ABSTRACT

Computational Transportation is an emerging discipline that poses many data management challenges. Computational transportation is characterized by the existence of a massive number of moving objects, moving sensors, and moving queries. This paper highlights important data management challenges for computational transportation and promising approaches towards addressing them.

Categories and Subject Descriptors
H. [Information Systems]: H.2 [Database Management]: H.2.4 [Systems] – Query Processing, Distributed Databases, H.2.8 [Database Applications]: Spatial Databases and GIS.

General Terms

Keywords
Intelligent Transportation Systems, Spatiotemporal Databases, Moving Object Databases, Computational Transportation.

1. INTRODUCTION

Nowadays, technology and low costs have spurred the use of location-detection devices in moving objects. Moreover, the infrastructure for the underlying transportation systems is being gradually computerized, automated, and empowered with many sensor devices of all sorts including location-detection devices, motion-detection devices, collision avoidance devices, RFIDs, RFID readers, and image and video cameras. Each of these devices generates a stream of readings that need to be captured and processed in real-time. In addition, the objects that move within the transportation infrastructure, e.g., cars and humans, also generate streams of spatiotemporal records as they move. Each generated record is of the form (Object or Sensor Identifier, Object or Sensor Location, Current Time, Object or Sensor Reading or Message), which represents the unique identifier of the moving object or sensor, the location of the object or the sensor when that record was generated -- location can be in the form of exact location coordinates (x, y, z), the time the object or sensor reports its reading, and finally the reading itself, e.g., the captured image or the captured RFID reading, etc.

The wealth of data and interesting problems that are generated from the above system are immense and give rise to the emerging field of Computational Transportation. In this paper, we focus on the data management and data processing challenges in computational transportation. More specifically, we focus on: constraint spatiotemporal data management, context awareness, annotation and provenance management, privacy, sensor data acquisition and phenomena detection and tracking, continuous query processing, indexing and handling of frequent updates of moving objects, similarity-based query processing, and scalable architectures. We highlight the challenges in each of these topics and highlight promising approaches towards addressing them.

2. QUERY PROCESSING GIVEN STATIC AND DYNAMIC CONSTRAINTS

Spatial object move in constrained motions in space, e.g., cars move on a road network, trains move on railroad networks, etc. These constrained types of motions should be taken into consideration when answering queries in computational transportation. For example, in computing nearest-neighbor queries, an object that is closest to a focal point based on Euclidian distance may actually be farther away when constrained using the underlying road network. Examples of algorithms that consider road networks for spatiotemporal query processing can be found in [5].

A road network is an example of what is termed "static constraints". Another important class of constraints is what is termed “dynamic constraints”. Dynamic constraints are ones that occur and disappear while processing a query, e.g., a continuous query that is reevaluated continuously over time. Dynamicity can either be in the time dimension, in the space dimension, or in both. Examples of dynamic constraints are car accidents, road blockages, road constructions, road congestion during some temporal durations, e.g., during rush hours, flood areas, pollution areas, sports events, races, etc. The effects of these examples are that they add temporarily additional constraints while queries are being processed. For example, when answering a continuous K-nearest-neighbor query, if a car accident occurs, it may cut out
and make unreachable one of closest objects, and hence that object is not among the \( K \) nearest to the query’s focal point. The query processor for continuous spatiotemporal queries needs to handle dynamic constraints as they occur.

Notice that constraints can be moving in space. For example, consider a constraint that represents a bike race and blockage of traffic due to the race. The location in space of the road blockage constraint keeps shifting as the bikes advance. The query processor needs to handle the change in location of the constraints while answering users’ queries.

Another important dimension is that of the shape of the constraints. While the constraint can claim any polygonal shape, it can also expand or shrink, e.g., pollution and flood regions. The shape of the constraint can either be rigid (i.e., does not change shape) or non-rigid (i.e., can change its shape). The query engine needs to be able to handle both rigid and non-rigid constraints while processing user queries.

3. CONTEXT-AWARENESS

When two users issue the same query, given each user’s context and preferences, the same query may report different answers to the two users because the two users have different contexts. Similarly, if the same user asks the same query twice but at different times, the user may get a different answer each time as the query’s context may have changed. For example, if the user requests a list of close-by restaurants when the weather is sunny versus when the weather is rainy, the user may get two different lists of restaurants.

Similarly, consider the case of a user issuing a query when alone versus the same user issuing that same query when accompanied by friends. Based on the preferences of each of the users and the aggregate context and the group of users, the query engine may return two different answers in each case.

In addition to the contexts of the query issuers, each of the queried objects in the database may have its own context. Matching the query’s context requirements with the database objects’ contexts is another important feature that the query engine should be able to support. So, even if a database object satisfies the query predicates, because the contexts of both the object and the query do not match, that object would not be returned as answer to the query.

Another important dimension that the query processor has to take into consideration is that these contexts may contradict each other, e.g., in the case of the group of users issuing the query as a group, the preferences of one user for a restaurant may contradict another user’s preferences. Even for the same user, the preference for low-cost vs. healthy-food restaurants may result in different rankings of the restaurants. Hence a form of prioritization and ordering needs to take place, e.g., via a multi-feature ranking mechanism or via a skyline mechanism.

4. ANNOTATIONS AND PROVENANCE MANAGEMENT

Users can comment on or attach unstructured or structured annotations to any subsets of the objects in space and at various granularities. The user can also attach annotations to only some attributes of a given object. Also, the user can attach annotations along the time dimension, e.g., an annotation that is only shown at dawn or sunset times, during rush hours, etc. Storage, management, and querying of these spatiotemporal annotations are important challenges, especially if one considers the issue of annotation granularity that is combinatorial in nature.

When a user queries moving objects or locations, their corresponding overlapping spatiotemporal annotations are retrieved and reported along with the query answers. Similarly, data objects can be queried by qualifications on their annotation values. Provenance issues are also very important to support along with annotations, e.g., who wrote such annotation, at what time, and in what context. Initial research efforts for supporting annotations have been conducted by [1,2,3,4].

5. PRIVACY

A major challenge in computational transportation is that of privacy. By enriching the transportation infrastructure with sensors, location-detection devices, and infrastructure servers, privacy of the users is challenged. The dilemma is as follows. In order for computational transportation to be of great benefit to users and to society, users and, more generally, all mobile objects need to reveal their locations. So, the question is: How to benefit from services provided by the computational transportation environment without compromising users’ and mobile objects’ privacy profiles? Several emerging efforts have been taken in these directions, e.g., see [22,23,24,25,26,27,28,38].

6. PHENOMENA-BASED QUERY PROCESSING

Sensors are ubiquitous in computational transportation systems. It is impossible and sometimes meaningless to process raw sensor data readings. It is mandatory to be able to summarize sensor data readings at a higher-level of abstraction. The notion of phenomenon has been proposed to address this need, e.g., see [29,30,31,32].

The user expresses or defines the phenomena of interest, e.g., in a format similar to that of select-from-where statement or a create view statement. The database server registers the phenomena specified by all database users. Upon arrival to the server, sensor data is scrutinized. Only data items relevant to any registered phenomena are retained. Higher-level dynamic models of phenomena are built on the fly, e.g., polygonal representations of areas of high temperatures or oil spill regions. Phenomena at this level of abstraction are detected and tracked as they move in space, or as their corresponding region(s) expand, or shrink.

In order for the query processor to scale to support a large number of queries and a large number of sensor reading streams, whenever a query is issued, it is directed to the phenomena relevant to the query. This pairing of queries and relevant phenomena can be performed in a variety of ways. One way is that the user specifies within the query which phenomena to direct the query to. Another way is that the query optimizer dynamically matches the query predicates with the phenomena definition predicates (in a way similar to that used for view matching) and direct the processing of the query to the appropriate phenomena. The server should have a phenomena detection and tracking module, e.g., similar to the ones in [30,34], that associates with
each phenomenon definition a region in the sensor network field where this phenomenon occurs or takes place, e.g., to those sensors that actually report the high temperature. Hence, the query is directed to the data generated by the corresponding sensor region to further process it. Some initial research efforts along these lines have been conducted in [29,30,31,32,34]. More research is needed to realize phenomenon-based query processing and optimization.

Several languages have been developed to express patterns of interest for data mining purposes. A useful application of these techniques is in the context of mining trajectories, which is very relevant for computational transportation [33].

7. CONTINUOUS QUERY PROCESSING

Many queries in computational transportation are continuous in nature. The fact that a query is continuously evaluated implies that the database server needs to store these issued queries inside the database in order to reevaluate them whenever needed.

Two possible approaches to answering continuous queries are: query reevaluation and incremental evaluation. In query reevaluation, the continuous query is reevaluated with a certain frequency and answers are reported with each reevaluation. In incremental evaluation, the query is evaluated once and the answer is stored with the user. Whenever the input changes, say $A_t$, only the changes are reevaluated. Only the changes in the answer, say $A_c$, from the previous stored answer are reported to the user. Incremental evaluation is advantageous since only the updates in both the inputs and the outputs are processed and not the whole tables. On the other hand, one disadvantage is that if the user gets out of sync, the deltas will be meaningless and some reevaluation from scratch may be needed to get back in sync. [18] is a disk-based approach to handling multiple continuous queries (moving or stationary) incrementally over moving objects. [18] uses an incremental query evaluation approach. [9,17] assume a streaming model and uses memory-based grid buffers to index both the moving and queries and incrementally evaluate the queries over time. Other spatiotemporal query processing and optimization challenges appear in [37].

8. ISSUES WITH INDEXING: HANDLING FREQUENT UPDATES

The need for indexing the locations of the moving objects is obvious. In order to speed up the processing of spatiotemporal queries, the query engine needs to look only into data objects that are relevant to the query location-wise and wants to avoid scanning all the data items with each query reevaluation (e.g., in the case of continuous queries). Hence, the locations of the moving objects should be indexed to speed up answering continuous and snapshot queries. However, although useful, this form of indexing introduces a new problem. As the objects move in space, their locations keep changing and hence the index needs to be updated with high frequency. Frequent index updates due to the change in objects’ locations incur a high update cost.

Several indexing techniques that deal with the issue of frequent updates have been proposed, e.g., [35,36,39,41,44,45,47,48,50,51,52,53,54]. Some of these index structures index the object at current time. In that case, old locations of moving objects are not preserved. However, in many computational transportation applications, there is a need to preserve the history of movements of an object, e.g., see [40,46,48,49]. Moreover, other application may need to predict the future location of objects. Indexes that support indexing the predicted locations of the objects in addition to their current locations, e.g., [40,42,43].

The indexes above are all disk-based with various memory structures or buffers to speed their operations. Sometimes, it is necessary to maintain and index the locations of all moving objects in main memory in order to answer continuous queries in a timely fashion. For example, in [9,17], a memory-based grid is used to index both the objects and their associated continuous queries. [55] enhances over [9,17] by allowing for dynamic grid sizes that address the issue of the variations in the distributions in space of moving objects over time.

9. SIMILARITY-BASED QUERY PROCESSING

There are many sources of inaccuracies in computational transportation data, e.g., inaccuracies due to lack of calibration in sensor data readings. Therefore, a query processor that performs exact matches would often return inaccurate and incorrect results. The query processor needs to perform similarity-based operations, e.g., similarity-based joins, similarity-based selects, similarity-based group-by, etc. While many similarity-based algorithms have been proposed in the literature, they are isolated in the sense that it is not known how multiple of them can be combined to answer complex similarity-based queries. An fully integrated similarity-based query processor needs to be designed along with its integrated similarity-based indexes and similarity-based query operators.

10. SCALABLE ARCHITECTURES

Due to the dispersion and diversity of the information to be collected, processed, and queried in computational transportation systems, one single data server cannot sustain excessive numbers of objects and sensors. There are several approaches to addressing this scalability challenge, mainly, (1) using distributed servers, (2) using a peer-to-peer setup, or (3) a hybrid approach. One example of a distributed server approach is the PLACE*. [21]. PLACE* is a distributed spatiotemporal data stream management system built on top of a set of regional PLACE servers [9, 16, 17, 18, 19, 20]. PLACE* supports continuous spatiotemporal queries over a set of regional servers where both queries and objects constantly move. A large or a dense area is usually divided into smaller geographical regions covered by a regional data server. A regional server communicates with only local objects and processes only local queries within its coverage region. The regional data servers form a server network. Figure 1a gives an example where the entire space is divided to six regions A -- F. Figure 1b illustrates a network of six regional data servers each of which covers a corresponding region given in Figure 1a. An object reports its location periodically to the server covering the object’s current location. As it moves, an object may switch the server it reports to based on the object’s location. Assume that in the event of a contamination incident, an official issues the query $q$: “Continuously inform first responder officer $i$ with all emergency personnel who are within five miles from the current location of a contamination region $f$.” Query $q$ has the following characteristics: (1) $q$ must be answered collectively and
continuously by regional servers whose coverage regions overlap q’s query region; (2) during f’s movement, the overlapping regions between q and regional servers continuously change as well as the set of regional servers that q hops among; (3) moving objects including i and f may change their regional servers as they move. Other distributed architectures include D-CAPE [15] and Aurora* [14]. D-CAPE is designed to distribute query plans and monitor the performance of each query processor with minimal communication cost. Aurora* [14] focuses on scalability in the communication infrastructure, adaptive load management, and high system availability.

The second alternative to address the issue of scalability is peer-to-peer technology, e.g., [10, 11, 12, 13]. Mobile peer-to-peer database architectures can scale very well. However, with the existence of a permanent intelligent infrastructure in the transportation system and with the presence of a massive number of mobile sensors, a hybrid approach is called for. In the hybrid approach, a network of distributed servers, e.g., similar to the one in [14, 15, 21], serves as the backbone for a collection of peer-to-peer sensor networks. Several hybrid architectures that integrate servers with server networks are being prototyped, e.g., [6, 7, 8].

![Diagram of a Network of Regional Servers](https://example.com/diagram.png)

**Fig. 1. A Network of Regional Servers**

11. CONCLUSIONS

The field of computational transportation is very exciting. In this paper, we have outlined important challenges for spatiotemporal data management in that field. Several other challenges remain to be addressed including scalable wireless networking, handling uncertainties and missing data, path planning and evacuation management, data mining and spatiotemporal periodicities detection.

12. REFERENCES


