Data Mining and Knowledge Discovery
https://doi.org/10.1007/s10618-019-00641-3

Multi-location visibility query processing using portion-based transactional modeling and pattern mining

Lakshmi Gangumalla[1] · P. Krishna Reddy[1] · Anirban Mondal[2]

Received: 26 November 2018 / Accepted: 23 June 2019

Abstract

Visibility computation is critical in spatial databases for realizing various interesting and diverse applications such as defence-related surveillance, identifying interesting spots in tourist places and online warcraft games. Existing works address the problem of identifying individual locations for maximizing the visibility of a given target object. However, in case of many applications, a set of locations may be more effective than just individual locations towards maximizing the visibility of the given target object. In this paper, we introduce the Multi-Location Visibility (MLV) query. An MLV query determines the top-k query locations from which the visibility of a given target object can be maximized. We propose a portion-based transactional framework and coverage pattern mining based algorithm to process MLV queries. Our performance evaluation with real datasets demonstrates the effectiveness of the proposed scheme in terms of query processing time, pruning efficiency and target object visibility w.r.t. a recent existing scheme.

Keywords Visibility query · Transaction-modeling · Pattern-mining · Portion based transactional framework

Responsible editor: Po-ling Loh, Evimaria Terzi, Antti Ukkonen, Karsten Borgwardt and Katharina Heinrich

✉ Lakshmi Gangumalla
lakshmi9414@gmail.com

P. Krishna Reddy
pkredddy@iiit.ac.in

Anirban Mondal
anirban.mondal@ashoka.edu.in

1 IIIT, Hyderabad, India

2 Ashoka University, Delhi, India

Published online: 26 July 2019

1 Introduction

Applications in spatial databases, computer graphics, computational geometry, geographic information systems and environmental modeling often require visibility computation. Given a set of obstacles in space (e.g., Euclidean space), two points in the space are said to be visible to each other if the line segment that joins them does not intersect any obstacles (Zhang et al. 2004). Visibility computation involves determining whether a given target object is visible from a specific query location, identifying the top-k individual query locations that maximize the visibility of the given target object and so on.

Visibility computation is critical in spatial databases for realizing various important applications (Gao and Zheng 2009; Gao et al. 2009a, b; Nutanong et al. 2010; Tao et al. 2002; Xu et al. 2010). Consider a defence-related surveillance application scenario. Here, the visibility of a given target object, which needs to be kept under observation, typically changes depending upon the environment (e.g., obstacles such as trees and the nature of the terrain) and the movement of vehicles such as tanks. Moreover, in a battlefield scenario, efficient visibility computation becomes a necessity to effectively plan the movement of vehicles (e.g., tanks) as well as army personnel across a given terrain in the presence of obstacles. In a similar vein, visibility computation is essential for surveillance applications such as monitoring car parking lots by means of CCTV cameras. Observe that all of these above application scenarios involve the presence of obstacles such as buildings, trees and so on.

Existing works (Gao and Zheng 2009; Gao et al. 2009a, b; Nutanong et al. 2010; Tao et al. 2002; Xu et al. 2010) have addressed the problem of visibility computation in spatial databases in the presence of obstacles by considering variants of k-nearest neighbour (kNN) queries. However, these works do not address visibility maximization of a given target object. In contrast, given a set of candidate locations, the *Maximum Visibility (MV) query* (Masud et al. 2013) determines a ranked list of the top-k locations from which the visibility of a given target object can be maximized in the presence of obstacles. However, existing approaches, such as (Masud et al. 2013), suffer from a drawback in that they try to maximize the visibility of a given target object from a *single* query location. However, in real-world scenarios, it often becomes practically infeasible to cover a significant percentage of any given target object from only a single query location, thereby narrowing the applicability of the existing approaches. Intuitively, a combination of query locations (e.g., a combination of CCTV cameras) could provide significantly higher visibility of a key enemy target area or a given expensive product as opposed to any one single query location (e.g., a single CCTV camera). However, given a target object and n query locations, the total number of potential combinations of these query locations (i.e., the candidate sets) would be $(2^n - 1)$, which is prohibitively expensive.

This work addresses the problem of determining top-k candidate sets of query locations from which the visibility of a given target object can be maximized in the presence of obstacles. We assume that the query locations are points, while the obstacles and the given target object are regions in space. Notably, in addition to the visibility consideration, there are two additional issues namely (a) *overlap* and (b) *continuity*, which arise as a consequence of considering multiple query locations in tandem.

Now we shall explain the notions of overlap and continuity. When a given target object T is viewed from multiple query locations, some regions of T may be visible from more than one query location. This results in *overlap* (repetitions/duplications) among the visible regions of T when viewed from different query locations. Thus, an important issue is to obtain a *minimal* set of query locations as far as possible by reducing the overlap for maximizing the visibility of T. Furthermore, applications may require the viewing of T in a continuous manner from multiple locations. We consider two query locations to be continuous if their corresponding visible regions of T have at least a non-zero overlap; otherwise, they are considered to be discontinuous.

To determine a set of multiple query locations to maximize the visibility of a given target object T, we introduce the *Multi-Location Visibility (MLV) query*. Given T, a set of query locations and a set of obstacles, the MLV query determines the top-k candidate sets of query locations. Each candidate set comprises possibly multiple query locations, which together maximize the visibility of T, while satisfying the threshold values of visibility and overlap and ensuring continuity. From these top-k candidate sets, users can choose one or more appropriate candidate sets depending upon the application requirements.

We divide a given target object T into equal-sized units, which we designate as *portions*. We also propose efficient approaches for computing visibility, overlap and continuity of a combination of query locations w.r.t. the portions of T. The size of a portion is essentially application-dependent. By dividing T into portions, T is represented as a set of portion identifiers (*pids*). Thus, now the visibility of T is based on portions as opposed to the area of the regions of T, thereby replacing complex computationally-intensive spatial operations associated with visibility computations by simpler set-based operations.

For efficiently extracting the combinations of query locations, we convert the proposed problem into a pattern extraction problem from a set of *portion transactions*. We form portion transactions by modelling the association of each portion with the corresponding query locations from which the portion is visible as a transaction. Thus, the portion transactions represent all of the possible associations of portions and query locations. Now the problem becomes similar to that of coverage pattern extraction (Srinivas et al. 2012) (see "Appendix A") from transactional databases subject to visibility, overlap and continuity requirements. Hence, for facilitating efficient pruning, we can draw from pattern extraction approaches (e.g., apriori Agrawal and Srikant 1994 or pattern growth Han et al. 2000). Overlap follows the *sorted closure property* (Liu et al. 1999), which we use in this work for pruning purposes. The key contributions of this work are three-fold:

1. We introduce the Multi-Location Visibility (MLV) query.
2. We introduce the concept of *portions* and propose a portion-based transactional framework for facilitating the efficient computation of visibility, overlap and continuity of a combination of query locations.
3. We present an efficient algorithm for processing MLV queries. The algorithm leverages our proposed portion-based transactional framework in conjunction with an apriori-based pruning strategy and coverage patterns.

Our performance evaluation using two real datasets demonstrates the effectiveness of the proposed scheme in terms of query processing time, pruning efficiency and target object visibility w.r.t. a recent existing scheme.

The remainder of the paper is organized as follows. In Sect. 2, we discuss related work and background information about visibility. In Sect. 3, we discuss the context of the problem. The proposed scheme is presented in Sect. 4. In Sect. 5, we report the performance evaluation. We provide discussion in Sect. 6. Finally, we conclude in Sect. 7.

2 Related work and background about visibility

Visibility computation approaches in spatial databases (Gao and Zheng 2009; Gao et al. 2009a, b; Nutanong et al. 2010; Tao et al. 2002; Xu et al. 2010) apply the concept of k nearest neighbours (kNN). A kNN query finds the k nearest points from a given query point based on a distance measure. The work in Gao and Zheng (2009) addressed continuous obstructed nearest neighbor queries by using the concept of control points in conjunction with an efficient quadratic-based split point computation algorithm for processing only the relevant data points and obstacles. The visible reverse nearest neighbor (VRNN) query (Gao et al. 2009a) considers the impact of obstacles on the visibility of objects by pruning away irrelevant data objects to save the traversal cost. Given a query point q, a VRNN query retrieves the points that have q as their visible nearest neighbor in the presence of obstacles. The work in Nutanong et al. (2010) incrementally computes visible neighbors, while enlarging the search space. The work in Gao et al. (2015) examined the Obstructed Reverse Nearest Neighbor query, which is a variant of the reverse nearest neighbour query that also considers the presence of obstacles.

Continuous nearest neighbor (CNN) queries (Tao et al. 2002) involve kNN query processing for a moving query point. CNN algorithms use branch-and-bound techniques for pruning. The work in Gao et al. (2009b) proposed variants of the continuous visible nearest neighbor search by indexing both data points and obstacles in R-trees separately. The work in Haider et al. (2016) introduced the k Continuous Maximum Visibility query, which determines the top-k query locations (from a set of candidate query locations) sorted in the order of visibility of a given target from these query locations. The Group Visible Nearest Neighbor query in Xu et al. (2010) considered both visibility and distance as constraints.

Visibility problems concerning terrains have been discussed in Floriani and Magillo (2003). Visibility computations on a terrain may involve either one or multiple viewpoints and range from visibility queries to the computation of structures that encode the visible portions of the surface. In Goodchild and Lee (1990), visibility coverage problems were defined by using visibility information derived from topographic surfaces. Furthermore, the HDoV-tree index (Shou et al. 2003) has been proposed to improve visual fidelity and performance in large virtual environments based on the degree of object visibility. The proposal in Asano et al. (1986) indicated how to create a data structure for computing the visibility polygon of a given point w.r.t. the set of polygons.

The concept of coverage has been used to solve node cover problem in graph theory (Garey et al. 1974) and set cover problem in set theory (Chvatal 1979). The model of coverage patterns and a level-wise pruning approach to extract coverage patterns from transactional databases was proposed in Srinivas et al. (2012). Pattern growth approach to extract coverage patterns was proposed in Gowtham Srinivas (2015).

Given a target object T and a set of query locations, the work in Masud et al. (2013) proposed the MV query. The MV query determines a ranked list of *individual query locations* that maximize the visibility of T in the presence of obstacles. The work in Masud et al. (2013) proposed some variants for processing MV queries. Among these variants, the Query centric Distance based variant (*QD*) outperformed the others. In *QD*, the obstacles are incrementally considered according to their non-decreasing distances from the query location currently under consideration. In this way, a query point that is severely blocked by a subset of obstacles can effectively be ruled out from the search process.

Approaches for maximizing visibility also include the works in Ali et al. (2017), Irtiza Tripto et al. (2019). The work in Irtiza Tripto et al. (2019) proposed the k Aggregate Maximum Visibility Trajectory query, which determines the top-k trajectories that provide the best view of the target objects in the presence of obstacles. The work in Ali et al. (2017) examines the Maximum Visibility Facility Selection query, which selects k locations (from a candidate set of locations) to place a facility (e.g., a surveillance camera) using which the visibility of a given region can be maximized.

Notably, *none* of these existing approaches has addressed the issue of determining a set of *multiple* query locations for maximizing the visibility of a given target object. Moreover, our work differs from that of the work in Masud et al. (2013), which only uses *individual* query locations for maximizing the visibility of a given target object. Additionally, we have proposed an efficient approach for processing MLV queries by means of a portion-based framework, forming portion transactions and extending the concept of coverage patterns.

2.1 Background information about visibility

Similar to the visibility computation approach in Masud et al. (2013), we adopt the definition of visibility from the works in Morling (2010), Seeds and Backman (2015). Given two point locations l_i, l_j and a set O of obstacles in space, l_i and l_j are considered to be visible to each other if the straight line connecting them does not intersect with any of the obstacles in O. Let Q be the set of all point query locations i.e., $Q = \{q_1, q_2, \ldots, q_n\}$, where n is the total number of query locations. A region V is considered to be visible w.r.t. a given query location q_i iff every point v in V has point-to-point visibility w.r.t. q_i. The visibility of an object T from query location q_i (designated as $vis(q_i, T)$) is defined as the ratio of the angular size *AS(T)* of T to the original size *OS(T)* of T (see Eq. 1).

$$vis(q_i, T) = \begin{cases} AS(T) \,/\, OS(T) & \text{if} \quad d \leq max\,D \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

In Eq. 1, observe that for T to be visible from q_i, the minimum distance between T and q_i should be below a certain threshold, which we designate as $maxD$. Here, $maxD$ is essentially application-dependent. For example, if a human is viewing T from q_i with his naked eye, the value of $maxD$ could be set to at most 1 km. On the other hand, the value of $maxD$ could be set in the range of a few km when the human is viewing T from q_i with binoculars.

Now let us define the notion of minimum visibility threshold $minV$. The value of $minV$ is specified by the users depending upon their requirements. Based on the $minV$ threshold, all individual query locations, for which $vis(q_i, T) \geq minV$, will be part of the result.

We denote the set of the given region(s) of T visible from q_i as $VR(q_i, T)$. (Multiple regions of T may be visible from a given q_i). We use Area() function to represent the spatial area covered by a spatial object or a set of spatial regions. Consider a set ψ of n query locations, where q_i represents the i^{th} query location. We define $vis(\psi, T)$ (i.e., the visibility of T w.r.t. ψ) as follows:

$$vis(\psi, T) = Area\left(\bigcup_{i=1}^{n} VR(q_i, T)\right) / Area(T) \tag{2}$$

In Eq. 2, the term $Area\big(\bigcup_{i=1}^{n}(VR(q_i, T))\big)$ represents the area of the union of the visible regions of q_i, where $i = 1$ to n. Consequently, for ψ to contribute to the MLV query result, $vis(\psi, T) \geq minV$.

3 Context of the problem

Consider a target object T, a set Q of query locations, a set O of obstacles, a minimum visibility threshold $minV$, a maximum overlap threshold $maxO$ and $continuity = true$. The MLV query returns a set C of top-k candidate sets. Each candidate set c_i (where $c_i \in C$) comprises a set of query locations, which maximize the visibility of T, while satisfying the $minV$, $maxO$ and *continuity* criteria. Recall our discussions about the $minV$ constraint in Sect. 2.1. We now explain the $maxO$ and *continuity* constraints.

Regarding overlap, it is possible for the same region of T to be visible from multiple query locations. This requires approaches for identifying combinations of query locations that would minimize overlap among the visible regions of T w.r.t. individual query locations. We designate the overlap for candidate set c_i as $overlap(c_i, T)$. The value of $overlap(c_i, T)$ is computed below:

$$overlap(c_i, T) = Area\left(\sum_{i=1}^{n_i} VR(q_i, T)\right) - Area\left(\bigcup_{i=1}^{n_i} VR(q_i, T)\right) \tag{3}$$

In Eq. 3, $\sum_{i=1}^{n_i} Area(VR(q_i, T))$ is the sum of the areas of visible regions of q_i, where $i = 1$ to n_i. For c_i to be in the MLV query result, $overlap(c_i, T) \leq maxO$, where $maxO$ is the user-defined maximum overlap threshold.

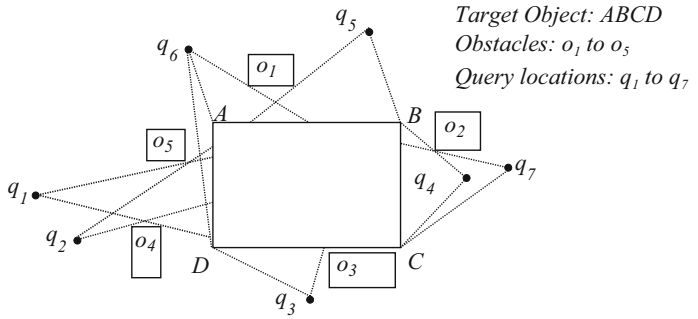


Fig. 1 Illustrative example of overlap and continuity

Regarding $continuity$, given query locations q_i and q_j from the set Q of n query locations, q_i and q_j form one-to-one continuity relationship if there is a non-zero overlap between $VR(q_i, T)$ and $VR(q_j, T)$. We consider q_i and q_j to be $continuous$ if there is a transitive-closure of one-to-one continuity relationship between q_i and q_j over Q. We designate the continuity for candidate set c_i as $continuity(c_i, T)$. Thus, $continuity(c_i, T)$ is $true$ iff the visible regions of every pair of query locations in c_i are continuous; otherwise, it is $false$.

Multiple candidate sets may qualify with the given $minV$, $maxO$ and continuity constraints. We perform top-k ranking of these candidate sets as follows. Since visibility maximization is our key focus, we primarily rank the candidate sets in descending order of their respective values of visibility. If more than one candidate set has the same value of visibility, priority is given to the candidate set with lower overlap. If multiple candidate sets have equal values for both visibility and overlap, ties are resolved based on the number of query locations in the set i.e., the one with lower number of query locations is ranked higher.

Let us use Fig. 1 to understand the notions of overlap and continuity. Figure 1 depicts a target object *ABCD* with sides *AB*, *BC*, *CD*, *DA* and seven query locations q_1 to q_7. Each query location covers only certain regions in $ABCD$ i.e., no query location is individually able to cover the entire region encompassed by $ABCD$ due to the presence of obstacles o_1 to o_5. Notably, existing approaches, such as (Masud et al. 2013), would extract q_4 since it provides the maximum view of $ABCD$. However, q_4 is able to cover only about 25% of the region in $ABCD$ due to the presence of obstacle o_2. Thus, we require a combination of query locations to improve the visibility of $ABCD$ w.r.t. the visibility (of $ABCD$) that is provided by q_4. For example, the combinations of query locations $\{q_1, q_3, q_5\}$ cover approximately 45% of $ABCD$ in the presence of obstacles $\{o_1, o_3, o_4, o_5\}$, while $\{q_2, q_3, q_4, q_6\}$ cover about 60% of $ABCD$ in the presence of obstacles $\{o_1, o_3, o_4, o_5\}$.

Moreover, the visible regions covered by $\{q_1, q_2\}$ and $\{q_4, q_7\}$ overlap, while the visible regions covered by $\{q_1, q_3\}$ and $\{q_4, q_5\}$ do not overlap. Regarding continuity, the visible regions covered by $\{q_5, q_6\}$, $\{q_6, q_1\}$ and $\{q_5, q_6, q_1\}$ are continuous, while the visible regions covered by $\{q_2, q_3\}$ and $\{q_3, q_4\}$ are discontinuous. Now we define the MLV query as follows:

Definition 1 (*Multi Location Visibility (MLV) Query*) Given a target object T, a set $Q = \{q_1, q_2, \ldots, q_n\}$ of n query locations, a set O of obstacles and the user-specified threshold values of $minV, maxO, k$, the *MLV* query returns a set $C = \{c_1, c_2, \ldots, c_k\}$ of top-k candidate sets, where each candidate set c_i comprises a set of query locations such that $vis(c_i, T) \geq minV$, $overlap(c_i, T) \leq maxO$ and $continuity(c_i, T) = true$.

Notably, the value of $vis(c_i, T)$ can be computed using Eq. 2. Furthermore, the value of $overlap(c_i, T)$ can be computed using Eq. 3. Given that c_i contains n query locations, the combined set $VR(c_i, T)$ of visible regions is computed below based on (Tanton 2005).

$$VR(c_i, T) = Area\left(\bigcup_{i=1}^{n_i} VR(q_i, T)\right) = \sum_{t=1}^{n_i} (-1)^{t-1} N_t \tag{4}$$

In Eq. 4, the term $\sum_{t=1}^{n_i} (-1)^{t-1} N_t$ denotes the summation of all combinations of visible regions from query locations, N_t denotes the sum of areas of all possible intersections of visible regions of t number of query locations (Tanton 2005). Observe that Eq. 4 is computation-intensive.

Determining $continuity(c_i, T)$ is also complex as it depends upon the computation of $overlap(c_i, T)$. As the number of query locations in c_i increases, determining $VR(c_i, T)$ exponentially increases since computations are based on spatial intersections and unions. Existing approaches (Bentley and Ottmann 1979; Bentley and Wood 1980), which only verify whether a given set of rectangular regions intersect, incur a complexity of $O(n \log n)$ using Bentley and Ottmann's algorithm (Bentley and Wood 1980) and Shamos and Hoey's algorithm (Bentley and Ottmann 1979). Furthermore, the plane sweep algorithm proposed in Mount (2004) for determining the area of intersection incurs a complexity of $O(n \log n)$.

4 Proposed approach

This section presents our proposed approach.

4.1 Basic idea

Our goal is to develop an approach for determining combinations (i.e., candidate sets) of query locations to satisfy a given MLV query. We shall use the terms **combination of query locations** and **candidate set** interchangeably.

Determining potential candidate sets from n query locations poses scalability issues as n increases. Given n query locations, we need to examine $2^n - 1$ combinations of these query locations to determine the candidate sets that satisfy the constraints of the MLV query. As n increases, the number N_{MLV} of combinations of query locations to be examined would essentially explode. Hence, it becomes imperative to use heuristics for *efficiently* pruning the search space to minimize the value of N_{MLV}. Notably,

the evaluation of each combination of query locations is in itself non-trivial because checking the constraints associated with visibility, overlap and continuity requires computationally expensive spatial computations (see Eqs. 2, 3 and 4).

The basic idea is as follows. We divide a given target object T into equal-sized units, which we designate as *portions*. Thus, we model T as a set of portions. Each portion has a unique identifier *pid* and the size of these portions is application-dependent. Then we propose an efficient methodology to compute the visibility, overlap and continuity of each candidate set in terms of portions. We refer to this framework as Portion-Based Framework (PBF).

Intuitively, PBF provides opportunities for efficient determination of top-k candidate sets as well as for efficient computations of visibility, overlap and continuity for each candidate set. This is because by considering a given target object T as a set of *pid*s, the computation of the visible regions of T from a given query location becomes equivalent to computing the corresponding set of visible *pid*s. Furthermore, the visibility of T from a combination of query locations can be computed as the union of the corresponding sets of the visible *pid*s of T. Thus, we are essentially replacing complex and computationally expensive spatial operations by set-based operations, which are typically faster by several orders of magnitude. In a similar vein, computations of overlap and continuity are based on set-based operations, which are also faster by several orders of magnitude as compared to that of complex spatial operations.

We model the association of each portion and the corresponding query locations from which that portion is visible as a **transaction**. Thus, we form a set of **portion transactions**. Hence, the problem of extracting candidate sets becomes the problem of extracting combinations of query locations from the set of portion transactions. Thus, we propose a pattern-based extraction method by exploiting an overlap-related pruning heuristic, which we shall discuss now.

Incidentally, maximum overlap threshold constraint satisfies the *sorted closure property* (Liu et al. 1999). Consider a candidate set $c_i = \{q_i, q_j, \ldots, q_n\}$, where the query locations are sorted in descending order of visibility of T. If $overlap(\{q_i, q_j\}, T)$ fails to satisfy the maximum overlap threshold constraint, any combination (of query locations) containing $\{q_i, q_j\}$ cannot possibly satisfy the constraint. Hence, we can avoid generating any further candidate sets comprising $\{q_i, q_j\}$.

We use this heuristic in our proposed approach for effective pruning as follows. First, we sort all of the query locations in descending order of their values of visibility of T. Then, starting from individual query locations, we continue to generate candidate sets (of query locations) of progressively larger sizes, while using the pruning heuristic to avoid unnecessary candidate set generation. After all the candidate sets have been generated, we sort the candidate sets in descending order of *visibility*. In case of ties, the candidate set with lower amount of *overlap* is ranked higher. In case of multiple candidate sets with same values of *visibility* and *overlap*, ties are resolved based on number of query locations in the set i.e., the one with lower number of query locations is ranked higher. Thus, we obtain a top-k ranking of the candidate sets.

4.2 MLV query processing under the portion-based framework

Now we discuss our proposed MLV query processing approach as follows:

1. Portion-based framework (PBF)
2. Determining the top-k candidate sets of a given MLV query using PBF

4.2.1 Portion-based framework (PBF)

We first explain the concept of $portions$. The target object T is divided into a set P of n_p equal-sized units, which we designate as portions. Thus, $P = \{p_1, p_2, \ldots, p_{n_p}\}$, where p_i represents a given portion with unique identifier pid_i. For any pair of p_i and p_j, $p_i \cap p_j = \varnothing$. Moreover, $p_1 \cup p_2 \cup \ldots \cup p_{n_p} = T$.

For simplicity, the notion of a portion in 2D is as follows. As the visible area of T in 2D forms a line segment, a portion in 2D is part of a line segment. Thus, we denote p_i as a pair of two points. Portion p_i is represented as a line segment with the pair of two coordinates. So, $p_i = \langle (x_{i_1}, y_{i_1}), (x_{i_2}, y_{i_2}))\rangle$, where (x_{i_1}, y_{i_1}) is the start point of p_i, represented by $sp(p_i)$ and (x_{i_2}, y_{i_2}) is the end-point of p_i, represented by $ep(p_i)$. So, $p_i = \langle sp(p_i), ep(p_i)\rangle$. Notably, the definition of a portion can also be extended to 3D albeit with some modifications.

Algorithm 1 depicts the steps of dividing T into n_p portions. We approximate T by its MBR. We divide each of the four sides of T into portions (see Lines 4–6 and Lines 8-10). The *pids* are assigned to portions by adding 1 to the preceding *pid*, starting from 0.

Algorithm 1 : Division_of_T(T,n_p)

Input: Target object T, number of portions n_p
Output: Set P of portions of target object T

side in $T \leftarrow \langle$ *sp(s)*, *ep(s)*$\rangle$, *sp(s)*, *ep(s)* are start and end-points of side *s*
Portion size $PS \leftarrow T/n_p$

```
1. Initialize P to empty set; (cur.x,cur.y) to (sp(s).x,sp(s).y)
2. for each side ⟨sp(s), ep(s)⟩ in T do
3.     if sp(s).x = ep(s).x then
4.         while cur.y <= ep(s).y do
5.             P = P ∪ ⟨(cur.x,cur.y),(cur.x,cur.y+PS)⟩
6.             (cur.x,cur.y) = (cur.x,cur.y+PS)
7.     else
8.         while cur.x <= ep(s).x do
9.             P = P ∪ ⟨(cur.x,cur.y),(cur.x+PS,cur.y)⟩
10.            (cur.x,cur.y) = (cur.x+PS,cur.y)
11. return P
```

We discuss the visibility $vis(q_i, p_j)$ of a portion p_j from a query location q_i and the visible region $VR(q_i, T)$ of q_i. Next, we explain visibility, overlap and continuity w.r.t. portions.

(i) ***vis*(q_i, p_j)** Given a query location q_i, we consider $vis(q_i, p_j) = true$ if the entire portion p_j is visible from q_i; otherwise, $vis(q_i, p_j)$ is *false* (see Eq. 1 in Sect. 2.1).

(ii) ***VR*(q_i, T)** The visible region $VR(q_i, T)$ of T from q_i is the union of all the portions of T that are visible from q_i. Hence, $VR(q_i, T) = \{p_j | p_j \in P$ and $vis(q_i, p_j) = true\}$. To compute $VR(q_i, T) \forall q_i \in Q$, we apply the algorithm proposed in Masud et al. (2013) for computing the VRs in terms of portions. The number of portions in $VR(q_i, T)$ is impacted by obstacles. Only the obstacles, which obstruct the line of sight from q_i to p_j, impact $vis(q_i, p_j)$ where $p_i \in P$. In the approach in Masud et al. (2013), all the obstacles are indexed using an R*-tree (Beckmann et al. 1990). The obstacles, which fall in the line of sight of p_j and q_i, are extracted from the R*-tree in the order of nearest distance to p_j. This is because the nearest obstacle impacts visibility maximally, thereby enabling efficient determination of $VR(q_i)$.

(iii) ***vis*(c_i, T)** The visibility $vis(c_i, T)$ of candidate set c_i consists of n_i query locations is the ratio of number of portions visible from all of the query locations in set c_i to the total number of portions of the target object T.

$$vis(c_i, T) = \left| \bigcup_{i=1}^{n_i} (VR(q_i, T)) \right| \Big/ |P| \tag{5}$$

The range of $vis(c_i, T)$ is 0 to 1. Recall that $minV$ in Sect. 2.1 represents the user-specified minimum visibility threshold. For c_i to be the candidate set of MLV query, $vis(c_i, T) \geq minV$.

(iv) ***overlap*(c_i, T)** Let the *VR* of a candidate set c_i of query locations be $VR(c_i, T)$. Suppose we want to capture more visibility of T by adding a new query location q_j to c_i. Let the VR of q_j be $VR(q_j, T)$. Notably, adding q_j to c_i could result in overlap between $VR(c_i, T)$ and $VR(q_j, T)$. If the overlap is high, $vis(\{c_i \cup q_j\}, T)$ may not increase significantly over $vis(c_i, T)$. Hence, from a visibility perspective, adding q_j to c_i would not be interesting. In essence, adding a new query location q_j to set c_i would be interesting only if there is a minimal overlap between the visible regions of c_i and q_j.

The overlap $overlap(c_i, T)$ of c_i is defined as follows. Let $c_i = c_j \cup q_k$. Then, $overlap(c_i, T)$ is equal to the ratio of the number of portions common in $VR(c_j, T)$ and $VR(q_k, T)$ to the number of portions in $VR(q_k, T)$.

$$overlap(c_i, T) = (|VR(c_j, T) \cap VR(q_k), T|) / |VR(q_k, T|) \tag{6}$$

The range of $overlap(c_i, T)$ is 0 to 1. Recall that $maxO$ represents the user-specified maximum overlap threshold. For c_i to be the candidate set of MLV query, $overlap(c_i, T) \leq maxO$.

Regarding *continuity* among portions, given p_i, p_j from the set P, p_i and p_j form one-to-one continuity relationship iff $sp(p_i) = ep(p_j)$ or $sp(p_j) = ep(p_i)$. We consider p_i and p_j to be *continuous* if there is a transitive-closure of one-to-one continuity relationship between p_i and p_j over P. We designate the continuity for

candidate set c_i as $continuity(c_i, T)$. Thus, $continuity(c_i, T)$ is $true$ iff for every pair of portions $\{p_i, p_j\} \in VR(c_i, T)$ are continuous. Otherwise, $continuity(c_i, T)$ is $false$.

4.2.2 Determining the top-k candidate sets of a given MLV query under PBF

Given set P of the portions of T and the methods for computing $vis(c_i, T)$, $overlap(c_i, T)$ and $continuity(c_i, T)$, we shall now explain how to determine the top-k candidate sets for a given MLV query subject to the constraints of $minV$, $maxO$ and *continuity*.

The brute-force approach is as follows. (i) Compute *VRs* of each $q_i \in Q$. (ii) Generate all of the $2^n - 1$ combinations of query locations, where n is the total number of query locations. (iii) Examine each combination to determine whether it satisfies the $minV$, $maxO$ and *continuity* constraints. (iv) Rank the combinations basing on *visibility* to get top-K combinations. Thus, for n query locations, the brute-force approach requires computing $vis(c_i, T)$, $overlap(c_i, T)$ and $continuity(c_i, T)$ for each of the $2^n - 1$ combinations, thereby making it prohibitively expensive at relatively larger values of n.

To address the limitations of the brute-force approach, we shall now present an efficient approach, which models the association between portions and the corresponding query locations from where a given portion is visible as a **portion transaction**. We define a portion transaction as follows.

Definition 2 (*Portion transaction*) Consider a set Q of n query locations $\{q_i, q_j ... q_n\}$ and a set P of w portions $\{p_i, p_j, \ldots, p_w\}$ of T. S is the set of portion transactions, where each portion transaction pt_i for portion p_i is a set $Q' = \{q_j \mid q_j \in Q' \text{ and } vis(p_i, q_j) = true\}$. Therefore, portion transaction pt_i is of the form $\langle pid_i, Q' \rangle$, where pid_i is the identifier of pt_i.

Notably, the number of portion transactions formed is equal to the number of portions. We extend the notion of coverage patterns and develop an efficient approach to extract top-k candidate sets pertaining to a given MLV query. The interestingness of a coverage pattern is determined with the values of *coverage support*, *overlap ratio* and *relative frequency* (see "Appendix A" for further details). The interestingness of a candidate set is determined based on $vis(c_i, T)$, $overlap(c_i, T)$, $continuity(c_i, T)$ under PBF. Thus, a candidate set is interesting iff $vis(c_i, T) \geq minV$, $overlap(c_i, T) \leq maxO$ and $continuity(c_i, T) = true$.

Given a set S of portion transactions, our problem now is to extract top-k candidate sets c_i such that for each c_i, $vis(c_i, T) \geq minV$, $overlap(c_i, T) \leq maxO$ and $continuity(c_i, T) = true$.

Incidentally, the downward closure property (Han et al. 2007) states that if a set satisfies a given property, all of its subsets also satisfy it. The visibility measure does not satisfy the downward closure property. Although a candidate set may satisfy the $minV$ constraint, all of its subsets need not necessarily satisfy it.

Moreover, the overlap measure also does not satisfy the downward closure property if the candidate sets are unordered sets of query locations. However, if the candidate sets are ordered, the overlap measure satisfies the downward closure property. This

Algorithm 2 : Compute_*MLV*(*T, Q, O*)

Input: Target object T, set Q of query locations, set O of obstacles
Output: *Top-K MLV candidate sets*

1. P = Divison_of_T(T, n_p)
2. S = Generate_Portion-Transactions(P, Q, O)
3. *MLV top-k candidate sets*=Levelwise_MLV($S, minV, max\,O$)
4. **return** *MLV top-k candidate sets*

Algorithm 3 : Generate_Portion-Transactions(P, Q, O)

Input: Set P of portions, set Q of query locations, set O of obstacles
Output: Set S of Portion transactions

1. Initialize S to empty set
2. Compute *VRs* w.r.t. portions
3. **for** each *VR(i)* in *VR* **do**
4. **for** each portion *pid* in *VR(i)* **do** { $S[pid].append(q_i)$ }
5. **return** S

property is called the sorted closure property (Liu et al. 1999). Let Q' be a set of query locations sorted in the descending order of the number of portions of the corresponding VRs. Also, let candidate sets c_i, c_j be ordered based on Q' and $c_i \subset c_j$. Based on the sorted closure property, if $overlap(c_i, T) \geq max\,O$, $overlap(c_j, T) \geq max\,O$. In essence, if a given candidate set fails to satisfy the $max\,O$ constraint, all of its supersets ordered based on Q' can be safely pruned.

Rationale: Let q_i, q_j and q_k be the query locations having $VR(q_i, T) \geq VR(q_j, T), \ldots, \geq VR(q_k, T)$. If $overlap(\{q_i \cup q_k\}, T) \geq max\,OR$, $overlap(\{q_i \cup q_j \cup q_k\}, T) \geq max\,OR$, because

$$\frac{|(VR(q_i, T)) \cap (VR(q_k, T))|}{|VR(q_k, T)|} \leq \frac{|(VR(q_i, T) \cup VR(q_j, T)) \cap (VR(q_k, T))|}{|VR(q_k, T)|}.$$

Algorithm 2 depicts the proposed approach. In Line 1, we invoke Algorithm 1 for dividing T into portions. In Line 2, we invoke Algorithm 3 for generating the portion transactions. In Line 3, using the set of portion transactions, we determine the top-k candidate sets for a given MLV query by applying level-wise pruning-based pattern extraction approach as presented in Algorithm 4.

Algorithm 3 depicts the generation of portion transactions. The inputs are set P of portions of T, set O of obstacles and set Q of query locations. For generating set S of portion transactions, we extract $VR(q_i, T)$ for each $q_i \in Q$. We form transactions of the form $\langle pid_i, Q' \rangle$ by associating p_i and Q', where Q' contains all q_i such that $vis(q_i, p_i) = true$. For each portion with portion id pid_k in $VR(q_i, T)$ of each query location q_i, q_i is added to portion transaction with id as pid_k (see Lines 3–5).

Algorithm 4 depicts the generation of candidate sets. The input is a set S of portion transactions. Like any pattern mining method, we can propose level-wise approach (Srinivas et al. 2012) or pattern-growth approach (Gowtham Srinivas 2015). This work uses the level-wise approach. The algorithm is given in Algorithm 4. We apply the level-wise approach by pruning based on the sorted closure property of the overlap

Algorithm 4 : Levelwise_MLV(S, $minV$, $maxO$)

Input: Set S of Portion transactions, $minV$, $maxO$
Output: *Top-k MLV candidate sets*
$Q_i \leftarrow$ *Set of i-length query locations.*
$C_i \leftarrow$ *Set of candidate i-length query locations.*

1. Initialise MLV to empty set; Initialise i to 1
2. Sort Q in descending order according to *VRs*
3. $C_2 = Q \bowtie Q$
4. **while** C_{i+1} is not empty **do**
5. **for** each c in C_{i+1} **do**
6. **if** *overlap(c, T)* $\leq maxO$ **then** { $Q_{i+1} = Q_{i+1} \cup c$ }
7. **if** *vis(c, T)* $\geq minV$ & *continuity(c, T)* = *true* **then** { $MLV = MLV \cup c$ }
8. $i = i$+1; $C_{i+1} = Q_i \bowtie Q_i$
9. Sort *MLV* in descending order according to *vis*, *overlap*
10. **return** *MLV top-k candidate sets*

measure. We identify the individual query locations and extend them to progressively larger sets of locations as long as those sets satisfy the constraints.

First, all the query locations are sorted based on size of the corresponding *VRs* (see Line 2). The sets are extended one query location at a time and are tested against the required threshold conditions. The sets, which fail to satisfy overlap measure, are pruned away. The other sets are extended to the next level (see Line 6). Moreover, the patterns, which satisfy the $minV$ and *continuity* constraints, are included in the candidate sets (see Line 7). Extraction of candidate sets terminates when no further successful extensions are found (see Line 4). Extracted candidate sets are sorted based on *visibility* (computed based on Eq. 5) for returning the top-k candidate sets see Line 9. In case of ties, the candidate set with less *overlap* (computed based on Eq. 6) is ranked higher. In case of multiple candidate sets with same values of *visibility* and *overlap*, ties are resolved such that the set with lower number of query locations is ranked higher.

Computation of visibility, overlap and continuity are relatively fast set-based union and intersection operations. Complexity for level-wise approach to extract the candidate sets of the MLV query from portion transactions database is same as that of the complexity of any *apriori* based approach (Agrawal and Srikant 1994).

Refer to "Appendix B" for illustrative example of the proposed approach.

5 Performance evaluation

The experiments were conducted on a core i5-4200U CPU @ 1. 60GHz x 4 PC with 4GB RAM running Ubuntu 14.04. We implemented the approaches in C++. We used two real datasets, namely the British Ordinance Survey dataset (Masud et al. 2013) (containing 5000 spatial objects) and the Boston Downtown dataset (Gangumalla 2019). (containing 10000 spatial objects). In these datasets the spatial objects are represented as 3D rectangles. In our experiments, we consider 2D objects; hence, we set the z-axis values of all the objects to 0. Among all of these objects, we randomly

Table 1 Parameters of the performance study

Parameter	Default	Variations
No. of query locations (N_Q)	25	20, 40, 60, 80, 100
No. of portions (N_P)	400	100, 200, 300, 500, 600
Min. visibility threshold ($minV$)	0.5	
Max. overlap threshold ($maxO$)	0.3	0.1, 0.5, 0.7, 0.9, 1.0

selected a single object as the target object. We divided the dataset into 2×2 grid cells and generated the query locations randomly in space such that the number of query locations is proportional to the total number of objects in each of these grid cells. The link to the code for the implementation is provided in the footnote.[1]

All obstacles are indexed by an R*-tree with the disk page size fixed at 4KB. We assume that each R*-tree node fits in a disk page i.e., each R*-tree node access corresponds to a disk I/O. Each rectangle was represented using two (x,y) coordinates; hence 16 bytes were used to represent each of the input rectangles with 4 bytes for each child pointer. We used a branching factor of 32, thereby resulting in (32*20) i.e., 640 bytes. Hence, the maximum space consumption of each R*-tree node is 640 bytes.

We selected the performance study parameters to closely reflect real-world scenarios based on existing works (Masud et al. 2013). Table 1 summarizes the performance study parameters. The performance metrics are processing time *PT* of processing MLV queries, the number N_{MLV} of candidate sets to be examined for processing an MLV query and the maximum achieved value of visibility *Vis*. Notably, *PT* concerns the time required for processing the MLV query once the disk pages have already been loaded into main memory. Thus, *PT* does not include the disk I/O cost. This is because our approach works on the portion transactions database, which can easily fit in the main memory. Furthermore, the retrieval of disk pages to create the portion transactions database is an *offline* process, which is not executed at run-time.

N_{MLV} represents the number of candidate sets that are to be examined for processing an MLV query. As we shall see shortly, the pruning heuristics used by our proposed approach preclude the need for examining every possible combination of query locations. This is in contrast with the reference approach, where all of the $(2^n - 1)$ possible combinations of query locations need to be examined. (Here, n is the total number of query locations.) *Vis* is the value of visibility for the top candidate set, when all of the candidate sets of a given MLV query are sorted in descending order of visibility.

Recall that given a target object T and a set of query locations, the MV query scheme (Masud et al. 2013) (discussed in Sect. 2) determines the ranked list of individual query locations that maximize the visibility of T in the presence of obstacles. However, the MV query scheme does not consider combinations of query locations for maximizing the visibility of T. Hence, as reference, for the purpose of meaningful comparison, we adapted the MV query scheme as follows. First, using the MV query scheme, we

[1] https://github.com/lakshmi9414/Multi-location-Visibility-Queries-Code.git.

compute the visibility regions (VRs) in terms of portions for each query location. Second, from the computed VRs, we compute the values of visibility, overlap and continuity for each combination of query locations. Third, the combinations (of query locations), which satisfy the visibility and overlap threshold values, are selected as the candidate sets for the given MLV query. We shall henceforth refer to this scheme as portion-based MV (pMV). In contrast with our proposed approach, since pMV does not use any heuristics for pruning, pMV needs to examine $(2^n - 1)$ combinations of query locations for answering an MLV query.

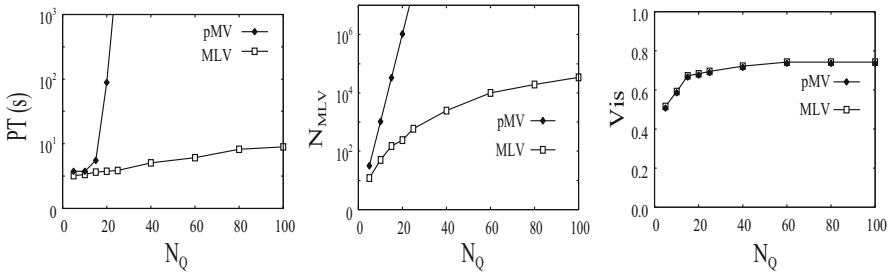
We ran each experiment at least 10 times and the results represent the average of those runs. The confidence interval of our results is 95%.

5.1 Varying the number N_Q of query locations

Figure 2 depicts the results for the British Ordinance Survey dataset. As N_Q increases, the results in Fig. 2a indicate that PT increases for both approaches. This occurs because with increase in N_Q, an increased number of candidate sets need to be examined, as shown by the results in Fig. 2b. However, observe that with increase in N_Q, the increase in PT is significantly lower for MLV than in the case of pMV. This occurs because the efficient pruning performed by MLV (based on sorted closure property of overlap ratio) leads to a significantly lower number of combinations of query locations (i.e., candidate sets) that need to be examined, as indicated by the results in Fig. 2b. On the other hand, pMV requires all possible combinations of query locations to be examined, thereby explaining why pMV explodes with increase in N_Q.

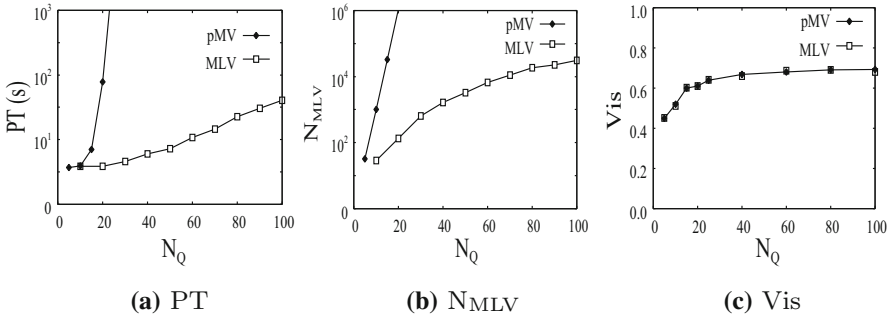
The results in Fig. 2c indicate that both the approaches behave the same way since the final output for both the approaches is same. The results indicate that as N_Q increases, *Vis* increases because an increased number of query locations implies higher visibility of the target object T as each of these query locations contributes to the visibility of T. However, both approaches eventually reach a saturation point essentially due to the presence of obstacles.

We observed similar trends in the results for the Boston Downtown dataset, as depicted in Fig. 3. The values differ due to difference in dataset sizes.



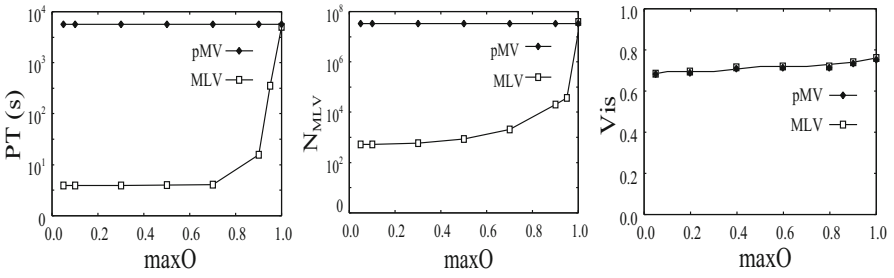
(a) PT **(b)** N_{MLV} **(c)** Vis

Fig. 2 Varying the no. of query locations for British ordinance survey dataset



(a) PT **(b)** N_{MLV} **(c)** Vis

Fig. 3 Varying the number of query locations for Boston downtown dataset



(a) PT **(b)** N **(c)** Vis

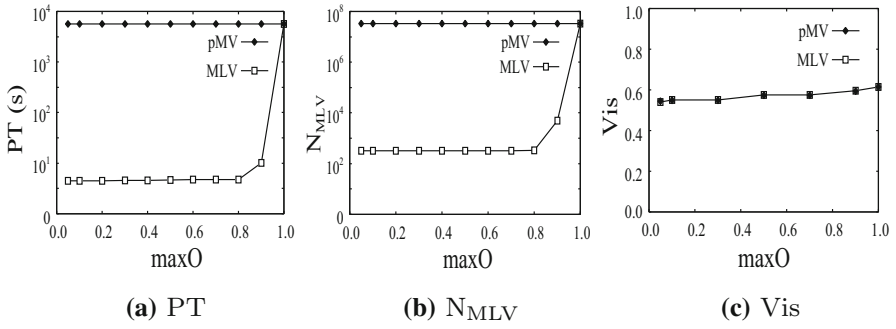
Fig. 4 Varying max. overlap threshold for British ordinance survey dataset

5.2 Varying the maximum overlap threshold *maxO*

Figure 4 depicts the results. When the value of $maxO$ is low (e.g., $maxO = 0.1$), the implication is that more constraints are being placed on the MLV query results. However, as $maxO$ increases, the constraints on the MLV query results are progressively being relaxed, thereby implying that more combinations potentially qualify as candidates for the MLV query results; hence, N_{MLV} also increases. From the results in Fig. 4b, observe that beyond values of $maxO = 0.8$ to 0.9, N_{MLV} increases substantially until at $maxO = 1.0$, the $maxO$ constraint is completely nullified. Detailed investigation of the experimental logs revealed that PT also increases albeit slightly with increase in $maxO$.

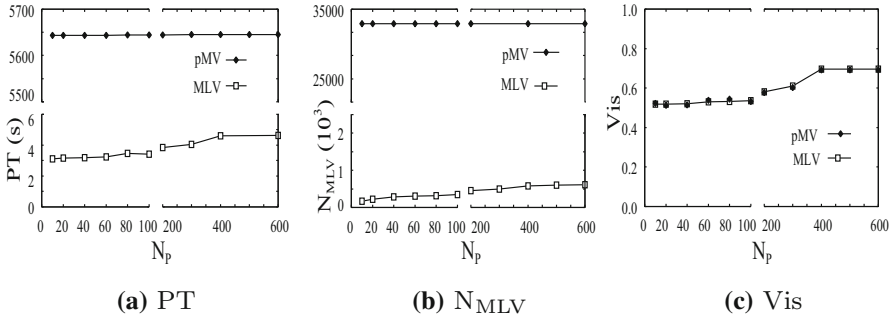
The results in Fig. 4c indicate that even when we apply very strict $maxO$ constraint (i.e., when $maxO < 0.1$), we still get some MLV query results that are close to the maximum possible *Vis* of 0.8. *Vis* is upper-limited to 0.8 due to the presence of obstacles. MLV significantly outperforms pMV because a considerable number of candidate sets get pruned away due to the sorted closure property of the $maxO$ measure. pMV shows comparable performance w.r.t. variations in $maxO$ as it does not incorporate the $maxO$ constraint.

We observed similar trends in the results for the Boston Downtown dataset, as depicted in Fig. 5. The values differ due to difference in dataset sizes.



(a) PT **(b)** N_{MLV} **(c)** Vis

Fig. 5 Varying max. overlap threshold for Boston Downtown dataset



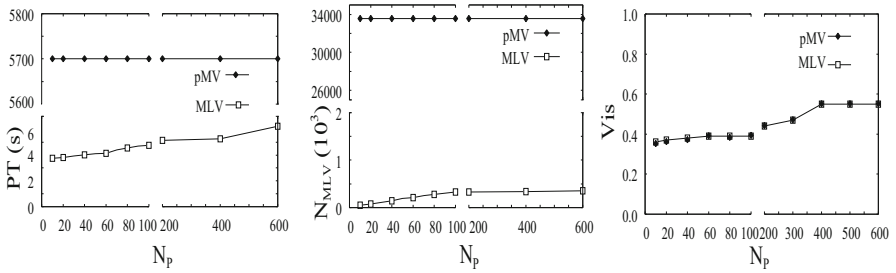
(a) PT **(b)** N_{MLV} **(c)** Vis

Fig. 6 Varying the number of portions for British ordinance survey dataset

5.3 Varying the number of portions

Figure 6 depicts the results of varying the number N_P of portions for the British Ordinance Survey dataset. As N_P increases, the number of portion transactions increases. Hence, the number of combinations to be examined also increases, thereby increasing the values of N_{MLV} as well as PT. However, when the value of N_P exceeds 400, N_{MLV} as well as PT exhibit a saturation effect. This is due to the effect of the pruning-based strategy of MLV becoming more pronounced because there are more combinations, which provide increased opportunities for pruning. Investigation of experimental logs revealed that *Vis* provided by MLV was 0.5 when the number of portions was 10. The results in Fig. 6c indicate that both the approaches behave comparably since their final output is the same. As the results in Fig. 6c show, *Vis* increases significantly from 0.5 to 0.8. However, beyond values of $N_P = 400$, visibility exhibits a saturation effect as it is upper-limited by the presence of obstacles.

We observed similar trends in the results for the Boston Downtown dataset, as depicted in Fig. 7. The values differ due to difference in dataset sizes.



(a) PT **(b)** N_{MLV} **(c)** Vis

Fig. 7 Varying the number of portions for Boston Downtown dataset

6 Discussion

In this section, we discuss some of the limitations w.r.t. the implementation of our proposed approach as well as some observations. Limitations concern how to set the values of the parameters such as the number of portions as well as the threshold values of $minV$ and $maxO$. Hence, we shall now discuss some guidelines for setting the parameters of our proposed approach.

6.1 Determination of number n_p of portions

If n_p is large, visibility can be computed at a finer granularity albeit at the cost of a higher number of portion transactions and consequently, higher MLV query processing overhead. Conversely, if n_p is small, the number of portion transactions would become relatively low; however, the approach may not achieve the required visibility. In essence, the value of n_p is application-dependent. As a heuristic, if the number of obstacles is large, we should set the value of n_p to be relatively high; this allows the target object to be viewed at *finer granularity*. If the number of obstacles is low, we should set the value of n_p to be relatively low.

6.2 Specifying the threshold values of *minV* and *maxO* for an MLV query

Recall that the goal of the MLV query is to *ideally* achieve the maximum possible visibility of the target object, while minimizing the overlap to zero. Hence, as a heuristic, we could start with the value of $minV$ equal to 1 and then progressively keep decreasing the value of $minV$ until a desired number of candidate sets can be obtained for the MLV query. Moreover, we could start with $maxO$ equal to 0 and then progressively keep increasing the value of $maxO$ until a desired number of candidate sets can be obtained. Furthermore, the values of $minV$ and $maxO$ need to be *experimentally* set in tandem with each other, especially given the wide gamut of real spatial datasets.

6.3 Observations about our proposed approach

Our approach is heuristic-based and approximate. The approximation of the spatial region with MBR enabled the formation of portion transactions and the efficient identification of query combinations. However, as we approximated spatial region with MBR, the exactness of the result of *MLV* query is also approximate. We believe that this result can be used after further refinements to find the exact solution based on the application requirement. This will be investigated as part of future work. Furthermore, since MBRs consider the minimum bounding rectangle corresponding to a given spatial object, approximating spatial objects using MBRs generally does not have a significant impact on the accuracy.

In our proposed scheme, the number of query locations is close to the optimum. This is because we sort the query locations based on the number of visible portions and then start identifying query combinations starting from the highest visible query location; hence, query locations are added incrementally. There could also be alternative solutions, which could further improve the accuracy by considering portions of varied sizes. Other alternative solutions could involve using computational geometry algorithms for addressing finer-grained spatial intersection/union operations; however, such solutions would incur prohibitively expensive computational cost.

7 Conclusion

Visibility computation is important in spatial databases for realizing various applications. While existing works address the problem of identifying individual query locations for maximizing the visibility of a given target object T, we consider the problem of determining multiple query locations for maximizing the visibility of T. We have introduced the Multi-Location Visibility (MLV) query and proposed a portion-based transactional framework and coverage pattern mining based algorithm to process MLV queries. Our performance evaluation with real datasets demonstrates the effectiveness of the proposed scheme in terms of query processing time, pruning efficiency and target object visibility w.r.t. a recent existing scheme. As visibility computation is an active research area, due to efficiency and flexibility, the proposed portion-based transactional framework provides the scope to explore efficient approaches for improving robotic and drone based surveillance applications.

Appendix: A coverage patterns

The concept of *coverage* has been used to the solve set cover problem in set theory (Chvatal 1979) and node cover problem in graph theory (Garey et al. 1974). By extending the notion of coverage, a model for extracting the knowledge of coverage patterns (CPs) (Srinivas et al. 2012) was proposed to extract coverage value of distinct sets of data items or patterns from a transactional database.

Now we shall explain the concept of coverage patterns. Given a transactional database D, each transaction is a subset of a set I of m items $\{i_1, i_2, i_3, \ldots, i_m\}$.

Table 2 Transactional database

TID	ITEMS	TID	ITEMS
1	a, b, c	6	a, e, g, h
2	b, c, e	7	b, c, d, e
3	e, g	8	a, d, h
4	b, h	9	a, c, h
5	c, d, g	10	c, f

A pattern P is a subset of items in I i.e., $P \subseteq I$ where $P = \{i_p, i_q, \ldots, i_r\}$ and $1 \leq p \leq q \leq r \leq m$ and p, q, r are integers. T^{i_r} is the set of transactions in which item i_r is present, hence $|T^{i_r}|$ is the cardinality of T^{i_r}.

The fraction of transactions containing a given item i_r is designated as *Relative Frequency* $RF(i_r)$ of i_r. $RF(i_r) = |T^{i_r}|/|D|$. A given item is considered as *frequent* if its relative frequency is greater than that of a threshold value, which we designate as $minRF$. *Coverage Set* CSet(P) of a pattern $P = \{i_p, i_q, \ldots, i_r\}$ is the set of all the transactions that contain at least one item from the pattern P i.e., $CSet(P) = T^{i_p} \cup T^{i_q}, \ldots, \cup T^{i_r}$. *Coverage Support* CS($P$) is the ratio of the size of CSet(P) to the size of D i.e., $CS(P) = |CSet(P)|/|D|$.

In order to add a new item to the pattern P such that the coverage support increases significantly, the notion of overlap ratio is introduced. (This is possible in the case when the number of transactions, which are common to the new item and the pattern P, is low.) *Overlap ratio* $OR(P)$ of a pattern P is the ratio of the number of transactions that are common between CSet($P - i_r$) and T^{i_r} to the number of transactions in T^{i_r}, i.e., $OR(P) = (|CSet(P - i_r) \cap (T^{i_r})|)/(|T^{i_r}|)$. A high CS value indicates more number of transactions and a low OR value means less repetitions among the transactions. A pattern is *interesting* if its CS is greater than or equal to the user-specified minimum CS threshold value (i.e., $minCS$) and its OR is less than or equal to user-specified maximum OR threshold value (i.e., $maxOR$). Given $minRF$, $minCS$ and $maxOR$, pattern $P = \{i_p, i_q, \ldots, i_r\}$ is regarded as a *coverage pattern CP* if $RF(i_k) \geq minRF \, \forall i_k \in P, CS(P) \geq minCS$ and $OR(P) \leq maxOR$.

For example, consider D depicted in Table 2 to understand the notion of coverage patterns. In D, the set of items $I = \{a, b, c, d, e, f, g, h\}$ and the number of transactions i.e., $|D| = 10$ with TIDs $= 1, 2, 3, \ldots, 10$. Let $minRF = 0.3$, $minCS = 0.6$, $maxOR = 0.8$. Consider pattern $P = \{b, c, d\}$; $T^b = \{1, 2, 4, 7\}$, $T^c = \{1, 2, 5, 7, 9, 10\}$, $T^d = \{5, 7, 8\}$. $|T^b| = 4$, $|T^c| = 6$, $|T^d| = 3$; $RF(b) = 4/10 = 0.4$, $RF(c) = 6/10 = 0.6$, $RF(d) = 3/10 = 0.3$. CSet(P) $= \{1, 2, 4, 5, 7, 8, 9, 10\}$. $RF(b) \geq minRF$, $RF(c) \geq minRF$, $RF(d) \geq minRF$; $CS(P) = 8/10 = 0.8 \geq minCS$; $OR(P) = (|(T^b \cup T^c) \cap (T^d)|)/(|T^d|)$; $OR(P) = 2/3 = 0.6 \leq maxOR$. Hence, $\{b, c, d\}$ is a coverage pattern.

An apriori-like approach to extract complete sets of CPs from transactional database has been proposed in Srinivas et al. (2012). A pattern growth-like approach (Gowtham Srinivas 2015) has also been proposed to extract the complete set of CPs in an efficient manner.

Appendix: B Illustrative example of our approach

Figure 8 depicts an illustrative example of the proposed approach. Consider the target object $ABCD$ having four sides $\{AB, BC, CD, DA\}$ with query locations (q_1 to q_{10}) and obstacles (o_1 to o_4). $ABCD$ is divided into portions (p_1 to p_{10}).

We compute the *VRs* corresponding to each query location, as shown in Table 3. Observe that p_1 is visible from query location q_1, p_5 is visible from $\{q_4, q_6\}$ and p_7 is visible from $\{q_6, q_7, q_8\}$. Now using the computed *VRs*, we generate portion transactions. Hence, the query location in portion transaction p_1 is q_1. Similarly, in the portion transaction p_7, the query locations are $\{q_6, q_7, q_8\}$, as depicted in Table 4.

Now let us consider a candidate set $c_i = \{q_4, q_6, q_9\}$. Let the values of $minV$ and $maxO$ be 0.7 and 0.4 respectively. $VR(q_4, T) = \{p_3, p_4, p_5\}$, $VR(q_6, T) = \{p_5, p_6, p_7\}$, $VR(q_9, T) = \{p_8, p_9\}$. $vis(c_i, T) = |VR(q_4, T) \cup$

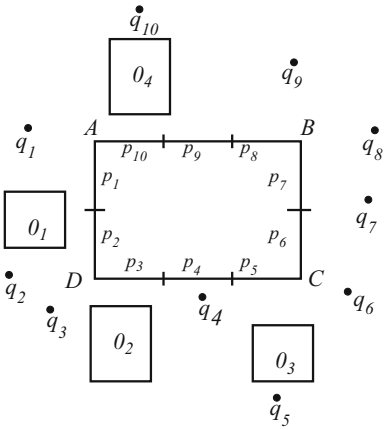


Fig. 8 Illustrative example of our approach

Table 3 Visible regions (*VRs*) from query locations

q_i	$VR(q_i, T)$	q_i	$VR(q_i, T)$
q_1	p_1	q_6	p_5, p_6, p_7
q_2	–	q_7	p_6, p_7
q_3	p_1, p_2	q_8	p_7
q_4	p_3, p_4, p_5	q_9	p_8, p_9
q_5	–	q_{10}	–

Table 4 Portion transactions

pid	Q'	pid	Q'
p_1	q_1, q_3	p_6	q_6, q_7
p_2	q_3	p_7	q_6, q_7, q_8
p_3	q_4	p_8	q_9
p_4	q_4	p_9	q_9
p_5	q_4, q_6	p_{10}	–

$VR(q_6, T) \cup VR(q_9, T)|/|S| = \{p_3, p_4, p_5, p_6, p_7, p_8, p_9\}/10$ i.e., $vis(c_i, T) = 0.8$. $|VR(q_4, T)| \geq |VR(q_6, T)| \geq |VR(q_9, T)|$. Thus, $overlap(c_i, T) = |(VR(q_4, T) \cup VR(q_6, T)) \cap VR(q_9, T)|/|VR(q_9, T)| = 0/2$ i.e., $overlap(c_i, T) = 0$. Here, $continuity(c_i, T) = true$ since portions p_3 to p_9 from the *VRs* of query locations in c_i are continuous. Since $vis(c_i, T) \geq minV$, $overlap(c_i, T) \leq maxO$ and $continuity(c_i, T) = true$, c_i qualifies as a candidate set.

Now let us consider $c_i = \{q_6, q_8\}$. $VR(q_6, T) = \{p_5, p_6, p_7\}$, $VR(q_8, T) = \{p_7\}$. $vis(c_i, T) = |VR(q_6, T) \cup VR(q_8, T)|/|S| = \{p_5, p_6, p_7\}$ /10 i.e., $vis(c_i, T) = 0.3$.$|VR(q_6, T)| \geq |VR(q_8, T)|$.

Thus, $overlap(c_i, T) = |(VR(q_6, T) \cap VR(q_8, T)|/|VR(q_8, T)| = 1/1$ i.e., $overlap(c_i, T) = 1$. Here, $continuity(c_i, T) = true$ since portions p_5 to p_7 from the *VRs* of query locations in c_i are continuous. Since $vis(c_i, T)$ and $overlap(c_i, T)$ do not satisfy the $minV$ and $maxO$ constraints, c_i would not qualify as a candidate set.

References

Agrawal R, Srikant R (1994) Fast algorithms for mining association rules in large databases. In: Proceedings of VLDB, pp 487–499

Ali D, Eunus M et al (2017) Efficient processing of maximum visibility facility selection query in spatial databases. Masters Thesis, BUET

Asano T, Asano T, Guibas L, Hershberger J, Imai H (1986) Visibility of disjoint polygons. Algorithmica 1(1):49–63

Beckmann N, Kriegel HP, Schneider R, Seeger B (1990) The R*-tree: an efficient and robust access method for points and rectangles. In: Proceedings of ACM SIGMOD

Bentley JL, Ottmann TA (1979) Algorithms for reporting and counting geometric intersections. IEEE Trans Comput 28(9):643–647

Bentley JL, Wood D (1980) An optimal worst case algorithm for reporting intersections of rectangles. IEEE Trans Comput 29(7):571–577

Chvatal V (1979) A greedy heuristic for the set-covering problem. Math Oper Res 4(3):233–235

Floriani L, Magillo P (2003) Algorithms for visibility computation on terrains: a survey. Environ Plan B Plan Des 30(5):709–728

Gangumalla L (2019) Datasets used in the paper. https://github.com/lakshmi9414/Datasets.git. Accessed 20 June 2019

Gao Y, Liu Q, Miao X, Yang J (2015) Reverse k-nearest neighbor search in the presence of obstacles. Inf Sci 330:274–292

Gao Y, Zheng B (2009) Continuous obstructed nearest neighbor queries in spatial databases. In: Proceedings of ACM SIGMOD, pp 577–590

Gao Y, Zheng B, Chen G, Lee WC, Lee KCK, Li Q (2009a) Visible reverse k-nearest neighbor queries. In: Proceedings ICDE, pp 1203–1206

Gao Y, Zheng B, Lee WC, Chen G (2009b) Continuous visible nearest neighbor queries. In: Proceedings of EDBT, pp 144–155

Garey MR, Johnson DS, Stockmeyer L (1974) Some simplified NP-complete problems. In: Proceedings of ACM symposium on theory of computing, pp 47–63

Goodchild MF, Lee J (1990) Coverage problems and visibility regions on topographic surfaces. Ann Oper Res 18(1–4):175–186

Gowtham Srinivas P et al (2015) Mining coverage patterns from transactional databases. J Intell Inf Syst 45(3):423–439

Haider CMR, Arman A, Ali ME, Choudhury FM (2016) Continuous maximum visibility query for a moving target. In: Proceedings of ADC, pp 82–94

Han J, Cheng H, Xin D, Yan X (2007) Frequent pattern mining: current status and future directions. Data Min Knowl Discov 15(1):55–86

Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. SIGMOD Rec. 29(2):1–12
Irtiza Tripto N, Nahar M, Ali ME, Choudhury F, Culpepper J, Sellis T (2019) Top-k trajectories with the best view. GeoInformatica 141:1–41. https://doi.org/10.1007/s10707-019-00343-4
Liu B, Hsu W, Ma Y (1999) Mining association rules with multiple minimum supports. In: Proceedings of ACM SIGKDD, pp 337–341
Masud S, Choudhury FM, Ali ME, Nutanong S (2013) Maximum visibility queries in spatial databases. In: Proc. ICDE, pp. 637–648
Morling K (2010) Geometric and engineering drawing 3E. Routledge, London
Mount DM (2004) Geometric intersection. Citeseer
Nutanong S, Tanin E, Zhang R (2010) Incremental evaluation of visible nearest neighbor queries. IEEE TKDE 22(5):665–681
Seeds MA, Backman D (2015) Stars and galaxies. Cengage Learning, Boston
Shou L, Huang Z, Tan KL (2003) HDoV-tree: the structure, the storage, the speed. In: Proceedings of ICDE, pp 557–568
Srinivas PG et al (2012) Discovering coverage patterns for banner advertisement placement. In: Proceedings of PAKDD, pp 133–144
Tanton J (2005) Encyclopedia of mathematics. In: Tanton 2005 encyclopedia. Facts On File, inc
Tao Y, Papadias D, Shen Q (2002) Continuous nearest neighbor search. In: Proceedings of VLDB, pp 287–298
Xu H, Li Z, Lu Y, Deng K, Zhou X (2010) Group visible nearest neighbor queries in spatial databases. In: Proceedings of WAIM, pp 333–344
Zhang J, Papadias D, Mouratidis K, Zhu M (2004) Spatial queries in the presence of obstacles. In: Proceedings of EDBT, pp 366–384