VizQ: A System for Scalable Processing of Visibility Queries in 3D Spatial Databases

Arif Arman[1], Mohammed Eunus Ali[2], Farhana Murtaza Choudhury[3], and Kaysar Abdullah[4]
[1−2,4]Bangladesh University of Engineering and Technology, Bangladesh
[3]RMIT University, Australia
arman@cse.uiu.ac.bd,eunus@cse.buet.ac.bd,farhana.choudhury@rmit.edu.au,kaysar@cse.buet.ac.bd

ABSTRACT

In this demonstration, we present *VizQ*, an efficient, scalable, and interactive system to process and visualize a comprehensive collection of novel visibility queries in the presence of obstacles in 3D space. Specifically, we demonstrate *four* types of query processing: (i) *k Maximum Visibility Query* (*k*MVQ), that finds *k* locations with the maximum visibility of a target object (ii) *Visibility Color Map (VCM)*, where each point in the space is assigned a color value denoting the visibility measure of the target (iii) *Continuous Maximum Visibility (CMV)* that continuously finds the location that provides the best view of a moving target, and (iv) *Text Visibility Color Map (TVCM)*, where *VCM* is generated considering readability of text data displayed on a target. We are the first to propose efficient algorithms to run all of the above four types of visibility queries in the context of a large number of 3D obstacle database. We exploit human visibility metrics to design our data structures and algorithms to efficiently process queries, and our approaches outperform baseline approaches in several order of magnitude both in terms of I/Os and processing time. The link of our demonstration video is https://youtu.be/rcizJtFvQfU.

CCS CONCEPTS

• **Information systems → Information retrieval query processing**; *Location based services*;

KEYWORDS

Visibility query; Spatial database; Location based services

1 INTRODUCTION

The 3D models of real-life urban structures are becoming widely available through the popular mapping services, such as Google Maps, Google Earth, and OpenStreetMap. Such availability enables us to address many practical applications that require visibility computation in the presence of 3D obstacles. For example, (i) a tourist may wish to find a location to enjoy the best view of a tourist attraction, or a hotel with the best view of the city skyline; (ii) a security company may want to find the suitable positions to place surveillance cameras in a city; (iii) the police may want to track a car with surveillance cameras, so they need to continuously switch to the camera that provides best visibility of the car; (iv) an advertisement company may want to check the visibility of a billboard from the surrounding areas to decide on the billboard's position and suitable font size. In these examples, several infrastructures such as buildings can obstruct the view of the cameras/viewers.

The existing studies in computation geometry and spatial databases ([5]) regard visibility as a binary notion, where a point is either visible or not from another point. In contrast, all of the aforementioned applications require visibility quantification as a continuous notion. Such quantification is important, because a billboard can be visible, but may not be readable from a viewpoint.

The visibility calculation and the processing of related queries are much more challenging than that of distance-based spatial queries due to the complex correlations among the objects, as the visibility of an object from a viewer depends on the presence of other objects in between them. Although there are many objects in the database, only a few objects around a viewer may obstruct the view of the other objects. So an efficient method to determine relevant obstacles is very crucial to the performance of the visibility based algorithms. In this paper, we demonstrate four visibility queries and their corresponding efficient processing algorithms that focus on retrieving only the obstacles necessary (e.g., affect the final result) for calculation, and thus reduce the total I/O cost and the processing time. The key ideas of all these approaches is to exploit human visibility metrics in designing smart data structures and algorithms that facilitate faster processing of visibility queries in the context of a large 3D obstacle database.

Specifically, in this demonstration, we present VizQ, an interactive system that allows users to process and visualize the steps of four novel visibility queries, namely (i) *k Maximum Visibility Query (kMVQ)*, (ii) *Visibility Color Map (VCM)*, (iii) *k Continuous Maximum Visibility (kCMV)*, and (iv) *Text Visibility Color Map (TVCM)*.

The algorithms of the first three queries are based on the previous works from our group. *k*MVQ [4] finds *k* locations from a set of query locations that maximize the visibility of a target object *T* in the presence of obstacles, i.e., addresses the first type of example applications mentioned above. To address the second type of example applications that require the visibility from/of every point of a continuous space, Choudhury et al. [1] propose a scalable technique to partition the space in a visually meaningful way and generate a heat-map of the space, the VCM, by assigning a color to each partition according to their visibility. The partial visibility is not taken into account in [1], i.e., *T* is considered as visible from a point *p* if *T* is entirely visible from *p*. Rabban et al. [6] proposed a technique that alleviates this limitation. Haider et al. [7] address the

CIKM'17 , November 6–10, 2017, Singapore, Singapore
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-4918-5/17/11…$15.00
https://doi.org/10.1145/3132847.3133190

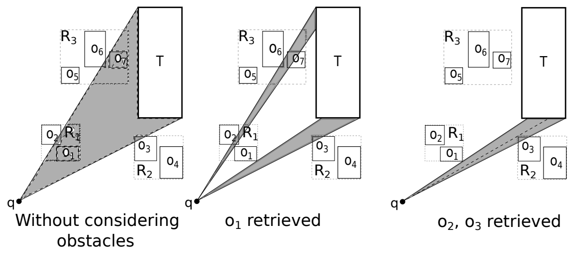


Figure 1: Updating visible region of a query

continuous version of the kMVQ query, denoted as kCMV, where the problem is to continuously report the k locations that maximize the visibility of a moving target (the third type of example application). *In addition to the queries presented in our previous works ([1, 4, 6, 7]), we include a new query type,* TVCM, *that generates VCM by considering the readability of text displayed on the target.*

We develop VizQ as an interactive system that demonstrates the details steps of the algorithms visually. It simulates the index of the objects, the object retrievals from disk, the pruning techniques, along with the computation of the visibility to answer the queries. The I/O costs for processing each query are also reported. The users are given options to upload their own datasets, and set different parameters, such as the target, the number of results to return, etc.

Note that, though some existing softwares [2, 3] used in urban planning and architecture facilitate platforms to create and render 3D objects with functionalities like animation and walk-through, they do not provide any functionality for visibility query processing. Hence, VizQ can be used as a complementary system with these softwares to enable them answering realistic visibility queries that require quantification of visibility in a 3D space.

2 OVERVIEW OF THE APPROACHES

The set O of obstacles are indexed with an R*-Tree. We first introduce a visibility metric, and then the notion of visible region, which is used to avoid retrieving the unnecessary obstacles that do not affect the result of a query.

Visibility metric. We consider the visual angle, i.e., the angle formed at a lens by an object as the measure of visibility. The value depends on the distance and the viewing angle between the object and the viewer, calculated as,

$$V = 2arctan(S_\alpha / D)$$

where $S_\alpha = \frac{\alpha}{90^o} \times S$. Here, S is the original length of T; D and α are the distance and angle between the viewer and the target object T, respectively.

Visible region (VR). A visible region w.r.t a target T and a query location q consists of the points in space such that any point p in VR, p is visible to both q and any point in T. Only the obstacles overlapping with VR can affect the visibility.

2.1 kMVQ

Three different approaches, namely, query centric distance (QD), query centric visible region (QV) and target centric distance (TD) based approach are proposed in [4], where the approaches differ in their retrieval orders of the obstacles. Starting from a query point, the QD and QV approaches retrieve obstacles incrementally based on their distances from the query point and based on their effects on visible region, respectively. The TD approach retrieves obstacles incrementally based on their distances from the target. As the experimental results reported in [4] show that QD outperforms the other approaches, we present the QD approach in this demonstration. The major steps of the *query centric distance based approach (QD)* to answer the kMVQ are:

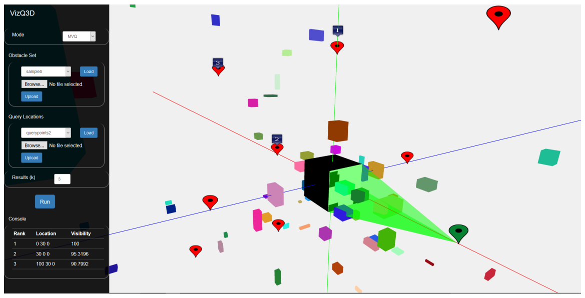
- **Best-first search:** A visibility estimate of each query point is computed by considering a subset of relevant obstacles, and is incrementally updated by considering one node/obstacle at a time. In each iteration, we consider the query location q with the maximum visibility estimation.
- **Obstacle pruning:** As a nearby obstacle is likely to reduce the VR more, we retrieve the obstacles in an increasing order of their distances from q, and update the VR. For each q, we maintain a priority queue of obstacles and MBRs based on their distances from q. If an MBR of the R*-tree or an obstacle is completely outside VR, it can be safely pruned.
- **Early termination:** If the priority queue of obstacles is empty for the current top ranked q, no other query location can have better visibility than q. We terminate the process when k such locations are found.

Figure 1 shows the visibility region computation of a target T w.r.t. a query point q. The objects that intersect with the initial visibility region are o_1, o_2, o_3, o_7, and o_8. The objects are retrieved based on the distance from q. When objects o_1 and o_2 are retrieved, they obstruct o_7 and o_8 from q, therefore discarded from further consideration.

2.2 VCM

We propose a space partitioning technique that avoids computing the visibility of a large number of points, and a retrieval technique to prune unnecessary obstacles to construct VCM. We also present two approximations of the partitions to improve efficiency at the cost of guaranteed error bounds. Our approach effectively prunes a large number of obstacles and significantly reduces I/Os while computing the VCM. VCM experiments with real and synthetic 3D datasets demonstrate that our approach outperforms the straightforward approach (that computes the visibility of every cell by considering all obstacles between the cell and the target) by $5 - 6$ orders of magnitude in terms of total processing time. We process the query in the following way:

- **Space partitioning:** We exploit the key insight that a human cannot differentiate the visual appearance of an object from spatially close set of points within a threshold. Thus, we partition the space based on the distance and angle from T into a set of cells such that the perceived visibility of T is indistinguishable from the points in a cell. This significantly reduces computational overhead in contrast to computing the visibility from each discrete point or from each cell.
- **Obstacle pruning:** We perform a plane sweep to determine the combined effect of multiple obstacles, and efficiently prune the unnecessary obstacles.
- **Visibility calculation:** We determine the visible portions of T from each cell that are either completely or partially visible to T, compute their corresponding visibility and represent



Mode: MVQ

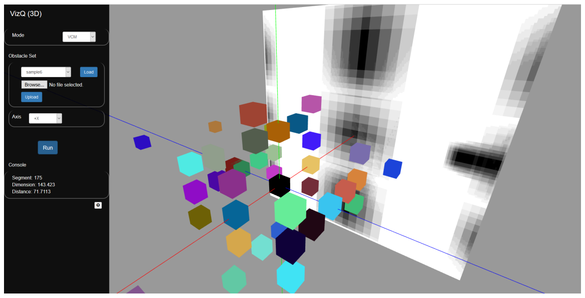
Obstacle Set: sample5 — No file selected.

Query Locations: querypoints2 — No file selected.

Results (k): 3

Rank	Location	Visibility
1	0 30 0	100
2	30 0 0	95.3196
3	100 30 0	90.7992

Figure 2: k Maximum Visibility Query



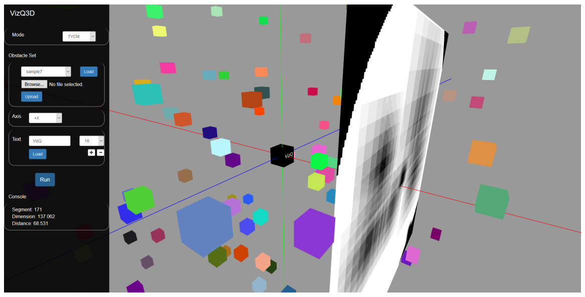
Mode: VCM

Obstacle Set: sample6 — No file selected.

Axis: +X

```
Segment: 175
Dimension: 143.423
Distance: 71.7113
```

Figure 3: Visibility Color Map



Mode: TVCM

Obstacle Set: sample7 — No file selected.

Axis: +X

Text: VizQ — 16

```
Segment: 171
Dimension: 137.062
Distance: 68.531
```

Figure 4: Text Visibility Color Map

the measure as a color. If a cell is completely invisible, the cell is colored as black.

2.3 kCMV

The key challenge of the kCMV query is that, as the target object T is continuously in motion, the visibility of T from each query location continuously changes, and we need to update their ranking and the results. The approach consists of a pre-processing step and an incremental update step.

- **Pre-processing step:** This step consists of generating the aggregated visible region (AVR) and blocking set (BS) of the query locations. AVR is a region formed by overlapping the VRs of one or more query points and is disjoint from all other AVRs. Any point within an AVR is visible only from those query points whose VRs formed that AVR. The blocking set of a query location q is the set of objects that lie in the VR of q, i.e., the objects that can affect the visibility of T from q. The aggregated blocking set (ABS) of an AVR is the set of obstacles that may affect visibility when target lies in that AVR. In the pre-processing step, the AVRs are constructed and indexed in a R*-Tree.
- **Incremental update step:** At any point of time, the target intersects with/resides in one or multiple AVRs. If target moves at least for a threshold amount, at first we check if the target has changed any of those AVRs. If there is no change, then we update the rank of the query points of those AVRs with the help of the objects in their blocking sets. If the target intersects with any new AVR, the visibility of T from the query points in that AVR is calculated and the ranking of the query locations are updated. Similarly, if T leaves any AVR, the query locations of that AVR is excluded from the result and the result needs to be updated.

2.4 TVCM

In this demonstration paper, we present a new extension of the VCM problem where we need to determine the readability of text data displayed on a target while constructing the VCM. We use Snellen chart [8] to determine the minimum font size that is readable from a certain distance when the observer is facing T at 90^o angle. Then, for each cell c of the VCM, we apply the oblique projection technique to determine the perceived font size required to be readable from c. If this perceived font size is larger than the user selected size, the text is considered as not readable from that cell. Here, we only need to consider the face of T that displays the text.

To incorporate a real billboard scenario, in TVCM we also incorporate multi-line texts, where each line of text has a different font size represented with a unique color. In that case, we create TVCM for each line of text separately and superimpose them to generate a combined TVCM, where each cell gets its color by superimposing the visibility color value for each line of text.

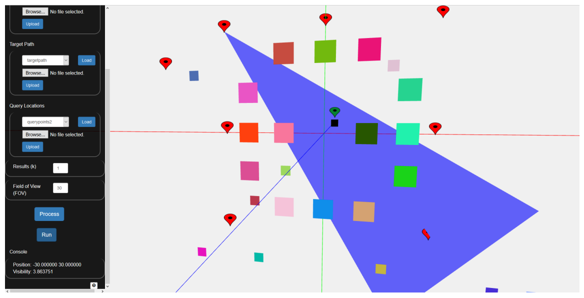
3 DEMONSTRATION

VizQ is implemented as a web application using Three.js WebGL framework with C++ backend. We demonstrate VizQ using 3D datasets. User interface of VizQ consists of two panels:

Settings Panel. This is the left panel of the user interface that includes options to select from the four modes as the query type to be processed, (i) kMVQ, (ii) VCM, (iii) TVCM, and (iv) kCMV. All modes require the user to select a spatial dataset of 3D objects. The user may select one from the list of existing datasets in the server or upload a dataset from local file system.

In kMVQ mode, VizQ allows the user to select a set of 3D query locations, either from the existing database, or by uploading a new set. VizQ also enables user to plot query locations in the *visualization panel* interactively through mouse-clicks. The user can click over an object from the dataset to select as the target T, and give an input for k. The rest of the objects are considered as obstacles. The *kMVQ* settings also include an input for k, the number of query locations to return.

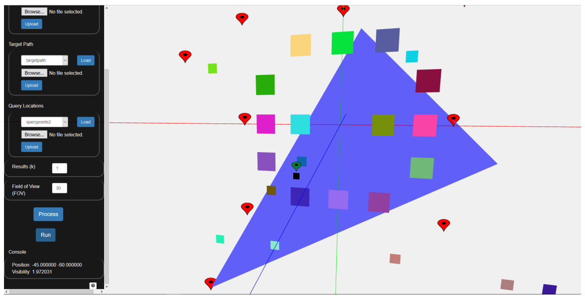
In *VCM* mode, the user can select an object as the target T and an axis along which *VCM* needs to be generated. In addition, the settings in *TVCM* mode allow the user to add text data using multiple text boxes (muti-line texts with varying fonts). These texts are considered as displayed in the face of T along the picked axis.



Results (k): 1
Field of View (FOV): 30

```
Position: -30.000000 30.000000
Visibility: 3.863751
```

(a)



Results (k): 1
Field of View (FOV): 30

```
Position: -45.000000 -60.000000
Visibility: 1.972031
```

(b)

Figure 5: Continuous Maximum Visibility Query

In *CMVQ* mode, the user can specify *field of view* and the direction of viewpoints to calculate visible region of each query location. The user can select a target path by uploading a file where path can be specified as a series of coordinates at different timestamps. The user is required to select an obstacle set, target path and query location set and click the *process* button for pre-computation. VizQ notifies the user via the console when pre-computation is complete.

The simulation of the selected query processing starts when the user clicks the *Run* button. If any required parameter is missing (e.g., target not set), the appropriate warning message is shown in the console. When a valid query is issued, VizQ updates the *visualization panel* accordingly.

Visualization Panel. This is the right panel of the user interface that displays a 3D world to the user. VizQ integrates a mouse-controlled dynamic camera including rotate, pan, and zoom operations. This provides the user the freedom to visualize from different viewing positions. When the user loads a dataset as the object set, VizQ displays each object with different colors. Once the dataset is loaded, VizQ allows the user to select an object as the target T through mouse-clicks. The selected target is shown in *black* and the information regarding T is shown in console.

In *k*MVQ mode, each query location q is shown by a *red location marker*. A transparent *green* region represents the VR of a q. VizQ presents the VR of one query location at a time, starting from the top ranked one according to visibility estimation. The user can view the VR of the subsequently ranked query location by pressing the key 'q'. The user can also view the VR of a q by clicking on the corresponding marker. The details of the selected q (e.g. rank, visibility) is displayed in the *settings panel* console. The visible portions of T from the selected q is colored as *white*, and the obstructed portions are colored as *black*.

Each cell in *VCM* is colored with a shade of *grey* where a *black* cell means T is completely invisible from that cell. The user can move the *VCM* plane back and forth (using 'v' or 'b' keys) to visualize the occlusion effects of the obstacles. The plane sweep algorithm can be better comprehended by continuously moving the plane away from T. In TVCM mode, VizQ displays the provided texts in their corresponding font sizes on the selected face of T. The color convention is the same as VCM. As the readability decreases with the increase of distance, the *TVCM* may turn completely black at some point if the user continues to move the plane farther away.

For *CMVQ*, at each timestamp, VizQ displays the updated location of the target, where the target is colored as *black* with a green marker. The visible region of the best viewpoint is shown in *blue*. VizQ also displays details such as the visibility measure of the best viewpoint in the console. If the best viewpoint changes due to target movement, VizQ updates the display accordingly in real time.

Demonstrated Scenarios. Fig. 2 shows a screenshot of *k*MVQ processing demonstration that shows: (i) The visible region of a selected query location in green, where the obstructed portion of the target is shown in black; (ii) The top-3 query locations are shown with a message bubble, and their visibility scores are shown in the console. Fig. 3 and Fig. 4 each shows a screenshot of VCM, where the visibility from different points of space are shown with shades of grey along a selected axis. In addition, Fig. 4 shows the text on a face of the target, and the VCM adjusted for the size of the text. The continuous maximum visibility is demonstrated in Fig. 5 for two different locations of the moving target, where the result is updated for the corresponding location.

4 CONCLUSION

We develop VizQ, a system that demonstrates the internal mechanism of the efficient algorithms of four important and novel visibility queries in 3D space. The system simulates the obstacle retrievals and the corresponding changes in visibility. The users can issue the queries in an interactive way and monitor different levels of details. The source code is publicly available in github (https://github.com/arif-arman/vizq), which can be adapted in real-life applications.

ACKNOWLEDGMENTS

This research is funded by ICT Division - Government of the People's Republic of Bangladesh.

REFERENCES

[1] Farhana Murtaza Choudhury, Mohammed Eunus Ali, Sarah Masud, Suman Nath, and Ishat E. Rabban. 2014. Scalable visibility color map construction in spatial databases. *Inf. Syst.* 42 (2014), 89–106.
[2] Cityengine. 2008. http://www.esri.com/software/cityengine. (2008).
[3] Lumion. 2012. http://lumion3d.com/urban-planning-software. (2012).
[4] Sarah Masud, Farhana Murtaza Choudhury, Mohammed Eunus Ali, and Sarana Nutanong. 2013. Maximum visibility queries in spatial databases. In *ICDE*. 637–648.
[5] S. Nutanong, E. Tanin, and R. Zhang. 2010. Incremental Evaluation of Visible Nearest Neighbor Queries. *TKDE* 22, 5 (2010), 665–681.
[6] Ishat E. Rabban, Kaysar Abdullah, Mohammed Eunus Ali, and Muhammad Aamir Cheema. 2015. Visibility Color Map for a Fixed or Moving Target in Spatial Databases. In *SSTD*. 197–215.
[7] Ch. Md. Rakin Haider, Arif Arman, Mohammed Eunus Ali, and Farhana Murtaza Choudhury. 2016. Continuous Maximum Visibility Query for a Moving Target. In *ADC*. 82–94.
[8] H. Snellen. 1862. Probebuchstaben zur Bestimmung der Sehschärfe. *Utrecht* (1862).