vector field.

下图为 对 Attractor 矢量场用密集导向流线算法(左)与 LIC 算法(右)进行可视化结果比对。

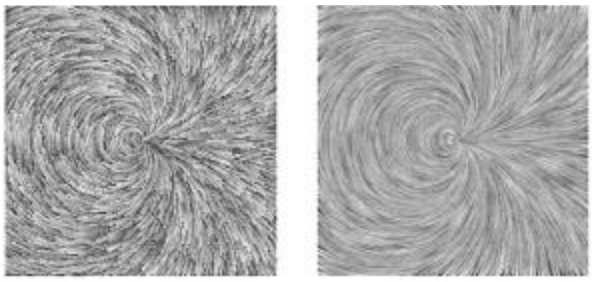

图 **4 Attractor** 矢量场中,密集导向流线算法(左)与 **LIC** 算法(右)的结果对比

Fig.4　Comparison of thick oriented stream-line(L) VS. LIC(R) in attractor field

下图为 对 Two points 矢量场用密集导向流线算法(左)与 LIC 算法(右)进行可视化结果比对。

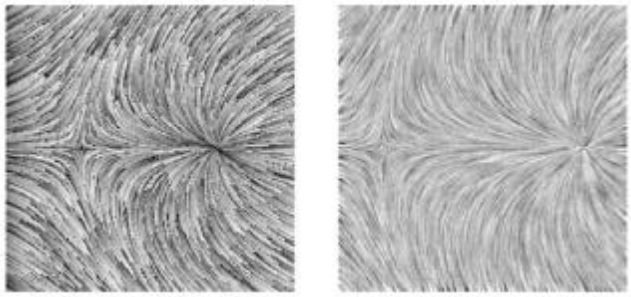

图 **5 Two points** 矢量场中,密集导向流线算法(左)与 **LIC** 算法(右)结果比对

Fig.5　Comparison of thick oriented stream-line(L) VS. LIC(R) in two points vector field.

由上面的三组可视化结果对比图说明密集导向流线算法比 LIC 算法可以更清晰和细致地表示出矢量场。

参考文献:

[1]　Slepian D, Wolf J. Noiseless coding of correlated information sources[J]. IEEE Transactions on Information Theory,1973,19(4) : 471-480.

[2]　Wyner A,Ziv J. The rate-distortion function for source coding with side information at the decoder[J]. IEEE Transactions on Information Theory,1976,22(01) : 1-10.

[3]　Van Wijk J J. Spot noise: texture synthesis for data visualization [J] . Comp uter Graphics, 1991, 25(04) : 309-318.

[4]　Cabral B, Leedom L. Imaging vector fields using line integral convolution[C] MComputer Graphics Proceedings, volume 27 of Annual Conference Series. New York, 1993: 263- 270.

[5]　唐泽圣.三维数据场可视化[M]. 北京: 清华大学出版社,1999.

[6]　胡星, 杨光. 流线可视化技术研究与进展[J].计算机应用研

究, 2002(05) : 8-11.

[7] Knight D, Mallinson G. A 3 -d streamline tracking algorithm using dual stream functions [C] MProceedings Visualization. 92. Boston: IEEE Computer Society Press, 1992: 62- 68.

[8] Cagnazzo M, Maugey T, Pesquet B. A differential motion estimation method for image interpolation in distributed video coding [C]/ / Proceedings of ICASSP2009. Washington, DC, USA: IEEE Computer Society Press, 2009: 1861-1864.

[9] Macchiavello B, Brandi F, Peixoto E, et al. Side -information generation for temporal and spatial scalable Wyner -Ziv codecs [J]. EURASIP Journal on Image and Video Processing, 2009, 2009: 1-11.

[10] 干宗良, 齐丽娜, 朱秀昌. Wyner-Ziv 视频系统中解码算法研究[J]. 信息处理, 2008, 24(04) : 609-613.

[11] Sühring K, Tourapis A. H264/AVC Reference Software J M16.0 [CP/OL]. [2010-04-14]. http: / /iphome. hhi. de/ suehring/tml / download / .

A Study on the Thick Oriented Stream - Line algorithm

ZHANG Dong

(Fujian Polytechnic of Information Technology, Fuzhou, Fujian 350003)

Abstract: The visualization of dense vector elds has important applications for scientic purposes. Beyond the standard methods, such as arrows and particle tracing, texture - based methods are able to show almost all the details of a eld.This paper presents the Thick Oriented Stream - Line algorithm, which can show direction, orientation and local ow speed even for dense vector elds by simulating the convolution process. A practical comparison of the performances of Thick Oriented Stream - Line algorithm vs. other visualizations algorithms (LIC and fastLIC) shows that the proposed algorithm can provide output textures faster than the other considered techniques.

Key words: thick oriented stream - line; vector feld visualization; scientic visualization; LIC