

國立交通大學

網路工程研究所

碩士論文

針對路網週邊交通監控查詢探索動態扇形

Exploring Dynamic Fan Shapes for Nearby Traffic
Monitoring in Road Networks

研究生：林廷威

指導教授：彭文志 教授

中華民國 一 百 年 十 一 月

針對路網週邊交通監控查詢探索動態扇形
Exploring Dynamic Fan Shapes for Nearby Traffic
Monitoring in Road Networks

研究生：林廷威

Student : Ting-Wei Lin

指導教授：彭文志

Advisor : Wen-Chih Peng

國立交通大學
網路工程研究所

碩士論文

A Thesis

Submitted to Institute of Network Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

November 2011

Hsinchu, Taiwan, Republic of China

中華民國一十年十一月

針對路網週邊交通監控查詢探索動態扇形

學生：林廷威

指導教授：彭文志

國立交通大學網路工程研究所

摘 要

在本論文中，我們首先介紹 CarWeb 系統提供的即時交通路況預測服務，該服務針對使用者的移動行為即時的提供附近的交通路況預測。本論文著重在研究如何有效率又精準的提供使用者最適當的資訊，我們特別將問題聚焦在交通路網上移動的個體其連續的範圍查詢所涵蓋的路段上，為了解決這個問題，我們研究不同參數設定下圓形和扇形範圍查詢的特性與使用者移動行為的關係，我們提出了「動態扇形連續查詢架構」動態的改變扇型參數以有效率又不妥協系統效能的情況下取得較有用的查詢結果，我們最後採用真實世界所紀錄的交通軌跡並實作了廣泛的實驗以驗證該架構的適用性。

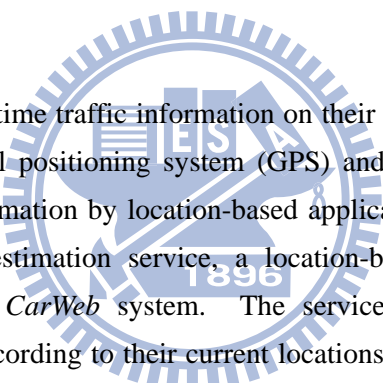
Exploring Dynamic Fan Shapes for Nearby Traffic Monitoring in Road Networks

Student : Ting-Wei Lin

Advisors : Dr. Wen-Chih Peng

Institute of Network Engineering
National Chiao Tung University

ABSTRACT



While people are driving, realtime traffic information on their way is informative and useful. With the prevalence of global positioning system (GPS) and portable devices, people can share and acquire traffic information by location-based applications. In this paper, we first introduce the traffic status estimation service, a location-based continuous range query application provided by the *CarWeb* system. The service provides users nearby with realtime traffic estimations according to their current locations and moving behaviors. This paper focuses on how to provide users with the most relevant traffic information efficiently and accurately. We emphasize the problem for objects moving on road networks requesting continuous range queries for traffic information of all road segments covered within the range. To tackle the problem, we study both circular and fan shaped range queries with different parameters comparing their coverage features with different moving behaviors of objects. We propose the Continuous Query with Dynamic Fan-Shape (*CQ-DFS*) framework which dynamically changes the parameters of the fan shaped range query to efficiently acquire useful traffic information without compromising the performance of the system. We conduct extensive experiments to demonstrate the effectiveness of the *CQ-DFS* framework using real world vehicle trajectories.

誌 謝

在前瞻資料庫系統實驗室的日子裡，學到了許多，生活過得非常充實。在研究上曾遭遇過許多困難，但很慶幸能經由大家的協助與討論，進而順利的完成碩士論文。

首先誠摯的感謝指導教授彭文志老師，老師用啟發式的教育方式，刺激學生獨立思考，從發現問題、分析問題、尋找既有資源、解決問題到論文撰寫技巧，獲益良多。除了課業以外，老師更在互動中教導我們許多人生的道理與待人處事的應對進退，在這段日子裡，不但充實了專業與研究技能，心性也成熟了許多。本論文的完成，亦得特別感謝鄭白樺教授與魏綾音學姊，在研究的過程中，不厭其煩地指導與協助各個階段所遭遇的問題，並在投稿的過程中一同努力。

在實驗室的這段時間，非常感謝魏綾音學姊、洪智傑學長、江孟芬學姊、雷伯瑞學長、廖忠訓學長的指導與協助。也非常感謝沈姿柔同學、張凱評同學、盧俊達同學、榮芊菡同學的協助與互相扶持。此外，也要感謝柯宇倫學弟、潘依琴學妹、許雅婷學妹、王堃瑋學弟、黃柏崑學弟在實驗室事務與口試安排的協助。

最後要特別感謝家人與女朋友的支持與包容，讓我能無後顧之憂地專心於課業。

謹以此文獻給我摯愛的你們。

Contents

1	Introduction	1
2	Related Work	5
2.1	Spatial Queries	5
2.2	Continuous Queries	6
2.3	Location Deviation	9
2.4	Location Prediction	10
3	Problem Definition	12
4	Design of Continuous Query with Dynamic Fan Shape	15
4.1	Exploring Fan Shapes for Continuous Queries	15
4.2	Deciding Fan Shapes On-The-Fly	17
4.3	Caching Overlapping Query Results	22
5	Performance Study	25
5.1	Comparing Query Shapes for Different Distance Complexities	29
5.2	Comparing Query Shapes for Different Angular Complexities	31
5.3	Comparing Query Shapes with a Full Dataset	33
5.4	Effectiveness of the Fan Shaped Range Formation	34
5.5	Query Response Time Study	35
5.6	Long Query Period Study	36
6	Conclusion	38



List of Figures

1.1	A nearby traffic monitoring application for a range query.	2
3.1	A continuous range query Q on nearby traffic monitoring in road networks. . .	14
4.1	Comparing circular shape to wide and narrow forms of fan shapes.	17
4.2	Angular complexity A_{c5} and distance complexity D_{c5} for $R = 5$	19
4.3	Expanding $\langle A_{c5}, D_{c5} \rangle$ forming the desired $\langle Ang_5, Rad_5 \rangle$ with $R = 5$	21
5.1	Experimental results of subset 1 and 2 comparing different query shapes. . . .	30
5.2	Experimental results of subsets 3 and 4 comparing different query shapes. . . .	32
5.3	Experimental results of full dataset.	33
5.4	Experimental results of subsets 1 and 2 comparing different fan shaped formation algorithms.	34
5.5	Experimental results of subsets 3 and 4 comparing different fan shaped formation algorithms.	35
5.6	Average query response time results of both previous experiment sets.	36
5.7	Experimental results of long period dataset.	37

Chapter 1

Introduction

With the technology advances in smart phones and positioning devices, location-based services (LBSs) are growing at a rapid speed. A wide variety of LBSs have been developed, such as cargo tracking, fleet controlling, and traffic status monitoring. The most important task in LBSs is to support spatial queries, such as range queries[32], K Nearest Neighbor (KNN) [8, 21, 37], spatial join [3, 28, 33, 34], and closest pair queries [11, 12, 20]. For example, one common query is to find the K nearest points of interest (POIs), where POIs could be gas stations, hotels, hospitals, to name a few. Consequently, the results of spatial queries include data whose geo-location should fulfill the predicate of the spatial queries.

The results of spatial queries may contain some real-time information. Note that some works solicit driving speeds from drivers and the speed data are detected by GPS devices. Prior works have explored the traffic estimation problems from speed data. Thus, another important spatial query is to provide drivers with the traffic status of nearby road segments. For example, given a spatial range as a purple circle in Figure 1.1 and a road network database, the traffic information of the roads located in the spatial range is returned as the range query result. Given a spatial range, the output of the range query is to return queried objects whose geo-locations are within the spatial range. Previous works have studied two kinds of results of range queries: one is the set of static objects whose geo-locations are always within the spatial range and the other is a set of dynamic objects whose geo-locations are updated with time. In this paper, we focus on applications of range queries with dynamic query ranges and

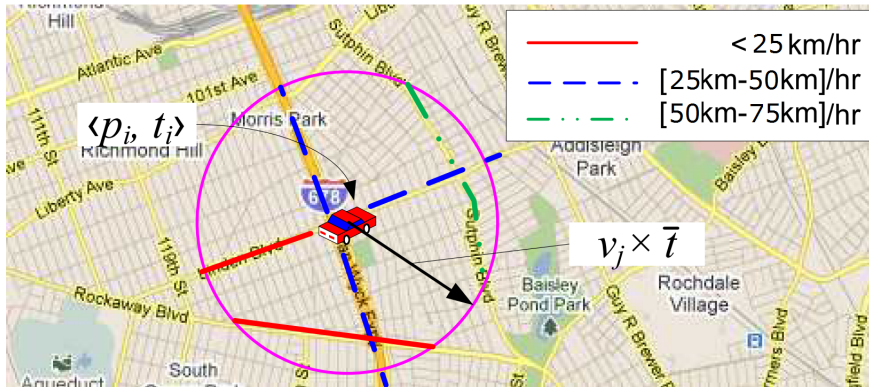


Figure 1.1: A nearby traffic monitoring application for a range query.

static objects. The objects could be POIs or road segments. For example, for nearby traffic monitoring applications of range queries, users request traffic information on road networks continuously, and then the server returns the nearby traffic status according to the users' updated locations. If a location of a queried range is dynamically changed, the range query is called a continuous range query. In this paper, we target our scenario as applications of range queries in vehicles. Because a location of a queried range refers to the location of a vehicle, we deal with continuous range query processing in this paper.

The existing continuous range queries specify a fixed shape as a query range without considering users' moving directions or movements. However, some query results satisfying the query range with a fixed shape would possibly be useless while users are moving. Take nearby traffic monitoring in road networks as an example of continuous range queries. The *CarWeb* system has been developed as a location-based service for traffic monitoring by smart phones [49]. When a user becomes active and starts reporting his/her locations to the server through a smart phone, the server forms a continuous range query exclusively for the user. Assume the user updates the server with his/her real-time locations periodically (e.g., once every 5 seconds) in the format of $\langle p_i, t_i \rangle$, where p_i denotes the user's location at time stamp t_i . The system treats each update $\langle p_i, t_i \rangle$ from the user as the input to form a range query. The system then fetches all the road segments satisfying the range query and delivers the traffic information of these road segments to the user. Although the user derives his/her nearby traffic information, the traffic information behind the user would be meaningless if the user moves straight ahead. This meaningless traffic information would waste a bandwidth of communication and the computation of servers. Specifically, consider the user in Figure 1.1

as an example. Because the user moves in a northeast direction, the road segments located to the southwest of the user are useless. In order to provide users with useful traffic information, we prefer to exclude those useless road segments from the query result. In addition, assume a user reports his/her location every \bar{t} seconds. Based on the user's moving velocity v , the server can predict the distance the user moves before the next reporting time $t_i + \bar{t}$, and then issues a circular range query centered at p_i with $v_j \times \bar{t}$ as the radius, i.e., the circle depicted in Figure 1.1. The traffic status of the road segments located inside the circular range are then delivered to the user. This approach faces one major deficiency. That is, a user usually only moves in one direction, and thus a circular range covers lots of dead space whose information is absolutely of no use to the user.

To provide moving users with useful information in time for such continuous range queries, there are two major issues. The first is to specify a query range such that the query results are useful to the users. The second issue is to shorten the query response time so that users could get query results on time or before moving to other locations. Consequently, in this paper, we propose a new Continuous Query with Dynamic Fan-Shape (*CQ-DFS*) framework to support accurate and real-time range queries. In addition, in this paper, we use nearby traffic monitoring in road networks as a case of this kind of range query. To quantify the performance of such range queries, the accuracy is measured via the *precision* and *recall* metrics. A relevant query result in the aforementioned example is the road segment that the user passes through in the next reporting location; and we prefer high precision and recall. On the other hand, a real-time estimation means a query result will be delivered to the user before the next location reporting, i.e., it guarantees that the response time is less than \bar{t} . These two requirements are contradictory. Returning more road segments tends to provide more useful information. However, querying more road segments requires a longer response time. We could only set up an underestimated upper bound for the number of road segments covered by each range query. Consequently, the key issue is how to effectively choose a set of road segments for each range query to maximize the precision and recall under the constraint that only a limited number of road segments could be included into a range query.

For a continuous range query, our framework *CQ-DFS* dynamically adjusts the range of each query range in a *fan-shape* based on users' movement patterns and locations on a road network. To be more specific, we take both the directions and recent trajectories of moving

users into consideration when deciding the settings of fan-shaped range queries in order to cover road segments which moving objects would pass by later. Note that a continuous range query for nearby traffic motoring in road networks has to return traffic information to a user before his/her next location reporting, i.e., a user should receive traffic estimation within \bar{t} . Consequently, the size of the range has to be appropriate since an estimation containing too much information incurs long processing-time and long delivery time. The framework *CQ-DFS* adjusts the radius/central angle of the fan-shape (i.e., circular sector) in order to maximize the precision and recall of the estimation, e.g., users moving fast along straight trajectories prefer a fan-shape with a large radius and narrow central angle, and users moving slowly on irregular trajectories prefer a fan-shape with a small radius and wide central angle.

The three-fold contributions of this paper are summarized as follows:

- We study both circular and fan shaped range queries with different parameters comparing their coverage features for different moving behaviors of users.
- We propose the Continuous Query with Dynamic Fan-Shape framework (*CQ-DFS*, for short) which dynamically changes parameters of fan shaped range query according to users' moving behaviors for more useful information.
- We conduct comprehensive experiments to show the effectiveness of the *CQ-DFS* framework by using real world vehicle trajectories.

The rest of this paper is organized as follows. In Section 2, we review related works about spatial queries, continuous queries, location deviation, and location prediction. Section 3 formally defines our problem. Section 4 gives details of the proposed *CQ-DFS* framework. Section 5 reports comprehensive experimental evaluations and Section 6 concludes this paper.

Chapter 2

Related Work

In this section, we review existing related works, including spatial queries, continuous queries, location deviation, and location prediction.

2.1 Spatial Queries

A spatial query is a type of database query that supports geographical data types such as points, lines, and polygons. In this section, we review range query, K Nearest Neighbor, spatial join, and closest pair query.

A range query refers to a spatial query retrieves the information of data objects inside a user defined 2D region. An easy representative example is finding all coffee shops within 10 miles from the user. In order to answer this kind of queries, various efficient data structures maintaining multi-dimensional geometric objects have been developed [15, 19, 31], as well as the popular R-tree [18] and its variation R*-tree [2]. [32] motivates four different user defined range query classes and derives a probabilistic model for each of them.

A K Nearest Neighbor (KNN) query refers to a spatial query that searches K data objects from a given location with the constraint that their distances between query location is the shortest compared with other data objects. KNN query is useful when the user is not familiar with the layout of data objects. One quick example is searching for 5 nearest restaurants from user's current location. [37] presents a branch-and-bound R-tree traversal algorithm to find

KNN, and introduces two metrics that can be used to guide an ordered depth first spatial search. [8] proposes an improved nearest neighbor search algorithm on the R-tree family, and challenges a previous work that two of the three heuristics should be removed. [21] solves incremental solution to nearest neighbor problem, compared with other works developed for k-d tree, LSD-tree, and PMR quadtree, R-tree is used alternatively.

A spatial join operation is used to combine spatial objects of two sets according to some spatial properties. For example, consider the spatial relations *campus* and *city* where an attribute in each relation represents the borders of campuses and cities respectively. A spatial join query could be finding all campuses which are in a city. [3] studies spatial join query processing by using R*-trees. Comparing to pairwise spatial join query, [28, 34] studies the multiway spatial join, which involves an arbitrary number of spatial inputs. Because of the exhaustive processing nature of spatial join query processing, [33] studies the retrieval of the best possible (exact or approximate) solutions within a time threshold, in order to provide fast retrieval for applications such as multimedia information in real time systems.

A K closest pairs query (K-CPQ) combines join and nearest neighbor queries that it discovers the K pairs of spatial objects formed from two datasets that have the K smallest distances between them. For example, the first dataset contains cultural landmarks, and the second dataset stores popular theme parks, A K-CPQ could discover the K closest pairs of landmarks and theme parks for efficient tour scheduling. [20] presents incremental algorithms for computing the distance join and distance semi-join, the K-CPQ is a variation from its problem setting. [11, 12] presents a pruning heuristic and two updating strategies for minimizing the pruning distance, and designs three non-incremental branch-and-bound algorithms and one iterative Best-First traversal algorithm solving K-CPQ.

2.2 Continuous Queries

A continuous query refers to a query that runs continuously over a specified period of time, and it is usually based on a dynamic scenario in which the query point and/or the queried objects are moving. In the following, we review continuous static queries over moving objects and moving queries over static objects. First, we assume that the queries are fixed while the

queried objects are moving. For example, a shopping mall wants to send e-coupons to all the potential customers who are nearby and it can issue a continuous range of queries during weekends. This kind of query assumes that the queries are known in advance, and tries to minimize the update cost of moving objects. In [23], the authors use a continuous window (CW) algorithm to monitor moving objects with updates, and the works presented in [29, 53] adopt incremental evaluation and shared execution strategies to improve the performance. Moreover, some other works have investigated this problem based on the concept of monitoring range, including CPM [30], SEA-CNN [52] and YPK-CNN [57].

Second, we assume that the queried objects are static, while the queries are not fixed. For example, a user who is shopping on Fifth Avenue submits a query to look for a nearby ATM until she reaches one. In this case, the queried objects (i.e., ATM) are static, while the user keeps moving and hence the query point is changing continuously. A typical continuous query in this context is continuous Nearest Neighbor Search (CNN). It looks for the nearest object to the user when the user is moving. Many solutions have been proposed to support CNN search, and some representatives are reviewed in the following. In [39], a sampling technique is employed to perform normal NN searches at some pre-defined sampling points and then an approximated range is derived to bound all the possible answers. However, its accuracy depends largely on the pre-defined sampling points. In order to enable an exact search, Tao et al. devised two search algorithms for CNN queries based on an R-tree. The first algorithm is based on the concept of time-parameterized (TP) queries, which treat a query line segment as the moving trajectory of a query point [45]. The second algorithm, proposed later in [44], navigates the R-tree based on certain heuristics. The whole answer set is obtained within one single navigation of the R-tree. We would like to highlight that the above works assume the moving trajectories of mobile users are known in advance. In addition to Euclidean spaces, CNN searching in road networks (i.e., network spaces) has been studied as well. The first solution to CNN queries in road networks is proposed in [14]. Other solutions include the work presented in [9], CNN retrieval via a continuous windowing algorithm, and the beach-line algorithm [26].

In addition to CNN queries, the problem of continuous range queries is also widely investigated. A continuous range query refers to a range query continuously updating the results within a certain period of time. Typically, continuous range queries consist of at least a

database server, and a large number of objects [59]. Some of them have static query ranges and dynamic moving objects, objects update their information such as locations to the server, then the server returns the information to the users who made the queries. The most common approach is making objects report their initial positions and velocity information, from which the server could estimate their future locations within a period of time. As a result, objects do not need to report their locations until the deviation between their actual locations and estimated locations exceed a threshold [50, 51, 56]. Some others have dynamic query ranges and static objects [54, 5, 58], users request their desired query region to the server, then the server responds with the information accordingly. [54] assumes the query client’s future trajectory is given apriori, and attempts to precompute query results in advance. [5] reports a safe region to the client in addition to the query result, so the client does not need to query again since the query results would remain the same inside the safe region. [58] employs the safe region idea, and further introduces the safe exits on road networks, where clients do not need to query again within the exits on the road network. Nevertheless, it is not a requirement for continuous range queries. For instance, [29, 17, 40] support moving queries over moving objects in server-based processing. [29] proposes the *SINA* that incrementally evaluates a set of concurrently executing queries. [17] introduces the concept of *motion-sensitive bounding boxes (MSBs)* to model moving objects and moving queries in order to decrease the number of updates to the indices. [40] focuses on server processing time optimization, and adopts a point-based update policy. Mobile clients update their new locations whenever they deviate from last reported locations. Apart from the central server architecture, the *MobiEyes* [16] and *MQM* [4] let mobile devices answer continuous range queries, leaving the central server as a mediator. In *MQM*, the *resident domain* is introduced as a subspace surrounding the moving object to process continuous queries. Queries are partitioned into *monitor regions*, where the monitor regions covered by the resident domain will be sent to the moving object.

Compared to the existing continuous queries, our framework *CQ-DFS* is unique and different from them because it dynamically changes the query shape according to the user’s different moving behaviors to maximize the usefulness of the range query while preserving the system performance. Being an instance of continuous range queries, *CQ-DFS* applies the central server architecture, and requests continuous range queries from mobile users, i.e., users request queries from their handsets when driving. The reason that *CQ-DFS* uses dynamic

query ranges instead of static ones is because it captures the complex moving behaviors of objects, and further adapts them on-the-fly. Fixed ranges could perform well in terms of usefulness in certain cases, but it would be impossible to achieve higher usefulness as well as *CQ-DFS* does. We leave further study between *CQ-DFS* and fixed ranges to Sections 5.1, 5.2, and 5.3.

2.3 Location Deviation

For moving users, their locations would change with time and the difficulty of retrieving the users' nearby information is increased. Although a server knows a user's reported location at present, the server would have no idea about the user's next location if he/she has not reported his/her next location after a period of time. To provide useful information for a user in time, a server has to predict where he/she will be after a period of time. Consequently, the estimated location might be different from the real one, which is clustered as location deviation related to data uncertainty or imprecise data.

In the literature, interpolation techniques have been used to determine the expected locations of moving objects in-between their two consecutive GPS records [36, 13]. [47] further models uncertain moving trajectories as a 3D cylindrical body to capture the expected deviation. [13] narrows the uncertainty problem by restricting trajectories following road networks, and analyzes the geometry of the uncertain trajectories. On the other hand, probabilistic approaches have also been employed to handle uncertain data. [6, 7] define lower and upper bounds and probability density functions of the values inside the bounds answering queries over uncertain data augmented by probability information. [41] follows the idea and proposes the *conservative functional box* binding the probabilities of objects for multi-dimensional uncertain data. [10] maps the uncertain movements to a dual space for indexing, and eliminates unqualified candidates to reduce the overall cost of query evaluations. Compared with works using interpolation techniques, *CQ-DFS* does not have information of future locations. Although they all provide a clustered region in which moving objects might appear, *CQ-DFS* only cares whether the next location of the moving object is covered by the range query while other works focus on discussing and analyzing the uncertain deviation regions. Compared with

works applying probabilistic approaches, *CQ-DFS* does not provide answers with validity but a dynamically determined fan shaped range.

2.4 Location Prediction

Location prediction frameworks predict future locations of moving objects by referencing their history trajectories. Some frameworks track a user’s most recent trajectory and predict the location after a short period of time, e.g., 5 seconds; the common navigation service is a representative example. Some other frameworks only care about rough locations and the transitions between regions. For example, say John usually leaves his office dropping by the supermarket on his way home. The transition model is formulated; thus whenever John leaves the office heading to the supermarket, it is expected with a higher possibility that he would go home afterwards. The two examples above are representatives of two common categories, *vector-based* and *pattern-based* predictions.

Vector-based predictions have two types, linear predictions [38, 46, 35, 24] and non-linear predictions [1, 42], which apply linear and non-linear mathematical functions to a model and derive future locations respectively. Among them, the *Recursive Motion Function (RMF)* [42] has the highest prediction accuracy. In *RMF*, location vectors of moving objects at time t are formulated as $l_t = \sum_{i=1}^f c_i l_{t-i}$ where c_i is a constant matrix and f is the number of the most recent locations used to determine the underlying motion pattern. The predictive location is derived according to the discovered motion pattern. *Pattern-based* predictions include the *Markov model*, and *association rules*. In the *Markov model* [22], the transition probabilities between locations are derived. It then looks at the current location and compares the transition probabilities of other locations to determine which is likely the next stop. In *association rules*, [43, 55, 48] model location transition and timestamp in the form $(p_i, t_i, c) \rightarrow (p_j, t_{i+1})$ with a confidence value c , where p_i, p_j are locations at time t_i, t_{i+1} respectively. The confidence value c is the probability that the user is likely to move from p_i at time t_i , to p_j at time t_{i+1} .

Beyond both *pattern-based* and *vector-based* predictions, the *Hybrid Prediction Model (HPM)* is proposed [25] incorporating both *association rules* and *RMF* to achieve higher accuracy and execution efficiency. However, *HPM* requires extra storage to store discovered moving pat-

terns and the Trajectory Pattern Tree of each user. Compared with all of the prediction frameworks reviewed above, *CQ-DFS* does not actually predict a user’s future location, but estimates a fan shaped region which is more likely to cover the user’s next location. Inspired by the *vector-based* prediction *RMF*, we borrow the concept applying the most recent R locations (i.e., *Reference Factor*) to determine a user’s moving behaviors, and furthermore decide suitable parameters for the fan shaped range query accordingly.



Chapter 3

Problem Definition

In this section, we first define a problem of a continuous range query. Without loss of generality, we adopt an application of nearby traffic monitoring in road networks as a case for this kind of continuous range query in this paper. Clearly, for other LBS services that provide nearby POIs, such as hotels, gas stations, and parking lots, one could apply the proposed continuous range query. Assume that a road network $G = (V, E)$ is given. Each user u_j drives in road networks and reports his/her location every \bar{t} seconds. Explicitly, if a user reports L times his/her own locations, the series of locations can form a trajectory. This trajectory consists of L location data points, represented by $\langle p_i, t_i \rangle$, where $t_{i+1} - t_i = \bar{t}$, and $1 \leq i \leq L$. In our nearby traffic monitoring scenario, when receiving $\langle p_i, t_i \rangle$ from user u_j , the server will return the query result to the user, denoted as a set E_i with $E_i = \{\langle s_{i1}, v_{i1} \rangle, \langle s_{i2}, v_{i2} \rangle, \dots, \langle s_{iM}, v_{iM} \rangle\}$. Here, each $\langle s_{ik}, v_{ik} \rangle$ represents that the estimated velocity range of the road segment s_{ik} is v_{ik} and $v_{ik} \in \{< 25km/hr, [25 - 50]km/hr, [50 - 75]km/hr, [75 - 100]km/hr, > 100km/hr\}$. For brevity of our presentation, we ignore the unit km/hr for velocity in the rest of this paper and time r_i is used to indicate the timestamp when user u_j receives E_i .

In the above example, since this user has L location data points, the server will issue a range query with a given location data point and the spatial range. The spatial range is a fan shape and the fan shape will be dynamically adjusted according to the moving speed of the user. Thus, the server returns to this user L query results, expressed by $\cup_{i=1}^L \langle p_i, t_i \rangle$. For each query location point, e.g., p_i , the server fetches all the road segments inside the range

specified and their corresponding estimated velocity ranges as a query result E_i . For each query location point, each query result should be returned to the user before he/she moves to the next location. Thus, one would like to have a constraint on the response time of the range queries issued. In this paper, we intend to guarantee that users could receive each query result within \bar{t} . In addition, the parameter τ is to restrict the maximum number of road segments retrieved. In this paper, to evaluate the performance of continuous range queries, we define two measurements, *precision* and *recall*, of the query results as follows:

Definition 1 (Precision) *Given a continuous range query Q , the precision of its query result, denoted as Ω_Q is defined as*

$$\Omega_Q = \frac{\sum_{i=2}^L \alpha_i}{\sum_{i=2}^L |E_{i-1}|} \quad (1)$$

where for $i \in [2, L]$, α_i is defined as

$$\alpha_i = \begin{cases} 1 & \text{if } r_{i-1} < t_i \wedge \exists \langle s_{(i-1)k}, v_{(i-1)k} \rangle \in E_{i-1} \text{ such that } p_i \in s_{(i-1)k} \\ 0 & \text{otherwise.} \end{cases}$$

Note that α_i is used to indicate whether the road segment that the user passes at t_i is included in the estimation E_{i-1} . In addition, α_1 cannot be measured because no estimation is available at the timestamp t_1 when the user reports its first position of a journey to the server.

Definition 2 (Recall) *Given a continuous range query Q , the recall of its query result, denoted as Γ_Q is defined as*

$$\Gamma_Q = \frac{\sum_{i=2}^L \alpha_i}{(L-1)}. \quad (2)$$

The scenario of nearby traffic estimation for a continuous range query is best understood by the example in Figure 3.1. We use the traffic status estimation service of the *CarWeb*

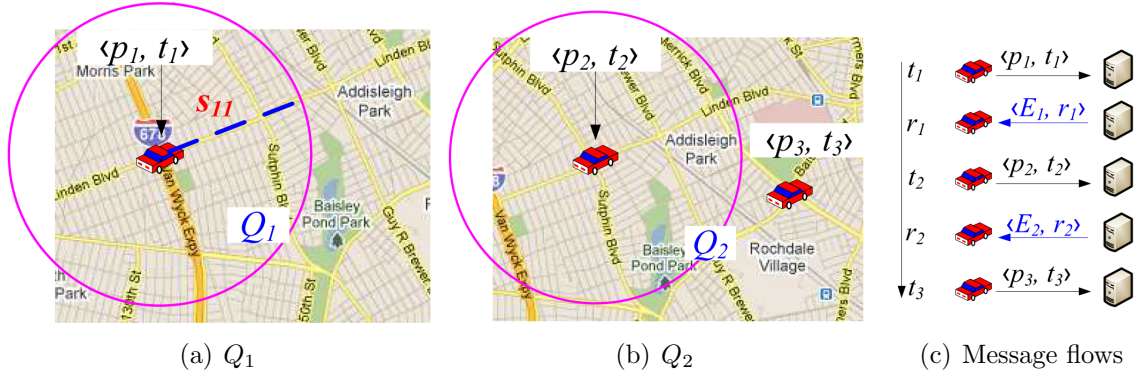


Figure 3.1: A continuous range query Q on nearby traffic monitoring in road networks.

Table I: Q_i s and corresponding E_i s.

Q_i	E_i	r_i
$Q_1 = \langle p_1, t_1 \rangle$	$\langle s_{11}, [25 - 50]km/hr \rangle, \langle s_{12}, < 25km/hr \rangle, \langle s_{13}, [25 - 50]km/hr \rangle,$ $\langle s_{14}, [75 - 100]km/hr \rangle, \langle s_{15}, [50 - 75]km/hr \rangle, \langle s_{16}, [50 - 75]km/hr \rangle$	$r_1(t_1 < r_1 < t_2)$
$Q_2 = \langle p_2, t_2 \rangle$	$\langle s_{21}, [25 - 50]km/hr \rangle, \langle s_{22}, [25 - 50]km/hr \rangle, \langle s_{23}, [25 - 50]km/hr \rangle,$ $\langle s_{24}, < 25km/hr \rangle, \langle s_{25}, [50 - 75]km/hr \rangle, \langle s_{26}, [50 - 75]km/hr \rangle,$ $\langle s_{27}, [75 - 100]km/hr \rangle$	$r_2(t_2 < r_2 < t_3)$

system as an example. For simplicity, we assume that a user only reports his/her position three times during a journey, i.e., $\langle p_1, t_1 \rangle$ shown in Figure 3.1(a), and $\langle p_2, t_2 \rangle$ and $\langle p_3, t_3 \rangle$ shown in Figure 3.1(b). Accordingly, the *CarWeb* system issues a continuous range query Q which contains two instances (i.e., Q_1 and Q_2) to estimate the traffic information that might be useful to the user. In this example, the spatial range is a circular range. The circles centered at the user shown in Figure 3.1(a) and Figure 3.1(b) are the queried ranges, corresponding to Q_1 and Q_2 , respectively. Accordingly, the server locates all the road segments inside the circular ranges, together with estimated traffic status, to form E_1/E_2 which will be returned to the user. The content of E_i s is shown in Table I. Notice that only road segment s_{11} of E_1 is explicitly depicted in Figure 3.1(a), and the remaining segments are ignored for presentation clarity. In Figure 3.1(c), the message flows between the user and the *CarWeb* system are presented.

For the performance of Q , $\alpha_2 = 1$ as p_2 is located on the road segment $s_{11} \in E_1$, and $\alpha_3 = 0$ as p_3 is not located on any road segment included in E_2 . For the number of road segments covered by range queries, $|E_1| = 6$ and $|E_2| = 7$. Consequently, the precision and recall of Q are $\frac{1+0}{6+7}$ and $\frac{1+0}{3-1}$, respectively (i.e., $\Omega_Q = 0.077$ and $\Gamma_Q = 0.5$).

Chapter 4

Design of Continuous Query with Dynamic Fan Shape

In this section, we propose a Continuous Query with Dynamic Fan-Shape (*CQ-DFS*) framework for continuous range query. First, we share some of the observations which motivate the design of *CQ-DFS*. Next, we explain the basic ideas of fan-shaped range queries. Then, we define two moving complexities of mobile users in the *CarWeb* system to facilitate the formation of proper fan-shaped range queries and present the detailed query formation algorithm. Finally, we present a caching scheme to further improve the performance.

4.1 Exploring Fan Shapes for Continuous Queries

A traditional approach for continuous range query on nearby traffic monitoring in road networks is illustrated in Figure 1.1. Specifically, we can use a circle with a fixed radius centered at a user's location to represent the region covering all road segments nearby. For static users on road networks, this is surely one of the best choices to set a range query. However, when users are moving, the location deviation issue arises in that moving users may go far beyond the circular range before query results are available if the circular range is not large enough. On the other hand, a very large radius of the circular shaped range can address the location

deviation issue, but may result in an estimation E_i containing too many road segments. Notice that a user would like to get information within a very short of period (i.e., 5 seconds in our implementation).

To provide more useful information (e.g., traffic estimation results) to moving users while preserving the system performance, the shape of a queried range is critical. To design an appropriate shape for a queried range which can maximize the precision and recall metric, we have made the following two observations. The first is that if it is predictable that a user is moving towards a specific direction, it is sure that the segments located in the opposite direction are not useful. Consequently, a query using a half circular shape as the query range is more beneficial, in terms of precision and recall under the restriction that at most τ segments could be included in an estimation. The second observation is that, while a user is moving on a certain route, the road segments within a query range may have different degrees of importance for the user. For example, if a user is moving very fast on a boulevard (say 80 km/hr), it is very likely that he/she is likely to move straight forward along the boulevard without any sudden turns. In other words, those road segments intersecting with the boulevard are less important than the boulevard itself, not to mention other road segments that are parallel to the boulevard or those farther ones. Consequently, these less important segments could be omitted in order to not violate the restriction of the maximal τ segments per estimation. However, we could not simply remove those less important segments directly from the range as it is inconsistent with the nature of range query. In addition, it is possible that the moving user would slow down and finally make a turn, so the road segments that are currently not important might become important later. The above observation suggests that when a user is moving fast, a somewhat long shape of range parallel to his/her current moving direction is preferred.

To provide moving users with useful information, we propose a Continuous Query with a Dynamic Fan-Shape (*CQ-DFS*) framework. In this framework, we adopt generalized fan shapes for range queries instead of circular shapes. The idea of adopting fan shapes for range queries is inspired by our daily experience. A flashlight illuminates unseen dark areas, and the areas illuminated by the flashlight of a midnight security patrol from a bird's-eye view are fan-shaped. Fan-shaped ranges match the dual requirements for a continuous range query. First, a moving user is situated in the center of a fan shaped range which matches the first observation

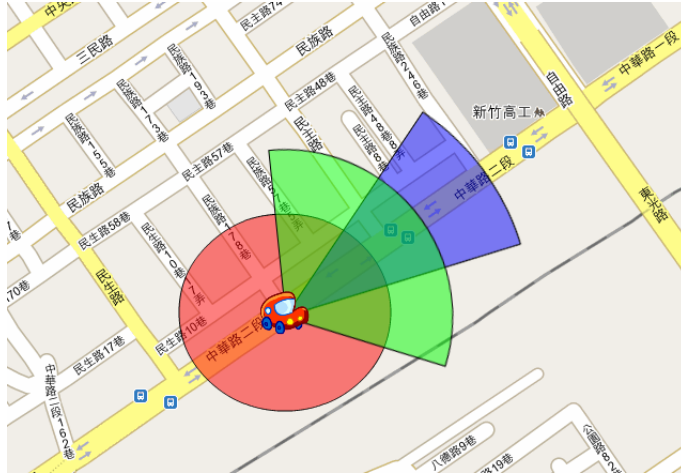


Figure 4.1: Comparing circular shape to wide and narrow forms of fan shapes.

that the road segments behind the moving user do not need to be included. Different from a circular shaped range query, a fan-shaped range query is direction aware. Second, given a fixed area of a query range, a fan-shaped form can be easily adjusted by changing the center angle and radius. For instance, a fan shape with a smaller center angle and a larger radius (i.e., a narrower fan shape) covers farther road segments ahead, while a fan shape with a larger center angle and a smaller radius (i.e., a wider fan shape) covers more nearby road segments. This easy-customization feature enables higher performance for different kinds of moving behaviors of users, as shown in Figure 4.1.

4.2 Deciding Fan Shapes On-The-Fly

To adapt our framework to different moving behaviors, we first define two kinds of moving complexities, *angular complexity* and *distance complexity*. For a movement, its speeds and route are widely used to capture its moving behavior. In addition, if we know a user's driving direction and speed, the user's movement can be captured. Therefore, in this paper, we define an angular complexity and a distance complexity to capture the directional changes of a movement and how far a movement is, respectively.

To capture a user's moving behavior, we observe his/her past movements, and then formulate the moving complexities. First, we use a *reference factor*, R , to decide how many past positions are to be taken into consideration when we calculate a user's moving complexities.

Specifically, the reference factor R is defined as the number of former locations reported by a user (i.e., $p_i, p_{i-1}, p_{i-2}, \dots, p_{i-R+1}$) that are considered when calculating the moving complexity of the user at position p_i . Note that for any p_i such that $i < R$, we set $R = i$. Based on a given reference factor R , the angular complexity and the distance complexity are defined in Definition 3 and Definition 4, respectively.

Definition 3 (Angular Complexity) *Given a user located at p_i and a reference factor R , let angle $a_k (= \angle p_k p_{i-R+1} p')$ be half of the center angle of the minimal fan shape range centered at p_{i-R+1} and meanwhile covering p_k with $k \in (i - R + 1, i]$. Here, vector $\overrightarrow{p_{i-R+1} p'}$ corresponds to the direction vector at p_{i-R+1} . The angular complexity, denoted as A_{ci} is set to the maximal angle a_k w.r.t. $k \in (i - R + 1, i]$. Mathematically, A_{ci} is expressed as*

$$A_{ci} = \text{MAX}(2 \cdot \text{MAX}_{k=i-R+2}^i \angle p_k p_{i-R+1} p', A_{min}). \quad (1)$$

Notice that in order to prevent the case where all the a_k s are too small (e.g., when users are moving along a straight line), A_{min} is used as the lower bound of A_{ci} . In our default parameter setting, we set $A_{min} = 30^\circ$.

Definition 4 (Distance Complexity) *Given a user located at p_i and a reference factor R , distance $d_k (= \text{dis}(p_{i-R+1}, p_k))$ is the radius of the minimal fan shaped range centered at p_{i-R+1} and meanwhile covering p_k . The distance complexity, denoted as D_{ci} is set to the maximal distance d_k w.r.t. $k \in (i - R + 1, i]$. Mathematically, D_{ci} is expressed as*

$$D_{ci} = \text{MAX}(\text{MAX}_{k=i-R+2}^i \text{dis}(p_{i-R+1}, p_k), D_{min}). \quad (2)$$

Notice that D_{min} is used as a lower bound of D_{ci} to prevent cases where d_k s are very small. In our implementation, we set $D_{min} = 30m$.

For example, in Figure 4.2, red flags represent the geographical points reported by a user,

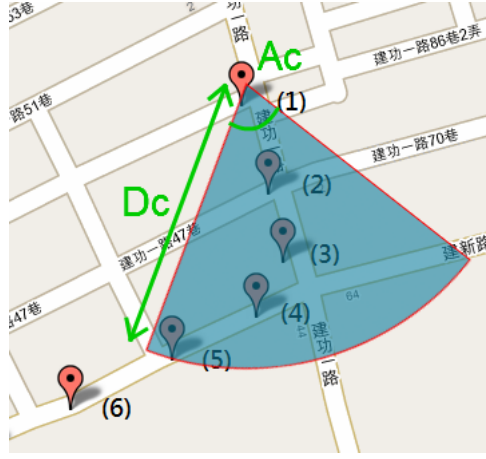


Figure 4.2: Angular complexity A_{c5} and distance complexity D_{c5} for $R = 5$.

numbered $p_1 \sim p_6$ in sequence with p_1 reported first and p_6 reported last. Assume reference factor $R = 5$, and the user is currently located at p_5 . To derive A_{c5} , we start from the location and the direction of p_1 looking for the minimum angle required to fully cover $p_1 \sim p_5$, which is illustrated as the center angle of the shaded angular sector. Similarly, the D_{c5} is derived starting from p_1 searching for the minimum distance required to cover $p_1 \sim p_5$, which is showed as the radius of the shaded angular sector. The reason A_{c5} is as large as Figure 4.2 shows is because the direction of p_1 is considered, which is approximately south-southeast. The shaded angular sector is the minimum angle required to cover $p_1 \sim p_5$ assuming the user requests a fan shaped range query from p_1 . Suppose the user moves to p_6 . It is observed that A_{c6} is much larger than A_{c5} . (A_{c6} is derived from $p_2 \sim p_6$, and the direction of p_2 is similar to p_1 's.) The philosophy of moving complexity actually assumes that moving objects would keep following current moving behaviors until the behaviors start to change. In our example, the user makes a right turn during $p_3 \sim p_4$. As a result, *CQ-DFS* assumes the user would keep making turns (requiring larger A_c) until the user starts to move straight.

The philosophy of angular and distance complexities is based on the assumption that whenever the user is moving fast on a boulevard or highway (i.e., a lower value of A_c and a higher value of D_c), he/she would keep moving fast until gradually slowing down (e.g., leaving the highway). On the contrary, if a user keeps wandering around the city turning left and right (i.e., a higher value of A_c and a lower value of D_c), he/she would keep doing so until gradually accelerating (e.g., leaving the city). The moving complexities significantly facilitate the understanding of different moving behaviors of users and the real-time formation of the

queried range.

However, it is possible that when users make sudden turns, a few following range queries would have sudden parameter distortion. For example, in Figure 4.2, if the user continues making turns at $p_5 \sim p_6$, the parameter distortion comes in handy. Here we are emphasizing that the changes between $A_{c4} \sim A_{c6}$ are so intense that when the fixed area of the range query is relatively small, the range query may only return a small number of segments which definitely affect the overall precision and recall. To capture whenever users are making turns, and meanwhile to avoid distortion, we apply a weighted moving average method to smooth the angular complexity and distance complexity, as defined in Definition 5 and Definition 6.

Definition 5 (Smoothed Angular Complexity) *Smoothed angular complexity SA_{ci} of p_i is defined as the weighted moving average of A_{ci} with parameter K , i.e.,*

$$SA_{ci} = \frac{K \cdot A_{ci} + (K - 1) \cdot A_{c(i-1)} + \dots + A_{c(i-K+1)}}{K + (K - 1) + \dots + 1}. \quad (3)$$

Note that for any p_i such that $i < K$, we set $K = i$.

Definition 6 (Smoothed Distance Complexity) *Smoothed distance complexity SD_{ci} of p_i is defined as the weighted moving average of D_{ci} with parameter K , i.e.,*

$$SD_{ci} = \frac{K \cdot D_{ci} + (K - 1) \cdot D_{c(i-1)} + \dots + D_{c(i-K+1)}}{K + (K - 1) + \dots + 1}. \quad (4)$$

Note that for any p_i such that $i < K$, we set $K = i$.

After defining moving complexities capturing different moving behaviors, we propose an approach to determine a proper fan-shaped range on-the-fly. Recall that to shorten a response time, the number of road segments in each estimation is limited on a server. We first set $Area_{fixed}$ as the upper bound size of the fan shaped ranges, and express it in Equ (5), where $Angle$ and $Radius$ represent the center angle and radius of the fan shape with area equals to $Area_{fixed}$.



Figure 4.3: Expanding $\langle A_{c5}, D_{c5} \rangle$ forming the desired $\langle Ang_5, Rad_5 \rangle$ with $R = 5$.

$$Area_{fixed} = \pi \cdot Radius^2 \cdot \frac{Angle}{360^\circ}. \quad (5)$$

Consequently, the desired fan shaped range could be represented as a two-tuple vector $\langle Angle, Radius \rangle = \langle Ang_i, Rad_i \rangle$. Based on moving complexities A_{ci} and D_{ci} , countless kinds of fan shaped ranges with the same area $Area_{fixed}$ could be formed, e.g., one ($\langle A_{ci}, D_{derived} \rangle$) has A_{ci} as the central angle and another ($\langle A_{derived}, D_{ci} \rangle$) has D_{ci} as the radius; the former type usually has a long narrow fan shape, and the latter one usually has a short wide fan shape. Given two fan shaped ranges at both extremes, *CQ-DFS* applies the concept of *Doctrine of the Mean* expanding both the central angle and the radius from A_{ci} and D_{ci} until the area of the expanded fan shape reaches $Area_{fixed}$. Figure 4.3 depicts this idea, where the small fan shape represents moving complexities $\langle A_{c5}, D_{c5} \rangle$, and the large fan shape $\langle Ang_5, Rad_5 \rangle$ is expanded from it. $\langle Ang_5, Rad_5 \rangle$ represents our desired fan shape for range query Q_5 . Note that the arrow starting from p_5 pointing northeast represents the direction that p_5 faces, and it is also the direction that *CQ-DFS* requests the desired Q_5 .

In order to examine the ratio between the given fixed area $Area_{fixed}$ and the original fan shape $\langle A_{ci}, D_{ci} \rangle$, we derive the *Area Multiplier* AM_i in Equ (6), which indicates the ratio by which the original fan shape could be expanded. With AM_i available, we multiply both $Radius^2$ and $Angle$ terms in Equ 5 with the square root of AM_i . The Ang_i and Rad_i of the desired fan-shaped range are derived by mapping the corresponding terms Rad_i^2 and Ang_i

respectively.

For example, suppose the $Area_{fixed} = 100,000 m^2$, and the area of the original fan shape $\langle A_{ci}, D_{ci} \rangle$ equals $50,000 m^2$, which makes the $AM_i = 2$. We multiply D_{ci}^2 and A_{ci} with $\sqrt{2}$, i.e., $100,000 = \pi \cdot D_{ci}^2 \cdot \sqrt{2} \cdot \frac{A_{ci} \cdot \sqrt{2}}{360^\circ}$; thus, the desired $\langle Ang_i, Rad_i \rangle = \langle A_{ci} \cdot \sqrt{2}, \sqrt{D_{ci}^2 \cdot \sqrt{2}} \rangle$.

$$AM_i = \frac{Area_{fixed}}{\pi \cdot D_{ci}^2 \cdot \frac{A_{ci}}{360^\circ}} \quad (6)$$

$$\begin{aligned} Area_{fixed} &= \left(\pi \cdot D_{ci}^2 \cdot \frac{A_{ci}}{360^\circ} \right) \cdot (AM_i) \\ &= \pi \cdot (D_{ci}^2 \cdot \sqrt{AM_i}) \cdot \left(\frac{A_{ci} \cdot \sqrt{AM_i}}{360^\circ} \right) \\ &= \pi \cdot (Rad_i^2) \cdot \left(\frac{Ang_i}{360^\circ} \right) \end{aligned}$$

$$\begin{aligned} \frac{Ang_i}{360^\circ} &= \frac{A_{ci} \cdot \sqrt{AM_i}}{360^\circ} \\ Ang_i &= A_{ci} \cdot \sqrt{AM_i} \quad (7) \end{aligned}$$

$$\begin{aligned} Rad_i^2 &= D_{ci}^2 \cdot \sqrt{AM_i} \\ Rad_i &= \sqrt{D_{ci}^2 \cdot \sqrt{AM_i}} \quad (8) \end{aligned}$$

4.3 Caching Overlapping Query Results

While driving, people request queries in short periods to avoid missing upcoming events. Frequent queries would induce heavy computations for servers. However, we observe that in certain short periods of continuous range query applications, consecutive queried ranges (i.e., Q_i and Q_{i+1}) possibly overlap geometrically, and the queried ranges issued by different users (i.e., Q_i corresponding to user u_j and $Q_{i'}$ corresponding to user $u_{j'}$) also possibly overlap geometrically. Consequently, caching of the result w.r.t. Q_i would benefit the processing of another range query.

However, the retrieved information would have an expiration time. For instance, the traffic status usually changes over time, so a traffic estimation has an expiration time. On the other hand, the evaluated traffic status remains the same, whenever any other user who has also issued overlapping query ranges for traffic monitoring services, the information could be

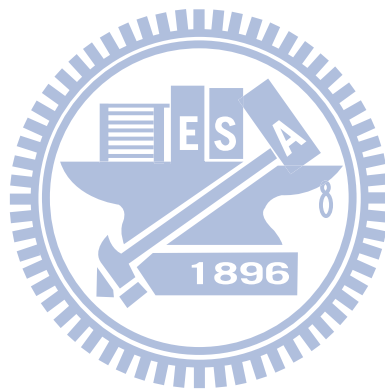
delivered directly as well. In this paper, we only focus on the benefit that a caching scheme would provide to the original user u_j , but do not detail an intra-user system architecture.

Similarly, we take an application of nearby traffic monitoring in road networks as an example. First, suppose $\tau = 40$, $Area_{fixed} = 50000m^2$ and the query Q_1 contains totally 40 road segments ($\leq \tau$), and the corresponding query result E_1 is delivered on time (i.e., $r_1 < t_2$). Assume that the query Q_2 also contains 40 road segments with 15 overlapping with Q_1 . Consequently, the server only needs to find out those 25 non-repeated road segments. As the query processing cost (to be more specific, the estimation of the traffic status of road segments) is the major performance bottleneck, caching of 15 road segments locally, together with their estimated traffic information, leaves extra space to retrieve another 15 road segments. In other words, the server is capable of evaluating a larger query shape which contains 55 road segments. Given the fact that 15 road segments are already known to the server, it only retrieves those non-repeated road segments.

To reasonably expand the fan shape when overlapping occurs, the idea is to proportionally increase the fixed area w.r.t. the *overlap ratio* ξ_i of Q_i , i.e., the ratio of the number of overlapping road segments to the total road segments contained in the range. Continuing the example above, Q_2 contains 40 road segments with 15 overlapping with Q_1 , and the given area $Area_{fixed} = 50000m^2$. The overlap ratio $\xi_2 = \frac{15}{40} = 0.375$, and hence the area of Q_2 will be expanded to $\frac{50000}{1-\xi_2} = 80000$ square meters.

With the caching capability of a server enabled, a range query request from a user is processed as follows. Take nearby traffic monitoring in road networks as an example. First, the server forms a range query Q_i by *CQ-DFS*. Second, it looks up the traffic estimation cache to determine how many road segments included in Q_i are actually locally available and derives the overlap ratio ξ_i accordingly. It then expands the search area of Q_i based on ξ_i and fetches those non-repeated road segments included in Q_i . Finally, an estimation E_i which contains all the road segments included in Q_i , i.e., both overlapping and non-overlapping ones, is delivered to the user to finish this service request. Note that this approach is based on the assumption that the density of the road network nearby is approximately equal. In other words, whenever the user passes from a sparse road network through a dense one, the total number of road segments covered by the certain expanded fan shape could be much more compared to traveling in the sparse density road network. Nevertheless, as we mentioned

earlier, compared to controlling the number of road segments, the fixed area scheme is already an underestimated approach. Thus, in *CQ-DFS*, the assumption of an equal density road network is acceptable.



Chapter 5

Performance Study

In this section, a comprehensive simulation is conducted to evaluate the performance of the *CQ-DFS* framework. We implement the *CarWeb* traffic status estimation service in the PHP programming language running on a *CarWeb* Ubuntu server powered by an Intel(R) Core 2 Duo 2.66GHz CPU with 8,960MB memory and PostgreSQL with PostGIS extension as the backend spatial database.

A real dataset extracted from the *CarWeb* repository [27] is used. The *CarWeb* repository records the user locations reported every 5 seconds by GPS devices. After filtering out noises like discontinuity, and short trajectories which contain less than 30 records, in total of 144 trajectories with 25,923 GPS records are retained. In order to study the moving behaviors of different types of users, we cluster those representative trajectories into four subsets. Subset 1 consists of all the trajectories with their distance complexity falling between 30 and 70 meters and hence it represents the slow moving group with velocity between $21.6 \sim 50.4$ *km/hr*. Subset 2 consists of all the trajectories with their distance complexity falling between 90 and 150 meters and hence it corresponds to the fast moving group with velocity between $64.8 \sim 108$ *km/hr*. Subset 3 consists of all the trajectories with their angular complexity falling between 30° and 90° and hence it represents the directional moving group. Subset 4 consists of all the trajectories with their angular complexity falling between 120° and 180° and hence it represents the undirectional moving group. Table I summarizes the properties of these four subsets. Each subset is filtered through the same procedure, which goes through

every trajectory and retains 50 consecutive GPS records conforming to the constraints of each subset, for example, searching through a highway trajectory to extract 50 consecutive points with angular complexity falling within $30 \sim 90$ degrees and clustered in Subset 3. Notice that the angular and distance complexities in subset constraints are the same as Definitions 1 and 2, and the default parameter settings of A_{min} and D_{min} are 30 degrees and 30 meters respectively. For reasons of fairness, we totally retain 50 trajectories in each subset, thus each subset contains 2,500 GPS records. In other words, the subset data generation process starts from the full dataset going through each real trajectory alphabetically, and retain 50 consecutive records as a trajectory. The process ends whenever 50 trajectories are clustered in each subset, or all trajectories from the full dataset have been visited.

Table I: Trajectory subsets details.

Subset Number	Feature	Distance Complexity	Angular Complexity
Subset 1	Slow Moving	30 ~ 70 meters	X
Subset 2	Fast Moving	90 ~ 150 meters	X
Subset 3	Directional Moving	X	30 ~ 90 degrees
Subset 4	Undirectional Moving	X	120 ~ 180 degrees

As defined in Section 3, *Precision* and *Recall* are the main performance metrics used in this work. In the simulation, we assume a user moves along a trajectory and updates its location every 5 seconds. Right after receiving a location update from a user, the server forms a fan-shaped range query on-the-fly, processes the query, and sends back the result to the user as a traffic estimation. For each estimation E_i , we record the response time between the user submitting a location update and the user receiving E_i , and those estimations with response time longer than 5 seconds will not contribute to the *Precision* and *Recall* metrics. Let us give a more detailed description of how we evaluate *Precision* and *Recall* for a trajectory which is composed of 50 GPS records. p_1 is the first location update from the user, the evaluation of both metrics starts from p_2 to p_{50} examining whether those locations are covered by the previous range queries whenever the query results are delivered on time. Going through $p_2 \sim p_{50}$, a *Precision* value and a *Recall* value are evaluated. Given a total of 50 trajectories in each data subset, 50 *Precision* and *Recall* values are evaluated respectively. The average *Precision* and *Recall* values are calculated, and graphed in all sets of experiments.

Six sets of experiments are conducted. The first three are to evaluate the performance of

CQ-DFS, compared with the performance of other continuous queries with different shapes, for different user moving behaviors. To be more specific, we evaluate the performance of *CQ-DFS* with the users of different moving speeds (i.e., different distance complexities) via using subset 1 and subset 2 in Section 5.1, evaluate the performance of *CQ-DFS* with the users of different moving directions (i.e., different angular complexities) via using subset 3 and subset 4 in Section 5.2, and evaluate the performance of *CQ-DFS* with the users of different moving behaviors (i.e., different distance complexities and different angular complexities) via using the full dataset in Section 5.3. In addition to *CQ-DFS* which forms fan-shaped range queries dynamically (denoted as **DynamicFan**), we also implement range queries of circular shapes, static fan shapes with a central angle of 120° , and static fan shapes with a central angle of 45° , referred to as **Circular**, **StaticFan120**, and **StaticFan45** respectively. Note that **Circular** is centered at the user’s location, while **StaticFan120**, **StaticFan45**, and **DynamicFan** request fan shaped range queries according to the user’s direction at each query point. For each round of simulation, given an area of fixed size, the parameters of each query shape are derived accordingly.

In the fourth set of experiments, we verify the formation of the fan shaped query. Given a fixed area, we can form a fan shape based on distance complexity D_{ci} or angular complexity A_{ci} , as mentioned in Section 4. However, we have proposed different approaches to improve the *Precision* and *Recall* of fan-shaped range queries, including the smoothed complexities introduced in Section 4.2, adopting the concept of Doctrine of the Mean to determine the central angle and the radius of the fan shape introduced in Section 4.2, and enabling the caching of overlapping traffic results in the server introduced in Section 4.3. For presentation simplicity, they are referred to as **Smo.Com.**, **Dyn.FanForm**, and **Caching** respectively in the following descriptions. Consequently, we want to demonstrate that the proposed approaches *do* improve the performance to a certain degree. In order to achieve this, we implement and evaluate five fan shaped formation algorithms, as summarized in Table II. The first two algorithms, i.e., **FanA_{ci}** and **FanD_{ci}** are purely based on either angular complexity A_{ci} or distance complexity D_{ci} , and they do not support caching. Algorithm **Fan** is still based on complexities A_{ci} and D_{ci} but it *does* consider both complexities and apply the concept of doctrine of the mean to form the fan shaped search range, i.e., supporting **Dyn.FanForm**. Instead of using complexities, the fourth algorithm **SFan** considers smoothed complexities

SA_{ci} and SD_{ci} , i.e., supporting both Dyn.FanForm and Smo.Com.. The last algorithm CSFan is almost the same as SFan but it enables the caching of overlapping traffic results, i.e., supporting Smo.Com., Dyn.FanForm, and Caching. The experimental results, to be represented later, will demonstrate the effectiveness of different approaches.

In the fifth experiment, we report the query response time of experiments in Sections 5.3 and 5.4, comparing different settings and different shapes of query ranges given the same query size. We also demonstrate the reasons we applied 80,000 m^2 as the maximum query size in this experiment.

In the sixth experiment, we further perform the same simulation of experiments in Sections 5.3 and 5.4 by using different period of location reporting datasets and different query sizes. The purpose of this experiment is to examine if *CQ-DFS* could be applied to different periods of moving trajectories.

Table II: Fan shaped formation algorithms for a fan with its area = $Area_{Fixed}$.

Approach	Ang_i	Rad_i	Caching
FanA _{ci}	A_{ci}	$\frac{Area_{fixed} \cdot 360^\circ}{\pi}$	No
FanD _{ci}	$\frac{Area_{fixed} \cdot 360^\circ}{\pi \cdot D_{ci}^2}$	D_{ci}	No
Fan	$A_{ci} \cdot \sqrt{AM_i}$	$D_{ci} \cdot AM_i^{\frac{1}{4}}$	No
SFan	$SA_{ci} \cdot \sqrt{AM_i}$	$SD_{ci} \cdot AM_i^{\frac{1}{4}}$	No
CSFan	$SA_{ci} \cdot \sqrt{AM_i}$	$SD_{ci} \cdot AM_i^{\frac{1}{4}}$	Yes

Please refer to Table III for the system parameter settings. Notice that in all experiments, we evaluate the *Precision* and *Recall* of different range queries with fixed areas ranging between 10,000 and 80,000 m^2 . We set 10,000 m^2 as the lower boundary for the *CarWeb* traffic status estimation service. For example, **Circular** with 10,000 m^2 area has a radius of 56.42 meters and **StaticFan45** with 10,000 m^2 area has a radius of 159.58 meters. Consequently, any smaller area would significantly drop the *Precision* and *Recall* no matter which query shape is applied. On the other hand, **Circular** with an 80,000 m^2 area has a radius of 159.58 meters and **StaticFan45** with an 80,000 m^2 area has a radius of 451.35 meters. Consequently, we set 80,000 m^2 as the upper boundary, as a larger area would cover too many road segments that could not guarantee that the query result would be delivered on time.

Table III: Environmental constants and parameter settings.

Name	Value
A_{min} : Min. Ang. Complexity	30°
D_{min} : Min. Dist. Complexity	30 m
Fixed Area	10,000 \sim 80,000 m^2
R : Reference Factor	3
K_{WMA}	3

5.1 Comparing Query Shapes for Different Distance Complexities

The first set of experiments is to evaluate the performance of continuous queries with different shapes in the cases where users have different distance complexities. First, we study their performance when the users are moving relatively slowly via using subset 1 with the results displayed in Figures 5.1(a) and 5.1(b). It is observed that **StaticFan45** performs the worst in both *Precision* and *Recall*. In *Precision*, **StaticFan45** performs significantly worse than the others no matter which area size is applied. In *Recall*, **StaticFan45** only achieves around 0.75 even when the area size reaches 80,000 m^2 . This is because subset 1 corresponds to the slow moving group; there are many trajectories in the central downtown area where moving objects keep changing moving directions. **StaticFan45**, due to its relatively narrow and long query range, does not match the moving behaviors of users within Subset 1. **Circular** performs slightly worse than **StaticFan120** when the area of the queried range is small (e.g., 10,000 \sim 40,000 m^2). **DynamicFan** performs the best in both *Precision* and *Recall*, especially in the cases where the query range is of small size (e.g., 10,000 \sim 30,000 m^2). The reason behind this is that *CQ-DFS* considers the former moving behaviors when predicting the behaviors, and changes the queried ranges when moving behaviors change. Consequently, **DynamicFan** has a higher chance of forming queries that truly match the real movements. In subset 1, **DynamicFan** usually widens the central angle to larger than 130° .

Next, we study their performance when the users are moving relatively fast via using subset 2 with the results displayed in Figures 5.1(c) and 5.1(d). Different from previous observations, **Circular** and **StaticFan120** have the lowest *Precision* and *Recall*. This is because subset 2 corresponds to users with relatively fast velocity. In other words, trajectories in subset

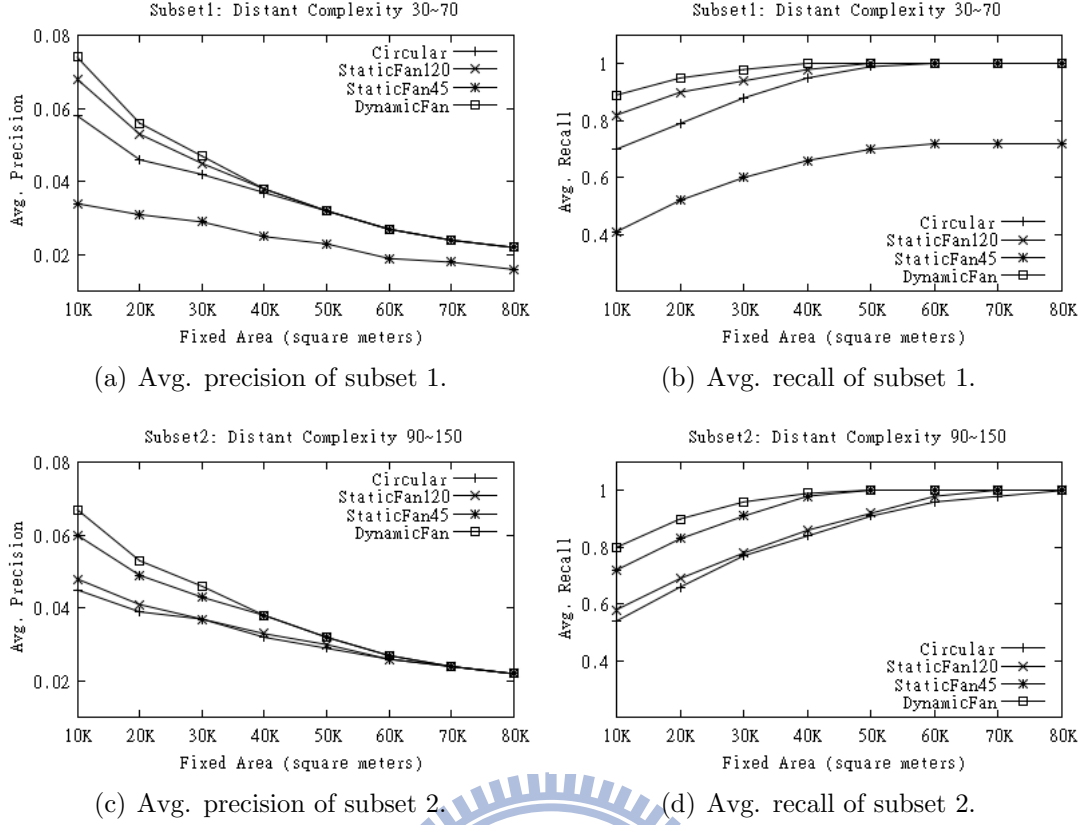


Figure 5.1: Experimental results of subset 1 and 2 comparing different query shapes.

2 are normally less complex and experience fewer turns. Consequently, direction-insensitive approaches like Circular and StaticFan120 do not match users' real movements, unless the fixed area reaches 70,000~80,000 m^2 . It is observed that StaticFan45 and DynamicFan perform much better in both *Precision* and *Recall*, while DynamicFan performs slightly better, due to its customizable feature that it is stretched to a narrow shape with a small angle between $30^\circ \sim 40^\circ$, which is slightly narrower and longer than StaticFan45. In brief, compared with continuous queries of other shapes, DynamicFan demonstrates the best resilience to the distance complexities, and it achieves the best *Precision* and *Recall* for both slow-moving users and fast moving users.

Note that all *Precision* results in our performance study appear strictly decreasing as the given query size increases. This is because a larger query range tends to cover more road segments, while there is only one of them which could possibly be considered as a relevant road segment in the *Precision* metric, as Definition 1 shows. On the other hand, all *Recall* results in our performance study appear to be increasing and converge to a specific value or 1

as the given query size increases. This is because a larger query range tends to have a higher possibility of covering the relevant road segment of each range query, while the number of relevant road segments of a trajectory is fixed, that is the length of the trip excluding the start point. We discuss these shared phenomena in this section, and will not repeat it in the following experiments.

5.2 Comparing Query Shapes for Different Angular Complexities

The second set of experiments is to evaluate the performance of continuous queries with different shapes in the cases where users have different angular complexities. First, we study their performance when the users are moving directionally via using subset 3 with the results displayed in Figures 5.2(a) and 5.2(b). It is observed that **Circular** and **StaticFan120** both have lower *Precision* and *Recall*, while **StaticFan45** and **DynamicFan** offer much better *Precision* and *Recall*. We also observe that the simulation result for subset 3 is similar to the result reported in Figures 5.1(c) and 5.1(d), which is expected since both subsets share a great portion of the fast moving trajectories on the highway.

Nevertheless, when the fixed area of the query range is small, **DynamicFan** outperforms **StaticFan45**. This is because around half of the trajectories are moving straight and fast and **DynamicFan** actually forms query ranges with a narrowed central angle (i.e., $< 45^\circ$) which successfully captures some fast moving behaviors that **StaticFan45** could not cover.

We also study their performance when the users are moving unidirectionally via using subset 4 with the result displayed in Figures 5.2(c) and 5.2(d). **StaticFan45** generates the worst *Precision* and *Recall*. Even when the fixed area is set to $80,000 \text{ m}^2$, its *Recall* is only around 0.65. **StaticFan120** performs much better as it reaches more than 0.9 in *Recall* when the fixed area is $80,000 \text{ m}^2$. We also observe that **StaticFan120** could not achieve 1 in *Recall*, which is very different from its performance in the above experiments. This is because subset 4 mainly consists of many complex moving behaviors. **DynamicFan** and **Circular** both perform very well as their *Precision* and *Recall* almost overlap and lie between $0.7 \sim 1$. This reveals

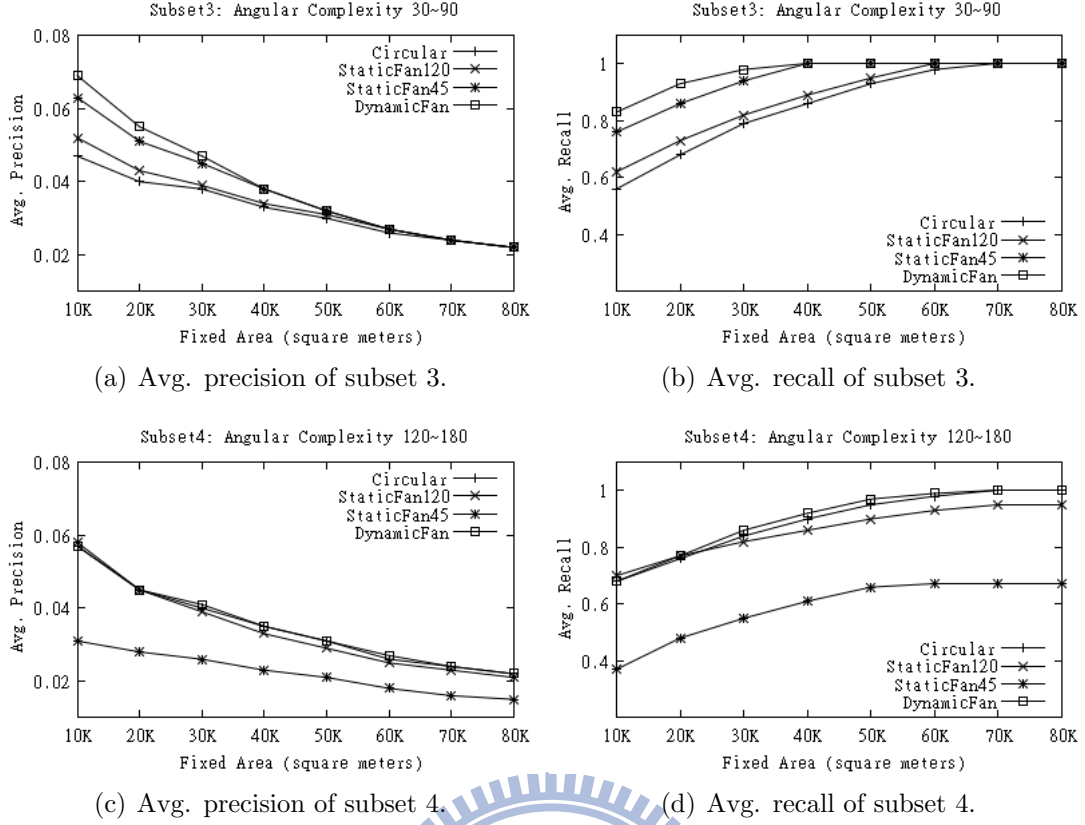


Figure 5.2: Experimental results of subsets 3 and 4 comparing different query shapes.

two important findings. Firstly, the angular complexity required to completely accommodate subset 4 is larger than 120° , since **StaticFan120** does not achieve 1 in *Recall* while **Circular** does. Secondly, **Circular** performs so well when the fixed area is large enough (e.g., when the area is larger than $60,000 m^2$). This implies that complex trajectories usually have high angular and short distance complexities. Consequently, location deviation is not significant. It is intuitive as it is almost impossible that a fast-driving user actually generates S-shaped or V-shaped complex trajectories and we do not consider car race track trajectories.

Note that we emphasize that **Circular** could handle troublesome cases well only if the fixed area is large enough. This brings up another observation made from Figure 5.2(d) that when the fixed area is small (e.g., $10,000 m^2$), **StaticFan120** performs slightly better than both **DynamicFan** and **Circular**. We find that those trajectories which enable **StaticFan120** to outperform others have relatively faster velocity that deviate out of the **DynamicFan** and **Circular** ranges.

We also find that, when the query range is set to $10,000 m^2$, those critical trajectories

causing **StaticFan120** which outperform **DynamicFan** all start with high angular complexities, followed by sudden acceleration. Consequently, query shapes insensitive to location deviation could not react soon enough. The main reason behind this is that $10,000 m^2$ is too small and so does not offer sufficient room for **DynamicFan** to adjust its shape flexibly. In the *CQ-DFS* framework, **DynamicFan** would expand the angle and radius simultaneously, while in certain cases, **DynamicFan** is expecting complex trajectories at the moment moving objects accelerate. Consequently, **DynamicFan** maintains a fan shape larger than 120° , which did not successfully cover the following records in the trajectories. However, when the fixed area reaches at least $20,000 m^2$, **DynamicFan** performs best again.

5.3 Comparing Query Shapes with a Full Dataset

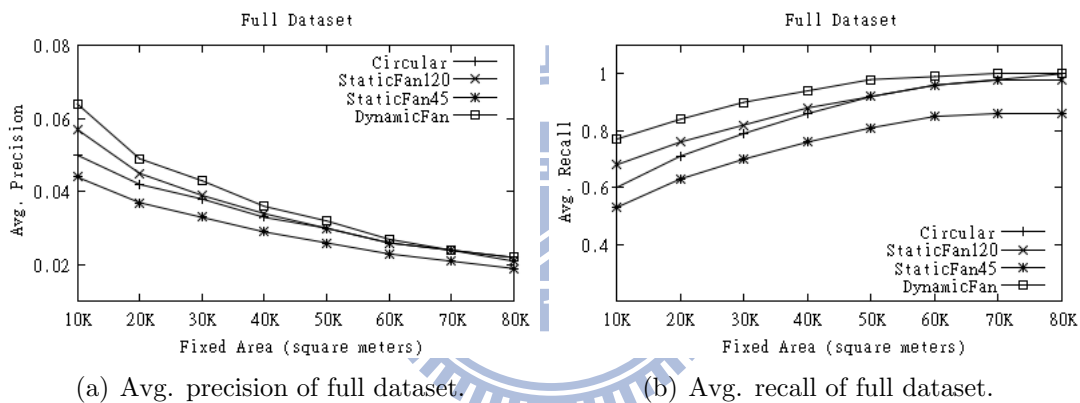


Figure 5.3: Experimental results of full dataset.

The third set of experiments is to evaluate the performance of continuous queries with different shapes in the cases where different user movement patterns are captured via using the full dataset with the results reported in Figure 5.3. No matter which fixed area is used, **DynamicFan** always performs the best, while **StaticFan45** performs the worst. **StaticFan120** and **Circular** perform similarly. Only when the area is smaller (e.g., $10,000 \sim 40,000 m^2$), does **StaticFan120** perform slightly better. This reflects that **Circular** experiences location deviation when the area is small. However, **Circular** could handle complex trajectories better than **StaticFan120** when the area is larger. According to the experimental results obtained from Sections 5.1, 5.2, and 5.3, **StaticFan45** could not achieve 1 in *Recall* in subset 1, **StaticFan45**

and StaticFan120 could not achieve 1 in *Recall* in subset 4 which explains why both StaticFan45 and StaticFan120 could not successfully achieve 1 in *Recall* in the full dataset.

5.4 Effectiveness of the Fan Shaped Range Formation

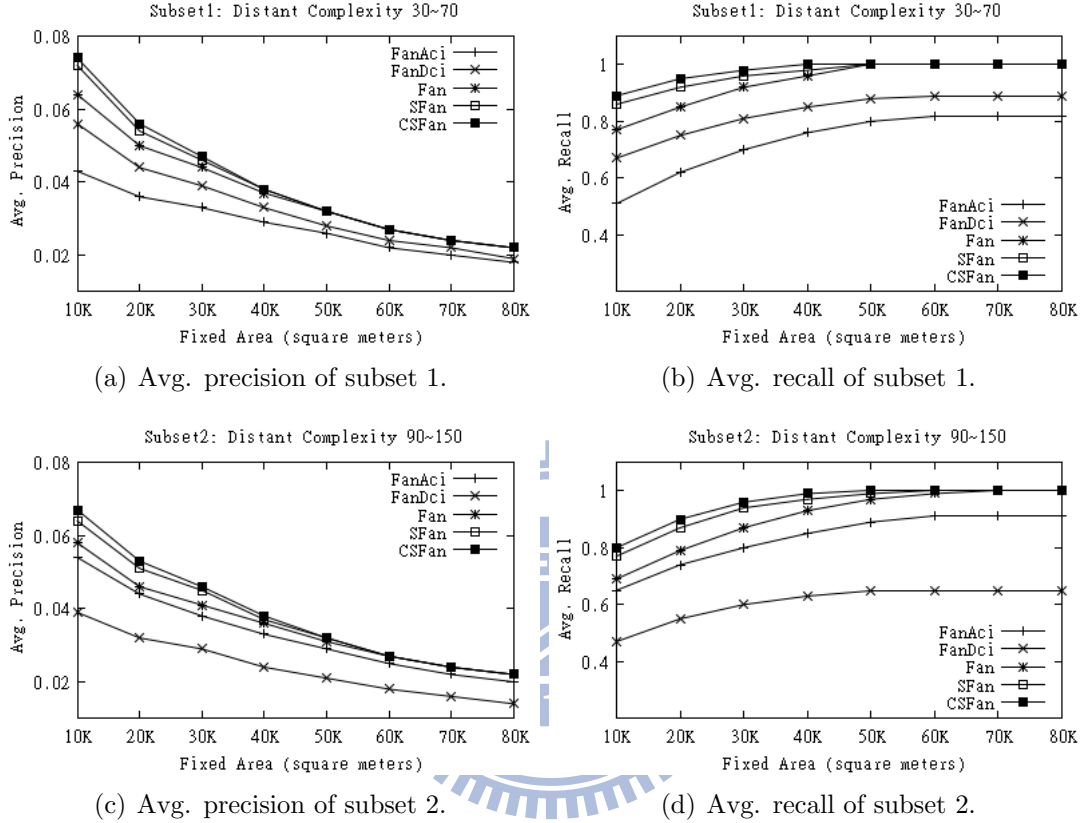


Figure 5.4: Experimental results of subsets 1 and 2 comparing different fan shaped formation algorithms.

The fourth set of experiments is to evaluate the effectiveness of the approaches Smo.Com., Dyn.FanForm, and Caching proposed in Section 4. We evaluate the *Precision* and *Recall* of different fan shaped formation algorithms, as reported in Figures 5.4 and 5.5. In general, we could observe that FanD_{ci} and FanA_{ci} which do not implement any of the proposed approaches in forming the fan shape query range perform the worst. On the contrary, the Smo.Com., Dyn.FanForm, and Caching approaches do improve the performance, and the algorithm CSFan that supports all three approaches performs the best. We also observe that as the area of the fan shape range increases, the improvement brought by our proposed approaches decreases.

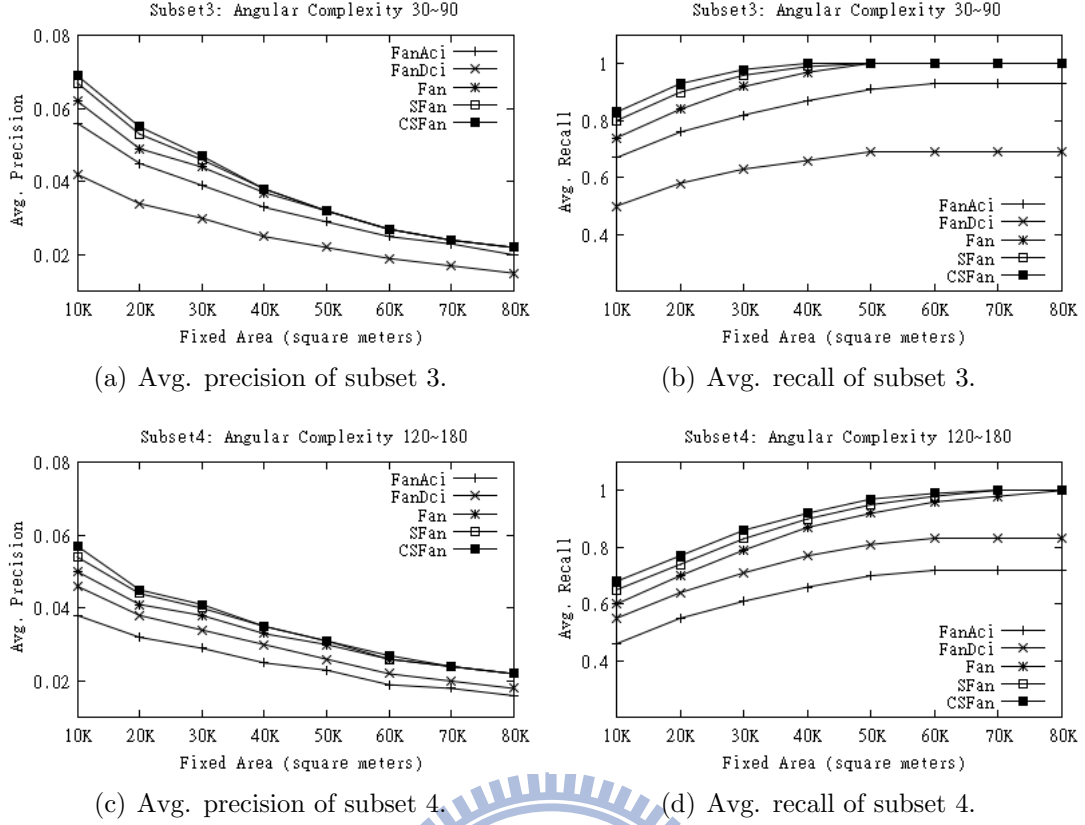
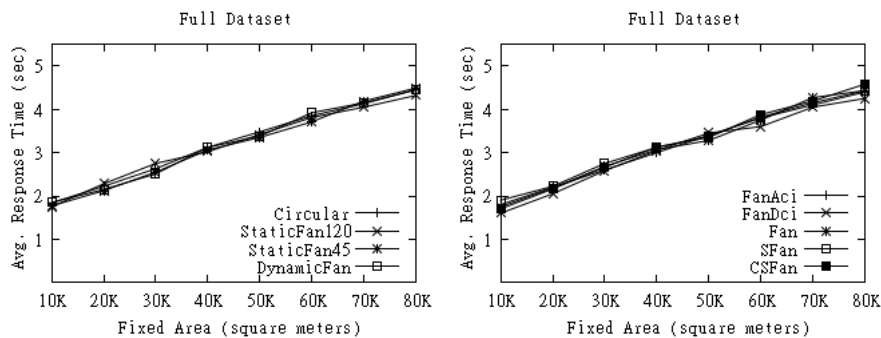


Figure 5.5: Experimental results of subsets 3 and 4 comparing different fan shaped formation algorithms.

This is because, when the area of the query range reaches a certain size, the algorithm Fan has already achieved 1 in *Recall* and hence there is no room for further improvement brought by Smo.Com. and Caching. Since CSFan has the best performance, our *CQ-DFS* adopts it as the algorithm to form the fan-shaped query range dynamically, and the notation DynamicFan used in the above experiments refers to CSFan.

5.5 Query Response Time Study

In order to report the efficiency of the *CQ-DFS* framework, we present the average response time of the experiments in Sections 5.3, and 5.4 by using the full dataset in Figures 5.6(a) and 5.6(b) respectively. We found that different query shapes and fan shaped formation algorithms do not have much difference in response time given the same query size. When the size equals 10K m^2 , all of them have around 2 seconds response time. As the size increases, the response



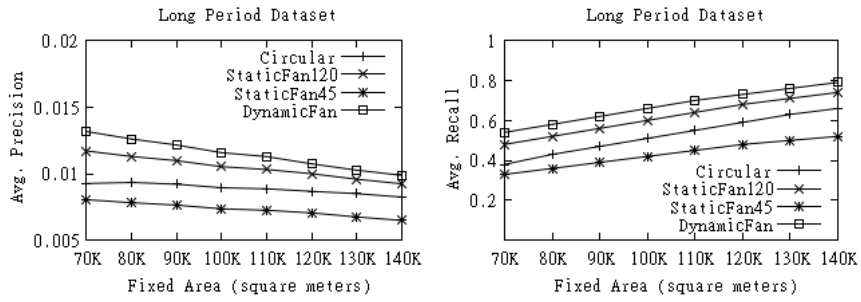
(a) Avg. response time of different shapes. (b) Avg. response time of different fans.

Figure 5.6: Average query response time results of both previous experiment sets.

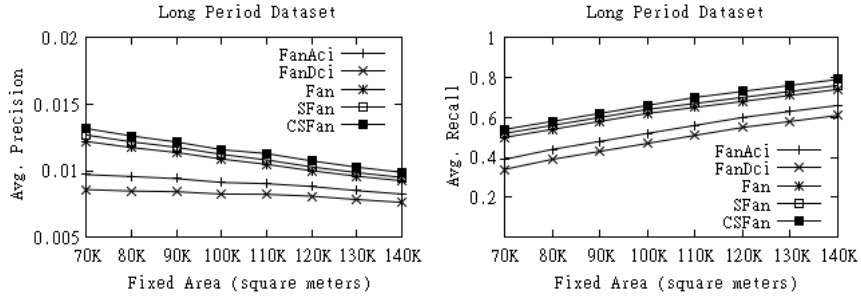
time increases as well, which demonstrates the fact that the more road segments that are covered, the more calculation cost is required. As we briefly mentioned earlier, we set $80K m^2$ as the maximum range size, which takes around 4.5 seconds on average. Any larger size, e.g., $90K m^2$, would risk the situation that query results could not be successfully delivered to the user within 5 seconds.

5.6 Long Query Period Study

The trajectory data of all previous experiments are recorded every 5 seconds. In this experiment, we would like to examine the performance of the *CQ-DFS* framework when the data period is longer. We simply retain the odd points and omit the even points of each trajectory that generates a dataset with 10 seconds per period. The performance of the experiment is reported in Figure 5.7. We apply the query range size between $70K$ and $140K m^2$ where $140K m^2$ query range size usually covers around 80 road segments, which takes around 9.5 seconds of response time. Any larger size would be too much for the dataset with 10 seconds per period. From Figure 5.7, we observe that in Figures 5.7(a) and 5.7(b), *DynamicFan* performs the best, followed by *StaticFan120*, *Circular*, and *StaticFan45* one after another. In Figures 5.7(c) and 5.7(d), *CSFan*, *SFan*, and *Fan* perform better than *FanA_{ci}* and *FanD_{ci}*. All average recalls of each shape could only achieve at most 0.8 even when the query size reaches $140K m^2$, which reveals that the current long period dataset is much more complicated than the original one. This is acceptable because user’s location reporting becomes more scattered,



(a) Avg. precision of different shapes. (b) Avg. recall of different shapes.



(c) Avg. precision of different fans. (d) Avg. recall of different fans.

Figure 5.7: Experimental results of long period dataset.

which implies that moving behaviors would be more difficult to predict. This experimental result also presents us with the suggestion that the *CQ-DFS* framework could be applied to different periods of trajectories by trying different query range sizes.

Chapter 6

Conclusion

In this paper, we study the real-time traffic status estimation service provided by the *CarWeb* system which tries to return the traffic information of a set of road segments that the user is very likely to pass by next. We would like the estimation service to be received by the users before their next location updating, and meanwhile we would like the estimation service to be accurate such that it *does* cover the road segment that the users actually pass by next. In order to fulfill these two requirements, we propose the Continuous Query with Dynamic Fan-Shape framework (*CQ-DFS*). It implements the traffic estimation service as a continuous range query with each query instance formed dynamically based on the users' last reporting locations and their moving behaviors. A comprehensive set of experiments has been conducted via using real vehicle trajectories to demonstrate the superior performance of *CQ-DFS*.

Bibliography

- [1] Charu C. Aggarwal and Dakshi Agrawal. On nearest neighbor indexing of nonlinear trajectories. In *Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 252–259, 2003.
- [2] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The r*-tree: an efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, pages 322–331, 1990.
- [3] Thomas Brinkhoff, Hans-Peter Kriegel, and Bernhard Seeger. Efficient processing of spatial joins using r-trees. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 237–246, 1993.
- [4] Ying Cai, Kien A. Hua, and Guohong Cao. Processing range-monitoring queries on heterogeneous mobile objects. In *Proceedings of the The Fifth International Conference on Mobile Data Management*, pages 27–38, 2004.
- [5] Muhammad Aamir Cheema, Ljiljana Brankovic, Xuemin Lin, Wenjie Zhang, and Wei Wang. Continuous monitoring of distance-based range queries. *IEEE Trans. on Knowl. and Data Eng.*, 23:1182–1199, August 2011.
- [6] Reynold Cheng, Dmitri V. Kalashnikov, and Sunil Prabhakar. Evaluating probabilistic queries over imprecise data. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 551–562, 2003.
- [7] Reynold Cheng, Yuni Xia, Sunil Prabhakar, Rahul Shah, and Jeffrey Scott Vitter. Efficient indexing methods for probabilistic threshold queries over uncertain data. In *Pro-*

ceedings of the 30th International Conference on Very Large Data Bases - Volume 30, pages 876–887, 2004.

- [8] King Lum Cheung and Ada Wai-Chee Fu. Enhanced nearest neighbour search on the r-tree. *SIGMOD Rec.*, 27:16–21, September 1998.
- [9] Hyung-Ju Cho and Chin-Wan Chung. An efficient and scalable approach to cnn queries in a road network. In *Proceedings of the 31st International Conference on Very Large Data Bases*, pages 865–876, 2005.
- [10] Bruce S. E. Chung, Wang-Chien Lee, and Arbee L. P. Chen. Processing probabilistic spatio-temporal range queries over moving objects with uncertainty. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, pages 60–71, 2009.
- [11] Antonio Corral, Yannis Manolopoulos, Yannis Theodoridis, and Michael Vassilakopoulos. Closest pair queries in spatial databases. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 189–200, 2000.
- [12] Antonio Corral, Yannis Manolopoulos, Yannis Theodoridis, and Michael Vassilakopoulos. Algorithms for processing k-closest-pair queries in spatial databases. *Data Knowl. Eng.*, 49:67–104, April 2004.
- [13] Victor Teixeira de Almeida and Ralf Hartmut Gutting. Supporting uncertainty in moving objects in network databases. In *Proceedings of the 13th Annual ACM International Workshop on Geographic Information Systems*, pages 31–40, 2005.
- [14] J. Feng and T. Watanabe. A fast method for continuous nearest target objects query on road network. In *Proceedings of the Eighth International Conference on Virtual Systems and Multimedia*, pages 182–191, 2002.
- [15] Michael Freeston. The bang file: A new kind of grid file. In *Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data*, pages 260–269, 1987.
- [16] Bugra Gedik and Ling Liu. Mobieyes: Distributed processing of continuously moving queries on moving objects in a mobile system. In *Proceedings of the Ninth International*

Conference on Extending Database Technology: Advances in Database Technology, pages 67–87, 2004.

- [17] Bugra Gedik, Kun-Lung Wu, Philip Yu, and Ling Liu. Motion adaptive indexing for moving continual queries over moving objects. In *Proceedings of the 13th ACM International Conference on Information and Knowledge Management*, pages 427–436, 2004.
- [18] Antonin Guttman. R-trees: a dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, pages 47–57, 1984.
- [19] A. Henrich, H.-W. Six, and P. Widmayer. The lsd-tree: Spatial access to multidimensional point and non-point objects. In *Proceedings of the 15th International Conference on Very Large Data Bases*, pages 45–53, 1989.
- [20] Gisli R. Hjaltason and Hanan Samet. Incremental distance join algorithms for spatial databases. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 237–248, 1998.
- [21] Gisli R. Hjaltason and Hanan Samet. Distance browsing in spatial databases. *ACM Trans. Database Syst.*, 24:265–318, June 1999.
- [22] Yoshiharu Ishikawa, Yuichi Tsukamoto, and Hiroyuki Kitagawa. Extracting mobility statistics from indexed spatio-temporal datasets. In *Proceedings of the 2004 International Workshop on Spatio-Temporal Database Management*, pages 9–16, 2004.
- [23] Glenn S. Iwerks, Hanan Samet, and Ken Smith. Continuous k-nearest neighbor queries for continuously moving points with updates. In *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, pages 512–523, 2003.
- [24] Christian S. Jensen, Dan Lin, and Beng Chin Ooi. Query and update efficient b+-tree based indexing of moving objects. In *Proceedings of the 30th International Conference on Very Large Data Bases - Volume 30*, pages 768–779, 2004.
- [25] Hoyoung Jeung, Qing Liu, Heng Tao Shen, and Xiaofang Zhou. A hybrid prediction model for moving objects. In *Proceedings of the 24th International Conference on Data Engineering*, pages 70–79, 2008.

- [26] Mohammad Kolahdouzan and Cyrus Shahabi. Voronoi-based k nearest neighbor search for spatial network databases. In *Proceedings of the 30th International Conference on Very Large Data Bases - Volume 30*, pages 840–851, 2004.
- [27] Chia-Hao Lo, Wen-Chih Peng, Chien-Wen Chen, Ting-Yu Lin, and Chun-Shuo Lin. Carweb: A traffic data collection platform. In *Proceedings of the Ninth International Conference on Mobile Data Management*, pages 221–222, 2008.
- [28] Nikos Mamoulis and Dimitris Papadias. Multiway spatial joins. *ACM Trans. Database Syst.*, 26:424–475, December 2001.
- [29] Mohamed F. Mokbel, Xiaopeing Xiong, and Walid G. Aref. Sina: Scalable incremental processing of continuous queries in spatio-temporal databases. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, pages 623–634, 2004.
- [30] Kyriakos Mouratidis, Dimitris Papadias, and Marios Hadjieleftheriou. Conceptual partitioning: An efficient method for continuous nearest neighbor monitoring. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pages 634–645, 2005.
- [31] J. Nievergelt, Hans Hinterberger, and Kenneth C. Sevcik. The grid file: An adaptable, symmetric multikey file structure. *ACM Trans. Database Syst.*, 9:38–71, March 1984.
- [32] Bernd-Uwe Pagel, Hans-Werner Six, Heinrich Toben, and Peter Widmayer. Towards an analysis of range query performance in spatial data structures. In *Proceedings of the 12th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 214–221, 1993.
- [33] Dimitris Papadias and Dinos Arkoumanis. Approximate processing of multiway spatial joins in very large databases. In *Proceedings of the Eighth International Conference on Extending Database Technology: Advances in Database Technology*, pages 179–196, 2002.
- [34] Dimitris Papadias, Nikos Mamoulis, and Yannis Theodoridis. Processing and optimization of multiway spatial joins using r-trees. In *Proceedings of the 18th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 44–55, 1999.

- [35] Jignesh M. Patel, Yun Chen, and V. Prasad Chakka. Stripes: An efficient index for predicted trajectories. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, pages 635–646, 2004.
- [36] Dieter Pfoser and Christian S. Jensen. Capturing the uncertainty of moving-object representations. In *Proceedings of the Sixth International Symposium on Advances in Spatial Databases*, pages 111–132, 1999.
- [37] Nick Roussopoulos, Stephen Kelley, and Frederic Vincent. Nearest neighbor queries. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 71–79, 1995.
- [38] Simonas Saltenis, Christian S. Jensen, Scott T. Leutenegger, and Mario A. Lopez. Indexing the positions of continuously moving objects. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 331–342, 2000.
- [39] Zhexuan Song and Nick Roussopoulos. K-nearest neighbor search for moving query point. In *Proceedings of the Seventh International Symposium on Advances in Spatial and Temporal Databases*, pages 79–96, 2001.
- [40] Dragan Stojanovic, Apostolos N. Papadopoulos, Bratislav Predic, Slobodanka Djordjevic-Kajan, and Alexandros Nanopoulos. Continuous range monitoring of mobile objects in road networks. *Data Knowl. Eng.*, 64:77–100, January 2008.
- [41] Yufei Tao, Reynold Cheng, Xiaokui Xiao, Wang Kay Ngai, Ben Kao, and Sunil Prabhakar. Indexing multi-dimensional uncertain data with arbitrary probability density functions. In *Proceedings of the 31st International Conference on Very Large Data Bases*, pages 922–933, 2005.
- [42] Yufei Tao, Christos Faloutsos, Dimitris Papadias, and Bin Liu. Prediction and indexing of moving objects with unknown motion patterns. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, pages 611–622, 2004.
- [43] Yufei Tao, George Kollios, Jeffrey Considine, Feifei Li, and Dimitris Papadias. Spatio-temporal aggregation using sketches. In *Proceedings of the 20th International Conference on Data Engineering*, pages 214–226, 2004.

- [44] Yufei Tao and Dimitris Papadias. Time parameterized queries in spatio-temporal databases. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pages 334–345, 2002.
- [45] Yufei Tao, Dimitris Papadias, and Qiongmao Shen. Continuous nearest neighbor search. In *Proceedings of the 28th International Conference on Very Large Data Bases*, pages 79–96, 2002.
- [46] Yufei Tao, Dimitris Papadias, and Jimeng Sun. The tpr*-tree: An optimized spatio-temporal access method for predictive queries. In *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, pages 790–801, 2003.
- [47] Goce Trajcevski, Ouri Wolfson, Klaus Hinrichs, and Sam Chamberlain. Managing uncertainty in moving objects databases. *ACM Trans. Database Syst.*, 29:463–507, September 2004.
- [48] Florian Verhein and Sanjay Chawla. Mining spatio-temporal association rules, sources, sinks, stationary regions and thoroughfares in object mobility databases. In *Proceedings of the 11th International Conference on Database Systems for Advanced Applications*, pages 187–201, 2006.
- [49] Ling-Yin Wei, Wen-Chih Peng, Chun-Shuo Lin, and Chen-Hen Jung. Exploring spatio-temporal features for traffic estimation on road networks. In *Proceedings of the 11th International Symposium on Advances in Spatial and Temporal Databases*, pages 399–404, 2009.
- [50] Ouri Wolfson, Sam Chamberlain, Son Dao, Liqin Jiang, and Gisela Mendez. Cost and imprecision in modeling the position of moving objects. In *Proceedings of the 14th International Conference on Data Engineering*, pages 588–596, 1998.
- [51] Ouri Wolfson, Bo Xu, Sam Chamberlain, and Liqin Jiang. Moving objects databases: Issues and solutions. In *Proceedings of the Tenth International Conference on Scientific and Statistical Database Management*, pages 111–122, 1998.

- [52] Xiaopeng Xiong, Mohamed F. Mokbel, and Walid G. Aref. Sea-cnn: Scalable processing of continuous k-nearest neighbor queries in spatio-temporal databases. In *Proceedings of the 21st International Conference on Data Engineering*, pages 643–654, 2005.
- [53] Xiaopeng Xiong, Mohamed F. Mokbel, Walid G. Aref, Susanne E. Hambrusch, and Sunil Prabhakar. Scalable spatio-temporal continuous query processing for location-aware services. In *Proceedings of the 16th International Conference on Scientific and Statistical Database Management*, pages 317–326, 2004.
- [54] Kefeng Xuan, Geng Zhao, David Taniar, and Bala Srinivasan. Continuous range search query processing in mobile navigation. In *Proceedings of the 14th IEEE International Conference on Parallel and Distributed Systems*, pages 361–368, 2008.
- [55] Gokhan Yavas, Dimitrios Katsaros, Ozgur Ulusoy, and Yannis Manolopoulos. A data mining approach for location prediction in mobile environments. *Data Knowl. Eng.*, 54:121–146, August 2005.
- [56] Kam yiu Lam, Ozgur Ulusoy, Tony S. H. Lee, Edward Chan, and Guohui Li. An efficient method for generating location updates for processing of location-dependent continuous queries. In *Proceedings of the Seventh International Conference on Database Systems for Advanced Applications*, pages 218–225, 2001.
- [57] Xiaohui Yu, Ken Q. Pu, and Nick Koudas. Monitoring k-nearest neighbor queries over moving objects. In *Proceedings of the 21st International Conference on Data Engineering*, pages 631–642, 2005.
- [58] Duncan Yung, Man Lung Yiu, and Eric Lo. A safe-exit approach for efficient network-based moving range queries. *Data Knowl. Eng.*, 72:126–147, February 2012.
- [59] Jun Zhang, Manli Zhu, Dimitris Papadias, Yufei Tao, and Dik Lun Lee. Location-based spatial queries. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 443–454, 2003.