DOI:10.16515/j.cnki.32-1722/n.2021.02.003

一种基于方位角的轨迹数据压缩方法研究

苗丽娟,徐尚瑜,王洪欣,严 冬

(金陵科技学院软件工程学院,江苏 南京 211169)

摘 要:实时应用对海量轨迹数据的压缩时间要求越来越高,为了更好地平衡压缩时间和压缩率,提出一种基于方位角的轨迹压缩(azimuth-based trajectory simplification,ATS)算法。该算法利用轨迹点的方位角变化量控制整条轨迹的方向误差,支持在线压缩,具有线性时间复杂度。在真实的车辆轨迹数据集上进行实验,实验结果表明,ATS算法压缩率虽然略低于传统的最短路径(shortest path,SP)算法,但计算效率大幅度提升,辅助缓存少,而且能够很好地处理大数量级的轨迹,具有较强的伸缩性。

关键词:海量数据;轨迹压缩;线段简化;方位角;误差计算

中图分类号:TP348 **文献标识码**:A **文章编号**:1672-755X(2021)02-0012-07

An Azimuth-based Trajectory Data Simplification Method

MIAO Li-juan, XU Shang-yu, WANG Hong-xin, YAN Dong

(Jinling Institute of Technology, Nanjing 211169, China)

Abstract: The demand for compression time of massive trajectory data is increasing. In order to better balance compression time and compression ratio, an azimuth-based trajectory simplification (ATS) is proposed. The algorithm controls the direction error of the whole trajectory by using the azimuth variation of the trajectory point, supports on-line compression, and has linear time complexity. Experiments are carried out on real vehicle trajectory data sets. Experimental results demonstrate that the compression rate of ATS algorithm is slightly lower than that of traditional shortest path(SP) algorithm. However, the calculation efficiency of ATS algorithm has been greatly improved. The auxiliary cache is less. Further, ATS algorithm performs well on large data level trajectories and has a good scalability.

Key words: massive data; trajectory simplification; line simplification; azimuth; error calculation

随着物联网技术的发展和移动终端的普及,产生了海量的移动对象轨迹数据[1]。现代物流企业为了加强对运输车辆的有效监控和跟踪,多采用定位终端采集车辆轨迹数据。然而庞大的轨迹数据势必带来存储量大、传输慢和前端展示延时等问题。

目前线段简化压缩方法在轨迹数据压缩中应用较为广泛[2-4]。线段简化压缩方法又可以分为两类:一类是保持位置的轨迹简化(position-preserving trajectory simplification, PPTS)[5],另一类是保持方向轨迹简化(direction-preserving trajectory simplification, DPTS)[6]。常见的PPTS有Split[2-3]、Greedy[7]、Dead-Reckoning[8]等方法。Split方法的主要思想是在原始轨迹中找到一个满足给定条件的点,并将轨迹

收稿日期:2021-03-12
基金项目:江苏省农业科技自主创新资金项目(CX(17)2015)
作者简介:苗丽娟(1985—),女,江苏南京人,讲师,硕士,主要从事数据处理与分析、区块链技术应用方面的研究。

一分为二，然后继续对每条子线段迭代执行此操作。Greedy 方法的思想是判断当前轨迹点与第一轨迹点连接的线段是否超过给定条件，如果没有超过，则继续对下一轨迹点执行该操作；否则保留当前点，并以当前点作为第一轨迹点，继续执行以上操作。Dead-Reckoning 是一种在线方法，顺序读取每个轨迹点，根据轨迹点是否满足给定条件决定是否丢弃该点。以上这些方法容易丢失方向上产生剧烈变化的点，而这些点对于评估驾驶行为等具有非常重要的作用。相对于只考虑距离误差的 PPTS 方法，DPTS 方法近几年来受到了人们的关注，Long 等首次提出了 DPTS 方法，并证明 DPTS 方法在约束方向误差的同时，对距离误差也进行了制约[6]。Ke 等指出了 PPTS 方法存在的不足，并提出了解决角度偏差的方法，该方法具有线性时间复杂度，然而实验结果表明该方法的压缩率并不理想[9-10]。

针对以上算法存在的问题，本文提出了一种基于方位角的轨迹数据压缩方法，即 ATS 算法。ATS 算法的优点为：1）使得物体（例如飞机、车辆等）在运动过程中的每个位置点都具有方向信息；2）通过监控物体方位角的变化，以实现轨迹数据的有效压缩；3）支持实时在线压缩，且有很好的伸缩性。此外，实验结果表明无论在弯曲的城市路段，还是在大数据级轨迹上，ATS 算法的时间复杂度都为 $O(n)$，辅助缓存都为常量 C。

1　方向误差计算与传统算法

1.1　方向误差计算

PPTS 方法通常采用垂直欧氏距离[3]或同步欧氏距离[4]表示轨迹的压缩误差，DPTS 方法则采用方向误差[6]表示轨迹的压缩误差。

定义 1　矢量的方向[6]。由 X 轴正轴方向沿逆时针方向旋转至矢量 $\boldsymbol{P}_i\boldsymbol{P}_{i+1}$ 的角度，称之为 $\boldsymbol{P}_i\boldsymbol{P}_{i+1}$ 的方向，表示为 $\theta(\boldsymbol{P}_i\boldsymbol{P}_{i+1})$，取值范围为 $[0,2\pi)$。

2 个矢量的方向差 $\Delta(\theta_1,\theta_2)$，取值为 θ_1 沿逆时针方向旋转至 θ_2 的角度和 θ_2 沿逆时针方向旋转至 θ_1 的角度的最小值，即 $\Delta(\theta_1,\theta_2)=\min\{|\theta_1-\theta_2|,2\pi-|\theta_1-\theta_2|\}$。

定义 2　方向误差计算[6]。假设 $T'=\{P_{s_1},P_{s_2},\cdots,P_{s_m}\}$ 是原始轨迹 $T=\{P_1,P_2,\cdots,P_n\}$ 压缩后的轨迹，其中 $m\leqslant n$ 且 $1=s_1<s_2<\cdots<s_m=n$，$\boldsymbol{P}_{s_k}\boldsymbol{P}_{s_{k+1}}(s_1\leqslant s_k<s_{k+1}\leqslant s_m)$ 是 T' 上的 2 个相邻点构成的矢量，那么 $\boldsymbol{P}_{s_k}\boldsymbol{P}_{s_{k+1}}$ 产生的方向误差 $\varepsilon(\boldsymbol{P}_{s_k}\boldsymbol{P}_{s_{k+1}})$ 等于 $\boldsymbol{P}_{s_k}\boldsymbol{P}_{s_{k+1}}$ 与 T 上被 $\boldsymbol{P}_{s_k}\boldsymbol{P}_{s_{k+1}}$ 近似的所有矢量 $\boldsymbol{P}_h\boldsymbol{P}_{h+1}$ 方向差的最大值，表示如下：

$$\varepsilon(\boldsymbol{P}_{s_k}\boldsymbol{P}_{s_{k+1}})=\max_{s_k\leqslant h<s_{k+1}}\Delta[\theta(\boldsymbol{P}_{s_k}\boldsymbol{P}_{s_{k+1}}),\theta(\boldsymbol{P}_h\boldsymbol{P}_{h+1})] \tag{1}$$

压缩轨迹 T' 产生的误差 $\varepsilon(T')$ 表示如下：

$$\varepsilon(T')=\max_{1\leqslant k<m}\varepsilon(\boldsymbol{P}_{s_k}\boldsymbol{P}_{s_{k+1}}) \tag{2}$$

引理 1　有界的位置误差[6]。假设 T' 是原始轨迹 T 压缩后的轨迹，方向误差为 $\varepsilon(T')<\pi/2$，则 T 上的任意一点 $P_i(1\leqslant i\leqslant n)$ 的垂直欧氏距离 $\mathrm{dis}(P_i,P_i')\leqslant 0.5\times\tan(\varepsilon(T'))\times L_{\max}$，其中 $L_{\max}$ 为 T' 上两相邻点之间长度的最大值。

引理 1 使得 DPTS 方法控制方向误差的同时，也能对距离误差产生约束。因此方向误差压缩方法既限制了方向误差，也控制了距离误差。

1.2　SP 算法介绍

Long 等最早提出 DPTS 概念以及 SP(shortest path)算法[6]，SP 算法的主要思路是在方向误差阈值内找到路径最短的轨迹，这里路径最短的轨迹表示路径上包含的轨迹点个数最少。

定义 3　H_l 集合[6]。原始轨迹 $T=\{P_1,P_2,\cdots,P_n\}$，其中 $P_i(1\leqslant i\leqslant n)$ 是 T 上的点，如果 P_1 到 P_i 经过的最短边长为 l（l 为非负整数），则 P_i 表示长度为 l 上的某个点，T 中所有长度为 l 的点集合用 H_l 表示。

SP 算法在实际应用中，$H_l(l=0,1,2,\cdots,n)$ 集合存放长度为 l 的点，U 存放未处理的点。基于 H_l 集合迭代计算出 H_{l+1}，从 H_l 中取出每个点 P_i 依次与 U 中的每个点 P_j 计算出 $\varepsilon(\boldsymbol{P}_i\boldsymbol{P}_j)$，并判断 $\varepsilon(\boldsymbol{P}_i\boldsymbol{P}_j)$ 是否小于方向误差值，如果小于方向误差值，则 P_j 从 U 中移除并作为 H_{l+1} 中的点，直到找到终点 P_n，终止迭代计算。

从上述算法描述可以看出，SP 算法需要多次迭代计算矢量方向的方差，其时间复杂度为 $O(n^3)$，空间复杂度为 $O(n)$，计算过程复杂。且 SP 算法是一种离线数据压缩算法，不能处理在线数据流。

2 ATS 算法模型与分析

2.1 相关概念

SP 算法在计算方向误差时，涉及的轨迹点比较多，然而确定某两点连线的方向误差时需要迭代计算方向夹角，因此计算效率低。本节提出的 ATS 算法在兼顾方向性的同时，对局部范围内的点进行计算比较，从而提高算法的计算效率。下面给出 ATS 算法涉及的相关概念。

轨迹上任何一个点，都具有走势和方向信息，方位角是测绘中的概念，本文给出方位角的形式化定义。

定义 4 已知轨迹上的 2 个相邻点 O 和 P，点 P 作为点 O 的下一时刻相邻点，以点 O 的指北方向线为射线，按照顺时针方向旋转至目标点 P 时，则旋转的角度称为点 O 的方位角，表示为 $\mathrm{azm}(O)$。

从定义可知，方位角是具有方向性的，任何一点的方位角取值范围为 $[0°,360°)$，方位角可视为轨迹点的一个属性。

定义 5 轨迹上的 2 个相邻点 A 和 B，点 B 相对于点 A 的方向变化量表示为 $\Delta_{\mathrm{azm}}(B|A)$。$\Delta_{\mathrm{azm}}(B|A)$ 计算公式如下：

$$\Delta_{\mathrm{azm}}(B|A)=\begin{cases}\mathrm{azm}(B)-\mathrm{azm}(A) & |\mathrm{azm}(B)-\mathrm{azm}(A)|\leqslant 180°,\\ \mathrm{azm}(B)-\mathrm{azm}(A)-360° & \mathrm{azm}(B)-\mathrm{azm}(A)>180°,\\ \mathrm{azm}(B)-\mathrm{azm}(A)+360° & \mathrm{azm}(B)-\mathrm{azm}(A)<-180°\end{cases} \tag{3}$$

从定义 4 可知，相邻点方向的变化量有正有负，沿着顺时针方向的变化为正，沿着逆时针方向的变化为负。结合定义 4 和定义 5 可知，轨迹点的方向变化可以用方位角变化来衡量。

推论 1 给定一条轨迹点序列 $T=\{P_1,P_2,\cdots,P_n\}$，设点 $P_i(1\leqslant i\leqslant n)$ 的方位角为 α_i，那么相邻点 P_{i+1} 相对于点 P_i 的方向变化量为 $\Delta_{\mathrm{azm}}(P_{i+1}\mid P_i)$，任意点 $P_j(j>i)$ 相对于点 P_i 的方向变化量为 $P_i,P_{i+1},\cdots,P_{j-1},P_j$ 中两两相邻点方向变化量的累计和为 $\Delta_{\mathrm{azm}}(P_j\mid P_i)=\sum_{i+1\leqslant k\leqslant j}\Delta_{\mathrm{azm}}(P_k\mid P_{k-1})$。

证明：假设任意相邻点之间的方位角差小于等于 180°，即 $|\mathrm{azm}(P_{i+1})-\mathrm{azm}(P_i)|\leqslant 180°$，根据定义 4 和式(3) 可知，点 P_2 相对于 P_1 的方位角差 $\Delta_{\mathrm{azm}}(P_2\mid P_1)=\alpha_2-\alpha_1$，点 P_3 相对于 P_2 的方位角差 $\Delta_{\mathrm{azm}}(P_3\mid P_2)=\alpha_3-\alpha_2$。由图 1 可知，根据平行线定理和补角定理，点 P_3 与 P_1 的方向变化量 $\Delta_{\mathrm{azm}}(P_3\mid P_1)=\alpha_3-\alpha_1=(\alpha_3-\alpha_2)+(\alpha_2-\alpha_1)$，又因为 $(\alpha_3-\alpha_2)+(\alpha_2-\alpha_1)=\Delta_{\mathrm{azm}}(P_3\mid P_2)+\Delta_{\mathrm{azm}}(P_2\mid P_1)$，因此点 P_3 相对 P_1 的方向变化量等于相邻点 P_1、P_2、P_3 的方向变化量的累计和，即 $\Delta_{\mathrm{azm}}(P_3\mid P_1)=\Delta_{\mathrm{azm}}(P_3\mid P_2)+\Delta_{\mathrm{azm}}(P_2\mid P_1)$。同理可推，任意点 $P_j(j>i)$ 相对点 P_i 的方向变化量 $\Delta_{\mathrm{azm}}(P_j\mid P_i)=\alpha_j-\alpha_i=(\alpha_j-\alpha_{j-1})+(\alpha_{j-1}-\alpha_{j-2})+\cdots+(\alpha_{i+1}-\alpha_i)$，又因为 $(\alpha_j-\alpha_{j-1})+(\alpha_{j-1}-\alpha_{j-2})+\cdots+(\alpha_{i+1}-\alpha_i)=\Delta_{\mathrm{azm}}(P_j\mid P_{j-1})+\Delta_{\mathrm{azm}}(P_{j-1}\mid P_{j-2})+\cdots+\Delta_{\mathrm{azm}}(P_{i+1}\mid P_i)=\sum_{i+1\leqslant k\leqslant j}\Delta_{\mathrm{azm}}(P_k\mid P_{k-1})$，因此点 P_j 相对点 P_i 方向变化量等于点 $P_i,P_{i+1},\cdots,P_{j-1},P_j$ 之间所有相邻点的方向变化量的累计和，即 $\Delta_{\mathrm{azm}}(P_j\mid P_i)=\sum_{i+1\leqslant k\leqslant j}\Delta_{\mathrm{azm}}(P_k\mid P_{k-1})$。

2.2 算法步骤与分析

根据上述方位角的特性，本文提出了基于方位角的轨迹压缩方法 ATS 算法，总体思路是比较当前点和前一相邻点的方向变化量与阈值的大小，同时也比较当前点和上一特征点的方向变化量与阈值的大小。如果两者的方向变化量都小于阈值，则丢弃当前点，继续对下一个相邻点执行该步骤；否则，将当前点视为特征点，并保留特征点，继续对下一个相邻点执行该步骤。ATS 算法伪代码描述如下。

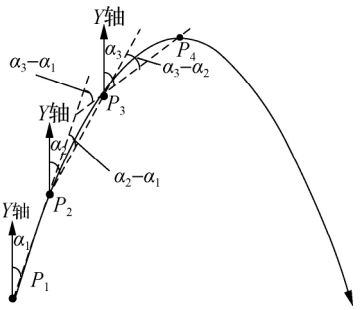


图 1 轨迹点之间的方向变化与方位角的关系

算法 1：ATS 算法

输入：原始轨迹点流 $T=\{P_1,P_2,\cdots\}$，方向阈值 δ

输出：压缩后的轨迹点序列集 T'

1. **初始化**：$T'\leftarrow\{P_1\}$，$P_s\leftarrow P_1$，$P_l\leftarrow P_1$，$P_i\leftarrow P_2$；
2. for P_i in T
3. 　if $P_{i+1}==null$ then
4. 　　$T'\leftarrow T'U\{P_l\}$；
5. 　else
6. 　　计算点 P_i 与前一轨迹点 P_l 的方向变化量 $\Delta_{azm}(P_i|P_l)$；
7. 　　计算特征点 P_i 与上一特征点 P_s 的方向变化量 $\Delta_{azm}(P_i|P_s)$；
8. 　　if $|\Delta_{azm}(P_i|P_l)|\geqslant\delta$ or $|\Delta_{azm}(P_i|P_s)|\geqslant\delta$ then
9. 　　$T'\leftarrow T'U\{P_i\}$，$P_s\leftarrow P_i$；
10. 　　$P_l\leftarrow P_i$；
11. return T'；

在上述算法的伪代码中，P_s 变量保存上一特征点的值，P_l 变量保存前一相邻点的值。对于轨迹起点，直接将其视为特征点，P_s 变量赋值为 P_1，P_l 赋值为 P_1，继续对下一轨迹点进行压缩处理。根据推论 1，当前点 P_i 相对于上一特征点的累计方向变化量 $\Delta_{azm}(P_i|P_s)$ 等于 P_s 与 P_i 的累计和，辅助缓存只需保存当前点、上一特征点 P_s、前一相邻点 P_l 以及当前点相对于上一特征点的方向变化量。ATS 算法的时间复杂度为 $O(n)$。从伪代码描述可以看出，ATS 算法既考虑了相邻点的方向变化量，又考虑方向的累计变化量，无论相邻点的方向变化量还是累计方向变化量超过系统允许的误差，都将当前点视为特征点，否则丢弃。相对于 SP 算法，ATS 算法处理流程较为简单，且支持实时轨迹数据流的在线压缩。

下面通过示例分析 ATS 算法与 SP 算法的不同。

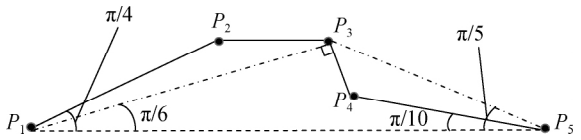


图 2　两种压缩方法的分析示图

示例：如图 2 所示，原始轨迹 $T=\{P_1,P_2,P_3,P_4,P_5\}$，假设阈值 $\delta=30°$即$\frac{\pi}{6}$。使用 SP 算法压缩的过程如下：首先 $H_0=\{P_1\}$，$U=\{P_2,P_3,P_4,P_5\}$，然后将 H_0 中的 P_1 与 U 中的 P_5 形成矢量 $\boldsymbol{P_1P_5}$，根据式(1)判断 $\varepsilon(\boldsymbol{P_1P_5})=\frac{2\pi}{3}>\delta$。继续将 H_0 中的 P_1 与 U 中 P_4 形成矢量 $\boldsymbol{P_1P_4}$，根据式(1)判断 $\varepsilon(\boldsymbol{P_1P_4})=\frac{\pi}{3}>\delta$。继续将 H_0 中的 P_1 与 U 中的 P_3 形成矢量 $\boldsymbol{P_1P_3}$，根据式(1)判断 $\varepsilon(\boldsymbol{P_1P_3})=\frac{\pi}{6}=\delta$，因此 $H_1=\{P_3\}$，$U=\{P_2,P_4,P_5\}$。同样将 H_1 中的 P_3 与 U 中的 P_5 形成矢量 $\boldsymbol{P_3P_5}$，根据式(1)判断 $\varepsilon(\boldsymbol{P_3P_5})=\frac{2\pi}{15}<\delta$，且 P_5 是终点，因此找到最短路径$\{P_1,P_3,P_5\}$即为压缩的轨迹点，循环结束。

使用 ATS 算法压缩的过程如下：根据方位角的定义 4 可知，图 2 中 P_1 点的方位角为 $45°$，P_2 的方位角为 $90°$，$|\Delta_{azm}(P_2|P_1)|=45°>\delta$，因此保留 P_2，并将上一特征点赋值为 P_2，继续执行 P_3。P_3 的方位角为 $150°$，$|\Delta_{azm}(P_3|P_2)|=60°>\delta$，因此保留 P_3，并将上一特征点赋值为 P_3，继续执行 P_4。P_4 的方位角为 $108°$，$|\Delta_{azm}(P_4|P_3)|=42°>\delta$，因此保留 P_4，并将上一特征点赋值为 P_4，继续执行 P_5。因为 P_5 为终点，所以保留点 P_5。因此压缩后轨迹为$\{P_1,P_2,P_3,P_4,P_5\}$。

从以上结果可以看出，SP 算法计算比较复杂，而且在同样精度的要求下，ATS 算法通过检测相邻轨迹点的方向变化以及方向的累积变化，能更敏锐地捕获角度变化的特征点，例如点 P_4 是发生方向明显变化的点。

3 实验与分析

3.1 实验配置

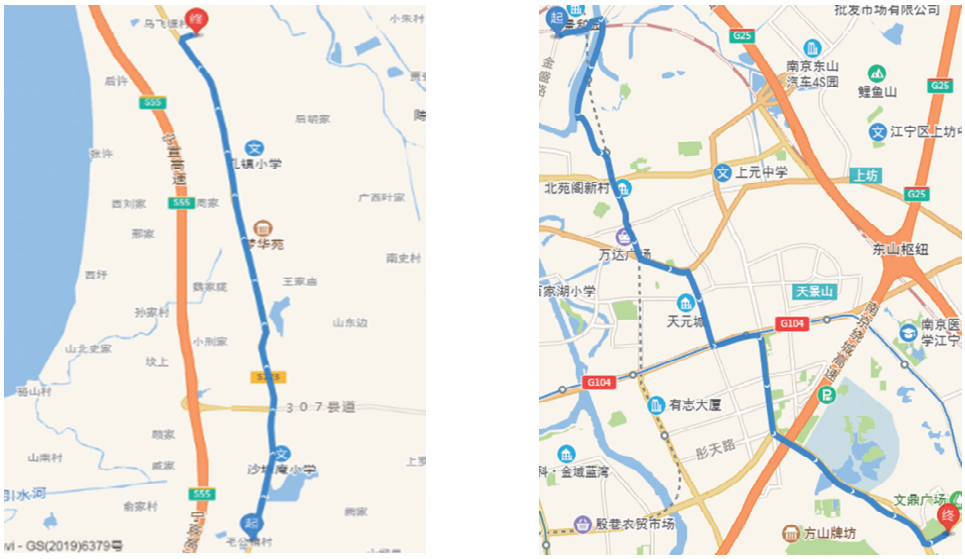
实验中算法的实现使用 Java 编程语言，使用的 jdk 版本为 jdk1.8.0_102。实验的硬件平台为 Intel Core i7@3.6 GHz CPU 和 8 GB RAM 的台式机，操作系统为 64 位 Window 7 系统。在南京某农产品基地的配送车辆上部署了一套车载监控终端并进行应用，实验数据来自该农产品基地配送车辆的轨迹数据集。实验中的压缩率表示压缩掉的轨迹点个数与原始轨迹的轨迹点个数比，用百分数的形式表示。

3.2 实验结果与分析

下面设置 3 个实验，对比本文提出的 ATS 算法与 SP 算法在 3 种条件下的压缩效果。

3.2.1 两种算法在两类路段的压缩效果对比分析

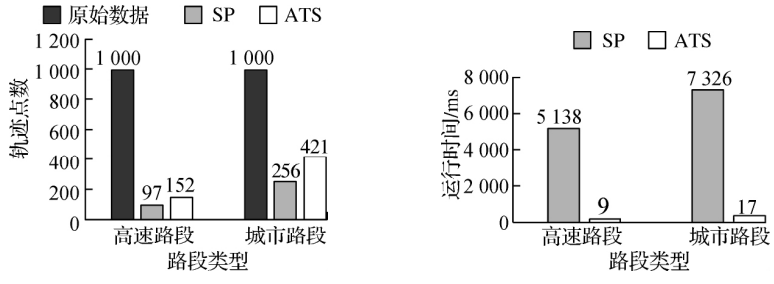
实验样本选取两类常见的车辆运行轨迹，即高速路段和城市路段（图 3）。高速路段较直，城市路段较为弯曲，两类运行轨迹采集的原始轨迹点数均设置为 1 000，方向误差阈值 δ 为 $15^\circ\left(\frac{\pi}{12}\right)$。图 3 是原始轨迹点集合的显示效果。



(a)高速路段　　(b)城市路段

图 3　两类路段原始轨迹点集合的显示效果

图 4 是两类运行轨迹分别采用 SP 算法和 ATS 算法压缩后的效果对比。从图 4(a)可以得出，高速路段使用 SP 算法压缩率为 90.3%，使用 ATS 算法压缩率为 84.8%；城市路段使用 SP 算法压缩率为 74.4%，使用 ATS 算法压缩率为 57.9%；同一算法高速路段的压缩率高于城市路段，表明算法的压缩率受运行轨迹弯曲程度的影响。ATS 算法的压缩率略低于 SP 算法，这是因为 ATS 算法能更加敏锐地捕获角度变化的特征点，与 2.2 节的分析一致，因此其压缩率要低于 SP 算法。从图 4(b)中可以看出，ATS 算法的运行效率远高于传统的 SP 算法，这是因为 SP 算法多次迭代计算过程比较复杂。



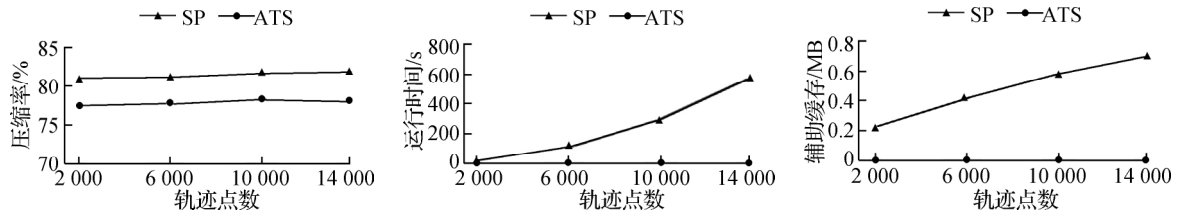
(a)轨迹点数　　(b)运行时间

图 4　两种算法在两类路段的压缩效果对比

3.2.2　两种算法在不同轨迹点数下的压缩效果对比分析

本实验不区分路段，由多条轨迹混合组成的不同数量级的轨迹进行实验，分别选取{2 000，6 000，10 000，14 000}的轨迹点构成轨迹，方向误差阈值 δ 为 $15°\left(\frac{\pi}{12}\right)$。图 5 是两种算法在不同轨迹点数下的压缩效果对比分析。

从图 5(a)中可以看出，ATS 算法的压缩率约为 77%，SP 算法的压缩率约为 81%。ATS 算法的压缩效率低于 SP 算法，这是因为 ATS 算法能更加敏锐地捕获角度变化的特征点。从图 5(b)可以看出，在 SP 算法中，随着轨迹点数的增多，运行时间呈急剧增加趋势；而在 ATS 算法中，运行时间随着轨迹点数的变化几乎没有波动，这是因为 SP 算法的循环迭代计算具有方向角度的复杂度，而 ATS 算法具有线性时间的复杂度。图 5(c)表明 ATS 算法的辅助缓存不会随着轨迹点数的增加而增加，而 SP 算法的辅助缓存随着轨迹点数的增加而增加。这是因为 ATS 算法仅需要缓存起点和上一相邻轨迹点信息，而 SP 算法是一种离线压缩算法，需要缓存所有的轨迹点，因此 SP 算法不太适用于大数量级的轨迹压缩，而 ATS 算法具有很好的伸缩性。

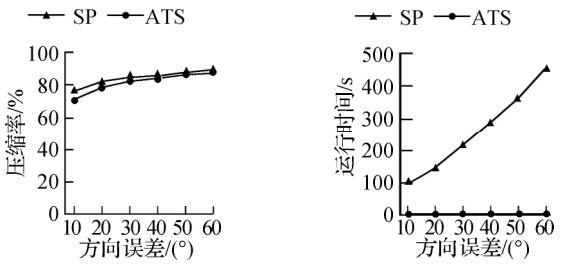


(a)压缩率　(b)运行时间　(c)辅助缓存

图 5　两种算法在不同轨迹点数下的压缩效果对比

3.2.3　两种算法在不同方向误差值上的压缩效果对比分析

本实验通过改变方向误差，对比 ATS 算法与 SP 算法的压缩效果，方向误差分别取值为{10°，20°，30°，40°，50°，60°}，轨迹点数固定为 5 000。图 6 是两种算法在不同方向误差值上的压缩效果对比。从图 6 可以看出，ATS 算法和 SP 算法的压缩率随着方向误差值的变大而呈增加趋势，SP 算法的运行时间随着方向误差值的变大而增大，但 ATS 算法的运行时间随着方向误差值的变大，几乎没有变化。因为 SP 算法随着方向误差值的变大，满足 $\varepsilon(\boldsymbol{P}_i\boldsymbol{P}_j)\leqslant\delta$ 的边数也逐渐变多，使得最短路径计算量变大；ATS 算法并不是通过计算方向误差，而是通过相邻方向变化量和方向累计变化量来实现的。



(a)压缩率　(b)运行时间

图 6　两种算法在不同方向误差值上的压缩效果对比

以上 3 个实验表明，轨迹的压缩率受轨迹弯曲程度和方向误差阈值的影响，ATS 算法的压缩率略低于 SP 算法。SP 算法的计算效率会随着轨迹点数和方向误差值的增大而变慢，而 ATS 算法具有很高的计算效率，即使在处理大数量级的轨迹时，该算法的计算效率和占用的缓存空间也不会发生大的变化，因此 ATS 算法具有简单、高效、伸缩性强的特点。

4 结 语

SP算法通过迭代方法计算每个采集点的方向变化；本文提出的ATS算法，将方位角视为采集点的一个属性，利用当前采集点分别与前一个采集点和上一个特征点的方位角变化量计算移动对象的方向变化，进而判别是否为特征点。在真实轨迹数据集上的实验结果表明，ATS算法大大提升了运算效率，支持在线压缩，且伸缩性强。

参考文献：

[1] 高强，张凤荔，王瑞锦，等. 轨迹大数据：数据处理关键技术研究综述[J]. 软件学报，2017，28(4)：959－992

[2] DOUGLAS D H，PEUCKER T K. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature[J]. Cartographica：the International Journal for Geographic Information and Geovisualization，1973，10(2)：112－122

[3] MERATNIA N，BY R D. Spatiotemporal compression techniques for moving point objects[C]//Proceedings of the 9th international conference on extending database technology. Heraklion：EDBT，2004：561－562

[4] POTAMIAS M，PATROUMPAS K，SELLIS T，et al. Sampling trajectory streams with spatiotemporal criteria[C]//Proceedings of the 18th international conference on scientific and statistical database management. Vienna：SSDBM，2006：275－284

[5] MUCKELL J，HWANG J，PATIL V，et al. SQUISH：an online approach for GPS trajectory compression[C]//Proceedings of the 2nd international conference and exhibition on computing for geospatial research & application. Washington D C：Association for Computing Machinery，2011：1－8

[6] LONG C，WONG R C，JAGADISH H V，et al. Direction-preserving trajectory simplification[J]. Proc VLDB Endow，2013，6(10)：949－960

[7] KOLESNIKOV A. Efficient online algorithms for the polygonal approximation of trajectory data[C]//2011 IEEE 12th international conference on mobile data management. Sweden：IEEE Computer Society，2011：49－57

[8] LANGE R，FARRELL T，DURR F，et al. Remote real-time trajectory simplification[C]//IEEE international conference on pervasive computing and communications. Texas：IEEE Xplore，2009：1－10

[9] KE B Q，SHAO J，ZHANG Y，et al. An online approach for direction-based trajectory compression with error bound guarantee[C]//Proceedings of the 18th Asia-Pacific Web conference. Suzhou：Springer International Publishing，2016：79－91

[10] KE B Q，SHAO J，ZHANG D，et al. An efficient online approach for direction-preserving trajectory simplification with interval bounds[C]//Proceedings of the 18th IEEE international conference on mobile data management. Daejeon：IEEE，2017：50－55

（责任编辑：谭彩霞）