Continuous Perspective Query Processing for 3-D Objects on Road Networks

Joon-Seok Kim, Kyoung-Sook Kim, and Ki-Joune Li†

Department of Computer Science and Engineering
Pusan National University, Pusan 609-735, South Korea
{joonseok,ksookim,lik}@pnu.edu

Abstract. In order to provide a streaming service of 3-D spatial objects to the mobile clients on a street, we propose a new query type, called continuous perspective query. The perspective query differs from conventional spatial queries in that the levels of details (LOD) of query results depend on the distance between the query point and spatial objects. The objects in the vicinity are to be provided with a higher LOD than those far from the query point. We are dealing with continuous queries, and the LODs of the results are also changing according to the distance from the mobile query point. The LOD of the result for an object changes from a low LOD to a high LOD as the mobile query point approaches to the object. In this paper, we also propose a processing method for continuous perspective query to reduce the processing cost at the server and communication cost between the server and the mobile clients.

Key words: continuous perspective query, road network, LOD, 3-D objects

1 Introduction

Most location-based services provide mobile users with the information of spatial objects such as buildings and roads in 2-D or 2.5-D. However we may achieve a higher quality of service when the 3-D information is offered such as 3-D geometry, and colors and texture on the facets of the objects. This information enables more realistic visualization of objects than simple 2-D data.

One of the main problems in providing 3-D information to mobile clients comes from the large size of data. It is almost impossible to store a large volume of 3-D data in a mobile device such as PDA and mobile phone with a relatively small size of memory. Furthermore, the transmission of 3-D data is limited due to the narrow bandwidth of wireless communication.

The basic idea of this paper is based on the observation that we do not need a detail information on an object, which is far from the query point but only its simplified sketch. However, more detail information is progressively required as the query point becomes close to the object. In this paper, we propose a new type of query, called continuous perspective query for this purpose. This query is a kind of continuous query, where the levels of details (LOD) of query results

depend on the distance between the query point and spatial objects. The objects closely located to the query point are to be provided with a higher LOD than those far from the query point.

Since we are dealing with continuous queries, the LODs of the results are also changing according to the distance from the mobile query point. The LOD of the results should be upgraded as the mobile query point approaches to the object. And an efficient protocol of upgrades reduces the communication cost and processing cost at the server. For this reason, a method is proposed to process the continuous perspective query and upgrade LOD in this paper.

The remainder of paper is organized as follows. In section 2, we review related work in perspective query and continuous query. We describe the basic concept and define continuous perspective query in the next section. In section 4, we propose a processing method for the query for the mobile clients on the streets. An analysis of performance is presented in section 5 and we conclude the paper in section 6

2 Related work and motivation

A perspective query is to request data, which have different levels of detail according to the importance of data. In GIS and LBS, this type of query is used to optimize the processing of geo-visualization. We may simplify a certain level of geometries of objects that are relatively far from the viewpoint to reduce the size of data. For this type of query, several researches have been done since last few years. For example, a processing method was proposed for perspective query by selecting proper LOD(Level of Detail) based on the range from the viewpoint in [3]. By this method, the entire space is divided into three ranges according to the distance from the viewpoint and a proper LOD is assigned to each range from high LOD to low LOD.

LOD-R-trees was proposed to index spatial objects with LOD in [1], which is a new data structure combined with R-trees and LODs. It stores the graphic data of spatial objects at each node and returns properly simplified LOD according to the distance. Similarly, V-Reactive tree was proposed in [2], which is another variant of R-trees for multiple LODs. It can be used for implementing the generalization of multi-LODs technique.

However the mobility of viewpoint has not been considered by these work. In most applications of perspective query such as LBS and navigation services, the spatial query condition continuously changes for a given period of time. During this time period, the queries are to be repeatedly evaluated for providing the correct information as changes the spatial query condition. This query is called continuous query [4]. Recently, a lot of attention has been paid to continuous range query and continuous k-nearest neighbor query. They are summarized by table 1.

Among these work, a dynamic query type was defined as a temporally series of snapshot queries in [5], where the authors proposed an index data structure for the trajectories of moving objects and described how to evaluate dynamic

types / mobility of (query, object)	Range	K-NN
(static query, dynamic objects)	[6–8, 10]	[11, 15]
(dynamic query, static objects)	[5, 6, 10]	[11–14]
(dynamic query, dynamic objects)	[5, 6, 10]	[11]

Table 1. classification of movement patterns and types of continuous queries

queries efficiently which represent predictable or non-predictable movement of an observer. D. Stojanovic et al. proposed a method for continuous range query processing in [6], characterized by the mobility of objects and queries that follow paths in an underlying spatial network. They introduced an additional pre-refinement step which generates main memory data structures to support periodical and incremental refinement steps.

In [7], a velocity constrained indexing and query indexing (Q-index) have been proposed by S. Prabhakar et. al. to evaluate continuous range queries. By indexing queries instead of objects, we reduce frequent updates of the index structure and expensive maintenance of index structure. By assuming that there is a sufficient amount of memory and computation power, Ying Cai et al. claimed in [8] that range queries relevant to an object can be sent to it to quickly determine whether an update be needed.

A scheme called Motion Adaptive Indexing (MAI) was proposed by B. Gedick in [9]. This enables optimization of continuous query evaluation according to the dynamic motion behavior of the objects. In this paper, the authors introduced the concept of motion sensitive bounding boxes (MSB) to model and index both moving objects and moving queries.

Mokbel et al. proposed a method based on shared execution and incremental evaluation of continuous queries, called SINA in [10]. Shared execution is performed for query evaluation as a spatial join between the mobile objects and the queries. The incremental evaluation means that the query processing system produces only positive or negative updates from the previous result.

The continuous query and perspective query have been separately considered by the previous work. However we integrate these two types of query into a new one to reduce the communication and processing cost. In most cases, the streaming services of mobile clients are required on streets for pedestrians or car drivers. For this reason, we focus on the query of mobile clients who moves on road networks instead of Euclidian space.

3 Basic concepts of continuous perspective query

In this section, we define continuous perspective query through a combination of perspective query and continuous query. Figure 1 illustrates an example of continuous perspective query. The results of perspective query change as the

viewpoint moves from $t_1, t_2, t_3, \ldots$, to t_n. For example the LOD of object b upgrades from low LOD at t_1, medium LOD at t_2, to high LOD at t_3. Hence, continuous perspective query is considered as an extended perspective query type to reflect the mobility of query point.

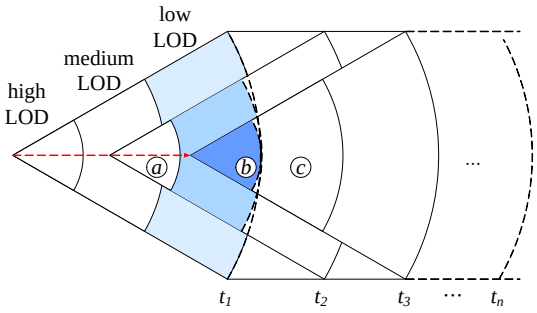


Fig. 1. Example of continuous perspective query

3.1 Perspective query

A perspective query is intended first to search spatial objects in view range, and second to determine their LOD. The query is given with three parameters, which are viewpoint, view direction, and view angle as follows,

- viewpoint: the query point in 3D space is given as $v_p = (x, y, z)$ in figure 2(a).
- view direction: the direction of the viewpoint in 3D space is given as $v_{dir} = (\varphi, \theta)$ in figure 2(a), where φ is the horizontal angle of view in the plane XY and θ is the vertical angle from the XY plane.
- view angle: the view angle is given as $v_{ang} = (\alpha, \beta)$, where α and β are the vertical and horizontal extents of view angle as figure 2(b).

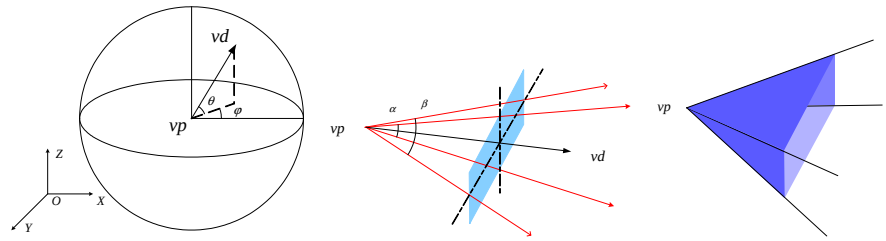
A query region, which is surrounded by four triangular bounding planes as figure 2(c), is defined as $q_{reg} = (v_p, v_{dir}, v_{ang})$. Then we give the definition of perspective query in terms of the result set of the query as follows;

Definition 1. *Perspective Query*

$$Q_p(q_{reg}) = \{(s, l) |\ s \in S, l \in L\}$$

S in this definition represents the set of spatial objects and $L = \{LOD_1, LOD_2, ..., LOD_n\}$ is the set of LODs.

Figure 3 shows an example of perspective query and its results. Given a set of six spatial objects a, b, c, d, e, f and a query region $q_{reg} = (v_p, v_{dir}, v_{ang})$,



(a) viewpoint, view direction (b) view angle (c) query region

Fig. 2. Parameters of perspective query and query region

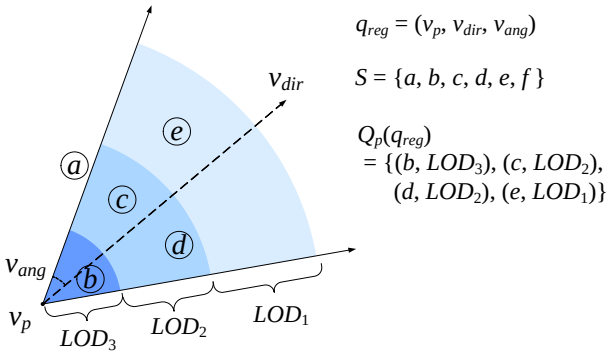
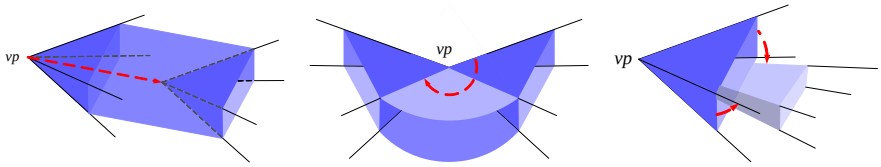


Fig. 3. Example of perspective query

then the result of the query is $\{(b, LOD_3), (c, LOD_2), (d, LOD_2), (e, LOD_3)\}$. The object a is not in the result because it is out of query region and object f does not belong to the result since the distance from the viewpoint exceeds the maximum threshold. Objects b, c and e are in the range of LOD_3, LOD_2 and LOD_1, respectively.

3.2 Continuous perspective query

Continuous perspective query is to continuously update results of perspective query as time goes by. As range query or k-NN query become continuous query by the movement of query point, a perspective query becomes a continuous perspective query according to the change of the parameters (v_p, v_{dir}, v_{ang}). This is presented in figure 4.



(a) change of viewpoint (b) change of view direction (c) change of view angle

Fig. 4. Changes of parameters in perspective query

Let $q_{reg}(t)$ be the query region at time t. Then we can represent perspective query as a function of time, $Q_p(q_{reg}(t))$ as follows, where T is a set of sampled time.

Definition 2. *Continuous Perspective Query*

$$Q_{cp}(T) = \{(t, Q_p(q_{reg}(t))) \mid t \in T\}$$

Figure 5 illustrates a continuous perspective query for the sampled times t_1, t_2, t_3, and t_4, where there are only two LODs, which are LOD_1 and LOD_2. While the query result at t_1 is $\{(a, LOD_1)\}$, the LOD is upgraded to LOD_2. It becomes $\{(a, LOD_2), (b, LOD_1)\}$ at t_2, since a new object b is included in the query region.

3.3 Continuous perspective query on road networks

In this paper, we restrict the continuous perspective query to road network space for two reasons. First, most real applications of perspective query are related to the navigation of vehicles and the information about 3-D spatial objects along a road is very useful to drivers.

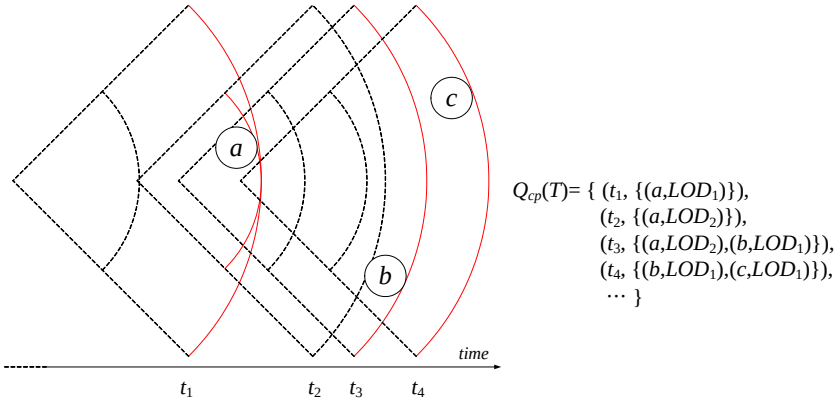


Fig. 5. Continuous perspective query

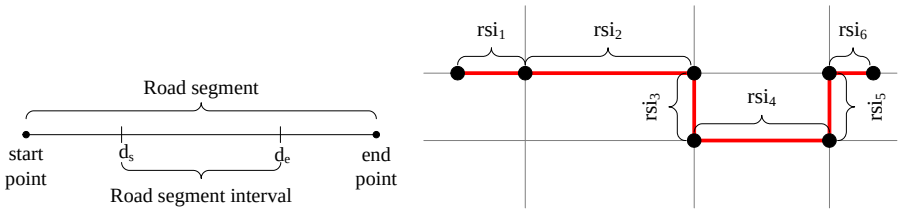
Second, we can easily obtain the view direction and angles of a vehicle on a road. Once a vehicle is on a given road, we can compute the view direction of the driver from the direction of the road. We can also define the view angle of the driver in spite of slight difference between drivers and vehicle types. This means that the third parameter of continuous perspective query, v_{ang} is fixed, and the second parameter v_{dir} depends on the road where the vehicle is located. For these reasons, we focus on road network space rather than Euclidian space.

Then we redefine the query region of continuous perspective query for road network space based on these assumptions and observations. First the position on road networks is given as (seg_{ID}, d), where seg_{ID} is the segment ID of a road segment and d is the displacement from the starting point of the road segment. As a consequence, the query region is defined, differently from the definition of the previous section, as $q_{reg} = (seg_{ID}, d, dir_{RS}, v_{ang})$, where dir_{RS} is the direction of the road segment, and v_{ang} is given a priori.

For the reason of convenience, we define rsi, which means road segment interval, as $rsi = (seg_{ID}, d_s, d_e)$, where d_s and d_e are the starting and ending displacements respectively within a road segment seg_{ID}. And we also define a sequence of rsi, $SEQ_{rsi} = (rsi_1, rsi_2, rsi_3, \ldots, rsi_n)$. In fact, a SEQ_{rsi} implies the route of a moving object on road networks as shown in figure 6(b). These definitions are illustrated in figure 6(a) and 6(b).

4 Query processing for continuous perspective query

In this section, we propose a query processing method for continuous perspective query on road networks. The objectives of proposed method are to reduce the processing and communication costs.



(a) rsi(road segment interval) (b) Sequence of rsi

Fig. 6. Road segment interval and sequence of it

4.1 Requirements for processing continuous perspective query

Before explaining the query processing method, we introduce a notion of *change point*, which will be required for the query processing. A set of change points $CP(rsi)$ on a road segment interval rsi is defined as a set of points where the result of a continuous perspective query should be changed as the view position is moving along the road segment interval rsi. In fact, the change of the query result takes place when the view position is passing though specific points on the road as shown in figure 7.

Then we define the continuous perspective query on road segment and road networks differently from the definition 2.

Definition 3. *Continuous Perspective Query on Road Segment*

$$Q_{cprs}(rsi) = \{(p, Q_p(p)) | p \in CP(rsi)\}$$

Definition 4. *Continuous Perspective Query on Road Networks*

$$Q_{cprn}(SEQ_{rsi}) = \bigcup_{ris \in SEQ_{rsi}} Q_{cprs}(rsi)$$

Figure 7 shows examples of change positions and continuous perspective query on a road segment. The sampling time in figure 5 is replaced with the change positions in figure 7.

The conceptual processing steps for continuous perspective query is summarized as follows. First, the view point from a mobile client is periodically reported to the server where the 3-D databases are stored. Second, the server sends the query result to the client. Since the query result is a stream, the server sends only the difference from the previous result to the mobile client. It means that the server does not need to send the query result if no difference of the result is found. And the server tells whether there is any difference by means of the change point. If the viewpoint is passing though a change point, it means that a difference is taking place and the difference is to be sent to the mobile client.

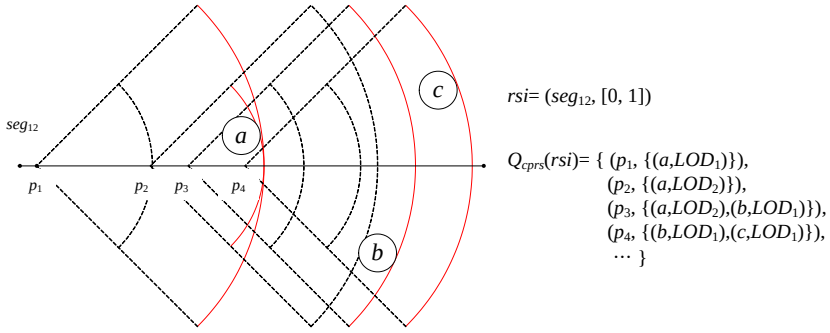


Fig. 7. Change positions and continuous perspective query on road segment

The information about the difference is described as an *update* $u = (p, s, LOD, op)$, where p is a change position, s and LOD are a 3-D object and its LOD respectively, and, op is an operation, which is one of `add`, `remove`, or `update`. This conceptual query processing steps are illustrated by figure 8 and summarized as follows,

- step 1: The viewpoint is sent to the server.
- step 2: The server checks if the viewpoint is passing through a change point. Then the server sends the update information u to the client.
- step 3: repeat step 1 and step 2

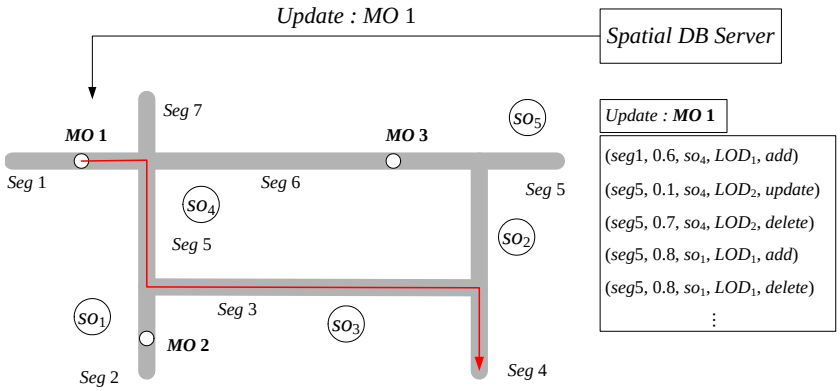


Fig. 8. System architecture of 3D database server

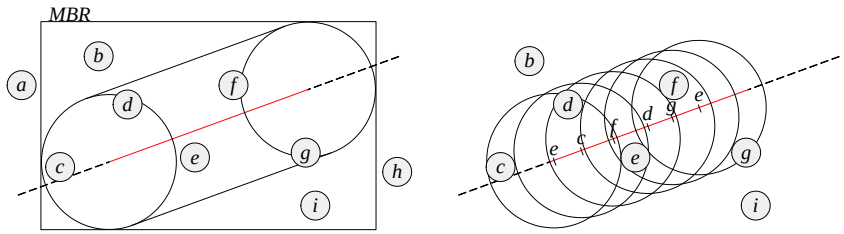
4.2 Query processing by pre-fetching

In order to reduce the communication between the server and clients, and to improve the response time at clients, we can pre-fetch possible updates when a

viewpoint is entering to a road segment interval. Then the continuous perspective query processing on a road segment consists of the following steps;

- step 1: the client sends the road segment interval (rsi) to the server when entering the rsi.
- step 2: the server computes the query region from the rsi.
- step 3: the server searches the change points contained in the query region.
- step 4: the server computes the update at each change point.
- step 5: the server sends the sorted set of updates on the rsi to the client.
- step 6: repeat from step 1 to step 5.

Figure 9 explains the query processing, where only one LOD is defined for the reason of simplicity. First figure 9(a) shows the step 2 and 3 that the server computes the query region and searches the spatial objects contained by the query region. The object a and h are excluded from the candidate objects because they are not contained by the query region. Figure 9(b) represents the processing step 4 and 5. The objects b and i are removed from the query result since they are out of the query region. And the change positions are computed from the searched objects (c, d, e, f, g). Finally, the sorted list of updates according to change positions is sent to the client.



(a) Filter step (b) Refinement step

Fig. 9. Query processing by rsi

After the client receives the the sorted list, it triggers each update in the sorted list, when the viewpoint passes through a change point in the list. In order to compute and send the sorted list of updates, we need a preprocessing at the server side, which will be explained in the next subsequent section.

4.3 Data structures for the preprocessed results

Since the computation of change positions and the corresponding updates are fixed once a road segment and 3-D objects are given, we do not need to repeat unnecessary computations. Instead we can simply get the pre-computed list of a given road segment and reduce the overhead on the server.

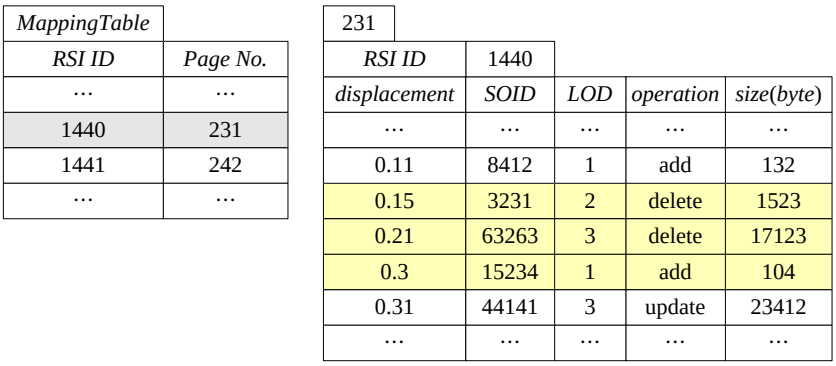


Fig. 10. Structure of sorted offset list

We propose the data structures of the preprocessed list of each road segment to efficiently retrieve the results and provide the stream of the results to mobile clients. Figure 10 shows the data structures of the sorted list of updates. First a mapping table contains the identifier of road segment interval and the disk page number where the pre-computed sorted list is stored. Second the pre-computed sorted list is stored at the corresponding disk page as shown by figure 10. The size of each object to send to client is also included as an attribute, since this information is useful for the transfer between the server and client.

5 Evaluation of performance

In this section, we will investigate the performance of the proposed query processing method. The processing cost of the query includes the cost of the server, the communication cost, and the cost at the client side. The performance bottleneck of the entire system comes from the costs of the server and communication, which we will discuss in this paper.

5.1 Environment of experiments

We prepared a set of 3-D spatial objects for the test data, extracted from the real building data in Daejon city area. This data set consists of three LODs, where LOD 1 describes only the simplified geometry of building objects, LOD 2 includes the detail geometry and texture of buildings, and the real facet image of a building is contained in LOD 3.

And in order to analyze the cost with several query ranges, we defined the several visible ranges of each LOD as table 2. For the LOD type 1, the lengths of query range are defined so that the LOD 1 of a 3-D object with 10 m height be displayed as a 5 mm object on a screen of 200 mm x 100 mm. For the LOD 2, this object should be displayed as an 1 cm object on the screen, and LOD 3 be as 5 cm object. Then for LOD type 1, the lengths of the query ranges for LOD 1,

LOD 2, and LOD 3 become 117 m, 58 m, and 11 m, respectively. Similarly, the lengths of query range for LOD type k are determined as k times of the lengths in LOD type 1 as shown by table 2.

	LOD1	LOD2	LOD3
LOD type1	117m	58m	11m
LOD type2	234m	116m	22m
LOD type3	351m	174m	33m
LOD type4	468m	232m	44m
LOD type5	585m	290m	55m

Table 2. Lengths of query ranges for several LOD types

5.2 Cost at the server

The cost at the server is mainly determined by the costs for searching and prefetching the sorted list, and reading the data of 3D objects, when the viewpoint is entering a new road segment interval. Let us assume that all data are stored on disk except the mapping table. Then the processing cost for new road segment interval C_{RSI} is given as

$$C_{RSI} = C_{MT} + C_{SDL} + C_{DATA}$$

C_{MT} is the cost for accessing the mapping table, C_{SDL} is for reading the disk pages of the sorted list for a given road segment interval, and C_{DATA} for retrieving the data of 3-D objects on the road segment interval. However the size of the mapping table is small and can be stored in main memory, we ignore the cost for the mapping table C_{MT}.

The cost for accessing the sorted list disk pages depends on the number of tuples on the road segment interval. If the length of the road segment interval is sufficiently short, the all change positions can be stored at a single disk page. But as grows the length, the number of disk pages increases. Figure 11 shows this relationship with several LOD types. Since the size of a tuple of the sorted list is 16 bytes, 256 tuples are stored at a disk page of 4 K bytes. It means that we need 4 disk pages to store a sorted list of LOD type 1 when the length of road segment interval is 200 meters. We observe that the number of tuples to transfer is proportional to the length of road segment interval, which is obvious when the density of 3-D objects is uniform.

The cost for retrieving the data of 3-D objects is determined by the size of 3D data and the placement on the disk. Table 3 shows approximate average sizes of a 3-D object at different LODs.

As shown in table 3, we see that the size of 3-D object for LOD 3 is quite large and it is practically the bottleneck of the server performance. The processing cost

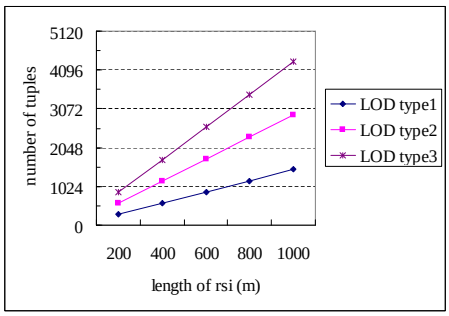


Fig. 11. Number of tuples at a road segment interval

	size
LOD 1	200 Bytes
LOD 2	1 K Bytes
LOD 3	100 K Bytes

Table 3. Approximate sizes of 3D object for different LODs

at the server is determined by C_{DATA} than C_{SDL} due to the large size of 3-D object of LOD 3. This fact is clearly illustrated in table 4, which shows the total cost C_{RSI} of the server.

length of rsi	200	400	600	800	1000
C_{Data}	468.6	937.2	1405.8	1874.5	2343.1
C_{SDL}	1	2	3	4	5
C_{Total}	469.6	939.2	1408.9	1878.5	2348.1

Table 4. Cost for retrieving data and the total cost at the server (disk page: 4K bytes)

5.3 Cost for communication

The communication cost is divided into two measures; the number of communications and the size of data to transfer. The number of communications from the server to mobile clients depends on the number of entrances to a new road segment interval. This means that we may reduce the number of communications by enlarging the length of road segment intervals. On the other side, a long road segment interval results in a large amount of data of a transfer.

The size of data to transfer depends on the number of updates. Figure 12 show the relation between the amount of data to transfer and the length of a road segment interval. For example, the server should transfer about 10 K bytes

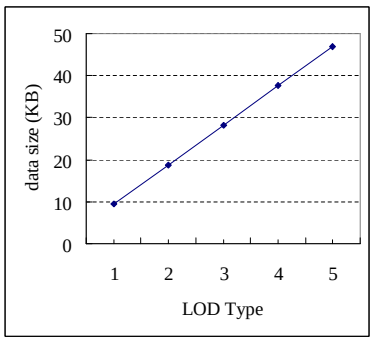


Fig. 12. Amount of data to transfer per 1 m

per 1 meters as shown in figure 12. Suppose that a vehicle is moving in 36 km/h. Then the server should transfer 100 K bytes per second. It means that we should either provide a high speed wireless communication media or reduce the size of LOD 3 data to support the continuous perspective query.

6 Conclusions

In this paper, we proposed a new type of query, called continuous perspective query to support stream services of 3D data to navigation users on the street. In comparison with the conventional spatial query such as range query, the results of perspective query depends on the distance between the query point and spatial objects and multiple LODs are applied to the results according to the distance. The contributions of our work are summarized as follows;

- we have introduced the continuous perspective query type. The query may be used for providing streaming service of 3-D data to navigation users along streets.
- an efficient query processing method has been proposed, which is based on pre-fetching and preprocessing techniques.

In this paper, we excluded the data modelling and progressive transfer issues from the scope of the study. We assumed that a complete data of a 3D object should be transferred when upgrading its LODs, for example from LOD 2 to LOD3. However, we could reduce the amount of data to transfer by a progressive transfer of 3D geometry data. And a careful data model is required to realize the progressive transfer. For this reason, our future work will include the data modelling and progressive transfer for continuous perspective query processing.

References

1. Michael Kofler, Michael Gervautz, and Michael Gruber.: "R-trees for Organizing and Visualizing 3D GIS Databases". Journal of Visualization and Computer Animation, vol. 11, No. 3 (2000) 129-143

2. Jun Li, Ning Jing, and Maoyin Sun.: "Spatial Database Techniques Oriented to Visualization in 3D GIS". In Proceedings of the 2nd International Symposium on Digital Earth 2001
3. Tomas Moller, Eric Haines, and Tomas Akenine-Moller.: "Real-Time Rendering", 2nd Edition, A. K. Peters 2002
4. Douglas B. Terry, David Goldberg, David Nichols, and Brian M. Oki, "Continuous Queries over Append-Only Databases", In Proceedings of ACM SIGMOD Conference 1992, 321-330,
5. Iosif Lazaridis, Kriengkrai Porkaew, and Sharad Mehrotra.: "Dynamic Queries over Mobile Objects". In Proceedings of EDBT Conference 2002, 269-286
6. Dragan Stojanovic, Slobodanka Djordjevic-Kajan, Apostolos N. Papadopoulos, and Alexandros Nanopoulos.: "Continuous Range Query Processing for Network Constrained Mobile Objects". In Proceedings of the 8th ICEIS 2006, 63-70
7. Sunil Prabhakar, Yuni Xia, Dimitri Kalashnikov, Walid Aref, and Susanne E. Hambrusch.: "Query Indexing and Velocity Constrained Indexing: Scalable Techniques for Continuous Queries on Moving Objects". IEEE Transactions on Computers, October, Vol. 51, No. 10 (2002)
8. Ying Cai, Kien A. Hua, and Guohong Cao.: "Processing Range-Monitoring Queries on Heterogeneous Mobile Objects". In Proceedings of Mobile Data Management Conference 2004
9. Bugra Gedik, Kun-Lung Wu, Philip Yu, and Ling Liu.: "Motion Adaptive Indexing for Moving Continual Queries over Moving Objects". In Proceedings of the CIKM Conference 2004, 427-436
10. Mohamed F. Mokbel, Xiaopeng Xiong, and Walid G. Aref.: "SINA: Scalable Incremental Processing of Continuous Queries in Spatio-temporal Databases". In Proceedings of SIGMOD Conference 2004, 623-634
11. 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 ICDE 2005, 643-654
12. Mohammad R. Kolahdouzan and Cyrus Shahabi.: "Continuous K-nearest neighbor queries in spatial network databases". In Proceedings of STDBM 2004, 33-40
13. Hyung-Ju Cho, and Chin-Wan Chung.: "An Efficient and Scalable Approach to CNN Queries in a Road Network". In Proceedings of VLDB 2005, 865-876
14. Yufei Tao, Dimitris Papadias, and Qiongmao Shen.: "Continuous Nearest Neighbor Search". In Proceedings of VLDB 2002
15. Kyriakos Mouratidis, Man L. Yiu, Dimitris Papadias, and Nikos Mamoulis.: "Continuous Nearest Neighbor Monitoring in Road Networks". In Proceedings of VLDB 2006