二维LIC矢量场可视化算法的研究及改进

詹芳芳[1] 胡 伟[1] 袁国栋[2]
(北京化工大学信息科学与技术学院 北京 100029)[1] (清华大学计算机科学与技术系 北京 100084)[2]

摘 要 线积分卷积(LIC)是一种针对矢量场的可视化方法。针对二维空间上的LIC算法进行了研究并提出了改进。首先,针对某些二维矢量场在用户关注区域矢量大小比较接近的问题,采用非线性的颜色映射法进行处理,最终的可视化结果可以突出用户感兴趣区域的矢量场特征。其次,从原始LIC算法的串行计算任务中提取出4个可以并行计算的子模块,并依托NVIDIA的CUDA架构实现了颜色增强LIC法的硬件加速。结果表明,加速后算法的加速比随着输入矢量场分辨率的增加而增加。因此,该算法适用于高分辨率二维矢量场的交互式可视化,且没有特别高的硬件要求,通用性较好。总之,新的算法较原始算法在视觉效果和性能上都有所改进。

关键词 矢量场可视化,线积分卷积,颜色增强,GPU加速
中图法分类号 TP391.41 **文献标识码** A

Improvement of 2D LIC Algorithm for Vector Field Visualization

ZHAN Fang-fang[1] HU Wei[1] YUAN Guo-dong[2]
(College of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China)[1]
(Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)[2]

Abstract Line integral convolution(LIC) is a method for vector field visualization. This paper presented an improved LIC technique for 2D vector field. Firstly, focused on the problem that some vector field's magnitude is too similar in regions of interest, we used nonlinear color mapping method to deal with it, and the final visualization result can highlight the vector field's characteristics in regions of interest. Secondly, we extracted four parallel sub-modules from the original LIC serial computing tasks and implemented color enhanced LIC's hardware acceleration relying on NVIDIA's CUDA architecture. The results show that, with the increase of the input vector fields' resolution, the speed of the accelerated algorithm increases as well. Therefore, the accelerated algorithm implemented in this paper can be well applied to interactive visualization of high-resolution 2D vector fields, and it has a good versatility, demanding no high hardware requirements. To conclude, the new method can improve visual quality with better performance.

Keywords Vector field visualization, Line integral convolution, Color enhancing, GPU acceleration

1 引言

矢量场可视化是科学计算可视化研究领域中的一个重要研究课题,它广泛应用于计算流体力学、航空航天、电磁场模拟、气象分析、地球系统模式等领域。

目前的矢量场可视化方法主要分为3类[1]:直接映射的矢量场可视化方法、基于积分流线的矢量场可视化方法以及基于纹理的矢量场可视化方法。各类方法都有自己的优缺点。直接映射的矢量场可视化方法,比如箭头法,容易实现,且可以同时表示矢量的大小和方向,但是由于空间分辨率的限制,该方法往往很难展现矢量场的细节信息。基于积分流线的矢量场可视化方法克服了箭头法的上述缺点,但是在该类方法的实现过程中,需要预先确定种子点的位置,一旦种子点的位置选择不合理,很可能会丢失用户感兴趣的特征[2]。基于纹理的矢量场可视化方法不仅可以展现矢量场的细节信息,而且不需要预先确定种子点,因此成为了研究热点之一。点噪声法[3]和LIC法(即线积分卷积法)[4]是两种比较常用的基于纹理的矢量场可视化方法。由于LIC法比点噪声法更能精确地表现矢量场的方向信息[5],因此本文针对二维空间上的LIC法进行了研究并提出了改进。

2 背景和相关工作

2.1 LIC算法原理

LIC(line integral convolution,即线积分卷积)[4],是由Brian Cabral和Leith Leedom在SIGGRAPH'93上提出的一种基于纹理的矢量场可视化方法。

到稿日期:2012-11-04 返修日期:2013-03-05 本文受国家高技术研究发展计划(863计划)(2010AA012402),国家自然科学基金项目(61003132)资助。

詹芳芳(1987—),女,硕士,主要研究方向为科学计算可视化,E-mail:fangfangzhan@gmail.com;胡 伟(1979—),男,副教授,主要研究方向为真实感图形实时绘制、照片与视频计算、基于多投影拼接的大屏幕组合显示;袁国栋(1977—),男,讲师,主要研究方向为计算机图形学、虚拟现实、数字图像处理。

算法的基本思想如图1所示，获取或生成原始矢量场之后，选择一个和原始矢量场分辨率一致的噪声(如白噪声)作为输入纹理，然后对该输入纹理沿着原始矢量场的切线方向进行低通滤波，得到的输出纹理便可以用来描述矢量场的方向信息。

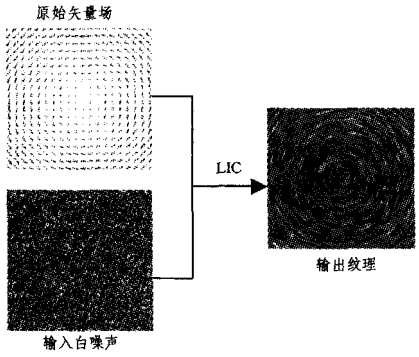


图1 LIC算法原理图

输出纹理中每个像素的灰度值均通过如下的计算得到。

首先，基于该像素沿其对应的矢量的正方向和反方向分别连续追踪，各生成一段长度为 L 的流线，再将流线上所有像素对应的输入噪声值按某个卷积核参与卷积，得到的结果即为该输出像素的灰度值。

对应于像素(x,y)的输出纹理灰度值的计算公式如下：

$$F(x,y)=\frac{\int_{-L}^{L}k(w)T(\rho(w))\mathrm{d}w}{\int_{-L}^{L}k(w)\mathrm{d}w} \tag{1}$$

式中，L 为事先确定的流线半长，$\rho(w)$ 为流线上的像素点$(-L\leqslant w\leqslant L)$，$T(\rho(w))$ 为流线上的点对应的输入纹理灰度值，$k(w)$ 为卷积核。

2.2 LIC算法的实现流程

根据上节的算法原理分析，算法实现的流程如下。

假设 C 表示卷积和，W 表示权重和，下标 p 和 n 分别表示沿矢量正方向和反方向进行积分卷积的过程，wgt 为当前噪声点对应的权重系数，$singleLen$ 为当前步的单步步长，$totalLen$ 为当前流线的总长，$step$ 表示当前方向上前进的步数，T 为输入纹理的灰度值，F 为输出纹理的灰度值。

对于输出图像中的每一个像素，计算灰度值。

1)初始化两个方向的卷积和以及权重和都为0：

$C_p=C_n=W_p=W_n=0$

2)分别计算两个方向的卷积和 C_p、C_n 及权重和 W_p、W_n：

a)初始化当前前进的步数和当前流线的总长：

$step=0$；$totalLen=0$

b)计算单步步长 $singleLen$；

c)更新当前流线的总长：$totalLen+=singleLen$；

d)根据卷积核的定义计算当前噪声点的权重系数 wgt；

e)更新当前的权重和：$W+=wgt$；

f)计算当前卷积和：$C+=T*wgt$；

g)更新前进的步数：$step+=1$；

h)若当前流线总长 $totalLen$ 小于之前确定的流线半长 L，继续步骤b)进行计算；

3)计算输出纹理的灰度值：$F=(C_p+C_n)/(W_p+W_n)$。

2.3 对LIC算法添加矢量强度信息的研究

由于在LIC算法的计算过程中要将矢量场的强度进行归一化，因此最终的可视化结果便丢失了矢量场的强度信息。

针对LIC不能显示矢量大小的不足，已有学者提出了一些改进。陆剑锋等人[6]通过对图像的对比度进行数量映射来显示矢量场大小；张文耀等[7]基于HSV颜色模型，将矢量场的强度大小映射为饱和度；陈丽娜[8]总结了多频噪声法、灰度表示法、色彩表示法、叠加矢量法等可以在LIC可视化结果的基础上显示矢量场大小的方法；朱宏玮等[9]详细阐述了基于RGB颜色空间的LIC增强显示法。

人眼对颜色的现实体验相当丰富，基于RGB颜色空间的增强显示法通常用暖色系的颜色代表大值，用冷色系的颜色代表小值，这也很符合人们的视觉习惯。

然而，基本的颜色增强LIC是将矢量场的大小线性映射到输出图像的颜色值上，当用户感兴趣区域的矢量场强度比较接近时，可视化结果中该区域的颜色便很相近，因此很难进一步区分感兴趣区域矢量场的大小分布。本文第3节针对此问题用非线性颜色映射方法对矢量场强度进行映射，最终的可视化结果在基本颜色映射法的基础上有所改进。

2.4 对提高LIC算法效率的研究

由LIC算法的原理可知，要生成LIC结果纹理，需要依次计算结果图像中每个像素的灰度值，且每次计算都要进行流线生成、积分卷积等步骤。因此，算法的效率严重依赖于输出图像的分辨率，远远不能满足高分辨率数据的交互式可视化需求。

Stalling和Hege[2]在LIC算法的基础上提出了FastLIC算法，将原始算法的效率提升了接近一个数量级。随后M. Zöckler等人[10]提出了并行LIC，该算法通过挖掘多帧图像中图像空间上的并行性来生成动画序列，使交互式矢量场可视化成为了可能。然而，该算法是在大规模并行分布式存储计算机上实现的，对硬件的要求过高，不具有通用性。M. Hlawatsch[11]等人提出的分级的线积分策略利用多核架构对算法中的积分计算过程进行加速，使得算法的计算时间从原先的和积分长度成线性增长关系下降到和积分长度成对数增长关系。尽管如此，当输入矢量场分辨率增加时，该算法的计算耗时依然呈线性增长。本文第4节应用NVIDIA的CUDA架构实现了LIC算法的GPU并行，最大化地利用了现代GPU的大量执行单元，大大提高了算法效率，且能很好地应用于高分辨率二维矢量场的可视化。

3 非线性颜色增强LIC

3.1 基本颜色增强LIC

3.1.1 基本颜色映射流程

设 $licValue$ 为输出图像中某个像素经过LIC计算得到的灰度值(0～255之间)，mag 为该像素对应的输入矢量的大小，$maxMag$ 和 $minMag$ 分别为输入矢量场中矢量强度的最大值和最小值，$colorTable$ 为预先定义的颜色映射表，则基本的颜色映射流程如下。

1)对基本LIC算法计算得到的灰度值进行归一化：

$nLicValue = licValue / 255$

2)对该像素对应的输入矢量的大小进行归一化：

$nMag = (mag - minMag)/(maxMag - minMag)$

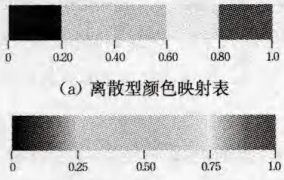
3)最终的输出颜色值计算如下：

$$outputRGB = colorTable[nMag] * nLicValue \quad (2)$$

其中输出颜色 $outputRGB$ 包含 3 个 0～255 的整数，分别表示该颜色的红、绿、蓝 3 个分量。颜色映射表 $colorTable$ 将归一化后的矢量大小 $nMag$ 映射到某个颜色值上。

3.1.2 颜色映射表的定义

上节所用到的颜色映射表分为离散型和渐变型两种，如图 2 所示。



(a) 离散型颜色映射表

(b) 渐变型颜色映射表

图 2 不同种类的颜色映射表

从图 2 中可以看出，离散型颜色映射表最终映射得到的颜色值将是颜色映射表中定义的 n 种颜色之一；而渐变型颜色映射表最终映射得到的颜色除了可以是颜色映射表中定义的 n 种颜色之外，还可以是相邻颜色通过插值计算得到的颜色。

设颜色映射表中有 n 种颜色，且存储在长度为 n 的一维数组 $color$ 中，即 $color[0]$－$color[n-1]$。其中每个元素都包含一种颜色的 RGB 值。$nMag$ 为经过归一化之后的矢量大小值，是一个 0～1 之间的浮点数。下面分两种情况将 $nMag$ 映射到一个颜色值。

当颜色映射表被定义为离散型时，映射过程如下：

1)如果 $nMag$ 为 1，则返回颜色映射表中表示最大值的颜色，即 $color[n-1]$；

2)否则，最终映射得到的颜色在颜色映射表中的序号为：

$destColorIndex = \lfloor nMag * n \rfloor$

则 $color[destColorIndex]$即为最终映射得到的颜色值。

当颜色映射表被定义为渐变型时，映射过程如下：

1)如果 $nMag$ 为 0，则返回颜色映射表中表示最小值的颜色，即 $color[0]$；

2)如果 $nMag$ 为 1，则返回颜色映射表中表示最大值的颜色，即 $color[n-1]$；

3)否则，设 $nMag$ 最终映射到 $color[i]$和 $color[i+1]$这两种相邻的颜色之间，$i \in [0, n-2]$，则 i 值可通过如下计算求得：

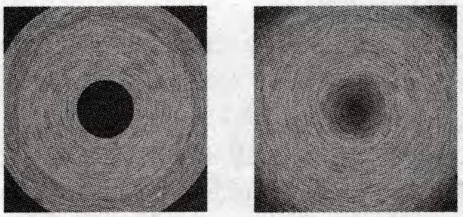
$i = \lfloor nMag * (n-1) \rfloor$

最终映射得到的颜色值为：

$$color = color[i] * (i+1-nMag * (n-1)) + color[i+1] * (nMag * (n-1) - i)$$

3.2 基本颜色增强 LIC 的效果及不足

图 3 所示为用基本颜色增强 LIC 法对同一个圆形矢量场进行可视化的效果图。该矢量场的矢量强度分布较均匀，因此可以很容易地区分出矢量场的强度分布。



(a) 使用离散颜色表　　(b) 使用渐变颜色表

图 3 对圆形矢量场用基本颜色增强 LIC 法可视化的效果图

但是，实际的矢量场强度往往分布不均匀，如图 4 所示。该图使用的数据为对中国东南沿海区域洋流进行模拟的结果，其中的纯黑色区域为陆地，其他区域为海洋。由于该矢量场的大小集中在值域较小的范围，因此可视化结果中大部分区域为蓝色(使用的颜色表中蓝色代表较小值)。在大片的蓝色区域里，很难再区分出矢量场的大小。

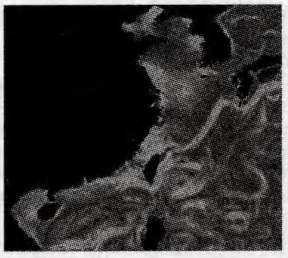


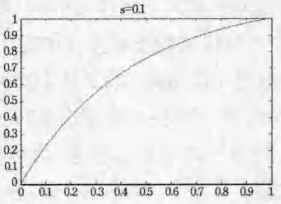
图 4 对一模拟洋流数据用基本颜色增强 LIC 可视化的效果图

3.3 非线性颜色增强 LIC

针对上述问题，本文对 3.1.1 节所述的基本颜色映射流程进行了修改，在对矢量场大小进行归一化之后，用式(3)所示的非线性函数对矢量场的强度 $nMag$ 重新进行映射，之后再将 $newMag$ 代入式(2)进行计算，得到最终的颜色值。

$$newMag = \frac{s^{nMag} - 1}{s - 1} \quad (3)$$

式中，s 为非线性映射因子(s 不等于 1)。式(3)的函数曲线如图 5 所示。



(a) $s = 0.1$

(b) $s = 10$

图 5 非线性映射函数曲线

从图 5 的函数曲线可以看出，当 s=0.1 时，0～0.5 的自变量映射到了一个较大的值域范围；而当 $s = 10$ 时，0～0.5 的自变量映射到了一个较小的值域范围。由于该函数严格单调递增，因此用其对矢量场大小进行映射不会改变原矢量场

中各矢量的大小关系。

在实际应用中，如果某个矢量场的大小集中在较小值附近，则可以使用小于 1 的 s 值对矢量大小进行映射，反之，则可以使用大于 1 的 s 值进行映射。对于图 4 中的矢量场，就可以使用小于 1 的 s 值将矢量强度中的小值映射到一个更大的范围，在本文的第 5 节将给出可视化结果。

4 基于 CUDA 架构的 GPU 加速 LIC

4.1 GPGPU 及应用于 LIC 算法的可能性

早期的图形处理单元(GPU)主要用来加速图形处理。但是随着 GPU 的发展，它也可以用来处理原本应该由 CPU 来完成的计算任务，因此人们提出了通用计算图形处理器(GPGPU)。现代的 GPU 不仅具有高度的可程序化能力，而且通常拥有相当高的内存带宽以及大量的执行单元，使得 GPU 被越来越广泛地应用在非图形处理运算领域。

虽然 GPU 可以用来加速通用计算，但是它只适合于处理那些可以高度并行化的简单任务。对于那些不能高度并行化的工作，应用 GPU 可能不会达到提高效率的目的。

LIC 算法的实现过程是高度并行化的。就其中的卷积计算过程来说，该过程对于输出图像中的每个像素依次进行计算，总的卷积计算时间和输出图像的分辨率几乎成正比。当利用 GPU 并行后，可以分别为每一个输出像素分配一个最小执行单元进行卷积计算。这样，所有输出像素的卷积计算过程可以同时进行，从而可以提高计算效率。除此之外，计算矢量场大小数组，对矢量场大小进行归一化，以及颜色映射等步骤，都是高度并行化的计算过程。因此，利用 GPU 对 LIC 算法进行加速是可能的。

4.2 CUDA 架构

CUDA[12]是 NVIDIA 的 GPGPU 模型。在 CUDA 架构下，一个程序被分为两个部分：在 CPU 上执行的部分以及在显示芯片上执行的部分。在显示芯片上执行的程序又称为核心程序(kernel)，也就是被提炼出来的高度并行的程序。每个核心程序都要被指明有多少个线程(thread)来执行。

4.3 LIC 算法在 CUDA 架构上的并行实现

对于颜色增强的 LIC 算法，本文从中提取了 4 个高度并行的计算任务，分别用一个核心程序来执行。设输出图像的分辨率为 $w \times h$，则对于每个核心程序，可以分别创建 $w \times h$ 个线程来执行它们。这 4 个计算任务分别是：

1)计算矢量场大小数组

由于在进行颜色映射时需要用到矢量场大小数组，因此要在对矢量场大小进行归一化之前将矢量场的原始大小保存到数组中。

2)对矢量场大小进行归一化

该步通过修正矢量场中矢量的两个分量，将矢量场中所有矢量的大小都转化为 1。

3)流线生成及卷积计算

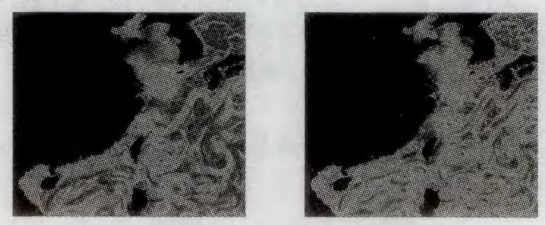
此步计算是 LIC 算法的核心，即对输入的矢量场和噪声纹理进行处理，生成 LIC 灰度纹理。

4)颜色映射

此步通过对矢量大小进行颜色映射，并叠加上 LIC 灰度纹理，输出最终的彩色纹理。

5 实验结果和分析

本文第 3 节的实验结果如图 6 所示，其中的两幅图像都是用非线性颜色增强 LIC 法对图 4 中的同一矢量场进行可视化的结果，区别在于它们使用了不同的非线性映射因子，分别为 0.1(a 图)和 0.01(b 图)。



(a) 非线性映射因子为 0.1　　(b) 非线性映射因子为 0.01

图 6　对图 4 中相同的矢量场数据用非线性颜色增强 LIC 可视化的效果图

对比图 6 和图 4 的可视化结果可以看出，使用非线性颜色增强后，图像中的颜色分布较之前更均匀。随着非线性映射因子的逐渐减小，之前的大片蓝色区域也逐渐减少，从而在矢量场强度比较集中的区域也能区分出矢量场的相对大小。

本文第 4 节的实验使用的数据是不同分辨率的圆形矢量场，分辨率范围从 100×100 到 3000×3000，横向分辨率每增加 100 测试一次。硬件平台为台式电脑，intel(R). Core(TM) i5-2300 处理器，2.80GHz 主频，4.0GB DDR3 内存，内存频率 1333MHz，NVIDIA GeForce GTS 450 显卡，GDDR5 1024MB 显存。软件平台为 Windows 7 Ultimate SP1(x86)操作系统，VC++2008、OpenGL 和 NVIDIA CUDA Toolkit 4.2。

图 7 所示为 GPU 加速前后算法的效率对比图。

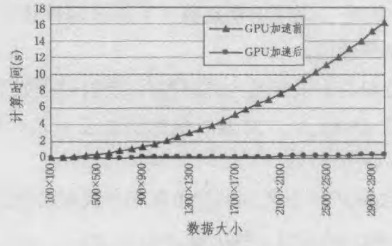


图 7　GPU 加速前后算法的计算效率对比

从图 7 可以看出，GPU 加速前，LIC 计算所需时间随着数据分辨率的增长几乎呈正比例增长；而 GPU 加速之后，随着数据分辨率的增加，LIC 计算时间只有微小增加。因此，相比 M. Hlawatsch 等人的分级的线积分策略对于 LIC 法的加速，本文的算法更适合于处理高分辨率数据。

图 8 所示为 GPU 加速之后算法加速比的变化趋势。

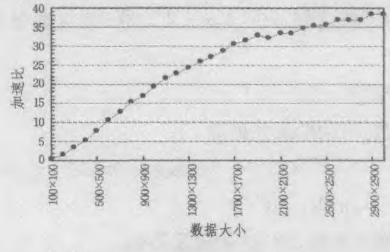


图 8　GPU 加速后算法的加速比

从图 8 可以看出，随着数据分辨率的增加，应用 GPU 加速后算法的加速比也增加(有个别例外)。当数据分辨率为

600×600 时，加速比就已经超过了 10 倍，因此当原始矢量场分辨率较高时，本文实现的算法比 FastLIC 更快。

相对于 M. Zöckler 等人的并行 LIC 对硬件的较高要求，本文实现的算法具有相对较好的通用性。

然而，从图 8 中也可以看到，当矢量场分辨率为 100×100 时，算法的加速比小于 1，GPU 的应用没能提高程序的效率。这是由于在执行核心程序之前要将核心程序所需要的数据从 CPU 的内存中复制到显示芯片的内存中，而在最后计算完成后又要将计算结果从显示芯片的内存复制到 CPU 的内存。在数据规模较小、本来的计算时间很短的情况下，数据来回复制的过程便占据了很大部分时间，使得应用 GPU 之后的效率反而有所下降。

从上文的图示和分析可知，基于 CUDA 架构的 GPU 加速 LIC 对于分辨率较小的矢量场（如采样点在 40000 个以下的矢量场）加速效果不明显甚至没有，但随着数据分辨率的增大，加速比逐渐增加，当数据分辨率达到一定时（如采样点在 360000 个以上），加速比超过 10，比 FastLIC 更快。因此，本文实现的 LIC 加速算法更适合于高分辨二维矢量场的可视化，且没有特别高的硬件要求，通用性较好。

结束语 本文首先改进了基本颜色映射 LIC 法，针对用户感兴趣区域矢量场强度比较接近的问题，利用非线性颜色增强 LIC 法进行可视化，使得在矢量场大小比较集中的区域也能区分出矢量的大小。接着利用 NVIDIA 的 CUDA 架构实现了 LIC 算法的 GPU 加速，加速后的算法尤其适用于高分辨率二维矢量场的交互式可视化，且对硬件的要求不高，通用性较好。本文所实现的改进算法还只适用于二维平面空间上的矢量场，对于三维空间场的情况尚没有进行研究。后续的研究工作可以针对三维空间上的 LIC 算法展开，以满足更多的应用需求。

参 考 文 献

[1] Laramee R S, Hauser H, Doleisch H, et al. The state of the art in flow visualization: dense and texture-based techniques[J]. Computer Graphics Forum, 2004, 23(2): 203-221

[2] Stalling D, Hege H. Fast and resolution independent line integral convolution[C]// Proceedings of the ACM SigGraph'95. New York ACM SIGGRAPH, 1995: 249-256

[3] Van Wijk J J. Spot noise: texture synthesis for data visualization [J]. Computer Graphics, 1991, 25(4): 309-318

[4] Cabral B, Leedom C. Imaging vector fields using line integral convolution[J]. Computer Graphics, 1993, 27(4): 263-272

[5] Leeuw W D, Liere R V. Comparing LIC and spot noise[C]// Proceedings of IEEE Visualization'98, IEEE Computer Society. 1998: 359-365

[6] 陆剑锋，潘志庚，张明敏. 基于图像对比度数量映射的矢量场可视化算法[J]. 系统仿真学报，2004，16(7)：1502-1505

[7] 张文耀，蒋凌霜. 基于 HSV 颜色模型的二维流场可视化[J]. 北京理工大学学报，2010，30(3)：302-306

[8] 陈丽娜. 基于线积分卷积可视化矢量场大小的研究[J]. 陕西理工学院学报：自然科学版，2009，25(3)：50-52，64

[9] 朱宏玮，姜国华，王宝智. 矢量场可视化线积分卷积方法研究与系统设计[J]. 计算机应用与软件，2010，27(4)

[10] Zöckler M, Stalling D, Hege H. Parallel line integral convolution [J]. Parallel Computing, 1997, 23(7): 975-989

[11] Hlawatsch M, Sadlo F, Weiskopf D. Hierarchical line integration [J]. IEEE Transactions on Visualization and Computer Graphics, 2011, 17(8): 1148-1163

[12] NVIDIA Corporation. NVIDIA CUDA C programming guide [EB/OL]. http://developer.nvidia.com/nvidia-gpu-computing-documentation, 2012-07-06

（上接第 242 页）

文中提出的支持换乘的多车辆合乘匹配问题还处在相对静态的阶段，即没有加入道路的信息，未来可以在该多车辆合乘匹配模型中加入交通流的影响，实现动态的多车辆合乘匹配；(2)对于算法的改进，该算法中的参数组合有待进一步的提高，需要大量的实验对算法进行优化。

参 考 文 献

[1] 李玲，谷寒雨，陈坚. 复杂 PDPTW 问题的插入启发式算法[J]. 计算机工程，2003，16(2)：65-69

[2] 贾永基，谷寒雨，席裕庚. 求解 PDPTW 问题的一种快速禁忌搜索算法[J]. 控制与决策，2004(01)

[3] Cristián E, Cortés. The pickup and delivery problem with transfers Formulation and a branch-and-cut solution method[J]. European Journal of Operational Research, 2010, 200(3): 711-724

[4] Parragh S N. Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem [J]. Transportation Research Part C-emerging Technologies, 2011, 19(5): 912-930

[5] Häme L. An adaptive insertion algorithm for the single-vehicle dial-a-ride problem with narrow time windows [J]. European Journal of Operational Research, 2011, 209(1): 11-22

[6] Herbawi W, Weber M. Comparison of multiobjective evolutionary algorithms for solving the multiobjective route planning in dynamic multi-hop ridesharing[A]// IEEE CEC[C]. New Orleans, June 2011

[7] 李文勇，王炜，陈学武. 公交出行路径蚂蚁算法[J]. 交通运输工程学报，2004(04)

[8] 柯良军，章鹤，尚可，等. 一类求解带时间窗的团队定向问题的改进蚁群算法[J]. 计算机科学，2012，39(4)：214-216

[9] Balseiro S R. An ant Colony algorithm hybridized with insertion heuristics for the Time Dependent Vehicle Routing Problem with Time Windows[J]. Computers & Operations Research, 2011, 38(6): 954-965

[10] 邵增珍，王洪国，刘弘，等. 多车辆合乘问题的两阶段聚类启发式优化算法[J]. 计算机研究与发展，2013

[11] Bullnheimer B, Hartl R F, Strauss C. An Improved Ant system Algorithm for the Vehicle Routing Problem[J]. Annals of Operations Research, 1999, 89(10): 319-328

[11] 黄永青，梁昌勇，张祥德. 基于均匀设计的蚁群算法参数设定[J]. 控制与决策，2006，21(1)