基于城市交通监控大数据的行程时间估计

李文明[1]，刘芳[1]，吕鹏[1]，于彦伟[2]
1. 烟台大学计算机与控制工程学院，山东 烟台 264005；
2. 中国海洋大学计算机科学与技术系，山东 青岛 266100

摘要
随着智慧交通的发展，越来越多的监控摄像头被安装在城市道路路口，这使得利用城市交通监控大数据进行车辆行程时间估计和路径查询成为可能。针对城市出行的行程时间估计问题，提出一种基于城市交通监控大数据的行程时间估计方法UTSD。首先，将交通监控摄像头映射到城市路网，并根据交通监控数据记录构建有向加权的城市路网图；然后，针对行程时间估计，构建时空索引和反向索引结构，时空索引用于快速检索所有车辆的摄像头记录，反向索引用于快速获取每辆车辆的行程时间和经过的摄像头轨迹，这两个索引大大提升了数据查询和行程时间估计的效率；最后，基于构建的索引，给出一种有效的行程时间估计和路径查询方法，根据出发时间、出发地和目的地，在时空索引结构上匹配出发地与目的地共有的车辆，再利用反向索引，快速获得行程时间估计与车辆路线。使用某省会城市的真实交通监控大数据进行实验评估，所提方法UTSD的准确率比基于有向图的Dijkstra最短路径算法和百度算法分别提高了65.02%和40.94%，且UTSD在以7天监控数据作为历史数据的情况下，平均查询时间低于0.3 s，验证了所提方法的有效性和高效性。

关键词
城市交通监控大数据；时空索引结构；行程时间估计；路线推荐

中图分类号：TP391 文献标识码：A doi: 10.11959/j.issn.2096-0271.2021008

Travel time estimation based on urban traffic surveillance data

LI Wenming[1], LIU Fang[1], LYU Peng[1], YU Yanwei[2]
1. School of Computer and Control Engineering, Yantai University, Yantai 264005, China
2. Department of Computer Science and Technology, Ocean University of China, Qingdao 266100, China

Abstract
With the development of intelligent transportation, more and more surveillance cameras are deployed at the intersections of urban roads, which makes it possible to use the urban traffic surveillance data to estimate the vehicle travel time and query the route. Aiming at the problem of urban travel time estimation, a travel time estimation method based on the urban

traffic surveillance data was proposed, which is called UTSD. Firstly, the traffic surveillance cameras were mapped into the urban road network, and a directed weighted urban road network graph was constructed based on traffic monitoring data recording. Secondly, a spatio-temporal index and a reverse index structure were built for travel time estimation, the former was used to quick search the camera records of all vehicles, and the latter was used to fast obtain the travel time and the passing camera trajectory of each vehicle. These two indexes significantly improved the efficiency of data query and travel time estimation. Finally, based on the constructed indexing structures, an effective travel time estimation and path query method was given. According to the departure time, origin and destination, the vehicles with the same origin and destination were matched on the spatio-temporal index structure, and then the reverse index was used to quickly obtain the travel time estimate and vehicle route. Using the real traffic monitoring big data of a provincial capital city for experimental evaluation, compared with Dijkstra shortest path algorithm based on directed graph and Baidu algorithm, the accuracy rate of the proposed method UTSD is improved by 65.02% and 40.94%, respectively. In addition, the average query time of UTSD is less than 0.3 s when the 7-day monitoring data is used as historical data, which verifies the effectiveness and efficiency of the proposed method.

Key words
urban traffic surveillance data, spatio-temporal index structure, travel time estimation, route recommendation

1 引言

近年来，城市的快速发展，城市内的车辆数量不断增加，导致了交通拥堵、交通事故频发等一系列的交通问题。在这种环境下，如何提高日常生活中的城市出行效率成为出行用户的首要考虑问题，而作为交通服务中的一项基础功能，路径规划为人们尤其是不熟悉路况的人提供了重要的出行路线参考。在移动互联的大数据时代，大数据、人工智能、云计算、物联网、智能终端等先进技术的不断发展，为综合交通的一体化、智能化、智慧化发展提供了坚实的资源和技术支撑[1]。城市交通中的车辆路径查询和行程时间估计一直是交通行业的热门问题，目前的大部分路线推荐方法利用车辆的GPS轨迹数据进行车辆的路线规划以及行程时间估计[2]。而随着城市交通以及智慧交通的发展，越来越多的监控摄像头被安装在城市道路路口，以实时监控城市的交通状况，这些智能化的监控摄像头可以实时记录路口车辆的各种信息，如车牌号、经过时间以及行驶方向等。因此，无论车辆是否装有GPS等定位设备，相关人员都能够通过城市的交通监控系统获取整个城市所有车辆的行驶轨迹信息。利用城市交通监控中的摄像头数据进行城市出行路线规划和行程时间估计成为可能。

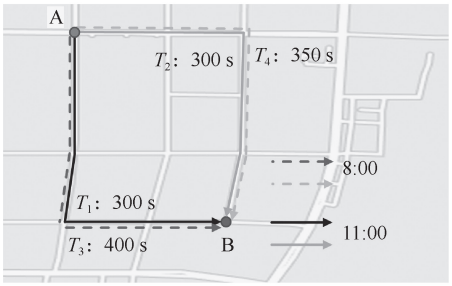
虽然城市交通监控系统的部署正在逐步完善，但是其安装和维护的成本等问题使得交通监控摄像头的数量及其覆盖的范围仍然有限。此外，交通监控数据是通过固定部署的监控摄像头获得的，观察到的车辆轨迹数据并不是完整的车辆行驶轨迹。因此利用交通监控摄像头的车辆数据进行查询时，会遇到以下3个挑战。

- 查询效率问题：路径推荐和时间估计是从某个范围区域内所有监控摄像头的车辆历史数据中查找车辆行驶轨迹和时间，涉及的数据规模非常大；此外，城市出行一般为即时查询，对查询效率有较高的要求。如果不能在短时间内对海量数据进行查询处理，就无法实时得到查询结果，

失去了城市出行的行程时间估计的意义。如何提高查询效率是使用交通监控大数据进行查询的一个亟待解决的问题。

● 路线选择问题：在实际生活中，车辆并不总是简单地从起始点出发，直接到达结束点，而是存在多条可能的轨迹路线。例如，车辆可能在行驶过程中停留，导致实际行程时间增加。如**图1（a）**所示，对于同样的起始点A和结束点B，3条轨迹（T_1、T_2、T_3）花费的时间不同，其中T_1为正常行驶轨迹，花费300 s；T_3由于绕行，花费了600 s。由此可知，并不是任意两个点之间都存在直接路线。在T_2和T_1轨迹长度相近的情况下，T_2的行驶时间为600 s，耗费的时间远大于T_1，这很可能是T_2的车辆在中途停留，导致行程时间增加。此外，车辆在不同的时间从相同的起始点到相同的结束点花费的时间也可能不同，这与对应路段的道路拥堵程度、交通事故情况以及其他的交通因素有关。在**图1（b）**中，对于同样的起始点A和结束点B以及相同的轨迹路线，在8:00出发比在11:00出发需要的行程时间更长。在11:00，从A点到B点，T_1和T_2的行程时间均为300 s；而在8:00，从A点到B点，T_3和T_4的行程时间分别为400 s和350 s，所花费的时间比11:00时更长。这很可能是因为人们大多集中在8:00去上班，道路拥堵，导致行程时间变长。

（a）车辆行驶轨迹不同



（b）车辆出发时间不同

图1 车辆出行的不同情况

● 噪声问题：由于一些原因，交通监控摄像头获取到的车辆轨迹信息会存在较多噪声。例如，在雾天或雨天，车牌号码识别不准确，导致车辆信息错误；部分摄像头因为故障没有记录到经过该路段的车辆信息，使得收集的车辆轨迹信息与车辆的真实轨迹不一致或车辆轨迹信息缺失等。

为了解决上述挑战，本文将城市道路路口的摄像头数据和聚类后的城市路网数据结合，将摄像头和车辆的轨迹信息匹配到城市路网中对应的路口上，形成路网数据库和摄像头数据库，包括摄像头的位置、编号，以及车辆ID、经过时间等数据。然后结合R树索引[3]构建时空索引和反向索引结构，时空索引用来根据位置信息和出发时间进行摄像头数据库的查询，反向索引用来快速获取每辆车辆的摄像头轨迹路线和行程时间。两种索引极大地提高了查询效率。当给定出发地和目的地位置信息及出发时间后，将查询位置信息与距其最近的摄像头进行匹配，对该摄像头及其所在路口摄像头的数据进行查询。根据摄像头的监控数据可以得到出发地和目的地所含有的共同车辆ID，再根据车辆ID利用反向索引得到其经过的摄像头编号和时间信息，经过时间排序，可以得到车辆的推荐路线和行程时间。

综上所述，本文主要贡献总结如下：

● 本文提出了一种基于城市交通监控

大数据的行程时间估计方法UTSD，可以实时进行城市出行的路线推荐和行程时间估计；

- 通过对城市监控摄像头数据构建时空索引和反向索引，加快车辆行驶轨迹的查询速度，从而快速得到车辆轨迹信息和对应的行程时间，极大地提高了监控大数据查询和行程时间估计的效率；
- 在某省会城市的真实交通监控摄像头数据上的实验结果验证了本文方法的有效性，相比对比算法，本文方法的性能有显著的提升。

2 相关工作

与本文相关的前期工作可以分为两类：时空数据管理和车辆行程时间估计。

2.1 时空数据管理

近年来，位置感知传感器在GPS、4G、5G网络等的应用中迅速普及，随着时间的推移，这些应用会产生大量的位置数据，这就需要合适的索引结构来实现对如此大的位置数据集的高效查询与处理。G树索引[4]在路网结构中应用较多，可以对路网进行有效的k近邻搜索[4]。R树被广泛应用于二维数据的索引[3,5]。同时，R树也可以将时间作为第三维度，变成三维R树（3DR-tree），如时空R树（STR-tree）和轨迹束树（TB-tree）[6]。而三维R树通过将时间维度划分为多个时间间隔，并链接到相应的空间索引，又衍生出多个版本的R树，如历史R树（HR-tree）[7]。此外，还有一些基于网格的索引，它们可以将一块区域空间划分为多个网格，如扁平起始树（CSE-tree）[8]和可扩展的高效轨迹索引（scalable and efficient trajectory index，SETI）[9]。划分后的网格由四叉树[10]或多维二叉树[11]构建，时间由混合B+树[12]或B树[13]进行索引。除了上述数据索引，反向索引[14]也是一种有效的轨迹查询索引。

有了合适的时空索引，才能有效地获取轨迹数据。这对于本文所要解决的城市交通监控大数据的数据查询问题更为重要。本文结合R树索引，构建了存储摄像头记录的时空索引，以及由车辆轨迹创建的反向索引，极大地提升了监控大数据的查询效率。

2.2 车辆行程时间估计

本文将关于车辆行程时间估计问题的相关工作分为3个主要类别：基于链路的行程时间估计、基于路径的行程时间估计和基于轨迹的行程时间估计。

（1）基于链路的行程时间估计

基于链路的行程时间估计方法是估计路网中车辆行程时间的经典方法。这种方法主要适用于静态的交通监控装置，如车辆感应装置[15]和摄像头装置[16]。对于浮动的车辆数据，如GPS数据，可以根据经过这些链路的车辆轨迹来推断各个链路的行驶时间。例如，Hofleitner A等人[17]基于交通流模型对链路的行程时间分布进行建模，并估计未来的行程时间。参考文献[18]使用最小二乘法根据仅包含结束点位置和有关行程的元信息（如行程距离）的出租车行程数据来估计链路的行程时间。还有一些方法[16-17]可被用来估计未来短期的链路行程时间，如动态贝叶斯网络算法[17]、模式匹配算法[16]、梯度增强回归树[19]和深度学习算法[20]等。

许多研究将估计的链路行程时间总和作为路径的行程时间，这种方法的缺点是没有考虑链路之间的时间花费，如车辆等待红绿灯的情况。为了有效解决该问题，Rahmani M等人[21]设计了一个非参数的链

路行程时间估计方法，以减少基于链路的行程时间估计模型的时间偏差。然而，该模型需要良好的道路网络动态覆盖，导致这种方法只能适用于特定的高速公路区域或一些特定的路线。

（2）基于路径的行程时间估计

参考文献[22]研究表明：直接测量道路上路径的行程时间比分开单独测量链路的行程时间准确度更高。然而并不是所有的路径都可以直接测量行程时间，因此基于大规模路径的行程时间估计需要将查询路径分解为较多的子路径。为了解决这个问题，Wang Y等人[2]研究了基于路径的行程时间估计子路径的长度和最小支持度的最优解。他们首先通过最小化子路径的总行程时间方差来计算最优解，并对每个子路径上的司机数量进行标准化；然后利用时空特征和驱动的张量分解，记录每个子路径的历史行程时间；最后构造了一个评价函数，并利用动态规划和后缀树优化来选择子路径的最佳组合。该方法取得了很好的实验效果，相比所有的基线算法和对比算法，其性能都有所提升。Jiang M Y等人[23]在参考文献[2]的基础上对算法进行改进，并用一种随机优化算法Adam（adaptive moment estimation）[24]替换了原算法中的SGD（stochastic gradient descent）算法[2]，进一步提升了算法的效率。

参考文献[25]提出了一种局部频繁共享算法，该算法从历史数据中学习一组频繁共享路径的局部拥堵模式。该算法可以从距离查询路段最近的轨迹中识别周围的当前拥堵模式，然后结合历史数据，估计未来的路径行程时间。该局部频繁共享算法的准确率比只使用历史轨迹的对比算法的准确率提高了20%~30%。

（3）基于轨迹的行程时间估计

基于轨迹的行程时间估计从历史数据中找到与所要查询的出发地、目的地和出发时间相近的历史轨迹，从而估计车辆的行程时间[26-27]。轨迹和路径的区别是：路径是指路网中某一段道路，轨迹是指车辆的行驶轨迹，可能会包含多段道路。通常假设相同端点之间的车辆轨迹相同或存在少量的替代轨迹。因此，这种方法更适用于预定路线的估计，如参考文献[28]的公交车出行。参考文献[27]根据匹配的历史轨迹计算查询行程的时间分布，并使用统计检验方法消除异常值。参考文献[26]结合周期性交通模式和与之匹配的历史轨迹对轨迹的行程时间进行调整估计。基于轨迹的行程时间估计也可以进行分层扩展，以实现某些路线的多样性。Yuan H T等人[29]提出了一种新的基于神经网络的估计模型，根据轨迹的起迄点（origin-destination，OD）以及轨迹的行程时间，设计了一种特殊的编码来与历史轨迹相关联，当输入OD进行查询时，可以通过编码直接得到对应的历史轨迹行程时间。

针对历史轨迹数据索引的查询，Ding Y C等人[30]提出了一种遍历轨迹聚合查询（traversal trajectory aggregate query）算法，对历史轨迹进行聚合存储，并提出了一种新的目标索引采样（targeted index sampling，TIS）框架，对数据进行采样查询，提高了查询效率和查询精度。

在参考文献[31]中，Yuan J等人将行程轨迹表示为一系列在城市热门地标之间的短途轨迹。行程时间为地标到地标的行程时间总和，再加上从起始点到第一个地标以及从最后一个地标到结束点所花费的时间。Yuan J等人[31]指出，尽管基于轨迹的行程时间估计的性能比基于链路的算法和基于路径的算法好，但在获得有用结果的同时，由于无法可靠地确定出租车行程的真实起始点和结束点，它无法被应用于没有位置标记的轨迹中。而本文提出的基于监控大数据的行程时间估计方法也是基于轨迹的行程时间估计方法，所提方法查询的

监控大数据为真实车辆数据，均可以根据摄像头记录来确定车辆的真实起始点位置和结束点位置（即轨迹经过的第一个交通监控摄像头位置和最后一个交通监控摄像头位置）。

3 问题定义

本节首先给出重要的概念定义，然后对基于交通监控大数据的行程时间估计问题进行定义。

定义1 路网。路网表示为$\mathcal{G}=(N,E)$，其中$N=\{n_1,n_2,\cdots,n_m\}$表示所有路口的集合，E表示路口之间所有路段的集合，$e_{i,j}\in E$表示从路口n_i到路口n_j的一条路段。需要注意的是，每个路段都是有方向的，也就是说，$e_{i,j}$不同于路段$e_{j,i}$。

定义2 摄像头记录。摄像头记录被定义为一个三元组$(\mathrm{veh_{id}},\mathrm{cam}_j,\mathrm{ts}_j)$，表示车辆$\mathrm{veh_{id}}$在$\mathrm{ts}_j$时刻经过摄像头$\mathrm{cam}_j$。

定义3 车辆轨迹。车辆$\mathrm{veh_{id}}$的轨迹是一个根据时间排序的摄像头记录序列，表示为$\mathrm{Tr_{id}}=\{\langle\mathrm{cam}_1,\mathrm{ts}_1\rangle,\langle\mathrm{cam}_2,\mathrm{ts}_2\rangle,\cdots,\langle\mathrm{cam}_i,\mathrm{ts}_i\rangle,\cdots,\langle\mathrm{cam}_n,\mathrm{ts}_n\rangle\}$，其中$\langle\mathrm{cam}_i,\mathrm{ts}_i\rangle$表示车辆$\mathrm{veh_{id}}$在$\mathrm{ts}_i$时刻经过摄像头$\mathrm{cam}_i$。

由定义3可知，车辆轨迹由车辆经过的所有摄像头的时间序列构成，本文将历史监控摄像头数据中所有含有同一车辆$\mathrm{veh_{id}}$的摄像头记录序列记为该车辆的车辆轨迹$\mathrm{TR_{id}}$，其部分车辆轨迹记为$\mathrm{Tr_{id}}$。所有车辆的轨迹集合记为TRs。

本文将基于城市交通监控大数据的行程时间估计问题定义如下。

问题定义：给定路网$\mathcal{G}$、路网上所有的摄像头记录（即所有的车辆轨迹集合TRs）、起始点位置$\mathrm{O}(\mathrm{lon_s},\mathrm{lat_s})$、结束点位置$\mathrm{D}(\mathrm{lon_e},\mathrm{lat_e})$以及出发时间ts，目标是得出路线推荐以及行程时间估计Time。

表1给出了本文用到的主要符号及其含义。

4 基于城市交通监控大数据的行程时间估计方法

4.1 总体框架

图2给出了本文提出的基于城市交通监控大数据的行程时间估计方法的总体框架，该框架主要包括三部分：数据预处理、构建数据索引以及行程时间估计。

（1）数据预处理

数据预处理是指在数据查询和构建数据索引之前对原始交通监控数据进行转换和筛选处理，主要包括将摄像头映射到路网上、从监控摄像头数据中提取车辆轨迹两部分。

（2）构建数据索引

当数据预处理完成之后，利用处理好的监控数据构建数据索引，提高数据查询效率。数据索引包括时空索引和反向索引两部分。时空索引是利用所有摄像头记录来检索车辆的索引，给定摄像头编号和时间，找到所有在该时间经过此摄像头的车辆。反向索引是根据数据预处理得到的车辆轨迹构建的，它是通过车辆来查询摄像头的索引，主要用于查询每辆车经过的摄像头组成的车辆轨迹和对应的行程时间。给定车辆ID和出发时间，可以找到该车辆在出发时间及以后所经过的摄像头。

（3）行程时间估计

给定起始点位置$\mathrm{O}(\mathrm{lon_s},\mathrm{lat_s})$、结束点位置$\mathrm{D}(\mathrm{lon_e},\mathrm{lat_e})$和出发时间ts，首先将起始点位置和结束点位置分别匹配到对应的摄像头；然后利用时空索引查询经过起始和结束摄像头的所有车辆数据，找出它们含有的车辆ID相同的车辆数据；接下来采用反

向索引，找到这些车辆经过起始摄像头之后的车辆轨迹；最后，从这些车辆轨迹中筛选出符合条件的车辆轨迹，每条车辆轨迹对应的行程时间是根据车辆经过起始摄像头和结束摄像头的时间差得到的。

4.2 数据预处理

4.2.1 摄像头映射

摄像头映射包括摄像头到路网位置的映射，以及摄像头位置到路网中对应路口的匹配两部分。首先，从开源地图平台OpenStreetMap[32]上获取真实路网。根据定义1，一个路网包括路口集合以及路口之间的路段集合。通常来说，监控摄像头被部署在邻近路口处的位置，用来获得所有经过此路段的车辆信息。因此，通过摄像头的位置信息（如经度和纬度）将摄像头映射到路网上，再匹配到相应路口上，就能够获得车辆在路网上对应的行驶记录。

4.2.2 提取车辆轨迹

根据定义2，在交通监控数据中，每个摄像头监控记录可表示为(veh_{id},cam_j,ts_j)，表示车辆veh_{id}在ts_j时刻经过了摄像头cam_j。由定义3可知，当车辆的ID相同时，根据时间排序后的摄像头记录序列可以表示车辆轨迹。为了保证车辆轨迹的连续性，本文将历史监控摄像头数据中每辆车经过的所有摄像头作为一条车辆轨迹（即TR_{id}）。

4.3 构建数据索引

4.3.1 构建时空索引

构建时空索引分为构建空间索引和构建时间索引两部分。

如**图3**所示，首先构建空间索引，生成城市路网后，先利用摄像头编号和位置信息，将摄像头映射在路网上，再将每个摄像头匹配到距离其最近的路口上，并创建摄像头编号至路口的索引，以便在读取监控数据时根据摄像头编号将摄像头记录存储到对应路口的摄像头中。

一般情况下人的活动是以天为周期的，很多研究[33-34]验证了这一点，相应的城市交通情况也会以天为周期出现变化。因此，在构建时间索引时，本文根据一天24 h将时间索引平均分为24个时隙层，每个时隙层索引的范围是1 h。每个时隙层中都包含一个完整的城市路网空间索引。如**图3**所示，在存储摄像头记录时，先根据时间确定摄像头记录所在时隙层，再根据摄像头编号将摄像头记录存储到对应路口的摄像头中。

表1 主要符号及其含义

符号	含义
lon_s、lat_s	起始点位置经度、纬度
lon_e、lat_e	结束点位置经度、纬度
veh_{id}	车辆ID
n_i	路网中的路口节点
n_o	起始点位置匹配到的路口节点
n_d	结束点位置匹配到的路口节点
ts	车辆的出发时间
te	车辆的结束时间
Time	路径对应的行程时间
$Path_{top\,n}$	最短路径对应的车辆轨迹集合

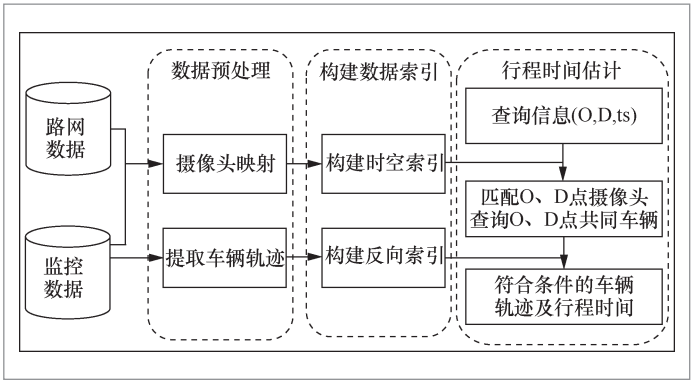


图2 总体框架

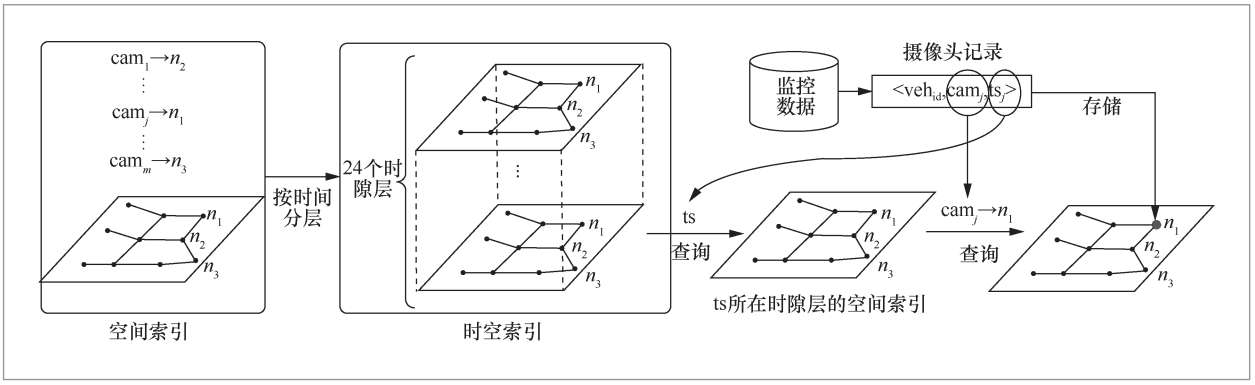


图 3 构建时空索引

4.3.2 构建反向索引

由定义2和定义3可知，对于车辆veh_{id}的车辆轨迹Tr_{id}，由车辆veh_{id}和经过时间ts_j可以唯一确定其经过的摄像头cam_j。此外，因为车辆轨迹是按时间排序的摄像头记录序列，还可以得到车辆veh_{id}在时间ts_j之后经过的所有摄像头以及对应的经过时间。因此，可以根据车辆轨迹的摄像头记录序列，建立根据车辆ID查找所经过摄像头的反向索引。

如**图4**所示，对于车辆veh_{id}的一条车辆轨迹 $Tr_{id}=\{\langle cam_1,ts_1\rangle,\langle cam_2,ts_2\rangle,\langle cam_3,ts_3\rangle,\langle cam_4,ts_4\rangle\}$，其中$cam_1$、$cam_2$、$cam_3$、$cam_4$分别匹配到路口$n_1$、$n_2$、$n_3$、$n_4$上，给定出发时间$ts_1$，可以得到车辆$veh_{id}$在时间$ts_1$经过的摄像头$cam_1$，以及在时间$ts_1$之后，车辆在时间$ts_2$、$ts_3$、$ts_4$分别经过的摄像头$cam_2$、$cam_3$、$cam_4$。

在构建反向索引前，对于车辆的所有摄像头记录，各个摄像头记录之间是相对独立的，无法确定单独的一条摄像头记录是否为噪声数据。然而，在构建反向索引之后，可以根据车辆轨迹中相邻摄像头记录之间的关系来过滤噪声数据，比如由于雾天导致车牌号码识别错误或者车辆在相邻摄像头记录之间有较长时间的停留等导致的噪声数据，详细描述见第4.4节。

4.4 行程时间估计

在时空索引和反向索引构建完成之后，给定起始点位置$O(lon_s,lat_s)$、结束点位置$D(lon_e,lat_e)$以及出发时间ts，就可以根据时空索引和反向索引对查询点进行车辆的路线推荐和行程时间估计。

在城市路网中，一个路口一般安装多个摄像头，因此在将摄像头映射到路网上时，可能有多个监控摄像头匹配到同一个路口。这些匹配到同一路口的摄像头产生的摄像头记录都表示车辆经过了摄像头所匹配的路口。因此本文将匹配到同一路口的摄像头组成一个摄像头集合，当车辆经过一个摄像头时，表示车辆经过了该摄像头所匹配路口的所有摄像头。设定$c_o=\{cam_o^1,cam_o^2,\cdots,cam_o^m\}$表示起始点位置$O(lon_s,lat_s)$匹配到的摄像头所在路口的摄像头集合；$c_d=\{cam_d^1,cam_d^2,\cdots,cam_d^n\}$表示结束点位置$D(lon_e,lat_e)$匹配到的摄像头所在路口的摄像头集合。

这样在查询OD匹配的摄像头时，可以查询到匹配摄像头所在路口的所有摄像头记录，提高了查询效率和查询范围。

算法1展示了路线推荐和行程时间估

计的伪代码。

算法1 路线推荐和行程时间估计算法

输入： 起始点位置O(lon_s,lat_s)、结束点位置D(lon_e,lat_e)、出发时间ts

输出： 推荐路径$Path_{top\boldsymbol{n}}$和对应的行程时间

1. Determine $\mathcal{G}_o=(N_o,E)$ at time slot ts;
2. Determine $\mathcal{G}_d=(N_d,E)$ at time slot te;
3. Map O to cam_o, map D to cam_d;
4. for n in N_o do
5. if cam_o in n do
6. $c_o \leftarrow n$;
7. for n in N_d do
8. if cam_d in n do
9. $c_d \leftarrow n$;
10. for each cam_o in c_o
11. for each rec_o in cam_o do
12. $Veh_o \leftarrow veh_{id}$;
13. for each cam_d in c_d
14. for each rec_d in cam_d do
15. $Veh_d \leftarrow veh_{id}$;
16. $Veh_c \leftarrow Veh_o \cap Veh_d$;
17. for each veh_{id} in Veh_c do
18. Path $\leftarrow Tr_{id}$;
19. calculate $Time_{id}$ of Tr_{id};
20. $Path_{top\boldsymbol{n}} \leftarrow$ Path with $Time_{id}$
21. for Tr_{id} in $Path_{top\boldsymbol{n}}$ do
22. $n_{id} \leftarrow cam_{id}$;
23. return $Path_{top\boldsymbol{n}}$, Time;

如算法1所示，首先，根据出发时间ts确定n_o所在时隙层的路网$\mathcal{G}_o=(N_o,E)$。确定ts对应的时隙层之后，会出现一个问题，即要查询的行程时间超过了一个时隙层的范围（即行程时间超过1 h），导致车辆实际的结束时间与出发时间不在同一个时隙层，而是在下一个时隙层或者后面的时隙层。此时，如果只查找ts所在时隙层路网中的摄像头记录，会影响查询精度或者无法得到车辆在结束时间所在时隙层的摄像头记录。为了解决这个问题，这里新增了一个结束时间te=ts+tp，用来预测n_d所在时隙层的路网$\mathcal{G}_d=(N_d,E)$。te是由出发时间ts和通过基于有向图的Dijkstra最短路径算法（以下简称最短路径算法）[35]得到的行程时间tp相加后得到的大致预测时间，并不是真实的行程时间。tp被用来预先估计OD的行程时间，预测te所在时隙层，以提高本文所提方法的查询精度。通过最短路径算法得到的行程时间tp的计算方式如下：给定起始点位置O(lon_s,lat_s)、结束点位置D(lon_e,lat_e)和路网$\mathcal{G}=(N,E)$，同时，路网中的每个路段$e_{i,j}$附带了距离权重和时间权重。距离权重表示从路口n_i到路口n_j的行驶距离，时间权重表示从路口n_i到路口n_j的行驶时间，这里的行驶时间为从路口n_i到路口n_j的历史车辆轨迹的平均行驶时间。在路网$\mathcal{G}=(N,E)$中分别匹配到距离O、D最近的路口为n_o和n_d。根据每个路段的时间权重，利用最短路径算法得到从路口n_i到路口n_j的行程时间tp。

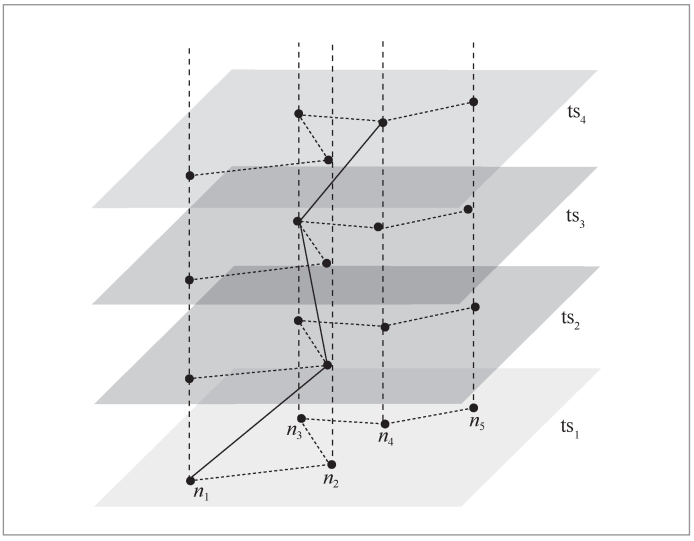


图4 反向索引的构建示例

例如，当出发时间为10:40时，查询n_o的时隙层范围为10:00—11:00的时隙层。若tp时长为50 min，那么te为11:30，查询

n_d的时隙层范围为10:00—11:00的时隙层以及11:00—12:00的时隙层。

如第3~9行所示，确定了要查询的时隙层的范围之后，根据所给的起始点位置O(lon_s,lat_s)和结束点位置D(lon_e,lat_e)，在路网中分别匹配到距离最近的起始点摄像头cam_o和结束点摄像头cam_d。然后，找到cam_o和cam_d所在路口的摄像头集合c_o和c_d。这里需要注意，从起始点位置到达起始点所匹配的摄像头位置以及从结束点所匹配的摄像头位置到达结束点位置的这两段路程，会分别根据ts和te所在时隙层，以及该路程在路网中所处的路段中所占的长度比例来计算时间，最后与主路程相加。

如第10~16行所示，在确定了c_o和c_d之后，分别查询c_o和c_d中各摄像头包含的摄像头记录rec_o和rec_d，并得到在该时隙层中经过起始点和结束点摄像头的车辆集合Veh_o和Veh_d。然后，求出集合Veh_o和Veh_d的交集Veh_c，即起始路口和结束路口的摄像头记录中共同含有的车辆ID。

如第17~23行所示，得到共同车辆ID之后，根据反向索引，可以得到每辆车veh_{id}的部分车辆轨迹Tr_{id}。这里 $\mathbf{Tr_{id}}=\{\langle \mathbf{cam_1},\mathbf{ts_1}\rangle,\langle \mathbf{cam_2},\mathbf{ts_2}\rangle,\cdots,\langle \mathbf{cam_i},\mathbf{ts_i}\rangle,\cdots,\langle \mathbf{cam_n},\mathbf{ts_n}\rangle\}$。$Tr_{id}$的起始摄像头$cam_1$为$c_o$中的摄像头，出发时间$ts_1$在ts确定的时隙层；结束摄像头$cam_n$为$c_d$中的摄像头，结束时间$ts_n$在te预测的时隙层。轨迹$Tr_{id}$的行程时间$Time_{id}$为结束时间$ts_n$与出发时间$ts_1$的差值。

同时，为了解决前文提到的监控摄像头的噪声数据问题，本文为每个路段$e_{i,j}$设置了一个时间阈值范围$th_{i,j}$，阈值范围根据该路段历史车辆轨迹的平均行驶时间来确定，本文取平均行驶时间的一半和3倍分别作为$th_{i,j}$的最小值和最大值。$th_{i,j}$用来判断前文得到的Veh_c中每辆车veh_{id}的部分车辆轨迹Tr_{id}中是否存在噪声数据（即可能是噪声数据的摄像头记录）。具体方法为，依次计算车辆轨迹Tr_{id}中相邻摄像头记录的时间差，并与摄像头记录所对应的路段上的时间阈值范围$th_{i,j}$进行比较，如果该时间差处于时间阈值范围外，就认为该相邻摄像头记录存在噪声数据，并过滤该条车辆轨迹。例如，对于轨迹 $\mathbf{Tr_{id}}=\{\langle \mathbf{cam_1},\mathbf{ts_1}\rangle,\langle \mathbf{cam_2},\mathbf{ts_2}\rangle,\cdots,\langle \mathbf{cam_i},\mathbf{ts_i}\rangle,\cdots,\langle \mathbf{cam_n},\mathbf{ts_n}\rangle\}$，根据相邻摄像头记录$\langle \mathbf{cam_1},\mathbf{ts_1}\rangle$和$\langle \mathbf{cam_2},\mathbf{ts_2}\rangle$计算$ts_1$和$ts_2$的时间差值，并与$cam_1$和$cam_2$所对应路段的时间阈值范围进行比较，然后依次计算$\langle \mathbf{cam_2},\mathbf{ts_2}\rangle$和$\langle \mathbf{cam_3},\mathbf{ts_3}\rangle$，直到$\langle \mathbf{cam_{n-1}},\mathbf{ts_{n-1}}\rangle$和$\langle \mathbf{cam_n},\mathbf{ts_n}\rangle$。

过滤噪声数据后，根据行程时间$Time_{id}$的大小，进行由小到大的排序，选出时间最短的前Top$n$条车辆轨迹，放入$Path_{top_n}$，然后将车辆轨迹经过的摄像头编号替换为该摄像头所匹配的路口，就可以得到车辆在路网上的推荐路线和对应的行程时间估计。这里由于摄像头覆盖率以及噪声数据等原因，得到的推荐路线可能会存在相邻路口不连续的情况，导致路线不够详细，如果想要得到更加详细的路线，可以在这些不相邻的路口间采用最短路径算法，进一步细化路线。

5 实验与结果

本节将在真实数据集上进行实验，以验证本文所提基于城市交通监控大数据的行程时间估计方法UTSD的有效性。

5.1 实验数据集

本文采用的实验数据集为某省会城市的真实交通监控数据集。该数据集包括

2016年8月1—31日共31天的从1 704个监控摄像头抓拍的4亿多条数据记录。路网是从开源地图OpenStreet Map[32]上采集得到的，本次实验区域为该城市部分市区，路网包括78个路口节点、149条路径，路网区域内有99个摄像头覆盖在39个路网节点上。对路网数据预处理（如去掉直行路段中多余的路口等）后，路网内有56个路网节点、125条路径，路网区域内有77个摄像头覆盖在33个路网节点上，摄像头覆盖率为59%。

5.2 实验设置

所有实验均在一台戴尔笔记本计算机上进行，系统为Windows10(64位)，配置为4核Intel(R) Core(TM)i5-5200U CPU @ 2.20 GHz，运行内存为8 GB。程序采用Python语言，编译器版本为Python 3.7.0 [MSC v.1912 64 bit (AMD64)]。每个实验运行1 000次，求出平均运行结果。

5.2.1 对比算法

由于本文方法为基于摄像头数据的大数据查询方法，目前暂时没有与该方法相关的大数据查询算法。因此本文使用基于有向图的Dijkstra最短路径算法[35]和百度地图的API查询算法（以下简称百度算法）作为对比算法。

基于有向图的Dijkstra最短路径算法是有向加权图中最基本的最短路径算法。基于有向图的Dijkstra最短路径算法可以表示为：给定加权有向图G和源点A，求A到G中其他顶点的最短路径，在求最短路径时，从起始点开始，采用贪心算法的策略，遍历距起始点最近且未访问过的顶点的邻接节点，直到遍历到结束点。那么在路网中求最短路径可以表示为：给定路网$\mathcal{G}=(N,E)$和起始点n_o，求n_o到路网中另一个路口n_d的最短路径。

百度地图的API查询算法是通过调用百度地图开发的API来查询起始点和结束点的算法。该算法运行在百度服务器上，实验中只能得到算法结果的返回值。

5.2.2 参数设置

实验中本文所提算法的默认参数设置如下：用来查询的监控数据为7天的历史数据，查询时隙层的范围是出发时间ts和结束时间te所在的时隙层，时间估计结果取前10%的数据的均值（即top$n$为前10%的数据）。

5.2.3 评估指标

本文采用行程时间估计常用的两种性能评估标准——行程时间的平均相对误差（MRE）和平均绝对误差（MAE）来评估算法的有效性。同时，为了减少数据集中存在的与大多数行程显著不同的异常行程对实验结果的影响，本文还增加了中值相对误差（MedRE）和中值绝对误差（MedAE）来评估算法的有效性。

平均相对误差的定义如下：

$$\mathrm{MRE}=\sum_{i=1}^{N}\frac{\left|t_{\mathrm{pre}}^{i}-t_{\mathrm{true}}^{i}\right|}{t_{\mathrm{true}}^{i}} \quad (1)$$

平均绝对误差的定义如下：

$$\mathrm{MAE}=\sum_{i=1}^{N}\frac{\left|t_{\mathrm{pre}}^{i}-t_{\mathrm{true}}^{i}\right|}{N} \quad (2)$$

中值相对误差的定义如下：

$$\mathrm{MedRE}=\mathrm{median}\left(\frac{\left|t_{\mathrm{pre}}^{i}-t_{\mathrm{true}}^{i}\right|}{t_{\mathrm{true}}^{i}}\right) \quad (3)$$

中值绝对误差的定义如下：

$$\mathrm{MedAE}=\mathrm{median}\left(\left|t_{\mathrm{pre}}^{i}-t_{\mathrm{true}}^{i}\right|\right) \quad (4)$$

其中，t_{pre}^{i}表示算法对行程 i的估计时间，t_{true}^{i}表示行程 i的真实时间。

5.3 性能评估

本节首先对3种算法在不同查询时隙下的相对误差和绝对误差进行评估，然后对3种算法的平均查询时间进行分析。

为了更好地评估算法的有效性，本文选取3个时隙：上下班的早高峰和晚高峰时隙以及查询较少的时隙，分别为8:00—9:00和18:00—19:00、0:00—1:00。

各算法的平均相对误差和中值相对误差分别如图5和图6所示。从实验结果可以看出，UTSD的平均相对误差和中值相对误差都小于最短路径算法和百度算法。在3个查询时隙中，最短路径算法的最小平均相对误差为79.66%，百度算法的最小平均相对误差为55.58%，而UTSD的最小平均相对误差为14.64%，UTSD的准确率比最短路径算法和百度算法分别提高了65.02%和40.94%。而在中值相对误差的比较中，UTSD的最小中值相对误差为14.03%，最短路径算法的最小中值相对误差为77.16%，百度算法的最小中值相对误差为51.11%，最短路径算法和百度算法的中值相对误差分别为UTSD的5.5倍和3.6倍。此外，UTSD在3个时隙上平均相对误差的最大值和最小值的差值为4.55%，比最短路径算法的28.55%和百度算法的7.61%都要低，这说明UTSD的稳定性更高。

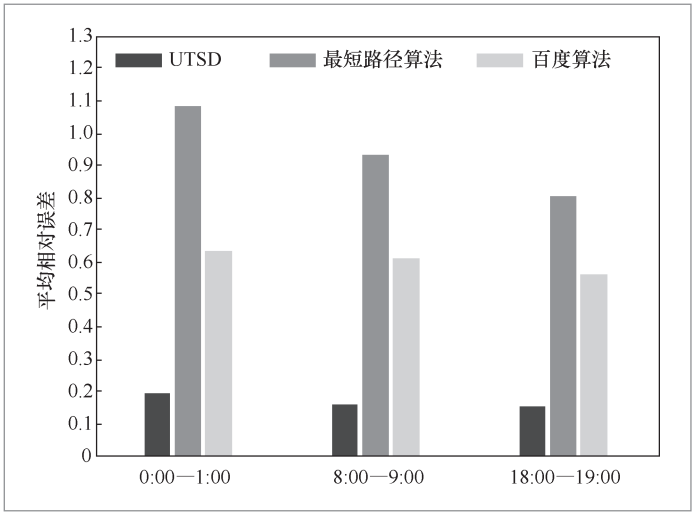


图 5 各算法的平均相对误差

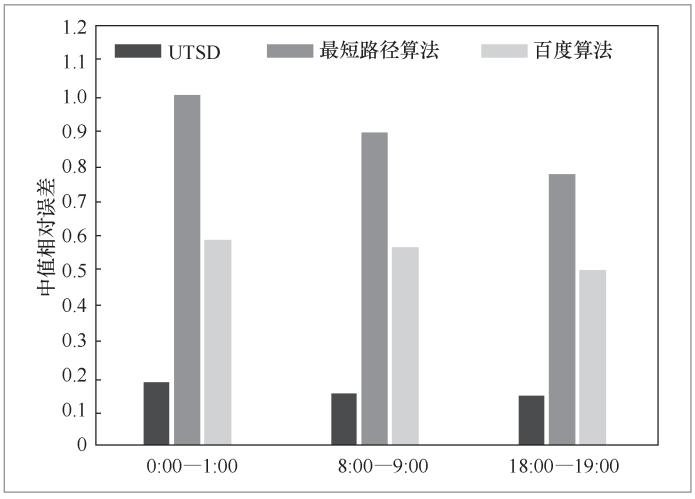


图 6 各算法的中值相对误差

在图7和图8中，UTSD、最短路径算法和百度算法的最小平均绝对误差分别为45 s、142 s和98 s。UTSD的平均绝对误差比最短路径算法和百度算法分别减少了97 s和53 s。因为实验采用的路网数据为城市路网的部分市区路网，查询的起始点和结束点均在该部分市区路网内，因此真实行程时间较短，绝对误差较小。若将路网区域扩大为监控数据覆盖的整个城市区域，真实行程时间增长，那么各算法的时间差值扩大，UTSD将更具优势。

从图5～图8可以看出，3种算法在时隙18:00—19:00的相对误差最小，在时隙8:00—9:00的相对误差居中，在时隙0:00—1:00的相对误差最大。最短路径算法和百度算法的绝对误差变化趋势和相对误差变化趋势相同，而UTSD的绝对误差在时隙18:00—19:00比时隙0:00—1:00要大，这可能是因为时隙18:00—19:00为车辆出行高峰期，虽然UTSD在时隙18:00—19:00的相对误差比在时隙0:00—1:00的相对误差要小，但是交通拥堵导致车辆行

程时间变长所带来的的影响更大，导致UTSD的绝对误差变大。

从**图5~图8**还可以看出，相比最短路径算法和百度算法，UTSD在行程时间估计结果中的误差较小（即准确度较高）。这是因为UTSD进行路径查询和对应的行程时间估计时，采用的都是经过筛选的真实历史车辆行程数据，而最短路径算法进行行程时间估计时，采用的只是历史车辆数据的平均时间数据，并没有对历史数据进行充分的利用，百度算法由于只采用接口进行查询，无法确定其采用的具体方法，但是可以看出其误差也比UTSD高。

各算法的平均查询时间见**表2**，第一列到第四列分别表示UTSD在1天、3天、5天、7天历史数据上的平均查询时间。从**表2**可以看出，最短路径算法的平均查询时间最短，为1.56×10^{-4} s，这是因为最短路径算法的时间复杂度很低，且路网构建的有向图含有的数据也很少，所以平均查询时间非常短。百度算法的平均查询时间为1.45×10^{-2} s，由于本文中的百度算法通过调用百度地图的API进行查询，算法没有在本地计算机上进行计算，而是在百度服务器上进行计算，运行算法时的硬件设备无法确定。UTSD在1天和3天的历史数据上的平均查询时间分别为3.17×10^{-2} s和9.61×10^{-2} s，与百度算法的平均查询时间在同一个数量级上，这表明UTSD在时间效率上与百度算法相近。随着历史数据天数增加，UTSD的平均查询时间也不断增加，这是由于算法要对大量的监控数据进行查询和处理必定会花费一定的时间，尽管如此，UTSD在以7天监控数据作为历史数据的情况下，平均查询时间低于0.3 s，完全可以满足城市出行的即时查询需求。

5.4 参数敏感性估计

本节将评估参数变化对所提方法的性能影响。本文所提方法是基于大数据的查询方法，因此主要影响因素是历史数据天数。

图9展示了本文所提方法在历史数据天数为3~17天的平均相对误差和中值相对误差的变化情况。从**图9**可以看出，在历史数据天数为3天时，所提方法的平均相对误差和中值相对误差均最高，这可能是

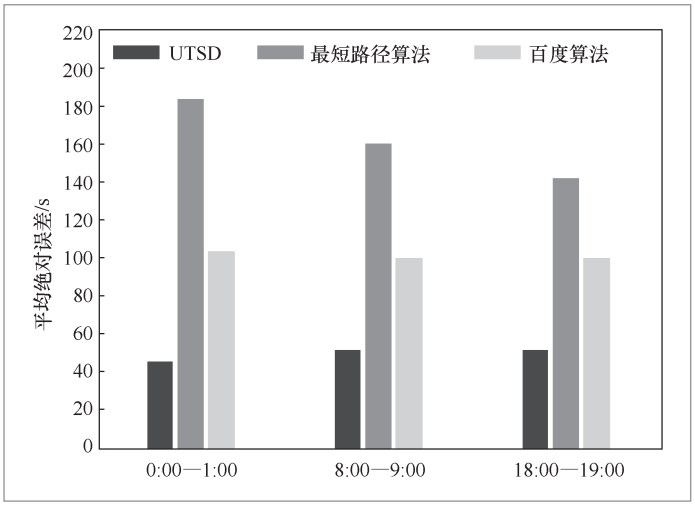


图 7 各算法的平均绝对误差

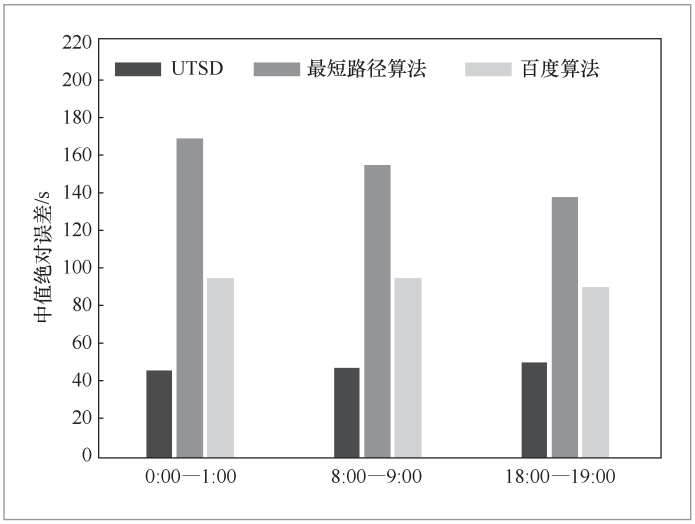


图 8 各算法中值绝对误差

表 2 各算法平均查询时间 /s

UTSD				最短路径算法	百度算法
1天	3天	5天	7天		
3.17×10^{-2}	9.61×10^{-2}	1.83×10^{-1}	2.98×10^{-1}	1.56×10^{-4}	1.45×10^{-2}

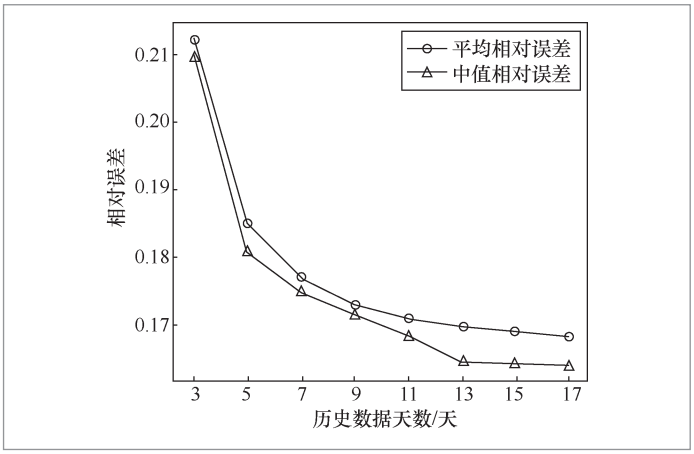


图 9 历史数据天数对相对误差的影响

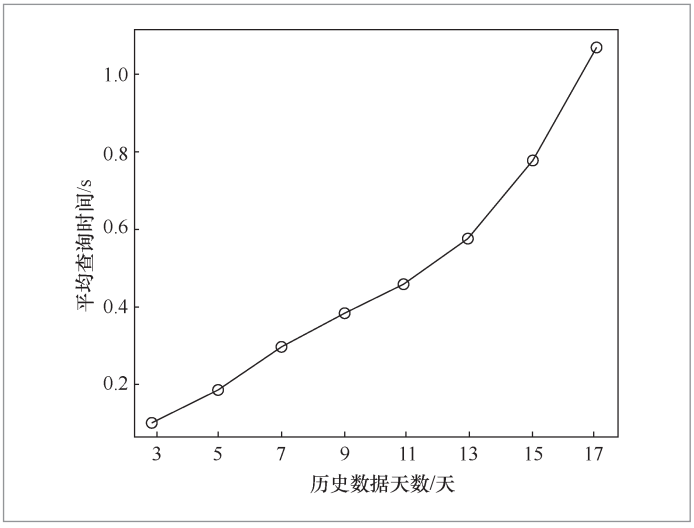


图 10 历史数据天数对 UTSD 的平均查询时间的影响

因为3天的历史数据所含有的历史车辆轨迹样本不够充足，形成的车辆轨迹不够完整，以及存在一些噪声数据，导致行程时间估计误差较大。随着历史数据天数的不断增加，本文所提方法的这两种误差不断减小，这说明随着历史数据天数的增加，该方法能够从历史数据中查询到更多符合查询条件的历史车辆轨迹，进而更好地进行路径推荐和对应行程时间的估计。当历史数据天数达到13天后，所提方法在平均相对误差和中值相对误差上的变化趋于稳定。随着历史数据天数增加到17天，所提方法的两种误差并没有明显减小，这说明当历史数据达到一定数量时，能够覆盖路网中所有的查询区域以及查询时间，并有足够的历史轨迹满足查询需求，再次增加新历史数据的同时，也会增加噪声数据，噪声数据增加的误差量不变，而新历史数据所能减少的误差量逐渐减小，导致算法误差的降低程度越来越小，进入“瓶颈期”。

图10展示了本文所提方法在历史数据天数为3~17天的平均查询时间的变化情况。从**图10**可以看出，随着历史数据天数的增加，算法的平均查询时间也在增加，而且随着天数的增加，平均查询时间增加的幅度越来越大。这一方面是因为随着历史数据数量的增加，读取、查询和处理数据的时间增大，进而导致运行时间增大；另一方面是因为本文所提方法本身的算法复杂度决定了算法随历史数据的增加所需要的运行时间会不断增加。

结合**图9**和**图10**可以看出，历史数据天数低于7天时，本文所提算法的误差较高，历史数据天数高于11天时，其误差与7天的误差相比降低较少，但平均查询时间增加较大。若要在较短的查询时间内得到较低的误差，使用7~11天的历史数据比较合适。

6 结束语

本文针对城市出行的时间估计问题，提出了一种基于城市交通监控大数据的行程时间估计方法UTSD。首先将道路网络建模为有向加权图，然后根据位置信息将摄像头映射到路网地图上，形成路网数据库和摄像头数据库。然后，结合R树构建时空索引和反向索引结构，时空索引用于快速检索所有车辆的摄像头记录，反向索引用于快速得到车辆的行程时间和经过的摄像头轨迹，大大提升了数据查询和行程时间估计的效率。

通过在某省会城市的真实交通监控数据上进行实验评估，验证了本文所提方法的有效性，且相比对比算法其准确性有显著的提升。

虽然城市路网中的现有监控摄像头数量较多，但是摄像头在路网中的覆盖率还不够高，提高摄像头在路网中的覆盖率显然无法做到，那么如何在摄像头覆盖率不变的情况下，进一步提升算法的精度和效率，是下一步需要进行的工作。

参考文献:

[1] 刘晓波，蒋阳升，唐优华，等. 综合交通大数据应用技术的发展展望[J]. 大数据，2019，5(3)：55-68.
LIU X B, JIANG Y S, TANG Y H, et al. Development prospect of integrated transportation big data application technology[J]. Big Data Research, 2019, 5(3): 55-68.

[2] WANG Y, ZHENG Y, XUE Y. Travel time estimation of a path using sparse trajectories[C]// The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM Press, 2014: 25-34.

[3] BECKMANN N, KRIEGEL H P, SCHNEIDER R, et al. The R*-tree: an efficient and robust access method for points and rectangles[C]// The 1990 ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 1990: 322-331.

[4] ZHONG R C, LI G L, TAN K L, et al. G-tree: an efficient index for KNN search on road networks[C]// The 22nd ACM International Conference on Information & Knowledge Management. New York: ACM Press, 2013: 39-48.

[5] KAMEL I, FALOUTSOS C. Hilbert R-tree: an improved R-tree using fractals[R]. 1993.

[6] PFOSER D, JENSEN C S, THEODORIDIS Y. Novel approaches to the indexing of moving object trajectories[C]// The 26th International Conference on Very Large Data Bases. [S.l.:s.n.], 2000: 395-406.

[7] TAO Y F, PAPADIAS D. Efficient historical R-trees[C]// The 13th International Conference on Scientific and Statistical Database Management. Piscataway: IEEE Press, 2001: 223-232.

[8] WANG L H, ZHENG Y, XIE X, et al. A flexible spatio-temporal indexing scheme for large-scale GPS track retrieval[C]// The 9th International Conference on Mobile Data Management. Piscataway: IEEE Press, 2008: 1-8.

[9] CHAKKA V P, EVERSPAUGH A C, PATEL J M. Indexing large trajectory data sets with SETI[C]// The 2003 CIDR Conference. [S.l.:s.n.], 2003: 75-86.

[10] FINKEL R A, BENTLEY J L. Quad trees a data structure for retrieval on composite keys[J]. Acta Informatica, 1974, 4(1): 1-9.

[11] BENTLEY J L. Multidimensional binary search trees used for associative searching[J]. Communications of the ACM, 1975, 18(9): 509-517.

[12] JENSEN C S, LIN D, OOI B C. Query and update efficient B+-tree based indexing of moving objects[C]// The 30th International Conference on Very Large Data Bases. [S.l.:s.n.], 2004: 768-779.

[13] COMER D. Ubiquitous B-tree[J]. ACM Computing Surveys, 1979, 11(2): 121-137.

[14] ZOBEL J, MOFFAT A, RAMAMOHANARAO K. Inverted files versus signature files for text indexing[J]. ACM Transactions on Database Systems, 1998, 23(4): 453-490.

[15] WU C H, HO J M, LEE D T. Travel-time prediction with support vector regression[J]. IEEE Transactions on Intelligent Transportation Systems, 2004, 5(4): 276-281.

[16] CHEN H, RAKHA H A, MCGHEE C C. Dynamic travel time prediction using pattern recognition[C]// The 20th World Congress on Intelligent Transportation Systems. [S.l.:s.n.], 2013.

[17] HOFLEITNER A, HERRING R, ABBEEL

P, et al. Learning the dynamics of arterial traffic from probe data using a dynamic Bayesian network[J]. IEEE Transactions on Intelligent Transportation Systems, 2012, 13(4): 1679–1693.

[18] ZHAN X Y, HASAN S, UKKUSURI S V, et al. Urban link travel time estimation using large-scale taxi data with partial information[J]. Transportation Research Part C: Emerging Technologies, 2013, 33: 37–49.

[19] ZHANG F M, ZHU X Y, HU T, et al. Urban link travel time prediction based on a gradient boosting method considering spatiotemporal correlations[J]. ISPRS International Journal of Geo-Information, 2016, 5(11): 201.

[20] NIU X G, ZHU Y, ZHANG X N. DeepSense: a novel learning mechanism for traffic prediction with taxi GPS traces[C]// 2014 IEEE Global Communications Conference. Piscataway: IEEE Press, 2014: 2745–2750.

[21] RAHMANI M, JENELIUS E, KOUTSOPOULOS H N. Route travel time estimation using low-frequency floating car data[C]// The 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013). Piscataway: IEEE Press, 2013: 2292–2297.

[22] CHEN M, CHIEN S I J. Dynamic freeway travel-time prediction with probe vehicle data: link based versus path based[J]. Transportation Research Record, 2001, 1768(1): 157–161.

[23] JIANG M Y, ZHAO T Q. Vehicle travel time estimation by sparse trajectories[C]// 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC). Piscataway: IEEE Press, 2019: 433–442.

[24] KINGMA D P, BA J. Adam: a method for stochastic optimization[J]. arXiv preprint, 2014, arXiv:1412.6980.

[25] LI Y, GUNOPULOS D, LU C W, et al. Urban travel time prediction using a small number of GPS floating cars[C]// The 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. New York: ACM Press, 2017: 1–10.

[26] WANG H J, TANG X F, KUO Y H, et al. A simple baseline for travel time estimation using large-scale trip data[J]. ACM Transactions on Intelligent Systems and Technology, 2019, 10(2): 1–22.

[27] JIANG Y J, LI X. Travel time prediction based on historical trajectory data[J]. Annals of GIS, 2013, 19(1): 27–35.

[28] 赖永炫, 杨旭, 曹琦, 等. 一种基于Gradient Boosting的公交车运行时长预测方法[J]. 大数据, 2019, 5(5): 58–78.
LAI Y X, YANG X, CAO Q, et al. A bus running length prediction method based on Gradient Boosting[J]. Big Data Research, 2019, 5(5): 58–78.

[29] YUAN H T, LI G L, BAO Z F, et al. Effective travel time estimation: when historical trajectories over road networks matter[C]// The 2020 ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 2020: 2135–2149.

[30] DING Y C, LI Y H, ZHOU X, et al. Sampling big trajectory data for traversal trajectory aggregate query[J]. IEEE Transactions on Big Data, 2018, 5(4): 550–563.

[31] YUAN J, ZHENG Y, ZHANG C Y, et al. T-drive: driving directions based on taxi trajectories[C]// The 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems. New York: ACM Press, 2010: 99–108.

[32] HAKLAY M, WEBER P. OpenStreetMap: user-generated street maps[J]. IEEE Pervasive Computing, 2008, 7(4): 12–18.

[33] LI Z H, WANG J J, HAN J W. Mining event periodicity from incomplete observations[C]// The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM Press, 2012: 444–452.

[34] LI Z H, WANG J J, HAN J W. ePeriodicity:

mining event periodicity from incomplete observations[J]. IEEE Transactions on Knowledge and Data Engineering, 2015, 27(5): 1219-1232.

[35] NOTO M, SATO H. A method for the shortest path search by extended Dijkstra algorithm[C]// The 2000 IEEE International Conference on Systems, Man and Cybernetics. Piscataway: IEEE Press, 2000: 2316-2320.

作者简介

李文明(1997-),男,烟台大学计算机与控制工程学院硕士生,主要研究方向为时空数据挖掘。

刘芳(1994-),女,烟台大学计算机与控制工程学院硕士生,主要研究方向为局部异常检测。

吕鹏(1995-),男,烟台大学计算机与控制工程学院硕士生,主要研究方向为高维数据异常检测。

于彦伟(1986-),男,博士,中国海洋大学计算机科学与技术系副教授,中国计算机学会会员,主要研究方向为时空数据挖掘、机器学习、分布式计算。

收稿日期: 2020-10-13

通信作者: 于彦伟, yuyanwei@ouc.edu.cn

基金项目: 国家自然科学基金资助项目(No.61773331)

Foundation Item: The National Natural Science Foundation of China (No.61773331)