D

Dynamic Nearest Neighbor Queries in Euclidean Space

Sarana Nutanong[1], Mohammed Eunus Ali[2], Egemen Tanin[3], and Kyriakos Mouratidis[4]
[1]City University of Hong Kong, Hong Kong, China
[2]Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh
[3]University of Melbourne, Melbourne, VIC, Australia
[4]Singapore Management University, Singapore, Singapore

Synonyms

Nearest neighbor monitoring; Temporal nearest neighbor query

Definition

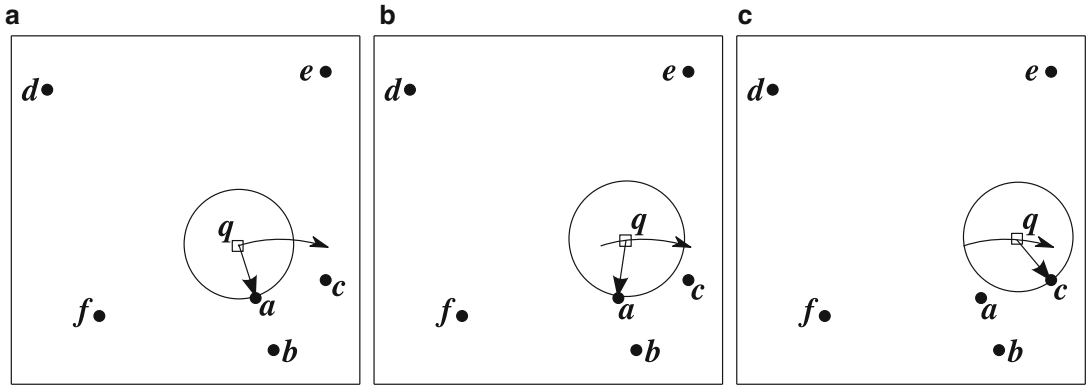
Given a query point **q** and a set $\mathcal{D}$ of data points, a nearest neighbor (NN) query returns the data point **p** in $\mathcal{D}$ that minimizes the distance $\text{DIST}(\boldsymbol{q},\boldsymbol{p})$, where the distance function DIST(,) is the L_2 norm. One important variant of this query type is *kNN query*, which returns k data points with the minimum distances. When taking the temporal dimension into account, the kNN query result may change over a period of time due to changes in locations of the query point and/or data points. Formally, the k-nearest neighbor (kNN) query is defined as follows.

Definition 1 (k-Nearest Neighbor (kNN) Query) Given a set $\mathcal{D}$ of data objects and a query point **q**, the kNN query finds a set $\mathcal{R}$ of objects such that: (i) $\mathcal{R}$ contains k objects from $\mathcal{D}$. (ii) for any object $\mathbf{x} \in \mathcal{R}$ and object $\mathbf{y} \in (\mathcal{D} - \mathcal{R})$, $\text{DIST}(\boldsymbol{q}, \boldsymbol{x}) \leq \text{DIST}(\boldsymbol{q},\boldsymbol{y})$.

A *dynamic kNN query in Euclidean space* returns kNN query results over a period of time in a dynamically changing environment.

Figure 1 provides an example of a dynamic 1NN query with a moving query point and a static dataset. The example shows that the query point **q** moves from left to right in three successive snapshots t_1, t_2, and t_3, where **a** is the nearest neighbor at times t_1 and t_2, and the result is updated to **c** at t_3. A straightforward approach to processing a dynamic kNN query is to issue multiple kNN queries repetitively. However, the result accuracy of this approach highly depends on the query frequency, and a higher query frequency incurs a greater query processing cost. In this example, if we assume that a kNN query is issued at t_1, at t_2, and at t_3, then there is a time period between t_2 and t_3 in which the result is obsolete.

S. Shekhar et al. (eds.), *Encyclopedia of GIS*,
DOI 10.1007/978-3-319-23519-6_1558-1



Dynamic Nearest Neighbor Queries in Euclidean Space, Fig. 1 Continuous k-nearest neighbor query with a moving query point **q** and a static dataset **{a, b, c, d, e, f}** in Euclidean space, where for $k = 1$, **a** is the nearest neighbor at times t_1 and t_2, and **b** is the nearest neighbor at time t_3. (**a**) Time t_1: NN(**q**) = **a**. (**b**) Time t_2: NN(**q**) = **a**. (**c**) Time t_3: NN(**q**) = **c**

Historical Background

The study of continuous kNN queries is generally concerned with deriving query processing techniques to reduce the query processing cost without sacrificing the result accuracy. Since the early 2000s, considerable research attention has been given techniques to process variants of continuous kNN queries for moving query points and moving data objects.

Scientific Fundamentals

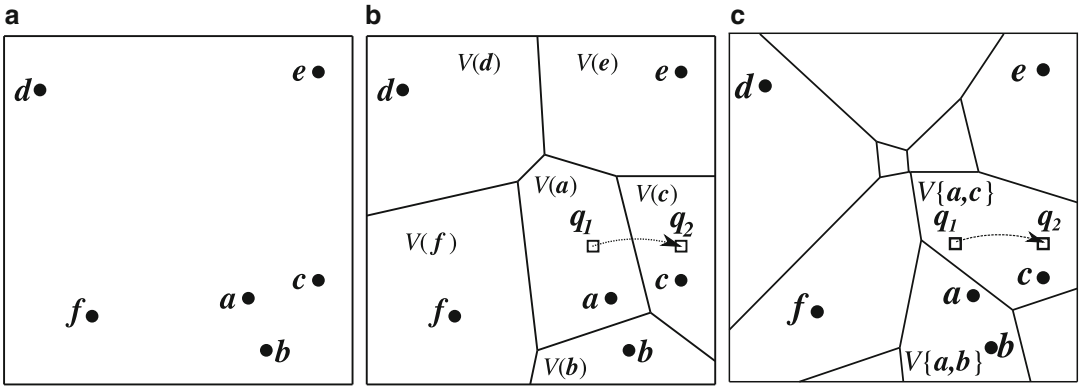
Continuous kNN processing relies on the same fundamental concept as other continuous query types, i.e., figuring out the conditions that may invalidate the current result set. For a continuous kNN query, the result set can be invalidated by having an object **y** that is not included in the current result set $\mathcal{R}$ coming closer to the query point **q** than an object in $\mathcal{R}$. The manner in which **y** can come closer to the query point than an object in $\mathcal{R}$ depends on whether the query point and/or the data objects are assumed to be moving.

Moving Query over Static Data Objects

This variant is also known as the *moving kNN* (MkNN) query. In this case, changes in the result set are caused by the query point **q** moving closer to an object **y** that is not in the result set $\mathcal{R}$ than any object **x** in $\mathcal{R}$. One popular processing approach for this variant is identifying all possible pairs of **x** and **y** in order to compute the boundaries $B_{x,y}$, where $B_{x,y}$ is formally defined as a set of points **p** such that $\text{DIST}(\boldsymbol{p}, \boldsymbol{x})$ is equal to $\text{DIST}(\boldsymbol{p}, \boldsymbol{y})$. The area inside these boundaries is also known as a *safe region*. As long as the movement of **q** is confined within the safe region, $\mathcal{R}$ remains valid.

Precomputing safe regions. A classic example of safe region-based techniques is the *Voronoi diagram* (Aurenhammer 1991; Okabe et al. 1992). The Voronoi diagram is a well-known space decomposition technique determined by distances to a given discrete set of objects, typically a set of points. Figure 2b shows a Voronoi diagram of six data objects $\{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}, \mathbf{f}\}$. Assume that the query point is originally at $\mathbf{q_1}$, the safe region is the area confined by five boundaries, $B_{a,b}$, $B_{a,c}$, $B_{a,d}$, $B_{a,e}$, and $B_{a,f}$. As long as the query point does not cross any of these boundaries, **a** remains the first NN. As exemplified in Fig. 2c, the Voronoi diagram can be generalized to the *kth-order Voronoi diagram (kVD)*, which can be used to help process kNN queries for any given location in the data space.

Processing an MkNN query using a kVD can be done by identifying the Voronoi cell in which the query point **q** is currently residing and monitoring the location of **q** constantly. The result set



Dynamic Nearest Neighbor Queries in Euclidean Space, Fig. 2 The Voronoi diagram and its generalizations. (**a**) Point set $\mathcal{D}$, {**a**, **b**, **c**, **d**, **e**, **f**}. (**b**) Voronoi diagram of $\mathcal{D}$. (**c**) Second-order Voronoi diagram of $\mathcal{D}$

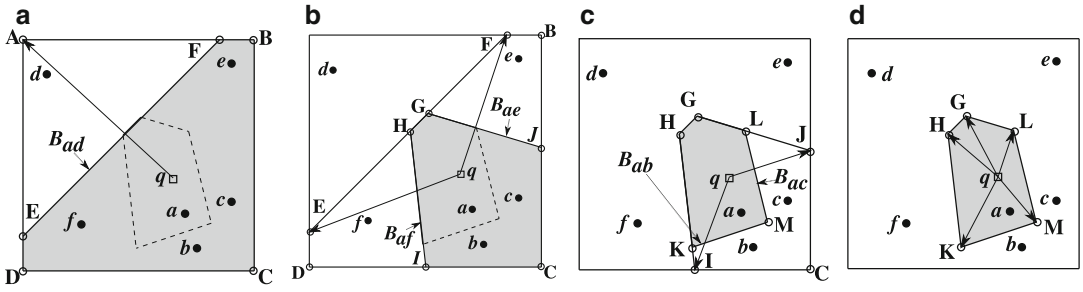
is updated only when **q** crosses a boundary. The main benefit of this approach is the query processing costs which are logarithmic with respect to the number of data objects for the initial lookup and constant for safe region checking (Aurenhammer 1991; Okabe et al. 1992).

The main drawback of using a kVD to process moving kNN queries is that the technique requires an evaluation of Voronoi cells of the entire dataset. This is considered undesirable especially when the movement of the query point **q** is confined in a small area with respect to the entire data space. Furthermore, one may require different kVDs for different needs. For example, a driver may need to find a gas station with a restroom facility, while another driver needs one with a special type of fuel. Precomputing kVDs for all possible scenarios is impractical.

Predefined query trajectories. When the trajectory is known in advance, the MkNN query processing problem can be simplified to identification of the point along the trajectory at which the kNN result set changes, i.e., where the trajectory intersects a safe region boundary. Tao and Papadias (2002) proposed the *time-parameterized kNN (TPkNN) query*. Assuming a linear trajectory of the query point, a TPkNN query finds (i) the current kNN set, (ii) a position on the trajectory where the kNN result set changes (the *influence point*), and (iii) the object that causes the change (the *influence object*). Finding the influence object is done by ranking candidate influence objects according to how early their corresponding influence points appear on the trajectory.

Another well-known method to handle an MkNN query with a predefined query trajectory is to use the *continuous kNN* (CkNN) query (Tao et al. 2002). CkNN query splits the query trajectory into segments where each segment corresponds to a particular kNN result set. This is done by identifying the influence points along the query trajectory. The main difference between CkNN and TPkNN is that CkNN obtains all kNN result sets along a given trajectory, but TPkNN provides only the segment corresponding to the current kNN result set.

Unknown query trajectories. One method to handle an MkNN query for an unknown trajectory is to locally construct the Voronoi cell that currently contains the query point. Since only one Voronoi cell is needed at a time, this method does not suffer from the same drawback as the Voronoi diagram method. Zhang et al. (2003) proposed a method that executes multiple instances of TP-kNN queries to discover all possible influence objects and cell boundaries around the query point (as illustrated in Fig. 3). Due to the convexity property of Voronoi cells generated from a set of data points, it is guaranteed that all boundaries are discovered when all safe region corners share the same kNN result set as the query point.



Dynamic Nearest Neighbor Queries in Euclidean Space, Fig. 3 Locally compute a kVD cell ($k = 1$). (**a**) Step 1. (**b**) Steps 2 and 3. (**c**) Steps 4 and 5. (**d**) Final steps

Result caching. Based on the principle of spatial locality of references, we can cache data objects around the query point and attempt to reproduce query results using the cached data objects as the query point moves (Nutanong et al. 2010; Song and Roussopoulos 2001). Once the cached data can no longer produce accurate query results, the cache is updated. One method to utilize cached data for processing an MkNN query is to apply the sampling-based approach (Song and Roussopoulos 2001), i.e., periodically ranking the cached data objects according to the distance from the query point. The cache is updated when the correctness of the kNN result set cannot be guaranteed.

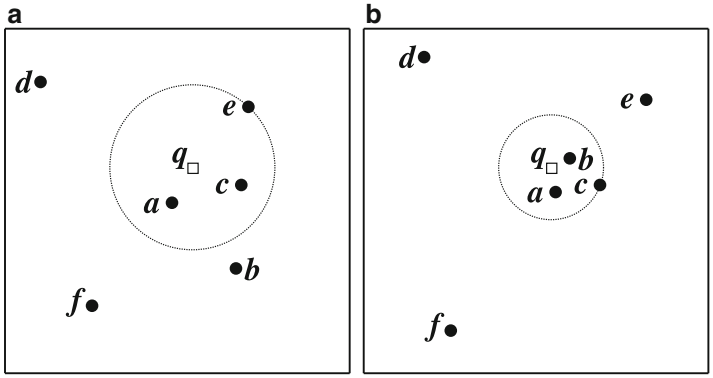
When truly continuous query results are required, cached objects can also be used to build a safe region. Nutanong et al. (2010) proposed a method which incrementally maintains two types of safe regions: (i) one that keeps the rank of cached data object constant and (ii) one that ensures the validity of the kNN obtained from the cache. Li et al. (2014) propose an incremental method to compute and to update a set of data objects that may invalidate the current kNN result set. The kNN result set is guaranteed to be valid as long as the query point is closer to kNN than any of the invalidating object. This method can also be regarded as a safe region-based method in the sense that we can compute the boundary between each pair of the invalidating object and its corresponding object in the kNN set. The region that is enclosed by these boundaries is a safe region.

The main difference between the sampling-based method (Song and Roussopoulos 2001) and the safe region-based method (Li et al. 2014; Nutanong et al. 2010) can be described as follows. The sampling-based method (Song and Roussopoulos 2001) does not produce truly continuous query answers; its main objective is to reduce the cost of each kNN query execution in order to accommodate a higher query frequency. On the other hand, the safe region-based method (Li et al. 2014; Nutanong et al. 2010) can produce continuous query results by keeping track of locations at which the query point crosses a safe region boundary.

Static Query over Moving Data Objects

When the query point is static and data objects are moving, processing a continuous kNN query involves keeping track of data objects moving in and out of the kNN result set. Figure 4 shows how the kNN result set changes from $\{\mathbf{a}, \mathbf{c}, \mathbf{e}\}$ to $\{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$ due to the movements of data objects in the dataset.

Predefined object trajectories. When object trajectories are known in advance, processing a continuous kNN query involves identifying object trajectories that may involve in the result set and ruling out those that are guaranteed to be outside the result set with respect to a given query location and time of interest. To accommodate querying in the temporal dimension, a time-parameterized data structure (Tao et al. 2003; Saltenis and Jensen 2002) is often used to index



Dynamic Nearest Neighbor Queries in Euclidean Space, Fig. 4 Continuous k-nearest neighbor query with a static query point **q** and a moving dataset $\{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}, \mathbf{f}\}$ in Euclidean space. (**a**) Time t_1: $3\text{NN}(\mathbf{q}) = \{\mathbf{a}, \mathbf{c}, \mathbf{e}\}$. (**b**) Time t_2: $3\text{NN}(\mathbf{q}) = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$

object trajectories. In this way, the location of each data object is represented as a function of time (i.e., its initial location and its velocity vector). Each object updates its location function only when it no longer accurately describes its movement or when the location function is older than a predefined threshold.

A notable example of techniques which utilize time parameterization is the *extended time parameterization (ETP)* algorithm, proposed by Iwerks et al. (2006). The ETP algorithm provides support for moving objects by extending the TP-kNN algorithm (Tao and Papadias 2002). Iwerks et al. (2006) also formulated an approach to processing a kNN query on moving objects by continuous evaluation of a range query. Their proposed approach is based on the observation that a continuous fixed-range query is easier to process than a continuous kNN query with moving objects. By imposing a condition that the scope of the range query must include at least $k + 1$ data objects, the k nearest objects can be evaluated based on only objects within the scope. The query scope is allowed to expand and to contract according to the current density of objects around the query point.

Unknown object trajectories. Unknown object trajectories are handled similarly to the case of an unknown query trajectory. Specifically, a safe region is associated with every data object. As long as all data objects remain inside their respective safe regions, the current kNN result set is guaranteed to be up to date. Mouratidis et al. (2005b) proposed a threshold-based approach to monitoring the k nearest objects in a setting for moving objects. Each monitored object is associated with a range of safe distances from the query point. An object cannot influence the query answer as long as it remains within the range of safe distances. Hu et al. (2005) proposed a safe region-based technique for *static* window and kNN queries on moving objects. Each moving object maintains its own safe region and only reports its new location if it may affect any query result.

Moving Query over Moving Data Objects

When both query and data objects are allowed to move, processing continuous kNN queries involves monitoring the locations of query points and moving objects periodically. Mouratidis et al. (2005b) showed that the threshold-based approach to handling moving objects can be extended to support multiple moving query points. Using this method, the server keeps track of the location of each query point, while each data object keeps track of the locations and threshold of all queries. The server checks the thresholds as location updates arrive from the objects and refreshes the query results when one or more threshold violations occur.

Gedik and Liu (2004) presented a distributed solution to continuous monitoring of moving queries and moving objects that utilize the computational power of the mobile devices attached to mobile objects. The authors proposed a technique which enables trade-offs between query precision and query processing cost (in terms of network bandwidth and energy consumption). In order to handle a large number

of queries, the authors also proposed a query-grouping mechanism to reduce the computational cost on the mobile device side. This mechanism allows continuous queries in proximity to be coprocessed.

Another stream of work drops the need for safe regions, as well as any assumptions about the movement patterns of objects and queries. Objects and queries move arbitrarily and unpredictably and report their new location to a processing server whenever they move. The aim of methods in this category is for the processing server to refresh the query results as quickly as possible in order to cater for the time-critical nature of monitoring applications. The dominant approach taken indexes the queries using a regular grid and augments that index with bookkeeping information to track the *influence region* of each query, i.e., the circular disk around the query location that includes its k nearest objects. Only objects leaving/entering the influence region of a query may affect its kNN result. The main representatives of this stream of work are YPK-CNN (Yu et al. 2005), SEA-CNN (Xiong et al. 2005), and CPM (Mouratidis et al. 2005a). A survey of techniques in this category can be found in Mouratidis (2009).

Key Applications

Aviation Safety

Keeping track of nearby locations at which an aircraft can land at all times is extremely crucial to aviation safety. An aircraft pilot can use the continuous kNN query to precompute the nearest emergency landing sites along a predefined flight path.

Location-Based Advertising

Location-based advertising (LBA) is a form of advertising that uses location information to help identify potential customers. A business establishment can use continuous kNN query to monitor movements of nearby LBA participants over a period of time in order to offer promotional deals to participants who frequently appear in the kNN result set.

Location-Based Tour Guide System

A location-based tour guide system provides related tourist information based on a user's location. The system can make use of the continuous kNN query type to continuously report a list of nearby tourist attractions. A user can browse the list and select to retrieve information about the attraction in which they are most interested.

Multiplayer Online Gaming

In a multiplayer online gaming environment where different groups of players compete against each other, complete awareness of the surroundings is very important to each player. The continuous kNN query type can be used to report nearby threats to each player at all times.

Future Directions

- **Privacy Issues.** In location-based services, users may choose to obfuscate their locations before submitting them to a service provider for greater privacy (Duckham and Kulik 2005; Gruteser and Grunwald 2003). Continuous queries require users to repetitively share their locations with the service provider. This may provide the opportunity for the service provider to infer the trajectory of a user from a set of altered locations that they share by applying physical constraints such as speed, road network topology, etc. (Chow and Mokbel 2007).
- **Probabilistic Queries.** One research direction is to capture the uncertain nature of object/query trajectories (Cheng et al. 2004; Niedermayer et al. 2013). Probabilistic continuous querying is concerned with presenting a number of possible results along with probability assessments to a user over a period of time.
- **Continuous kNN with Spatial Constraints.** Another important research direction is concerned with incorporating spatial constraints into problem modeling and query processing. For example, in the presence of obstacles, one may be interested in monitoring k nearest objects that are visible from the query (Gao

et al. 2011). In the context of objects and queries that move along the roads of a city, one would want to monitor the k nearest objects in terms of traveling distance within the road network (Mouratidis et al. 2006).

Cross-References

- Nearest Neighbor Query
- Queries in Spatio-temporal Databases, Time Parameterized

References

Aurenhammer F (1991) Voronoi diagrams – a survey of a fundamental geometric data structure. ACM Comput Surv 23(3):345–405

Cheng R, Kalashnikov DV, Prabhakar S (2004) Querying imprecise data in moving object environments. IEEE Trans Knowl Data Eng 16(9):1112–1127

Chow C, Mokbel MF (2007) Enabling private continuous queries for revealed user locations. In: SSTD, Boston, MA, USA, pp 258–275

Duckham M, Kulik L (2005) A formal model of obfuscation and negotiation for location privacy. In: Pervasive computing, third international conference, PERVASIVE 2005, Munich, 8–13 May 2005, Proceedings, pp 152–170

Gao Y, Zheng B, Chen G, Chen C, Li Q (2011) Continuous nearest-neighbor search in the presence of obstacles. ACM Trans Database Syst 36(2):9

Gedik B, Liu L (2004) Mobieyes: distributed processing of continuously moving queries on moving objects in a mobile system. In: EDBT, Heraklion, Crete, Greece, pp 67–87

Gruteser M, Grunwald D (2003) Anonymous usage of location-based services through spatial and temporal cloaking. In: Proceedings of the first international conference on mobile systems, applications, and services, MobiSys 2003, San Francisco, 5–8 May 2003

Hu H, Xu J, Lee DL (2005) A generic framework for monitoring continuous spatial queries over moving objects. In: SIGMOD, Baltimore, Maryland, USA, pp 479–490

Iwerks GS, Samet H, Smith KP (2006) Maintenance of k-nn and spatial join queries on continuously moving points. ACM Trans Database Syst 31(2):485–536

Li C, Gu Y, Qi J, Yu G, Zhang R, Yi W (2014) Processing moving knn queries using influential neighbor sets. PVLDB 8(2):113–124

Mouratidis K (2009) Continuous monitoring of spatial queries. In: Encyclopedia of database systems. Springer, New York, pp 479–484

Mouratidis K, Hadjieleftheriou M, Papadias D (2005a) Conceptual partitioning: an efficient method for continuous nearest neighbor monitoring. In: SIGMOD, Baltimore, Maryland, USA, pp 634–645

Mouratidis K, Papadias D, Bakiras S, Tao Y (2005b) A threshold-based algorithm for continuous monitoring of k nearest neighbors. IEEE Trans Knowl Data Eng 17(11):1451–1464

Mouratidis K, Yiu ML, Papadias D, Mamoulis N (2006) Continuous nearest neighbor monitoring in road networks. In: VLDB, Seoul, Korea, pp 43–54

Niedermayer J, Züfle A, Emrich T, Renz M, Mamoulis N, Chen L, Kriegel H (2013) Probabilistic nearest neighbor queries on uncertain moving object trajectories. PVLDB 7(3):205–216

Nutanong S, Zhang R, Tanin E, Kulik L (2010) Analysis and evaluation of v*-$k$nn: an efficient algorithm for moving $k$nn queries. VLDB J 19(3):307–332

Okabe A, Boots B, Sugihara K (1992) Spatial tessellations: concepts and applications of Voronoi diagrams. Wiley, New York

Saltenis S, Jensen CS (2002) Indexing of moving objects for location-based services. In: ICDE, San Jose, California, USA, pp 463–472

Song Z, Roussopoulos N (2001) K-nearest neighbor search for moving query point. In: SSTD, Redondo Beach, CA, USA, pp 79–96

Tao Y, Papadias D (2002) Time-parameterized queries in spatio-temporal databases. In SIGMOD, Madison, Wisconsin, USA, pp 334–345

Tao Y, Papadias D, Shen Q (2002) Continuous nearest neighbor search. In: VLDB, Hong Kong, China, pp 287–298

Tao Y, Papadias D, Sun J (2003) The tpr*-tree: an optimized spatio-temporal access method for predictive queries. In: VLDB, Berlin, Germany, pp 790–801

Xiong X, Mokbel MF, Aref WG (2005) SEA-CNN: scalable processing of continuous k-nearest neighbor queries in spatio-temporal databases. In: ICDE, Tokyo, Japan, pp 643–654

Yu X, Pu KQ, Koudas N (2005) Monitoring k-nearest neighbor queries over moving objects. In: ICDE, Tokyo, Japan, pp 631–642

Zhang J, Zhu M, Papadias D, Tao Y, Lee DL (2003) Location-based spatial queries. In: SIGMOD, San Diego, California, USA, pp 443–454