Jurnal EurekaMatika

Journal homepage: https://ejournal.upi.edu/index.php/JEM

K Nearest Neighbor in the Presence of a Circle as an Obstacle

Utari Wijayanti, Nar Herrhyanto*

Program Studi Matematika, FPMIPA Universitas Pendidikan Indonesia
*Correspondence: utari@upi.edu

A B S T R A C T

Nearest Neighbour (NN) and k-Nearest Neighbour (kNN) queries, along with their various types, have found widespread applications in numerous real-world scenarios. As a result, they have garnered significant research attention over the past few decades. However, spatial queries involving curved obstacles present unique challenges due to the mathematical complexity of the curves themselves.

Most existing algorithms address this issue by approximating curved obstacles as polygons and then applying visibility graphs to determine the shortest path. While effective in some cases, this polygonization approach becomes problematic when the obstacle contains a large number of polygonal points, which significantly increases the computational burden on both the visibility graph and Dijkstra's algorithm. Alternatively, representing curved obstacles directly as mathematical equations offers a more efficient solution, with linear complexity in shortest path computation. Despite its potential, this approach has not yet been explored in the context of NN and kNN queries.

In this research, we propose a novel approach to NN and kNN queries by modeling curved obstacles using curve equations, with circular shapes as the initial focus. Through theoretical analysis, we develop pruning techniques based on spatial relationships between query points, reference points, and obstacles. The correctness of our algorithms is rigorously validated through nine original lemmas, formulated independently without reference to existing papers or books.

ARTICLE INFORMATION

History:

Submitted April 3rd, 2025
Revised April 10, 2025
Approved April 26, 2025
Available online May 1st, 2025
Published May 1st, 2025

Keywords:
Curve obstacles,
Dijkstra algorithm,
K Nearest neighbor query,
Nearest Nearest Neighbor Query,
Poligonization.

1. INTRODUCTION

The widespread use of digital maps has transformed decision-making in daily life. These maps rely on spatial database systems that support geographic information systems and spatial queries such as Nearest Neighbour (NN) and k-Nearest Neighbour (kNN)(Güting, 1994; Papadias et al., 2003; Sharifzadeh et al., 2008; Taniar & Rahayu, 2013). These queries help users locate nearby facilities, like supermarkets or ports, based on proximity and travel direction.

In many cases, spatial queries must account for obstacles, making Euclidean distance insufficient. Instead, the shortest path that avoids obstacles is computed, as explored in various studies (Cruz & Encarnação, 2012; Kapoor & Maheshwari, 1988; Lozano-Pérez & Wesley, 1979; Storer & Reif, 1994; Welzl, 1985; Zhang et al., 2005). The challenge intensifies when obstacles are curved, due to their algebraic and geometric complexity. Yet, curved obstacles—such as trees or coastlines—are common in real-world scenarios, including maritime navigation. For instance, ships may need to reroute to the nearest port during adverse weather.

Most existing approaches convert curved obstacles into polygons and apply visibility graphs (Agarwal et al., 2009; Chang et al., 2006; Chazelle et al., 1994; Chew, 1985). However, even in simple cases like disjoint disks, this method quickly becomes computationally expensive.

A common approach for handling curved obstacles is to approximate them as polygons, then apply visibility graphs followed by Dijkstra's algorithm, which has a computational complexity of $O(n^2)$, where n is the number of polygon vertices. However, this method can lead to poor accuracy and high computational cost. For example, approximating a circular obstacle as a square yields only four vertices, resulting in a 57% loss of accuracy. Increasing the polygon order (e.g., to an octagon) improves accuracy but also quadruples the computational cost due to the quadratic complexity.

An alternative approach is to represent circular obstacles using their exact curve equations, which allows shortest path computations with linear complexity, as shown by (Hershberger et al., 2022). Building on this idea, our study focuses on circular obstacles and introduces a new method for NN and kNN queries using curve-based modeling. We develop pruning strategies grounded in the spatial relationships among query points, reference points, and obstacles. To ensure the validity of our approach, we present nine original lemmas, independently derived without relying on prior literature.

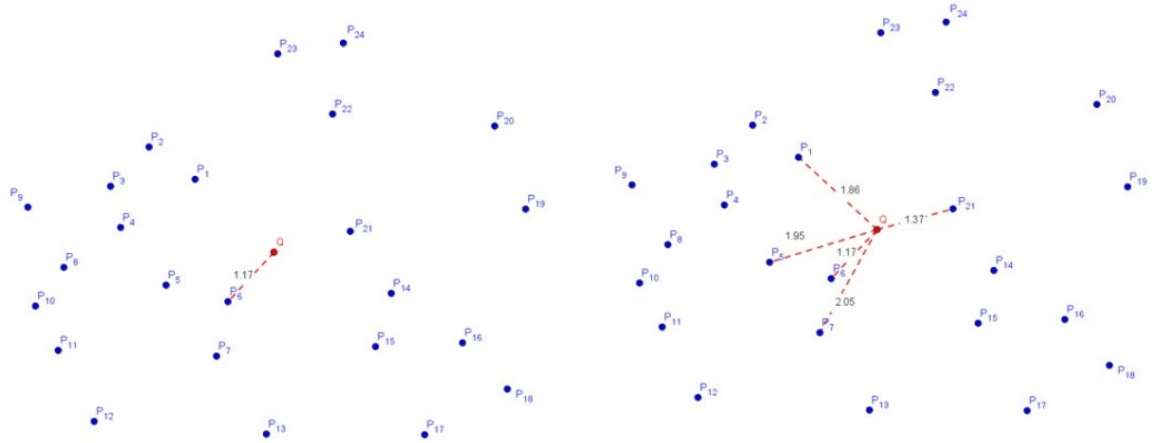
2. METHOD

2.1 Research Design

In this paper we work on the circle obstacles as a circle, then we find the shortest distance between two points by considering the circle. Lastly, we construct the pruning region for k-nearest neighbor queries and provide formal proof that no valid candidate is mistakenly excluded.

A nearest neighbor query retrieves the closest point to a given query point, as introduced by Evelyn Fix and Hodges Jr, J.L. in their project report *Discriminatory analysis-nonparametric discrimination: Small sample performance* (No. UCB11) year 1952. Its extension, the k-nearest neighbor query, returns the k closest objects to point q. These queries are widely

applied in spatial databases, data mining, image processing, and geographic information systems (Cover & Hart, 1967). Figures 1a and 1b illustrate the concept.



(a) Nearest Neighbor Query (b) k Nearest Neighbor Query

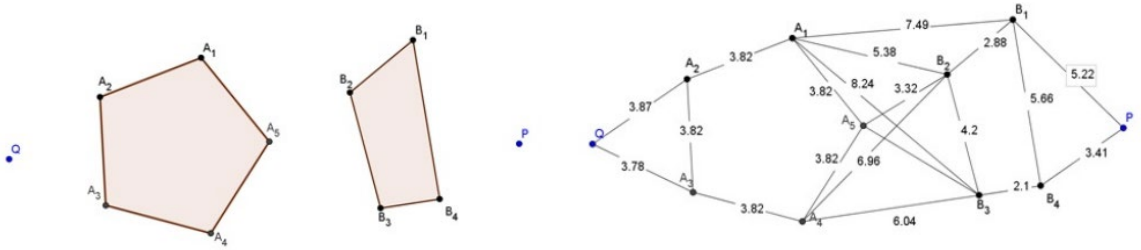
Figure 1. Nearest neighbor and k Nearest Neighbor Query

2.2 Shortest Path with Obstacles

To compute the shortest path around polygonal obstacles, the visibility graph method is widely used (Kapoor & Maheshwari, 1988; Storer & Reif, 1994; Zhang et al., 2004, 2005). This process involves converting obstacles into polygons or segment collections, then identifying all visible connections between segment points.

From the query point Q and destination P, edges are drawn to all visible segment points. Visibility means the connecting line does not intersect any obstacle. Edges are also added between visible point pairs from different polygons. Figures 2a and 2b illustrate this process.

Once the visibility graph is built, Dijkstra's algorithm is applied to find the shortest path from Q to P. The general complexity is $O(n^2)$, where n is the number of points. In simpler cases, such as paths within a triangulated simple polygon, the complexity can be linear (Guibas et al., 1987). For disjoint convex polygons with total complexity N, the runtime improves to $O(N + \frac{n}{\sqrt{\epsilon}} \log \frac{n}{\epsilon})$ (Hershberger & Suri, 1999).



a. Vertices P and Q among obstacles

a. Visibility graph of vertices P and Q with the obstacle vertices

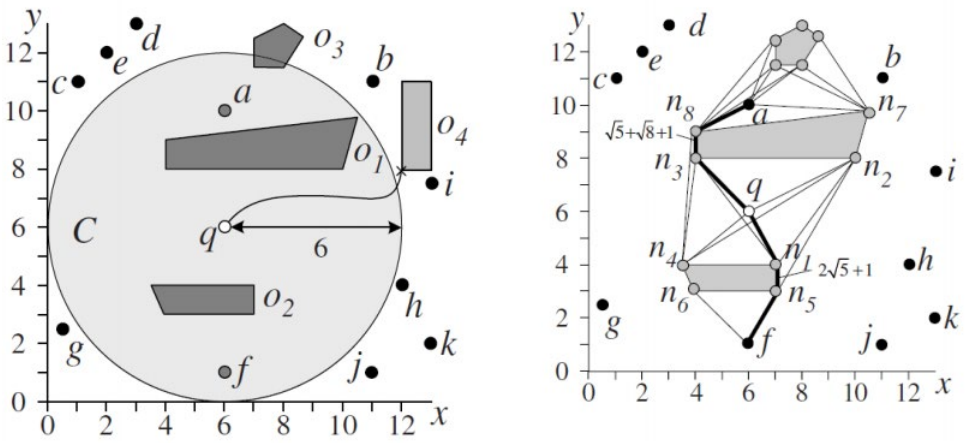
Figure 2. Visibility graph for finding shortest distance

Visibility graphs have been used broadly and applied in network optimization (Ghosh & Mount, 1991). However, when used with curved obstacles, the complexity increases significantly, resulting in poor performance (Mitchell et al., 1987). To address these

challenges, two optimization methods are proposed: Local Visibility Graph and Incremental Computation.

2.3 Local Visibility Graph

This method improves efficiency by using a range search query to find objects within obstructed distance e from query point q. The process involves retrieving candidate entities within Euclidean distance using an R-tree, identifying relevant obstacles, building a local visibility graph and removing false hits from the candidate set (Zhang et al., 2005). Figure 3 illustrates the workflow.



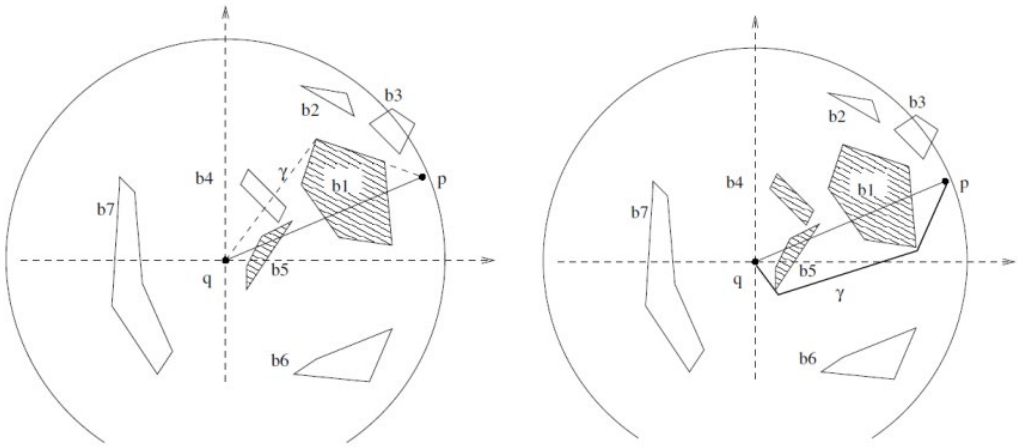
a. Query of range obstacle

b. Visibility graph which is constructed locally

Figure 3. Visibility graph for finding shortest distance

2.4 Incremental Computation of Visibility Graph

This approach builds the visibility graph step-by-step. Starting with a candidate path segment pq, if it intersects obstacles, a partial visibility graph is generated for those obstacles. The path is re-evaluated, and if new obstacles are crossed, the graph is expanded accordingly. This process repeats until no further updates are needed. Figure 4 shows the procedure.



(a) First iteration of incremental computation of VG

(b) Second iteration of incremental computation of VG

Figure 4: Incremental computation of VG

2.5 Visible Queries in Obstructed Space

Some queries use terms like "obstructed" or "visible," which relate to obstacle-aware processing but differ from our focus. These approaches consider only the points visible from the query point, ignoring invisible areas.

Examples include Visible nearest neighbor (Nutanong et al., 2007), Visible reverse nearest neighbor (Gao et al., 2009), Visible k-nearest neighbor (Y. Q. Wang et al., 2010), Visible group nearest neighbor (H. Xu et al., 2010), Continuous visible nearest neighbor (H. Xu et al., 2010), Continuous visible k-NN for moving objects (Y. Wang et al., 2014), and variants of basic visible queries (J. Xu & Güting, 2015).

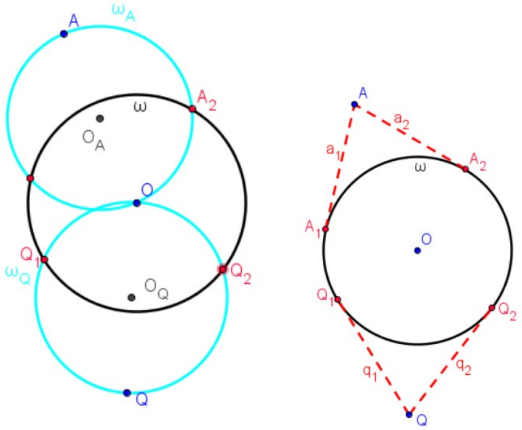
In summary, the visibility graph approach approximates curved obstacles as polygons, requiring many points for precision which is resulting in high complexity. Most existing studies follow this method. In contrast, representing obstacles with curve equations offers linear complexity, but has not yet been explored for k-nearest neighbor queries. This study proposes applying the curve equation approach to spatial queries involving curved obstacles.

3. RESULTS

3.1 Constructing Obstructed Distance Between A and Q

Given a circular obstacle ω centered at O, with query point Q and site A outside ω, the obstructed distance is computed as follows:

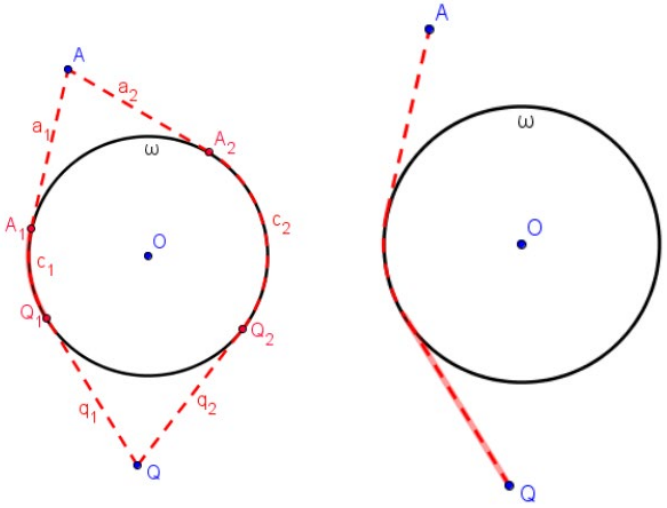
- Define midpoints OA and OQ as O_A and O_Q, respectively (Figure 5a).
- Construct circle ω_A and ω_Q centered at O_A and O_Q, passing through A and Q, resepectively (Figure 5b).
- Let line l_y pass through O and Q. Name the intersections of ω and ω_A as A_1 and A_2, they are the tangent points of A to ω. On the other hand, the intersections of ω and ω_Q are the tangent points of Q to ω. Name the point on the same half plane of l_y with A_1 and A_2 as Q_1 and Q_1, respectively (Figure 6a)
- Create segments q_1 and q_2 which connect Q with Q_1 and Q_2, respectively, also segments a_1 and a_2 which connect A with A_1 and A_2, respectively (Figure6b).
- Create circular arcs with center O which connect A_1 with Q_1, A_2 with Q_2 in counter clock wise direction in a way such that the degree of the circular arcs are less than π, name them as c_1 and c_2, respectively.
- Compute path lengths $w_{A_1} = q_1 + c_1 + a_1$ and $w_{A_2} = q_2 + c_2 + a_2$.
- The minimum of these two is the obstructed distance from Q to A, denoted as $weight(A)$.



- a. Tangent points of Q and A to circle ω b. Tangent segments
- Figure 5: Finding the tangents

3.2 Pruning Lemmas Based on Reference Site

To support the proposed pruning strategy for NN and kNN queries in obstructed spaces, we formulate nine original lemmas based on the geometric relationships between the query point, reference site, and circular obstacles. These lemmas are independently derived and serve as the theoretical foundation for validating the correctness of our pruning region.



a. Draw the alternative routes b. Measure the total length

Figure 6: Finding the route

Let ω be a circle centered at O with radius r, and N a point on ω . Rotating N by $\frac{\pi}{2}, \pi, \frac{3}{2}\pi$ yields points W, S, E. Let lines l_x and l_y pass through O and W, and O and N, respectively.

Lemma 1 Given λ_Pa circle with center Q and radius the obstructed distance of $d_O(Q,P)$. Then any point outside circle λ_P has a greater obstructed distance than $d_O(Q,P)$.
Proof: It is clear because the radius of $\lambda_P = d_O(Q,P)$, then the Euclidean distance any point outside λ_P is larger than $\lambda_P = d_O(Q,P)$. See Figure 7 for illustration.

On the next lemmas, it will be shown that every point on the halfplane of t_p which does not contain Q has obstructed distance larger than $d_O(Q,P)$. For illustration, see Fig. 8, the shading area with colour purple has obstructed distance larger than $d_O(Q,O)$. For the next three lemmas, that is lemma 2, 3 and 4 use Figure 9 as illustration.

Lemma 2 Given U the intersection between l_x and the tangent line of ω passing through T, then $UT \geq \widehat{WT}$
Proof: Let $\alpha = \angle WOT$. The length of $\widehat{WT} = r\alpha$ and the length of $UT = r\ tan\ \alpha$.
Because $tan\ \alpha \geq \alpha$ for $0 \leq \alpha \leq \frac{\pi}{2}$ than $UT \geq \widehat{WT}$.

Lemma 3 For point B on l_w, and line $l_B \parallel l_x$ through B, if C is the intersection of line TU and l_B, then $CU \geq BW + \widehat{WT}$.
Proof: If $U \neq W$ then $CU > BW$. Together with Lemma 2, we have $CU \geq BW + \widehat{WT}$

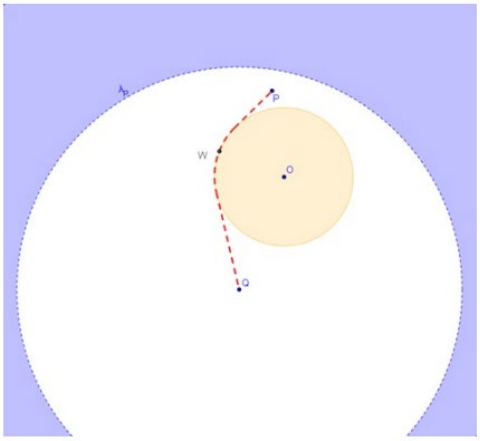


Figure 7. Circle λ_P

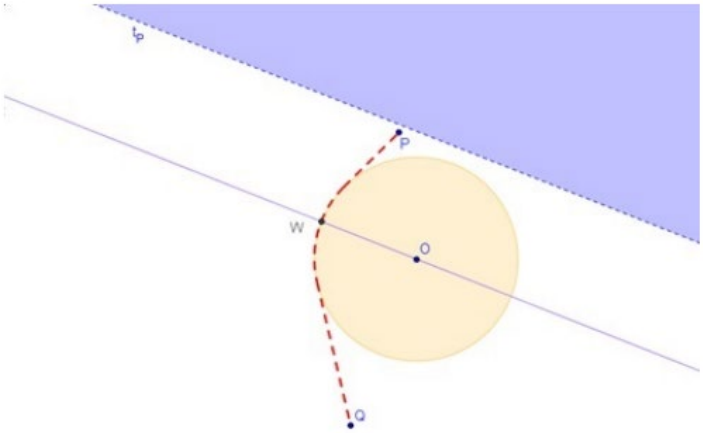


Figure 8. halfplane of t_P

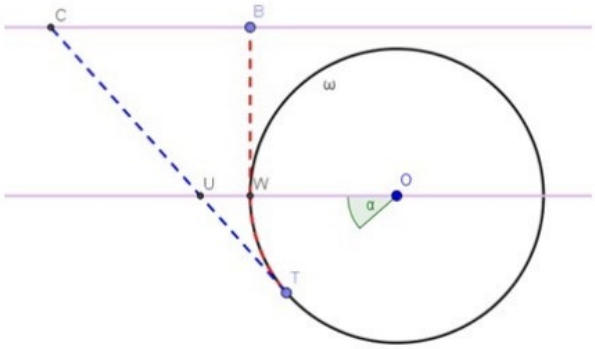


Figure 9. Evaluating the distance for tangent point below point W

Lemma 4 Let $d_O(B,Q) = d_O(P,Q)$. Then, all points above line l_B have obstructed distance greater than $d_O(P,Q)$.
Proof: This follows directly from Lemmas 2 and 3

For the next three lemmas, Lemma 5, Lemma 6 and Lemma 7, use Figure 10 as illustration

Lemma 5 Let T be a point in region $\widehat{NW}$, and F on line l_w such that $TF \parallel l_x$ then
$$FW < \widehat{TW}$$
Proof: It is clear that $FW < TW < \widehat{TW}$.

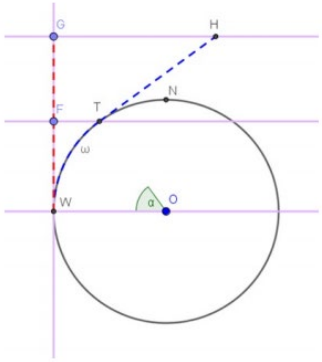


Figure 10. Evaluating the distance for tangent point above point W

Lemma 6 Let G be a point on l_w with $GW > FW$. Let $l_G \parallel l_x$ pass through G, and let H be the intersection of l_G with the tangent from T. Then $GW \leq HT + < \widehat{TW}$.
Proof: If $T \neq F$, then $HT > GF$. Combined with Lemma 5, the inequality follows.

Lemma 7 If $d_O(G, Q) = d_O(P, Q)$, then all points above line l_G have greater obstructed distance than $d_O(P, Q)$.
Proof: Follows from Lemmas 5 and 6.

Lemma 8 The union of the region outside circle λ_P and the half-plane of line t_P that does not contain Q forms the pruning region for reference point P.
Proof: From Lemma 1, points outside λ_P have greater obstructed distance. Lemmas 2–7 show the same for points above t_P. Thus, their union defines the pruning region.

Lemma 9 Given query point Q, reference point P, and obstacles $\omega_1, \omega_2, \ldots, \omega_k$, let t_{P_i} be the pruning line for each ω_i. Then the union of regions outside λ_P and all half-planes t_{P_i} not containing Q forms the pruning region. See Figure 11 for illustration.
Proof: Applying Lemma 8 to each ω_i yields a unique circle λ_P and pruning lines $t_{P_1}, t_{P_2}, \ldots t_{P_k}$. Their union defines the complete pruning region.

4. CONCLUSION

This study addresses the limitations of traditional visibility graph methods in handling curved obstacles, where polygonal approximations often lead to increased computational complexity and reduced geometric precision. To overcome these challenges, we propose a novel framework that models curved obstacles using their exact equations, with a specific focus on circular shapes. The advantage of computating obstructed distances of curve-based representation is linear complexity.

Building on this foundation, we introduce a pruning strategy for NN and kNN queries that significantly reduces the search space. The pruning mechanism is guided by geometric relationships among query points, reference sites, and circular obstacles. To ensure the correctness of this approach, we formulate nine original lemmas—each derived independently without reference to existing literature. These lemmas define the boundaries of pruning regions and guarantee that no valid candidate is mistakenly excluded during query processing.

The results demonstrate that curve-based modeling avoids the overhead of polygonal approximations, making it suitable for real-time or large-scale spatial queries. Pruning regions constructed from visibility constraints, tangent evaluations, and obstructed distance metrics effectively narrow the candidate set. Lastly, the mathematical validation through original lemmas reinforces the reliability and soundness of the proposed method.

Together, these contributions offer a scalable and theoretically grounded framework for spatial queries in environments with curved obstacles. The proposed method enhances computational efficiency and opens new avenues for applying curve-based spatial reasoning in domains such as geographic information systems (GIS) and robotics navigation. Future research may extend this model to more complex curves (e.g., ellipses or splines), dynamic obstacle scenarios, or higher-dimensional spatial spaces.

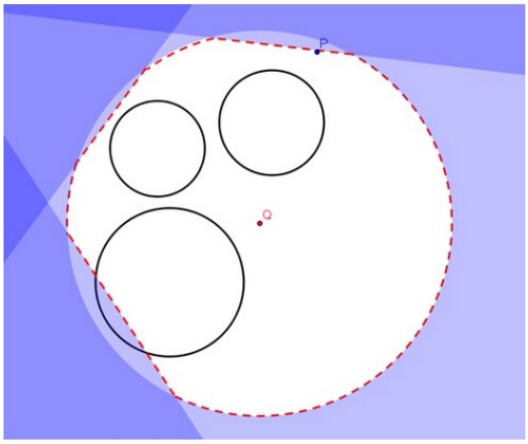


Figure 11. Pruning region with the presence of circles

5. REFERENCES

Agarwal, P. K., Sharathkumar, R., & Yu, H. (2009). Approximate Euclidean shortest paths amid convex obstacles. *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*.

Chang, E. C., Choi, S. W., Kwon, D. Y., Park, H., & Yap, C. K. (2006). Shortest path AMIDST disc obstacles is computable. *International Journal of Computational Geometry and Applications*, *16*(5), 567-590.

Chazelle, B., Edelsbrunner, H., Grigni, M., Guibas, L., Hershberger, J., Sharir, M., & Snoeyink, J. (1994). Ray shooting in polygons using geodesic triangulations. *Algorithmica*, *12*(1).

Chew, L. P. (1985). Planning the shortest path for a disc in O(n2 log n) time. *Proceedings of the 1st Annual Symposium on Computational Geometry, SCG 1985*.

Cover, T. M., & Hart, P. E. (1967). Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory*, *13*(1), 21-27.

Cruz, G. C. S., & Encarnação, P. M. M. (2012). Obstacle avoidance for unmanned aerial vehicles. *Journal of Intelligent and Robotic Systems: Theory and Applications*, *65*(1), 203-217.

Gao, Y., Zheng, B., Chen, G., Lee, W. C., Lee, K. C. K., & Li, Q. (2009). Visible reverse k-nearest neighbor query processing in spatial databases. *IEEE Transactions on Knowledge and Data Engineering*, *21*(9), 1314-1327.

Ghosh, S. K., & Mount, D. M. (1991). Output-sensitive algorithm for computing visibility graphs. *SIAM Journal on Computing*, *20*(5), 888-910.

Guibas, L., Hershberger, J., Leven, D., Sharir, M., & Tarjan, R. E. (1987). Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, *2*(1), 209-233.

Güting, R. H. (1994). An introduction to spatial database systems. *The VLDB Journal*, *3*(4), 357-399.

Hershberger, J., & Suri, S. (1999). An optimal algorithm for Euclidean shortest paths in the plane. *SIAM Journal on Computing*, *28*(6), 2215-2256.

Kapoor, S., & Maheshwari, S. N. (1988). Efficient algorithms for Euclidean shortest path and visibilty problems with polygonal obstacles. *Proceedings of the 4th Annual Symposium on Computational Geometry, SCG 1988, 172-182*.

Lozano-Pérez, T., & Wesley, M. A. (1979). An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, *22*(10), 560-570.

Mitchell, J. S., Mount, D. M., & Papadimitriou, C. H. (1987). The discrete geodesic problem. *SIAM Journal on Computing*, *16*(4), 647-668.

Nutanong, S., Tanin, E., & Zhang, R. (2007). Visible nearest neighbor queries. In *International Conference on Database Systems for Advanced Applications,* 876-883. Berlin, Heidelberg: Springer Berlin Heidelberg.

Papadias, D., Zhang, J., Mamoulis, N., & Tao, Y. (2003). Query processing in spatial network databases. In *Proceedings 2003 VLDB Conference: 29th International Conference on Very Large Databases (VLDB),* 802-813. Morgan Kaufmann.

Sharifzadeh, M., Kolahdouzan, M., & Shahabi, C. (2008). The optimal sequenced route query. *VLDB Journal*, *17*(4), 765-787.

Storer, J. A., & Reif, J. H. (1994). Shortest paths in the plane with polygonal obstacles. *Journal of the ACM (JACM)*, *41*(5), 982-1012.

Taniar, D., & Rahayu, W. (2013). A taxonomy for nearest neighbour queries in spatial databases. *Journal of Computer and System Sciences*, *79*(7), 1017-1039.

Wang, Y. Q., Xu, C. F., Yu, G., Gu, Y., & Chen, M. (2010). Visible k nearest neighbor queries over uncertain data. *Jisuanji Xuebao/Chinese Journal of Computers*, *33*(10), 1943-1952.

Wang, Y., Zhang, R., Xu, C., Qi, J., Gu, Y., & Yu, G. (2014). Continuous visible k nearest neighbor query on moving objects. *Information Systems*, *44,* 1-21.

Welzl, E. (1985). Constructing the visibility graph for n-line segments in O(n2) time. *Information Processing Letters*, *20*(4), 167-171.

Xu, H., Li, Z., Lu, Y., Deng, K., & Zhou, X. (2010). Group visible nearest neighbor queries in spatial databases. In *International Conference on Web-Age Information Management*, 333-344. Berlin, Heidelberg: Springer Berlin Heidelberg.

Xu, J., & Güting, R. H. (2015). Querying visible points in large obstructed space. *GeoInformatica*, *19*(3), 435-461.

Zhang, J., Papadias, D., Mouratidis, K., & Manli, Z. (2005). Query processing in spatial databases containing obstacles. *International Journal of Geographical Information Science*, *19*(10), 1091-1111.

Zhang, J., Papadias, D., Mouratidis, K., & Zhu, M. (2004). Spatial queries in the presence of obstacles. In *International conference on extending database technology – EDBT 2004 Proceedings*, 66-384. Berlin, Heidelberg: Springer Berlin Heidelberg.