Roadmap-Based Path Planning

Using the Voronoi Diagram for a Clearance-Based Shortest Path

©DIGITAL VISION

The path-planning problem was originally studied extensively in robotics, and, through this research, it has gained more relevance in areas such as computer graphics, simulations, geographic information systems (GIS), very-large-scale integration (VLSI) design, and games. Path planning still remains one of the core problems in modern robotic applications, such as the design of autonomous vehicles and perceptive systems [31], [39]. The basic path-planning problem is concerned with finding a good-quality path from a source point to a destination point that does not result in collision with any obstacles. Depending on the amount of the information available about the environment, which can be completely or partially known or unknown, the approaches to path planning vary considerably. Also, the definition of a good-quality path usually depends on the type of a mobile device (a robot) and the environment (space), which has fostered the development of a rich variety of path-planning algorithms, each catering to a particular set of needs. Latombe [28] provides a comprehensive survey of different path-planning algorithms.

Computational geometry plays a special role in path-planning developments. Extensive methodologies that rely on geometric representation of the space, reveal topological properties of the agents (robots and/or obstacles), and allow the efficient dynamic position tracking and updates have been brought forward from computational geometry to solve a specific set of path-planning problems. Such problems usually have a well-defined and deterministic set of objectives, regular geometric space representation, and specific functions that discribe robotic movements. The problems also include planning a path and optimizing it (based on selected criteria such as the length, the smoothness, the cost, etc.) [7], solving problems involving evolutionary algorithms and swarm intelligence [2], and studying the behavior of a network of mobile robot agents [17].

The traditional computational geometry-based approaches to path planning can be classified into three basic categories: the cell decomposition method [35], the roadmap method [1], and the potential field method [40]. If robots are represented by polygonal objects, an approach based on the Minkowski sum is often used [24]. Both the cell decomposition and the roadmap methods along with the Minkowski sum method have their roots in computational geometry.

The cell decomposition method uses nonoverlapping cells to represent the free-space (C_f) connectivity. The decomposition can be exact or approximate. An exact decomposition divides C_f exactly [4]. An approximation scheme Kambhampati discretizes C_f with cells. It decomposes the free space recursively, stopping when a cell is entirely in C_f or entirely inside an obstacle. Otherwise, the cell is further divided. Because of memory and time constraints, the recursive process stops when a certain degree of accuracy has been reached. The cell decomposition method, although simple to implement, seldom yields high-quality paths. The exact cell decomposition technique is faster than the

Digital Object Identifier 10.1109/MRA.2008.921540

BY PRIYADARSHI BHATTACHARYA AND MARINA L. GAVRILOVA

1070-9932/08/$25.00©2008 IEEE

approximate one, but the path obtained is not optimal. The approximate cell decomposition can yield near-optimal paths by increasing the grid resolution, but the computation time will increase drastically. There is also the known problem of digitization bias associated with using a grid. This stems from the fact that while searching for the shortest path in a grid, the grid distance is measured and not the Euclidean distance.

The roadmap method attempts to capture the free-space connectivity with a graph. This approach has several variations. The probabilistic roadmap method (PRM) [1], [23] represents the free-space connectivity with a graph whose vertices are generated randomly in free space and connected to the k-nearest neighboring vertices such that the connecting edges do not cross any obstacle. To expedite the graph creation, several sampling-based methods such as Ariadne's Clew algorithm [33], expansive space planner [20], random walk planner [11], rapidly exploring random tree [26], [27] have been proposed. For algorithmic details, refer to [29]. Some of the other popular roadmap-based approaches are based on computational geometry structures such as the visibility graph for the shortest path and the Voronoi diagram for a maximum clearance path.

The idea behind the potential field method is to assign a function similar to the electrostatic potential to each obstacle and then derive the topological structure of the free space in the form of minimum potential valleys. The robot is pulled toward the goal configuration as it generates a strong attractive force. In contrast, the obstacles generate a repulsive force to keep the robot from colliding with them. The path from the start to the goal can be found by following the direction of the steepest descent of the potential toward the goal [9]. The strength of this approach is that path planning can be done in real time by considering only the obstacles close to the robot. Information on the locations of all obstacles is not required beforehand. However, as only local properties are used in planning, the robot may get stuck at local minima and never reach the goal.

In this article, we chose the roadmap approach and utilized the Voronoi diagram to obtain a path that is a close approximation of the shortest path satisfying the required clearance value set by the user. A Voronoi diagram (a fundamental computational geometry structure) is defined as the partitioning of a plane with n points (generators) into convex polygons such that each polygon contains exactly one generator and every point in a given polygon is closer to its generator than to any other. For a more formal definition and properties, refer to [3]. The Voronoi diagram and its dual structure, the Delaunay triangulation, have been used in a wide variety of applications such as collision detection [14], extraction of crust and skeleton [16], swarm intelligence optimization [2], cluster analysis [7], and mobile robot agent network [17]. The Voronoi diagram is also a well-known roadmap in the path-planning literature, which has edges that provide a maximum clearance path among a set of disjoint polygonal obstacles. However, a path obtained directly from the Voronoi diagram may be far from optimal. It usually has many unnecessary turns, and the length of the path may be undesirably long at regions where the obstacles are far apart. In fact, it is worth noting that minimizing the path length and maximizing the clearance seemingly contradict each other, as increasing the clearance results in a longer path whereas reducing the path length necessarily reduces the clearance from obstacles. We thought that it would be highly beneficial for many applications if an algorithm could be developed that would accept the minimum clearance required as an input parameter and produce a path that would be shortest while satisfying the minimum clearance requirement. The shortest path problem on its own can be viewed as only a special case when we set the clearance required to zero. The practical usefulness of this kind of an algorithm is apparent for many applications, including marine GIS, ship route planning, VLSI design, oil drill path planning, etc.

Figure 1 illustrates the output of the proposed algorithm on a spatial dataset. Figure 1(a) is the path obtained directly from the Voronoi diagram. Figure 1(b) shows the shortest path obtained with $C_{\min} = 0$. Figure 1(c) shows the final optimal path with a user specified clearance value of two units ($\approx$20 m).

The advantage of the proposed technique versus alternative path-planning methods is in its simplicity, versatility, and efficiency. For example, for planning the path for translational robots of varied dimensions, it has the capability to set the minimum clearance to a proportional value and the algorithm will find an optimal path for that minimum clearance, if one exists. The reported path is of more practical value because it is optimal with respect to both length and clearance. A high-quality approximation of the shortest path can also be obtained by the developed algorithm in much less time than by the popular and very well-performing visibility graph approach when we set $C_{\min} = 0$. To prove this claim, we experimentally compared the lengths of the paths obtained by these techniques and presented results in the experimental section.

In the next section, we provide an extensive literature review. An outline of the proposed algorithm is provided, followed by the detailed description of Voronoi based methodology. Later, a comprehensive analysis of experimental results is provided. Finally, the last section draws conclusions and outlines future work.

Literature Review

As established in the previous section, the planning algorithms vary in the nature of the desired path and the information on

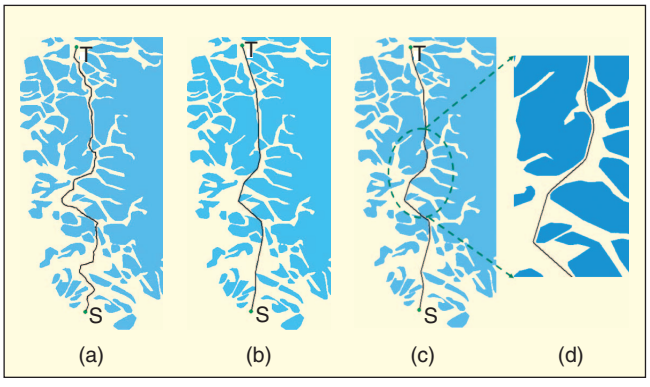

Figure 1. *Shortest path and path with a minimum clearance. The figure depicts a part of the South American coastline from the ESRI world low-resolution layer: (a) shortest path from Voronoi-diagram-based roadmap, (b) shortest path using proposed algorithm ($C_{\min} = 0$), (c) clearance based path using proposed algorithm ($C_{\min} = 2$), and (d) zoomed path.*

the environment. Recently, path planning in dynamic and complex environments has received considerable attention from researchers. In a dynamic environment, obstacles are allowed to move, and so, the environment can change dramatically over a period of time. An adaptive path-planning technique that takes cue from the previous situation can be found in [5]. In [10], the authors use an adaptive mesh for dynamic path planning based on a combination of graph- and grid-based representation of the environment. The PRM is very promising for dynamic path planning, as a big advantage of the PRM is that its complexity depends mostly on the difficulty of the path and to a much lesser extent on the global complexity of the environment or the dimension of the configuration space [36]. A recent algorithm based on the PRM for a dynamic environment can be found in [6]. However, in general, PRM-based approaches, being probabilistic in nature, do not meet any optimality criteria.

There has been research on planning algorithms coordinating motion of multiple robots [37]. The idea of this article is to perceive multiple robots as a single composite robot and then to determine a roadmap for this composite robot.

A highly interesting research on the utilization of computational geometry methods for coordination problems in dynamic systems was conducted by Cortés and Bullo [12]. They capitalize on the properties of geometric constructs (disk covering and sphere packing), nonsmooth analysis, and dynamic system approaches, studied in the context of networked robot interactions.

In this article, we focus on the roadmap-based path planning and utilize a powerful computational geometry data structure, the Voronoi diagram, to obtain a clearance-based shortest path. The advantage of using a Voronoi diagram as a roadmap over alternative methods, among which the visibility graph prevails, is its efficiency. The Voronoi diagram can be constructed in just $O(n\log n)$ time, whereas even the fastest known algorithm for constructing visibility graph [15] can take $O(n^2)$ time in the worst case when the visibility graph has $O(n^2)$ edges. Since a Voronoi diagram has $O(n)$ edges, querying for a path in a Voronoi diagram-based roadmap is also much faster than querying in a visibility graph. However, as mentioned before, the quality of path obtained directly from the Voronoi diagram may be far from optimal. Thus, improving the quality of the path (refining the path) is an important direction of research.

A general method for refining a path obtained from a roadmap based on classical numerical optimization techniques can be found in [25]. The authors apply costs to each edge and use an augmented Dijkstra's algorithm to determine an optimal path. The edges that are nearer to obstacles are assigned higher costs. However, there is no guarantee that the method will generate an optimal path, as the path is constrained to the edges in the roadmap. To improve the smoothness of the path obtained from a roadmap, a B-Spline approximation has been used in [21].

In [32], the authors combine the Voronoi diagram, visibility graph, and potential field approaches to path planning into a single algorithm to obtain a tradeoff between the optimal by safe and the shortest paths. The algorithm is fairly complicated, and although the path length is shorter than those obtained from the potential field method or the Voronoi diagram, it is still not optimal. The path exhibits bumps and rudimentary turns and is not smooth.

Another recent work on reducing the length of the path obtained from a Voronoi diagram [19] involves constructing polygons at the vertices in the roadmap where more than two Voronoi edges meet. The path is smoother and shorter than that obtained directly from the Voronoi diagram, but there has been no attempt to reach optimality.

In [41], the authors create a new diagram called the $VV^{(c)}$ diagram (the Visibility-Voronoi diagram for clearance c). The motivation behind their work is similar to ours, i.e., to obtain an optimal path for a specified clearance value. The diagram evolves from the visibility graph to the Voronoi diagram as the value of c increases. Unfortunately, as the method is visibility based, the processing time is $O(n^2\log n)$, which renders it impractical for large spatial datasets.

Our technique is able to generate near-optimal paths in just $O(n\log n)$ time. The paths are the shortest possible while maintaining just the amount of clearance required. We conjecture that the obtained path is satisfactory in the homotopy sense, i.e., it can be continuously transformed to the true-optimal path without crossing any obstacles. Thus, the obtained approximation can be always refined to a near-optimal path. Experimental results support this conjecture. Another interesting property of the path is that it can be extraordinarily smooth when going around smooth obstacles. The smoothness only seems to help reduce the path length in such cases.

Outline of the Method

The cornerstone of the methodology is the utilization of a powerful computational geometry data structure: the Voronoi diagram. We start by building the Voronoi diagram of the obstacles. The source and destination are dynamically inserted into the diagram, and they are connected to all Voronoi vertices of their Voronoi cells, respectively. Inserting the source and destination dynamically has two major advantages over simply connecting them to the nearest Voronoi vertex. There is no possibility of the connecting edges crossing an obstacle, as they are contained inside the Voronoi cell. Also, if we want to perform multiple queries, we can do so by simply dynamically deleting the old source and destination from the Voronoi diagram instead of rebuilding the diagram. We next remove all those edges in the resulting diagram that have a clearance less then the minimum clearance required ($C_{\min}$). The resulting graph represents the roadmap used by our algorithm. Any path obtained directly from this roadmap between the source and the destination is guaranteed to have a clearance $\geq C_{\min}$. We apply Dijkstra's algorithm to determine the shortest path in the roadmap, but as mentioned before, the path is far from optimal. Our algorithm then refines the obtained path by removing unnecessary turns, so that the path has a minimum number of links but at the same time satisfies the minimum clearance criterion. This path, however, is still not optimal and has sharp corners. We next add Steiner points along the edges of this path and use a novel corner-cutting paradigm to convert it to an optimal path. The flowchart in Figure 2 illustrates the steps involved. We next discuss each stage in detail.

Methodology in Detail

As established earlier, a powerful computational geometry data structure has been proposed to solve the problem of an optimal path generation between a source and a destination in the presence of simple disjoint polygonal obstacles. This method has a number of unique features, such as a novel application of the Voronoi diagram in the specified clearance context, the iterative refinement technique based on Steiner points for path optimization, and the possibility of performing dynamic updates on the structure during the path computation process. The main steps of the developed path generation algorithm are detailed next.

Voronoi Diagram Construction

The construction of the Voronoi diagram of a set of polygons in the plane is both complex and time consuming. Instead, we approximate the obstacles with points on the boundary edges and generate the Voronoi diagram of the approximating points. The approximation of generalized Voronoi diagrams was first proposed by Sugihara [38]. Removal of the Voronoi edges that cross obstacles from this diagram provides a nice approximation of the Voronoi diagram of the original obstacles. We first create the Delaunay triangulation and then generate the Voronoi diagram from it in $O(n)$ time.

We use the winged-edge data structure to represent the triangulation, as we found it more intuitive to use and update than the half-edge or quad-edge data structures. This structure is also more flexible, as the direction of the edges can be fixed arbitrarily. It maintains three collections—one for vertices, one for edges, and one for triangles. Each vertex stores the coordinates and the index of any one edge incident on it. A triangle stores the index of any one of the three bounding edges. Most of the topological information is stored inside an edge. It contains the indices of the end vertices, indices of triangles on left and right, and the clockwise and counterclockwise predecessors and successors of the edge.

The randomized incremental algorithm for construction of the Delaunay triangulation [18] proved helpful for the path planning problem, as it allows dynamic insertion of new points in $O(1)$ time without having to rebuild the entire triangulation. The incremental construction starts by creating a triangle that contains all the datapoints inside it. However, just containing the points is not a sufficient condition. The three corner points of this triangle should not lie inside the circumcircle of any of the triangles of the triangulation of the dataset. This is to ensure that the enclosing triangle does not influence the triangulation of the dataset.

We determine the minimum and maximum x and y values in the dataset as $x_{\min}$, $x_{\max}$, $y_{\min}$, and $y_{\max}$. The center point (x_0, y_0) of the dataset is then calculated as $x_0 = x_{\min} + (x_{\max} - x_{\min})/2$ and $y_0 = y_{\min} + (y_{\max} - y_{\min})/2$. We set M as $\text{Max}((x_{\max} - x_{\min})/2, (y_{\max} - y_{\min})/2)$. Then, the three corner points of the external triangle $\{p_1, p_2, p_3\}$ can be specified as

$$p_1 = (x_0 + 3 \times M,\ y_0)$$
$$p_2 = (x_0,\ y_0 + 3 \times M)$$
$$p_3 = (x_0 - 3 \times M,\ y_0 - 3 \times M).$$

Figure 3 shows the enclosing triangle for a simplified case where the center of the datapoints is $(0, 0)$ and the span of the data along both x and y axes equals $2 \times M$. Once the enclosing triangle is constructed, the datapoints are added to the triangulation one by one, and after each insertion, the required topological changes are performed to restore the Delaunay properties. A new datapoint p_i can lie inside a triangle or on a triangle edge of the current triangulation. On the basis of this, two kinds of topological updates are required. In Figure 4(a), the datapoint lies inside a triangle. The topological update involves the addition of three new edges. In Figure 4(b), the datapoint lies on an edge. In this case, the old edge is deleted, and four new edges are created. The new triangles created as a result of the topology update may not satisfy the empty circle property. If an edge is found to be illegal, it is flipped so that the Delaunay properties are restored.

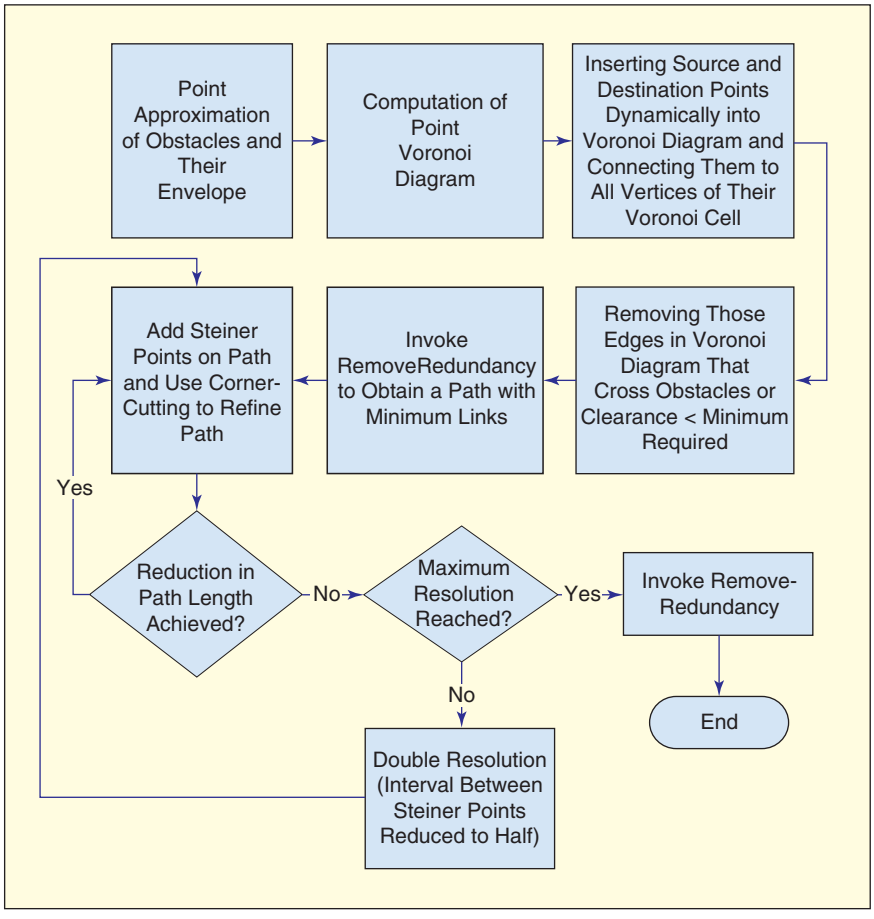

Figure 2. *Different stages of the proposed algorithm.*

The advantage of using a Voronoi diagram as a roadmap over alternative methods, among which the visibility graph prevails, is in efficiency.

Generation of Roadmap

Before generating the roadmap, we add the source and the destination to the Voronoi diagram dynamically. To requery, we dynamically delete the old source and destination and insert the new ones. Details on point location and dynamic point deletion are provided in next section. The roadmap is generated by removing the edges from the Voronoi diagram that have an obstacle clearance $< C_{\min}$. There is one issue, however. If we consider the obstacles only for the Voronoi diagram construction, the resulting roadmap will not be complete. This is evident in Figure 5(a). This can be resolved by determining the minimum bounding box (mbb) of all obstacle vertices in linear time and expanding the mbb in all four directions by at least twice the minimum clearance required. The Voronoi diagram is then constructed from the points approximating the obstacles and this expanded mbb. The roadmap obtained from such a diagram is shown in Figure 5(b). The quality of the diagram depends on the spacing between approximation points. As can be observed in [30], if the spacing is too large, the Voronoi edges exhibit a zigzag pattern. However, it has been observed in [38] that a finer initial approximation yields results practically indistinguishable from the original Voronoi diagram. The proof was the utilization of the results of the approximating generalized Voronoi diagrams by ordinary Voronoi diagrams in the framework of path planning. Thus, it is possible to find the fine approximation (in our case established experimentally) so that the path obtained for the set of approximation points always satisfies the clearance requirement for the original obstacle set.

In Figure 6(a), the roadmap is generated using an expanded mbb. It can be observed that because of the concavity of the obstacle, there is a large deviation in the shortest path obtained from the roadmap. There are two main disadvantages to this. The first one is that the unnecessarily large deviation may make a shorter path appear longer, and this may result in the selection of a longer path from the roadmap as the shortest path. The second disadvantage is that the optimization step will take longer to execute. To eradicate this problem, we retract the approximation points on the mbb toward the nearest obstacle point so that the distance is a little greater than $2 \times C_{\min}$. The Voronoi diagram is then constructed from the obstacle approximation points and the points on this retracted boundary. This results in a much improved shortest path obtained from the roadmap as evident in Figure 6(b).

Figure 7(a) shows the Voronoi diagram obtained from the point approximation of an obstacle and the retracted boundary for a dataset. Figure 7(b) shows the roadmap extracted from the Voronoi diagram. In practice, to reduce the computation time for the clearance check, we first build a quadtree of the mbbs of the obstacle edges. When checking for clearance of a Voronoi edge e, we first determine in $O(\log n)$ time all the

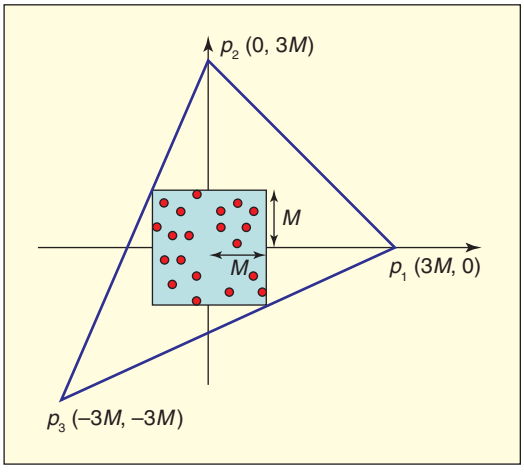

Figure 3. *The enclosing triangle containing all datapoints.*

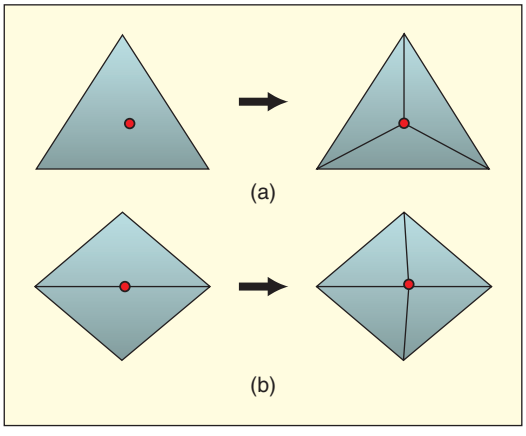

Figure 4. *Topological updates associated with insertion of a new datapoint: (a) datapoint lies inside a triangle and (b) datapoint lies on a triangle edge.*

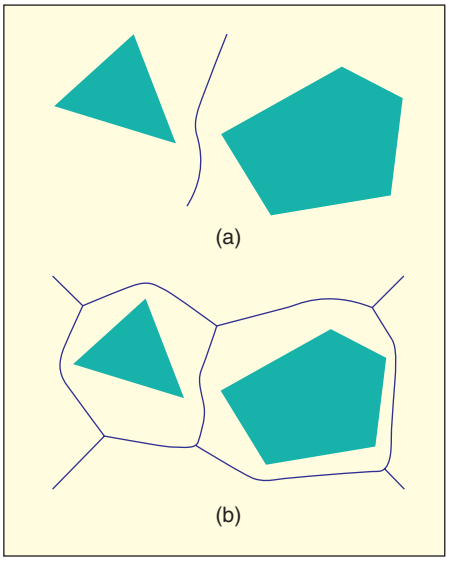

Figure 5. *Complete and incomplete roadmaps: (a) without outer boundary and (b) with outer boundary.*

obstacle edge indices whose mbbs overlap with the mbb of e, expanded in all four directions by $C_{\min}$. This yields a constant number of edges against which the actual clearance check is carried out. As the expanded mbbs of the Voronoi edges constituting the medial axis of the obstacle and some of the outer edges do not overlap with the mbb of any obstacle edge, they are not removed and remain in the roadmap. These edges do not influence the determination of the shortest path in any way.

Dynamic Insertion and Deletion of Source and Destination

To insert the source and destination into the triangulation, we perform a walk in the triangulation to locate the triangle containing the point. Algorithmic details about walking in a triangulation have been presented by Devillers et al. [13]. To allow requerying, we delete the old source and destination dynamically from the Voronoi diagram and dynamically insert the new ones. For dynamic deletion of points from the Delaunay triangulation, we followed the algorithm outlined in [34] with a little modification. Because of limitations of the floating-point arithmetic, the number of neighbors of the point (P) to be deleted may never get reduced to three, resulting in an infinite loop. To avoid this problem, the authors in [34] consider the circumcircle to be shrunk by a small amount while performing the Incircle test. We adopt a different approach. Even if some of the remaining neighbors of P fall inside the circumcircle of the potential triangle, we flip the edge. This guarantees that the number of neighboring vertices of P will always be reduced to three. We then perform a normal Incircle test on the flipped edges to ensure that the triangulation remains Delaunay. Figure 8 shows a program snapshot of the topological events in deleting the encircled point from a simple dataset.

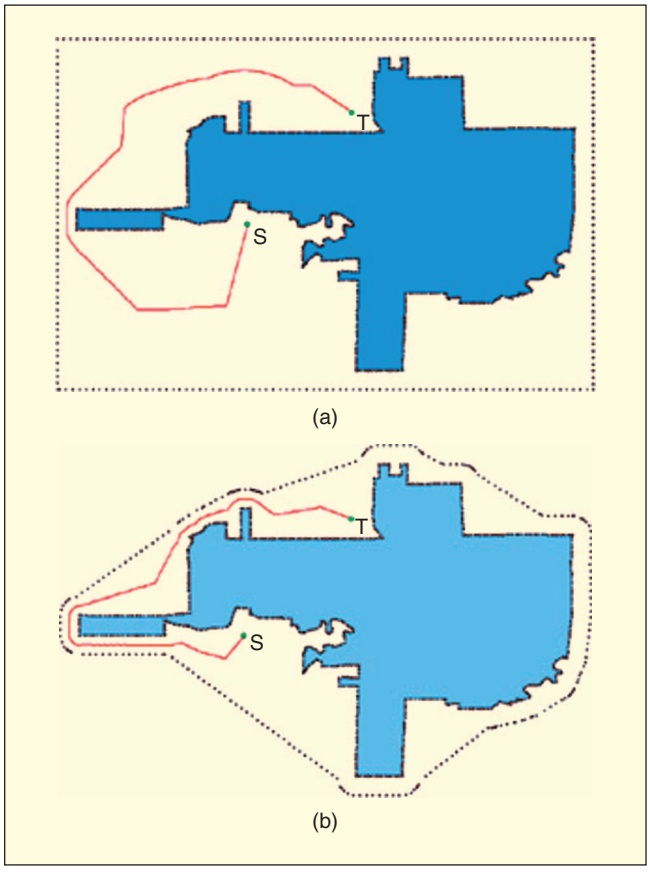

Figure 6. *Shortest paths obtained from (a) roadmap using expanded mbb and (b) roadmap using retracted boundary.*

Removal of Redundant Vertices and Obtaining a Path with Minimum Number of Links

The shortest path obtained in the previous step has many unnecessary turns and redundant vertices. This step removes all redundant vertices and generates a path that has a minimum number of links or edge connections. A minimum number of links ensures that the iterative refinement in next step consumes less time. The method is simple.

For a vertex v_i on the path ($i = \{1 \ldots n-2\}$), we check whether the line segment $\overline{v_i v_{i+2}}$ has a clearance $\geq C_{\min}$. If so, we remove v_{i+1} from shortest path and repeat the process from vertex v_{i+2}. If not, we retain v_{i+1} and consider it as the next vertex for processing. We provide the pseudocode in Algorithm 1.

To efficiently determine the clearance of an edge (e), we build a quadtree of the mbbs of the obstacle edges. We determine the mbb of e and expand it by $C_{\min}$ in all four directions. In $O(\log n)$ time, it is possible to determine the edges whose mbbs overlap with the expanded mbb of e, and the clearance check is carried out for only these constant number of edges. This ensures the complexity of this step for all edges is $O(n \log n)$.

Iterative Refinement Using a Corner-Cutting Technique

The main idea behind the iterative refinement of the path is the utilization of the Steiner points. We first add the Steiner points along the edges of the path at regular interval $\triangle$. For our

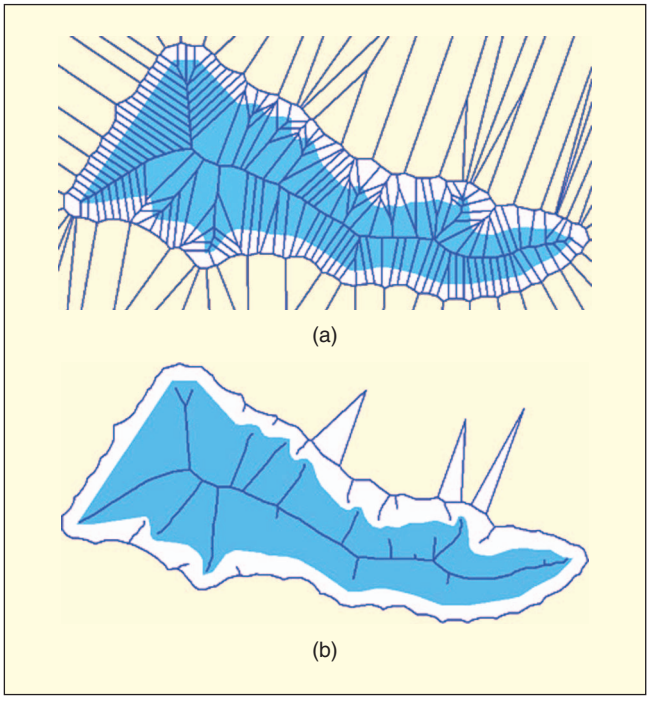

Figure 7. *Roadmap generation using point Voronoi diagram: (a) Voronoi diagram of points approximating object and retracted boundary and (b) roadmap extracted from Voronoi diagram.*

experiments, we set $\triangle$ to roughly one fourth the average obstacle edge length. This is two times the resolution we used to approximate the obstacles with points. Let V be a vertex on the shortest path other than source and destination. Let e_1 and e_2 be the two edges incident on V. We define the first Steiner point along e_1 as a Steiner point that lies on e_1 at $\triangle$ distance away from V, the second Steiner point is $2\triangle$ away from V, and so on. We try to connect the first Steiner point on e_1 with that on e_2. If the connecting edge satisfies the minimum clearance, we move to the second Steiner point along both edges and try to connect these. The process continues until an intersection is detected, or the clearance from obstacles falls below the required minimum clearance, or the endpoint of one of the incident edges on V is reached. We then replace V with the last Steiner points that we could successfully connect with an edge. If we failed to connect even the first Steiner points along the two incident edges, we retain V. The path cannot be shortened any further at point V at this resolution. When no more reduction in path length is possible for any of the vertices, we double the resolution (i.e., set the interval between Steiner points along the edges to $\triangle/2$) and repeat the process. The iteration continues until the resolution reaches a maximum precalculated value (Figure 2). Figure 9 shows a sample path obtained after the different stages.

Experimental Results

We now demonstrate that the path obtained by our algorithm is optimal with respect to length and clearance from obstacles for a specified value of minimum clearance ($C_{\min}$). The refined path is also smooth. We also show that the proposed algorithm outperforms the popular (and efficient) visibility graph-based approach with regard to both speed and quality of the path.

We have put our algorithm to test on a real-world data from the large spatial datasets. The spatial datasets considered are mostly represented in the form of Environmental Systems Research Institute (ESRI) shapefiles. When dealing with a spatial data, it is often required to substantially increase the resolution of the data so that extraordinary cases such as polygons degenerating into points can be avoided. After performing the computations, we scale down the result to display scale.

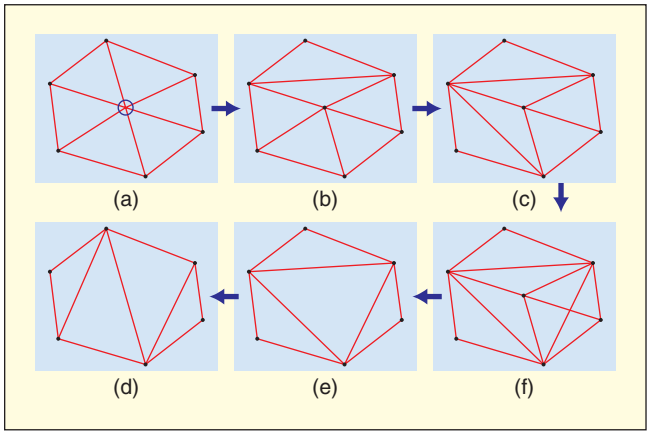

Figure 8. *Dynamic deletion of a point from the Delaunay triangulation.*

Algorithm 1. RemoveRedundancy($C_{\min}$, S_{old}, S_{new})

Input: S_{old}: Vertices on shortest path (in sequence) between source and destination. Obtained by applying Dijkstra's algorithm on roadmap.
$C_{\min}$: minimum clearance required
Output: S_{new}: Path obtained after processing

$N \leftarrow$ *NumberofVertices* (S_{old}) {*NumberofVertices* (S): number of vertices in S}
if $N = 2$ **then**
  $S_{\text{new}} \leftarrow S_{\text{old}}$ {trivial case}
  return
end if
repeat
  $N \leftarrow$ *NumberofVertices* (S_{old})
  $S_{\text{new}} \leftarrow \Phi$
  for $i = 1$ to $N - 2$ **do**
    $S_{\text{new}} \leftarrow S_{\text{new}} \cup S_{\text{old}}[i]$
    if $\overline{v_i v_{i+2}}$ has clearance $\geq C_{\min}$ **then**
      $i \leftarrow i + 1$ {skip over next vertex}
    end if
  end for
  for $j = i$ to N **do**
    $S_{\text{new}} \leftarrow S_{\text{new}} \cup S_{\text{old}}[j]$ {complete path}
  end for
  $S_{\text{old}} \leftarrow S_{\text{new}}$
until *NumberofVertices*(S_{new}) $= N$ {path remains unchanged}

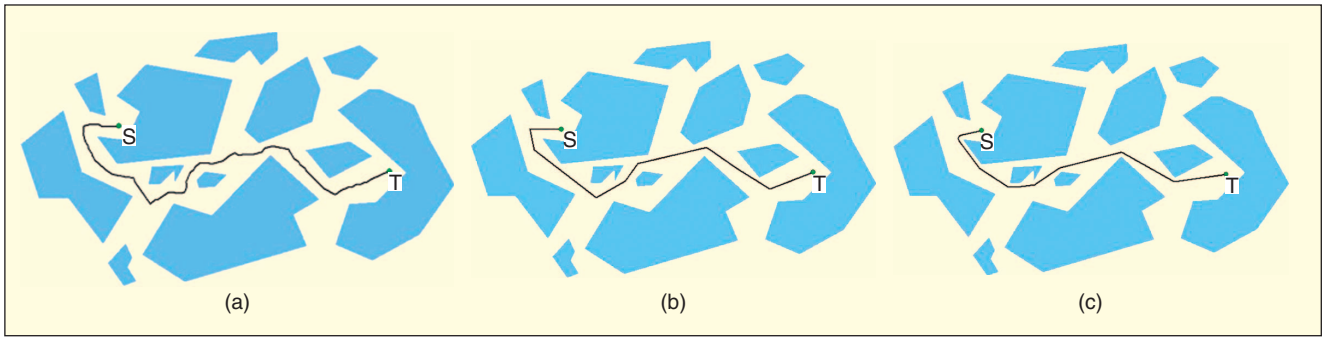

Figure 9. *Clearance-based path (different stages): (a) path from roadmap, (b) path after minimizing links in path, (c) path after iterative refinement.*

Users can zoom in on any part to see magnified view of a selected portion. All experiments have been carried out on a 1.6-GHz Intel Centrino Duo processor with 0.99-GB RAM. The programs have been implemented in Visual C++. The source and target are indicated with S and T in all figures.

In Figure 10, we visually compare the path obtained directly from the Voronoi diagram–based roadmap to that obtained after iterative refinement ($C_{min} = 0$) on a portion of the real-world data. The shortest path in Figure 10(b) can be observed to wrap around the obstacles very similar to a path obtained from a visibility graph. The result shows that our algorithm is effective even in complicated environments.

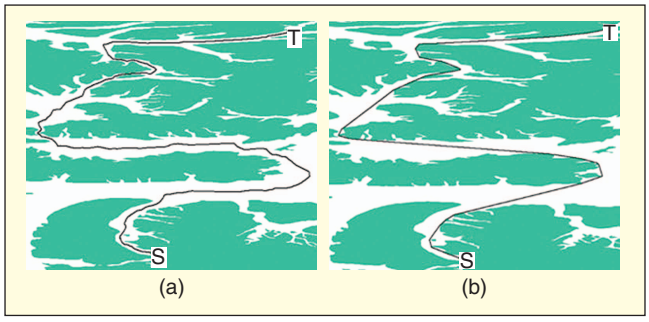

Figure 10. *Shortest path using proposed algorithm (number of vertices in dataset = 11,797): (a) shortest path obtained from Voronoi-diagram-based roadmap, (b) optimal path after iterative refinement ($C_{min} = 0$).*

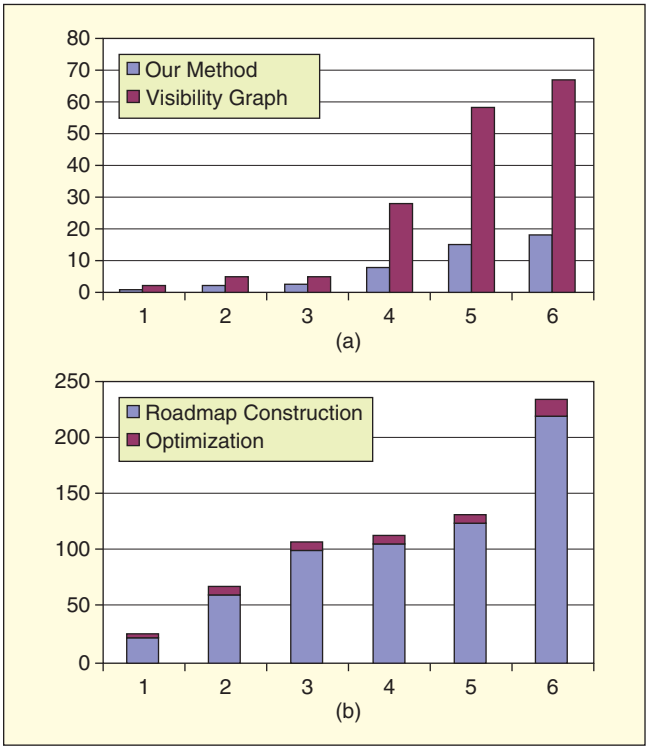

Figure 11. *CPU time estimates. x axis is the shapefile index and y axis is the time consumed in seconds: (a) CPU time of our method versus visibility graph approach, (b) breakup of CPU time.*

Table 1. Shapefile index versus number of vertices.

	S1	S2	S3	S4	S5	S6
Vertex cnt. [Fig. 11(a)]	155	338	355	739	939	1,866
Vertex cnt. [Fig. 11(b)]	2,599	4,065	4,769	5,365	8,070	11,797

We provide time estimates in Figure 11. The number of vertices for the six datasets for Figure 11(a) and 11(b) are mentioned in the first and second rows of Table 1, respectively. Figure 11(a) shows the time consumed by our algorithm and the visibility graph approach on a number of spatial datasets. The time (in seconds) includes the time to build the roadmap and determine the path between the source and the destination. The time estimated for our method includes the time for path optimization. It can be observed that the time difference becomes more pronounced as the number of vertices in the dataset increases. In Figure 11(b), we show the breakup of roadmap construction time and optimization time for some large datasets. The time consumed by the visibility graph approach for these datasets is extremely high, and therefore, we do not mention it here.

Conclusions

In this article, a computational geometry data structure has been proposed to solve the problem of an optimal path generation between a source and a destination in the presence of simple disjoint polygonal obstacles. The method has a number of unique features, such as a novel application of the Voronoi diagram in the specified clearance context, the iterative refinement technique based on the Steiner points for path optimization, and the possibility of performing dynamic updates on the structure during the path computation process. The obtained path is optimal with respect to length and clearance from the obstacles for a specified value of minimum clearance (C_{min}). The refined path is also observed to be smooth. The proposed algorithm outperforms the popular (and efficient) alternative approach with regard to speed as well as the quality of the path and is flexible to allow various required clearance settings and source/destination location changes.

Future studies will involve porting the algorithm to other platforms and applications (including ArcGIS compatibility and direct shape file visualization options) and extending the algorithm to work with three-dimensional data.

Acknowledgment

We would like to acknowledge GEOIDE and NSERC granting agencies for continuous support of this project.

Keywords

Path planning, Voronoi diagram, clearance-based path.

References

[1] N. Amato and Y. Wu, "A randomized roadmap method for path and manipulation planning," in *Proc. IEEE Int. Conf. Robotics and Automation*, 1996, pp. 113–120.

[2] R. Apu and M. L. Gavrilova, "Battle swarm: An evolutionary approach to complex swarm intelligence," in *2006 Proc. 9th Int. Conf. Computer Graphics and Artificial Intelligence*, Eurographics, Limoges, France, May 2006, pp. 139–150.

[3] F. Aurenhammer, "Voronoi diagrams—A survey of a fundamental geometric data structure," *ACM Comput. Surv.*, vol. 23, no. 3, pp. 345–405, 1991.

[4] F. Avnaim, J. D. Boissonnat, and B. Faverjon, "A practical exact motion planning algorithm for polygonal objects amidst polygonal obstacles," in *Proc. IEEE Int. Conf. Robotics and Automation*, Apr. 1988, vol. 3, pp. 1656–1661.
[5] J. Baltes and N. Hildreth, "Adaptive path planner for highly dynamic environments," *Lect. Notes Comput. Sci.*, vol. 2019, pp. 76–85, 2001.
[6] K. Belghith, F. Kabanza, L. Hartman, and R. Nkambou, "Anytime dynamic path-planning with flexible probabilistic roadmaps," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2006, pp. 2372–2377.
[7] P. Bhattacharya and M. L. Gavrilova, "CRYSTAL—A new density-based fast and efficient clustering algorithm," in *Proc. 3rd Int. Symp. Voronoi Diagrams in Science and Engineering*, 2006, pp. 102–111.
[8] P. Bhattacharya and M. L. Gavrilova, "Geometric algorithms for clearance based optimal path computation," in *Proc. 15th ACM Int. Symp. Advances in Geographic Information Systems (ACM GIS)*, 2007, pp. 208–311.
[9] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Trans. Syst., Man, Cybern.*, vol. 19, no. 5, pp. 1179–1187, 1998.
[10] P. Brož, I. Kolingerová, R. A. Apu, M. L. Gavrilova, and P. Zítka, "Path planning in dynamic environment using an adaptive mesh," in *Proc. Spring Conf. Computer Graphics*, 2007, pp. 172–178.
[11] S. Carpin and G. Pillonetto, "Robot motion planning using adaptive random walks," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2003, pp. 3809–3814.
[12] J. Cortés and F. Bullo, "Coordination and geometric optimization via distributed dynamical systems," *SIAM J. Control Optim.*, vol. 44, no. 5, pp. 1543–1574, 2005.
[13] O. Devillers, S. Pion, and M. Teillaud, "Walking in a triangulation," *Int. J. Found. Comput. Sci.*, vol. 13, no. 2, pp. 181–199, 2002.
[14] M. L. Gavrilova and J. G. Rokne, "Collision detection optimization in a multi-particle system," *Int. J. Comput. Geometry Appl.*, vol. 13, no. 4, pp. 279–301, 2003.
[15] S. K. Ghosh and D. M. Mount, "An output-sensitive algorithm for computing visibility graphs," *SIAM J. Comput.*, vol. 20, no. 5, pp. 888–910, 1991.
[16] C. Gold, "Crust and anti-crust: A one-step boundary and skeleton extraction algorithm," in *Proc. 15th ACM Symp. Computational Geometry*, 1999, pp. 189–196.
[17] R. Graham and J. Cortés, "Asymptotic optimality of multicenter Voronoi configurations for random field estimation," *IEEE Trans. Automat. Contr.*, submitted for publication.
[18] L. J. Guibas, D. E. Knuth, and M. Sharir, "Randomized incremental construction of Delaunay and Voronoi diagrams," *Algorithmica*, vol. 7, no. 1, pp. 381–413, 1992.
[19] D.-H. Yang and S.-K. Hong, "A roadmap construction algorithm for mobile robot path planning using skeleton maps," *Adv. Robotics*, vol. 21, no. 1, pp. 51–63, 2007.
[20] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," *Int. J. Comput. Geometry Appl.*, vol. 4, no. 4/5, pp. 495–512, 1999.
[21] J. M. Ibarra-Zannatha, J. H. Sossa-Azuela, and H. Gonzalez-Hernandez, "A new roadmap approach to automatic path planning for mobile robot navigation," in *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics (Humans, Information, and Technology)*, 1994, vol. 3. pp. 2803–2808.
[22] S. Kambhampati and L. S. Davis, "Multiresolution path planning for mobile robots," *IEEE J. Robot. Automat.*, vol. 2, no. 3, pp. 135–145, 1986.
[23] L. E. Kavraki and J.-C. Latombe, *Probabilistic Roadmaps for Robot Path Planning*. New York: Wiley, 1997.
[24] K. Kedem and M. Sharir, "An efficient motion planning algorithm for a convex rigid polygonal object in 2-dimensional polygonal space," *Discrete Comput. Geom.*, vol. 5, no. 1, pp. 43–75, 1990.
[25] J. Kim, R. A. Pearce, and N. M. Amato, "Extracting optimal paths from roadmaps for motion planning," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2003, pp. 2424–2429.
[26] J. Kuffner, Jr., and J.-C. Latombe, "Interactive manipulation planning for animated characters," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2000, pp. 417–418.
[27] J. Kuffner, Jr., and S. LaValle, "RRT-Connect: An efficient approach to single-query path planning," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2000, vol. 2. pp. 995–1001.
[28] J.-C. Latombe, *Robot Motion Planning*. Norwell, MA: Kluwer, 1991.
[29] S. M. LaValle. (2005). *Planning algorithms* [Online]. Available: http://msl.cs.uiuc.edu/planning
[30] R. Mahkovic and T. Slivnik, "Generalized local Voronoi diagram of visible region," in *Proc. IEEE Int. Conf. Robotics and Automation*, 1998, vol. 1. pp. 349–355.
[31] S. Martínez, J. Cortés, and F. Bullo, "Motion planning and control problems for underactuated robots," in *Control Problems in Robotics* (Springer Tracts in Advanced Robotics Series, vol. 4). New York: Springer-Verlag, 2003, pp. 59–74.
[32] E. Masehian and M. R. Amin-Naseri, "A Voronoi diagram-visibility graph-potential field compound algorithm for robot path planning," *J. Robot Syst.*, vol. 21, no. 6, pp. 275–300, 2004.
[33] E. Mazer, J. M. Ahuactzin, and P. Bessiére, "The Ariadne's clew algorithm," *J. Artif. Intell. Res.*, vol. 9, pp. 295–316, 1998.
[34] M. A. Mostafavi, C. Gold, and M. Dakowicz, "Delete and insert operations in Voronoi/Delaunay methods and applications," *Comput. Geosci.*, vol. 29, pp. 523–530, 2003.
[35] H. Nolborio, T. Naniwa, and S. Arimoto, "A quadtree-based path-planning algorithm for a mobile robot," *J. Robot Syst.*, vol. 7, no. 4, pp. 555–574, 1990.
[36] M. H. Overmars, "Recent developments in motion planning," *Lect. Notes Comput. Sci.*, vol. 2331, pp. 3–13, 2002.
[37] P. Švestka and M. Overmars, "Coordinated path planning for multiple robots," *Robot Autom. Syst.*, vol. 23, no. 3, pp. 125–152, 1998.
[38] K. Sugihara, "Approximation of generalized Voronoi diagrams by ordinary Voronoi diagrams," *CVGIP, Graph. Models Image Process.*, vol. 55, no. 6, pp. 522–531, 1992.
[39] G. Taylor and L. Kleeman, "Robust range data segmentation using geometric primitives for robotic applications," in *Proc. 5th IASTED Int. Conf. Signal and Image Processing*, 2003, pp. 467–472.
[40] C. W. Warren, "Global path planning using artificial potential fields," in *Proc. IEEE Conf. Robotics and Automation*, 1989, pp. 316–321.
[41] R. Wein, J. P. Van den Berg, and D. Halperin, "The Visibility-Voronoi complex and its applications," in *Proc. 21st Annu. Symp. Computational Geometry*, 2005, pp. 63–72.

Priyadarshi Bhattacharya is a senior software developer at Intermap Technologies Corporation, Calgary, Canada, developing software for GIS and 3-D visualization. He graduated with M.Sc. from the Department of Computer Science, University of Calgary, in 2007, where he worked in Spatial Analysis in Computational Sciences (SPARCS) Laboratory on projects related to optimal path planning in marine environments.

Marina L. Gavrilova is with the Department of Computer Science, University of Calgary. She is a founder and codirector of the SPARCS Laboratory and the Biometric Technologies Laboratory (BTLab). She has published over 100 journal and conference articles, books and book chapters, in addition to editing a number of journal special issues. In 2007, she became an editor-in-chief for *LNCS Transactions on Computational Science Journal*, Springer-Verlag. She currently serves on the Editorial Board for *International Journal of Computational Sciences and Engineering*, *Computer Graphics and CAD/CAM Journal*, and *Journal of Biometrics*. She is a Member of the IEEE.

Address for Correspondence: Priyadarshi Bhattacharya, SW Dev, Intermap Technologies Corp., 1074 Northmount Dr. NW, Calgary, AB T2L 0C2, Canada. E-mail: pbhattacharya@intermap.com.