applied sciences

MDPI

Article

Expanded Douglas–Peucker Polygonal Approximation and Opposite Angle-Based Exact Cell Decomposition for Path Planning with Curvilinear Obstacles

Jin-Woo Jung *, Byung-Chul So, Jin-Gu Kang, Dong-Woo Lim and Yunsik Son

Department of Computer Science and Engineering, Dongguk University, Seoul 04620, Korea; sbc10620@naver.com (B.-C.S.); kanggu12@dongguk.edu (J.-G.K.); aehddn@gmail.com (D.-W.L.); sonbug@dongguk.edu (Y.S.)

* Correspondence: jwjung@dongguk.edu; Tel.: +82-2-2260-3812

Received: 16 January 2019; Accepted: 11 February 2019; Published: 14 February 2019

Abstract: The Expanded Douglas–Peucker (EDP) polygonal approximation algorithm and its application method for the Opposite Angle-Based Exact Cell Decomposition (OAECD) are proposed for the mobile robot path-planning problem with curvilinear obstacles. The performance of the proposed algorithm is compared with the existing Douglas–Peucker (DP) polygonal approximation and vertical cell decomposition algorithm. The experimental results show that the path generated by the OAECD algorithm with EDP approximation appears much more natural and efficient than the path generated by the vertical cell decomposition algorithm with DP approximation.

Keywords: curvilinear obstacle; douglas–peuker polygonal approximation; opposite angle-based exact cell decomposition; path planning; mobile robot

1. Introduction

The path-planning process of a mobile robot aims at finding a collision-free path to move the robot from the current posture to the goal posture [1–3]. If there are multiple available paths, the optimal path in the sense of an objective function, such as the minimum distance, can be chosen. The algorithms for mobile-robot path planning can be grouped into four categories: roadmap approaches, such as the visibility graph or generalized Voronoi graph; cell decomposition approaches, such as the vertical cell decomposition (VCD) or approximate cell decomposition; sampling-based planning methods, such as the rapidly exploring random tree (RRT) or probabilistic roadmap (PRM), and potential field methods [2–5]. Among these methods, roadmap approaches are generally fast and easy to implement, but an intrinsic way to describe environmental information is not provided [1–3]. Sampling-based planning methods are more practical, but they do not provide completeness so we cannot recognize the non-existence of a path [2,3]. Potential field methods are useful to control the robot by generating a differentiable smooth path, but they cannot give explicit information on the roadmap and easily fall into a local minimum [1–3]. There are also some hybrid approaches, such as a potential field with RRT [6] or potential field with cell decomposition [7].

Cell decomposition, which is a classical and representative method for mobile-robot path planning, decomposes the given environment into several cells and finds a collision-free path based on the connectivity graph of these cells [2–10]. Here, each node of a connectivity graph is made by the representative point of each cell or its border line. Each link of the connectivity graph between the nodes indicates that the corresponding cell is adjacent to each other [11]. In various cell decomposition algorithms [12–17], one of the most widely known algorithms is vertical cell decomposition (VCD), but

it does not generate an efficient path because it uses too many cells [8]. Figure 1 shows an example of the previous exact cell decomposition using VCD. VCD makes vertical lines from the convex vertices to decompose the environment into cells. The adjacency relationship among the cells is represented by the connectivity graph and used to find a path using graph search algorithms. VCD does not guarantee the optimality of the number of decomposed cells since there is no consideration of the shape of the obstacle. Reducing the number of decomposed cells directly increases the efficiency in path planning, but finding the optimal decomposition case is known as an NP-hard problem [1–3].

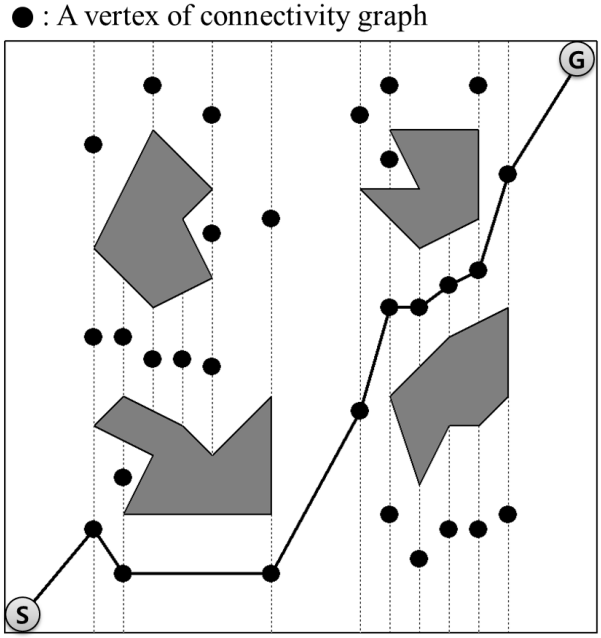


Figure 1. An example of Vertical Cell Decomposition (VCD) and its path.

To supplement this drawback, the Opposite Angle-based Exact Cell Decomposition (OAECD) [18] was proposed by using a type of greedy approach to minimize the number of cells and increase efficiency. However, the OAECD can only be applied for polygonal obstacles, since it is operated based on the relationship among the vertices of the obstacles. In other words, path planning in an environment with curvilinear obstacles is impossible by itself. Therefore, in this paper, a novel expanded polygonal approximation method based on Douglas–Peucker (DP) algorithm is proposed to apply OAECD path planning to the cases with curvilinear obstacles.

In Chapter 2, a novel polygonal approximation algorithm for curvilinear obstacles is addressed. Then, the algorithm is applied to a modified OAECD algorithm in Chapter 3. The experimental results are shown in Chapter 4, and the paper is concluded in Chapter 5.

2. Polygonal Approximation Algorithm of Curvilinear Obstacle

In this chapter, the Douglas–Peucker (DP) algorithm [19], which is one of the most popular algorithms for polygonal approximation, is reviewed. An Expanded Douglas–Peucker (EDP) algorithm for the application with curvilinear obstacles is proposed with mathematical validation on the circumscription of the EDP Algorithm.

2.1. Douglas-Peucker Algorithm

The DP algorithm is a representative method of polygonal approximation. The purpose of a DP algorithm is to find a similar piecewise linear curve with fewer points given a closed curve. The algorithm uses the concept of dissimilarity based on the maximum distance between the original curve points and their simplified piecewise linear curve [17,18].

The algorithm (Algorithm 1) measures the distance between each point of a curve and the base line, which is the line segment with the same first and last points with the curve, to find the farthest point from the line segment with the maximum perpendicular distance.

Here, C is the set of obstacle contours, which is the set of point lists of the obstacle; ε is the threshold value for the maximum dissimilarity tolerance; R is the final result of the polygonal approximation of all obstacles; c is a point list of the obstacle contour, which is arranged in counter-clockwise order; pl is a point list; r_1 and r_2 indicate each result of the recursiveDP procedure; r indicates the final result of the polygonal approximation of an obstacle by DP algorithm.

Algorithm 1. Pseudo Code of the Initial DP.

Input:
$C \leftarrow$ Set of point lists of the obstacle
$\varepsilon \leftarrow$ Threshold value for maximum dissimilarity tolerance
Output:
$R \leftarrow$ Final result of polygonal approximation of all obstacles represented as a set of point lists

Begin ***DP Procedure***
for each c **in** C **do**
  find two points in c, p_1 and p_2, which have the maximum distance from each other and p_1 is in front of p_2
  $r_1 \leftarrow$ ***recursiveDP***($pl[p_1 \ldots p_2]$, $\boldsymbol{\varepsilon}$) // pl: point list of a segment of obstacle contour
  $r_2 \leftarrow$ ***recursiveDP***($pl[p_2 \ldots$ starting point of $c \ldots p_1]$, $\boldsymbol{\varepsilon}$)
  $r \leftarrow$ point list$[r_1[0] \ldots r_1[\text{end}_1 - 1]\ r_2[0] \ldots r_2[\text{end}_2 - 1]]$ // end_i: number of points in r_i
  insert r ***to*** R
end for
End ***DP Procedure***

In the recursive procedure of DP algorithm (Algorithm 2), the line segment is further divided into two sub-line segments using the farthest point as the via-point whenever the maximum perpendicular distance is greater than or equal to the threshold value for maximum dissimilarity tolerance, ε. This process is recursively repeated until the maximum perpendicular distance is less than ε.

Here, r_1 and r_2 are each the result of the recursiveDP procedure, and r is the result of the polygonal approximation of the given curve.

Figure 2 shows the process of the DP algorithm. l_b is the base line, which is identical to $line_{base}$ in Algorithm 2. l_{b1} is determined with the starting point P_1 and ending point P_2, which makes the widest width of the given obstacle. Then, we check whether $dist_{max}$ is less than ε. If $dist_{max}$ is less than ε, two vertices of l_{b1} are inserted into the point list of the polygonal approximation. If $dist_{max}$ is greater than or equal to ε, a new base line l_{b2} with P_1 and P_3 is made, new $dist_{max}$ is found again based on l_{b2}, and the above procedures are repeated.

The obstacle approximation result by DP algorithm does not guarantee the circumscription of the original obstacle. In other words, some interior points of the obstacle region may not be included inside the polygon by the DP algorithm. Similarly, some exterior points of the obstacle region may be included inside the polygon by the DP algorithm.

Thus, if the result of the DP algorithm on a curvilinear obstacle is directly used for path planning, the generated path may penetrate inside the real obstacle region, and the robot may easily collide with the obstacle. To overcome this problem, a modified DP algorithm is proposed in this paper.

Algorithm 2. Pseudo Code of the Recursive DP.

Input:
$pl \leftarrow$ Point list of the given curve
$\varepsilon \leftarrow$ Threshold value for maximum dissimilarity tolerance
Output:
$r \leftarrow$ Result of the polygonal approximation of the given curve represented as a point list
Initialization:
$line_{base} \leftarrow$ line segment connected from pl[0] to pl[end] // end: number of points in pl
$dist_{max} \leftarrow -1$
$i_{max} \leftarrow -1$

Begin ***recursiveDP Procedure***
for each point p in $pl[1 \ldots \text{end}-1]$ ***do***
  $dist \leftarrow$ perpendicular distance between $line_{base}$ and p
  if $dist > dist_{max}$
    $dist_{max} \leftarrow dist$
    $i_{max} \leftarrow$ index of p
  end if
end for
if $dist_{max} >= \varepsilon$ ***then***
  $r_1 \leftarrow$ ***recursiveDP***$(pl[0 \ldots i_{max}], \varepsilon)$ // pl: point list of a segment of the given curve
  $r_2 \leftarrow$ ***recursiveDP***$(pl[i_{max} \ldots \text{end}], \varepsilon)$
  $r \leftarrow$ point list$[r_1[0] \ldots r_1[\text{end}_1 - 1]\ r_2[0] \ldots r_2[\text{end}_2]]$ // end_i: # of points in r_i
else
  insert pl[0] ***to*** r
  insert pl[end] ***to*** r
end if
End ***recursiveDP Procedure***

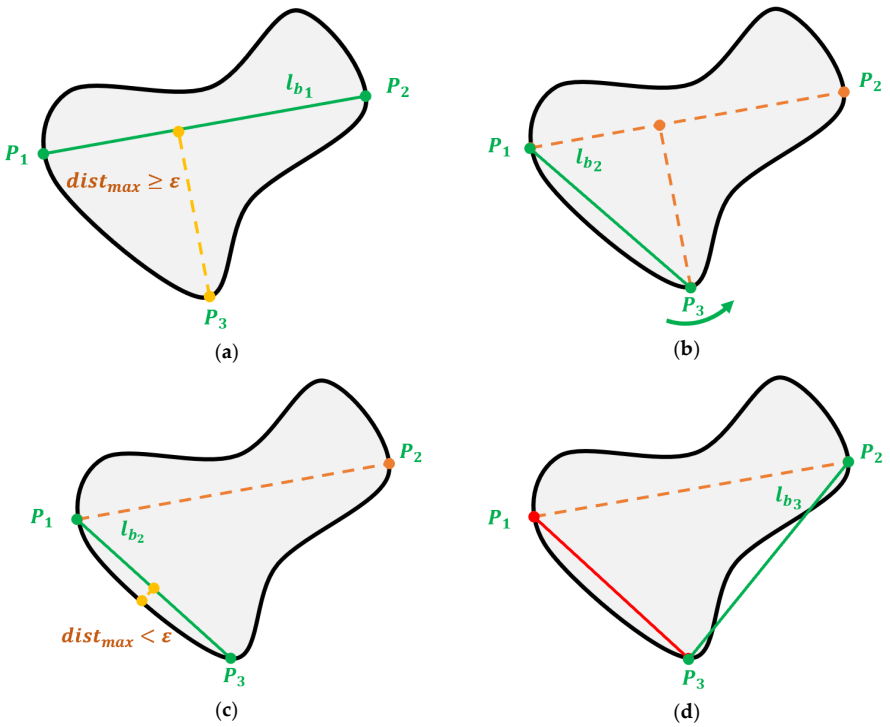


Figure 2. Douglas–Peucker (DP) process between P_1 and P_2: (**a**) If $dist_{max} \geq \varepsilon$, then (**b**) l_{b2} between P_1 and $dist_{max}$ point P_3 is used for the next base line; (**c**) If $dist_{max} < \varepsilon$, then (**d**) l_{b3} between P_3 and P_2 is used for the next base line.

2.2. Proposed Expanded Douglas-Peucker (EDP) Algorithm

The path planning for curvilinear obstacles using DP algorithm may not be collision-free. The basic philosophy of the EDP algorithm is to guarantee the circumscription of obstacles by expanding the polygon of the DP algorithm with the maximum dissimilarity tolerance. In addition, by appending additional points near the convex corner, the convex corner clearance is considered, where the robot control may become more difficult during path following.

Figure 3 shows the operation of EDP based on the counter-clockwise traversal. The first step is to perform the DP algorithm. If the half angle between line segments l_0 and l_1 is 0–90°, i.e., the corner of DP polygon is convex, the perpendicular half-lined to l_0 and l_1 are v_0 and v_1 at the corner point, respectively. Then, the points on the arc, with the center point as the convex corner and the radius of ε, are appended starting from v_0 and rotating with angle θ_{rot} until they meet v_1 in the counter-clockwise direction. These points are added for the convex corner clearance. If the half angle between line segments l_0 and l_1 is 90° or more, i.e., the corner of DP polygon is concave, the point far from the cross point of l_0 and l_1 with distance ε to the outer direction of the obstacle is appended. The above procedures are repeated for the remaining corner points of the DP polygon.

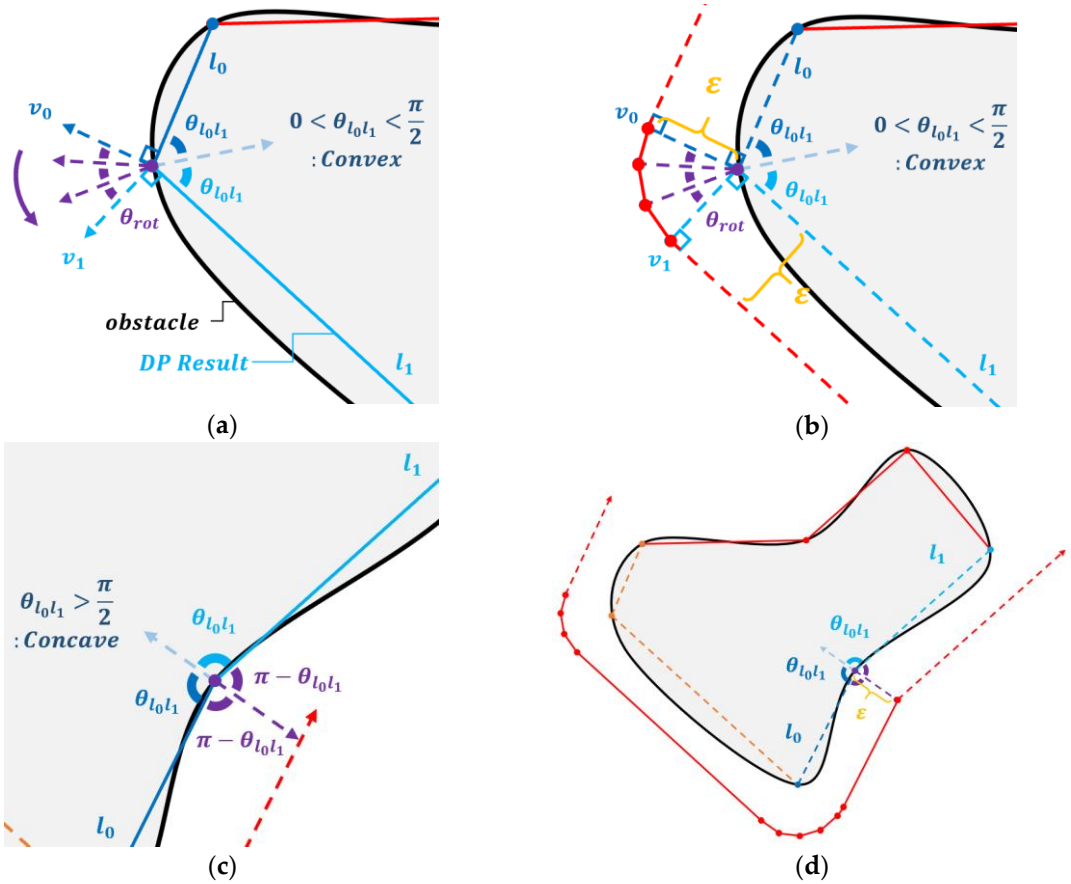


Figure 3. Abstract process of Expanded Douglas–Peucker (EDP) algorithm: (**a**) Before applying EDP for a convex corner; (**b**) After applying EDP for a convex corner; (**c**) Before applying EDP for a concave corner; (**d**) Overall appearance after applying EDP for a concave corner.

Algorithm 3 shows the abstract version of the pseudo code of the EDP algorithm.

Algorithm 3. Pseudo Code of EDP (Abstract version).

Input:
R_{DP} ← Result of DP polygonal approximation of all obstacles represented as a set of point lists
ε ← Threshold value for maximum dissimilarity tolerance
θ_{rot} ← Constant angle for obstacle corner clearance
Output:
R ← Final result of the EDP polygonal approximation represented as a set of point lists

Begin ***Expanded-DP Procedure***
for each r_{DP} ***in*** R_{DP} ***do***
  R ← null
  for each point p in r_{DP} ***do***
    if p is convex vertex ***then***
      θ ← half inner angle of p
      $dist$ ← ε
      l_0 ← line segment connected from the previous point of p to p
      l_1 ← line segment connected from p to the next point of p
      v_0 ← perpendicular half-line to l_1 started from p
      v_1 ← perpendicular half-line to l_2 started from p
      loop
        insert points p_{cv} on the arc with center point p and radius ε, which is consisted of the points according to θ_{rot} angle from v_0 to v_1, ***to*** R
      end loop
    else
      θ ← half outer angle of p
      $dist$ ← $\varepsilon/\sin\theta$
      insert point p_{cc} far from p with distance $dist$ to the outer direction of the obstacle ***to*** R
    end if
  end for
end for
End ***Expanded-DP Procedure***

Here, R_{DP} is the set of point lists of the polygonal approximation of the obstacles by the DP algorithm; ε is the threshold value for maximum dissimilarity tolerance; θ_{rot} is a constant angle for obstacle corner clearance; R is the final result of the polygonal approximation of all obstacles by the EDP algorithm, and r_{DP} is a point list of the obstacle polygonal approximation by the DP algorithm, which is arranged in counter-clockwise order.

Figure 4 illustrates the detailed version of the EDP algorithm (Algorithm 4).

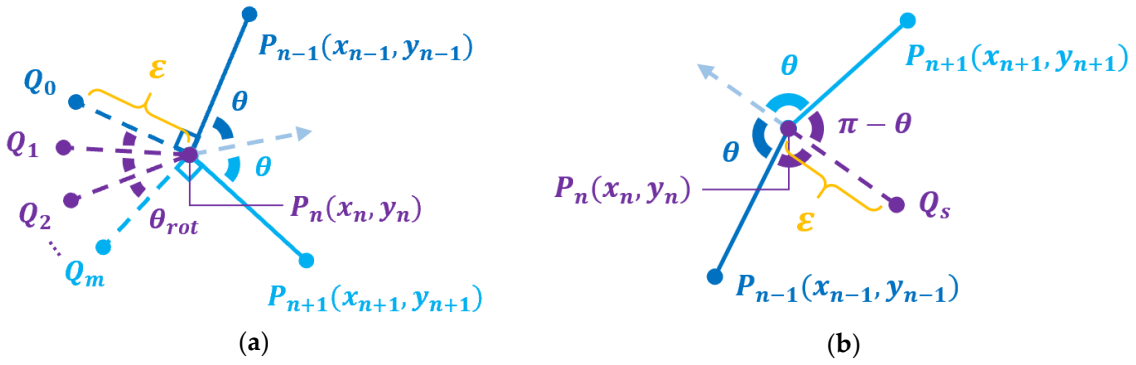


Figure 4. Detailed process of the EDP: (**a**) Convex; (**b**) Concave.

Since $\overrightarrow{P_{n-1}P_n} = (x_n - x_{n-1},\ y_n - y_{n-1})$ and $\overrightarrow{P_{n-1}P_n} \perp \overrightarrow{P_nQ_0}$,
unit vector of

$$\overrightarrow{P_nQ_0} = \frac{1}{\sqrt{(y_n - y_{n-1})^2 + (x_n - x_{n-1})^2}}(y_n - y_{n-1},\ -(x_n - x_{n-1}))^T \tag{1}$$

Therefore,

$$Q_0 = \frac{\varepsilon}{\sqrt{(y_n - y_{n-1})^2 + (x_n - x_{n-1})^2}}(y_n - y_{n-1},\ -(x_n - x_{n-1}))^T \tag{2}$$

In addition, since Q_i (i = 1, 2, . . . , $m - 1$) are on the arc with center point P_n and radius ε, rotating Q_i with angle θ_{rot} in the counter-clockwise direction yields

$$Q_i = \begin{pmatrix} \cos\theta_{rot} & -\sin\theta_{rot} \\ \sin\theta_{rot} & \cos\theta_{rot} \end{pmatrix}(Q_{i-1} - P_n) + P_n \tag{3}$$

Similarly, in the case of a concave corner,

$$\overrightarrow{OP_n} + \overrightarrow{P_nQ_s} = (x_n, y_n)^T + \begin{pmatrix} \cos(\pi - \theta) & -\sin(\pi - \theta) \\ \sin(\pi - \theta) & \cos(\pi - \theta) \end{pmatrix}(x_{n-1} - x_n,\ y_{n-1} - y_n)^T \tag{4}$$

Around the convex vertex of DP polygon, the concept of θ_{rot} is used for the convex corner clearance. When $\theta_{rot} \geq \angle Q_0P_nQ_m$, there is no additionally appending vertex, and $Q_m = Q_1$; when $\theta_{rot} < \angle Q_0P_nQ_m$, there are additionally appending vertices $Q_1, Q_2, \ldots$, which makes the effect of securing additional free space near the obstacle corner relatively difficult to path following.

Algorithm 4 shows the detailed version of the pseudo code of the EDP algorithm.

Algorithm 4. Pseudo Code of EDP (Detailed version).

Input:
$R_{DP} \leftarrow$ Result of DP polygonal approximation of all obstacles represented as a set of point lists
$\varepsilon \leftarrow$ Threshold value for maximum dissimilarity tolerance
$\theta_{rot} \leftarrow$ Constant angle for obstacle corner clearance
Output:
$R \leftarrow$ Final result of the EDP polygonal approximation represented as a set of point lists

Begin ***Expanded-DP Procedure***
for each r_{DP} in R_{DP} **do**
  for each point $P_n(x_n, y_n)$ in point list r_{DP} ***do***
    if $\angle P_{n+1}P_nP_{n+1} < \pi$ ***then***
      $\theta \leftarrow \frac{1}{2}\angle P_{n+1}P_nP_{n+1}$
      $dist \leftarrow \varepsilon$
      $Q_0 \leftarrow (x_n, y_n)^T + \frac{\varepsilon}{\sqrt{(y_n - y_{n-1})^2 + (x_n - x_{n-1})^2}}(y_n - y_{n-1},\ -(x_n - x_{n-1}))^T$
      $Q_m \leftarrow (x_{n+1}, y_{n+1})^T + \frac{\varepsilon}{\sqrt{(y_{n+1} - y_n)^2 + (x_{n+1} - x_n)^2}}(y_{n+1} - y_n,\ -(x_{n+1} - x_n))^T$
      loop until $i \leq \frac{\pi - 2\theta}{\theta_{rot}}$
        $Q_i \leftarrow \begin{pmatrix} \cos\theta_{rot} & -\sin\theta_{rot} \\ \sin\theta_{rot} & \cos\theta_{rot} \end{pmatrix}(Q_{i-1} - P_n) + P_n$
        insert Q_i to R
        $i \leftarrow i + 1$
      end loop
    else
      $\theta \leftarrow \frac{1}{2}\angle P_{n+1}P_nP_{n+1}$
      $dist \leftarrow \varepsilon / \sin\theta$
      $Q_s \leftarrow \overrightarrow{OP_n} + \overrightarrow{P_nQ_s} = (x_n, y_n)^T + \begin{pmatrix} -\cos(\theta) & -\sin(\theta) \\ \sin(\theta) & -\cos(\theta) \end{pmatrix}(x_{n-1} - x_n,\ y_{n-1} - y_n)^T$
      insert Q_s ***to*** R
    end if
  end for
end for
End ***Expanded-DP Procedure***

2.3. Mathematical Validation on the Circumscription of the EDP Algorithm

[*Theorem*] The polygon that consists of the resulting points from the expanded DP (EDP) algorithm on an obstacle includes all points of the original obstacle, i.e., no point of the obstacle region is outside of the polygon that consists of the resulting points from EDP on that obstacle.

(Proof) Let P_n be the n-th point from the DP polygonal approximation of an obstacle. Then, the following propositions are always true by the DP algorithm.

[P1] P_{n-1}, P_n, P_{n+1} are points included in the boundary line of the obstacle.

[P2] P_n is the farthest point from the line segment $\overline{P_{n-1}P_{n+1}}$ among the boundary points of the obstacle in [P_{n-1}, P_{n+1}].

[P3] Every boundary point of the obstacle in [P_{n-1}, P_n] has a shorter distance than ε from the line segment $\overline{P_{n-1}P_n}$.

[P4] Every boundary point of the obstacle in [P_n, P_{n+1}] has a shorter distance than ε from the line segment $\overline{P_nP_{n+1}}$

If P_n is a convex point.

Let us assume that there is a boundary point $c \in [P_{n-1}, P_{n+1}]$ of the obstacle such that c is located outside the polygon Q_m, Q'_0, Q'_1, ... , Q'_m, Q''_0 such as Figure 5a, which consists of the resulting points from the EDP algorithm on the obstacle. Since P_{n-1}, P_n, and P_{n+1} are points on the boundary

line of the obstacle by proposition [P1], boundary point c should be identical to P_n or included in the range of $[P_{n-1}, P_n]$ or (P_n, P_{n+1}).

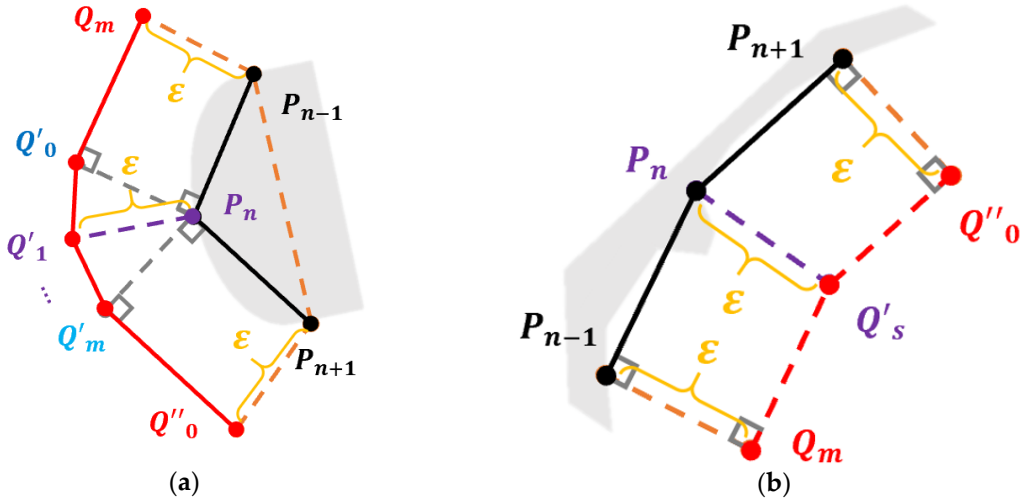


Figure 5. Circumscription of EDP: (**a**) Convex; (**b**) Concave.

If point c is identical to P_n, it is trivial that c cannot be located outside of the polygon with vertices Q_m, Q'_0, Q'_1, ... , Q'_m, and Q''_0, since P_n is inside the polygon.

If point c is included in the range of $[P_{n-1}, P_n)$, the distance from point c to the line segment $\overline{P_{n-1}P_n}$ should be less than ε by proposition [P3]. Therefore, c cannot be located outside of the polygon with vertices $Q_m \sim Q'_0$.

If point c is included in the range of $(P_n, P_{n+1}]$, the distance from point c to the line segment $\overline{P_nP_{n+1}}$ should be less than ε by proposition [P4]. Therefore, c cannot be located outside of the polygon with vertices $Q'_m \sim Q''_0$.

Therefore, no boundary point of the obstacle region is outside of the polygon with vertices Q_m, Q'_0, Q'_1, ... , Q'_m, and Q''_0 in the case of convex P_n.

If P_n is a concave point,

Let us assume that there is a boundary point $c \in [P_{n-1}, P_{n+1}]$ of the obstacle such that c is located outside of the polygon Q_m, Q'_s, Q''_0 such as Figure 5b, which consists of the resulting points from the EDP algorithm on the obstacle. Since P_{n-1}, P_n, and P_{n+1} are points on the boundary line of the obstacle by proposition [P1], boundary point c should be identical to P_n or included in the range of $[P_{n-1}, P_n]$ or (P_n, P_{n+1}).

If point c is identical to P_n, it is trivial that c cannot be located outside of the polygon with vertices Q_m, Q'_s, and Q''_0, since P_n is inside the polygon.

If point c is included in the range of $[P_{n-1}, P_n)$, the distance from point c to the line segment $\overline{P_{n-1}P_n}$ should be less than ε by proposition [P3]. Therefore, c cannot be located outside of the polygon with vertices Q_m and Q'_s.

If point c is included in the range of $(P_n, P_{n+1}]$, the distance from point c to the line segment $\overline{P_nP_{n+1}}$ should be less than ε by proposition [P4]. Therefore, c cannot be located outside of the polygon with vertices Q'_s and Q''_0.

Therefore, no boundary point of the obstacle region is outside the polygon with vertices Q_m, Q'_s, and Q''_0 in the case of concave P_n (*Q.E.D.*).

Once the polygonal approximation with the EDP algorithm is completed, it is possible to solve the obstacle collision problem that may occur when DP algorithm is used for the polygonal approximation of a curvilinear obstacle. Since the polygon created by the EDP algorithm can guarantee the circumscription of obstacles, this result of EDP can be used to apply OAECD in path planning with curvilinear obstacles.

3. Modified Opposite Angle-Based Exact Cell Decomposition (OAECD) Algorithm for Path Planning with Curvilinear Obstacles

The basic concept of the modified OAECD algorithm is to reduce as possible as many cells after the execution of the EDP algorithm and increase the calculation efficiency. Using the concept of inclusion of an opposite angle, OAECD first tries to connect the closest neighboring cell in the opposite angle region. OAECD does not randomly or sequentially process the points but processes in the order from a close pair to a far pair through three consecutive steps, which are similar to the human method. In addition, OAECD does not try to make an additional decomposition if the shape of the generating cell is convex.

The detailed algorithm for the modified OAECD consists of three steps [18]. Figure 6 shows how the decomposing lines are formed between the obstacles for each step of the modified OAECD algorithm. In the first step, the modified OAECD finds the closest neighboring vertex in the set of all vertices of other obstacles for every convex vertex (CV) of each obstacle. For every CV of each obstacle and its closest neighboring vertex (NV) in other obstacles, a new decomposing line is drawn if there is no existing decomposing line with the current CV, its closest NV is inside the region of the opposite angle of the current CV, and no intersection is made with other obstacles or other decomposing lines.

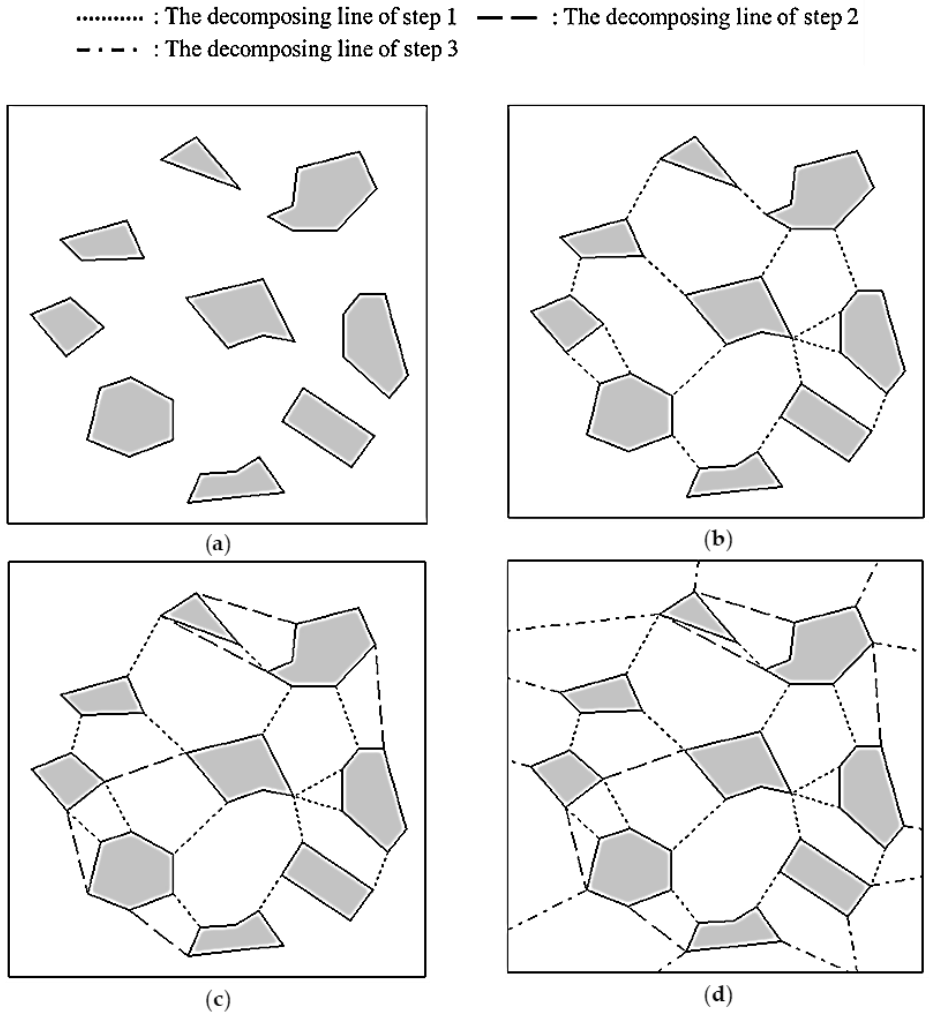


Figure 6. Decomposition results of each step in the modified Opposite Angle-Based Exact Cell Decomposition (OAECD) algorithm after the EDP algorithm: (**a**) Environment; (**b**) Result of step 1; (**c**) Result of step 2; (**d**) Result of step 3.

In the second step, for every CV among the remaining vertices and a vertex (AV) in another obstacle inside the region of the opposite angle of the current CV, a new decomposing line is drawn if there is no decomposing line with the current CV, the AV in another obstacle is the shortest one from the current CV among all available AVs, and there is no intersection with other obstacles or other decomposing lines.

Finally, in the third step, for every CV of the remaining vertices, a new decomposing line is drawn in the direction of the equiangular line of the opposite angle of the current CV until it intersects with the boundary of the environment, obstacle, or another decomposing line, if no decomposing line is connected with the current CV or more than one decomposing line is already connected from another CV, and the created angle near the current CV is larger than 180°.

The OAECD can be properly applied in a static environment. Although a single-sensed map may generate noise, it can sense multiple times for the same environment in a short period and remove the noise on the map by the moving average method [20] or the median method [21]. Based on these methods, one can plan the route. However, this process is only accurate in the static environment. In the dynamic environment, it is difficult to resolve the sensor noise completely by the moving average method or the median method for the same position because the obstacle moves.

Algorithms 5–7 show the pseudo code of the modified OAECD algorithm for each step in the process. Here, v_{c1} is a convex vertex of an obstacle; V_c is the set of point lists, which is the result of the EDP polygonal approximation of all obstacles; v_{a1} is the closest neighboring vertex from v_{c1}; V is the set of all vertices of the obstacles; V_{check} is the point set to check for processing.

Algorithm 5. Pseudo Code of the modified OAECD (Step 1).

Input:
$V_c \leftarrow R$ // R: Result of the EDP polygonal approximation of all obstacles represented as a set of point lists
$M \leftarrow$ Environment map
Output:
$V_{chec} \leftarrow$ Point set to check for processing
$M \leftarrow$ Environment map with decomposing lines of step 1

Begin ***OAECD-Algorithm Procedure***
for every convex vertex of each obstacle in V_c ***do***
    find the closest vertex in the set of all vertices of other obstacles by comparing the distances between the current vertex and other vertices
end for
for each v_{c1} in V_c ***do***
    if there is no decomposing line that is already connected with v_{c1} ***then***
        if v_{a1} is included in the region of the opposite angle of v_{c1} and ***line***(v_{c1}, v_{a1} in V) is not intersected with other obstacles or other lines ***then***
            draw decomposing line from v_{c1} ***to*** v_{a1} and ***insert*** v_{c1} ***to*** V_{check}
        else if
    else if all angles near v_{c1} are less than or equal to 180° ***then***
        insert v_{c1} ***to*** V_{check}
    end if
end for
End ***OAECD-Algorithm Procedure***

Generally, the cells created by the cell decomposition methods are shaped as a convex polygon to avoid being penetrated by obstacles. Therefore, only the convex vertices of the obstacles are considered in this step because the concave vertices of the obstacles obviously result in convex vertices of the free cell. In other words, there is no need to make an additional decomposing line from some vertices if they preserve the concaveness. In addition, the closest neighboring vertex is chosen to reduce the

number of possible cells and the possibility of crossing with other decomposing lines. Here, v_{c2} is a convex vertex of an obstacle in V_c-V_{check}, v_{a2} is a vertex in the region of the opposite angle of v_{c2}, and V_i is the set of all vertices inside the region of the opposite angle of v_{c2}.

Algorithm 6. Pseudo Code of the modified OAECD (Step 2).

Input:
$V_c \leftarrow R//R$: Result of the EDP polygonal approximation of all obstacles represented as a set of point lists
$M \leftarrow$ Environment map
$V_{check} \leftarrow$ Point set to check for processing (Result of previous step 1)
Output:
$V_{check} \leftarrow$ Point set to check for processing (Result of this step)
$M \leftarrow$ Environment map with the decomposing lines of step 1 and 2

Begin ***OAECD-Algorithm Procedure***
for each v_{c2} in V_c - V_{check} ***do***
  if there is no decomposing line that is already connected with v_{c2} ***then***
    if v_{a2} has the shortest distance with v_{c2} compared with those in V_i - $\{v_{a2}\}$ and ***line***(v_{c2}, v_{a2} in V_i) is not intersected with the obstacle and other lines ***then***
      draw decomposing line from v_{c2} ***to*** v_{a2} and ***insert*** v_{c2} ***to*** V_{check}
    end if
  else if all angles near v_{c2} are less than or equal to 180° ***then***
    insert v_{c2} ***to*** V_{check}
  end if
end for
End ***OAECD-Algorithm Procedure***

From Step 2, the decomposing line from v_{c2} to v_{a2} is drawn, and v_{c2} is inserted into V_{check} although v_{a2} is not the closest vertex in V. Here, v_{c3} is an unchecked convex vertex of each obstacle.

Algorithm 7. Pseudo Code of the modified OAECD (Step 3).

Input:
$V_c \leftarrow R//R$: Result of the EDP polygonal approximation of all obstacles represented as a set of point lists
$M \leftarrow$ Environment map
$V_{check} \leftarrow$ Point set to check for processing (Result of previous step 2)
Output:
$M \leftarrow$ The final environment map with decomposing lines of steps 1, 2, and 3

Begin ***OAECD-Algorithm Procedure***
for each v_{c3} in V_c - V_{check} ***do***
  if there is no decomposing line that is already connected with v_{c3} ***then***
    draw decomposing line from v_{c3} in the direction of half angle of the opposite angle of v_{c3} until it intersects with the boundary of the environment, other obstacles, or other lines and ***insert*** v_{c3} ***to*** V_{check}
  else if all angles near v_{c3} are less than or equal to 180° ***then***
    insert v_{c3} ***to*** V_{check}
  end if
end for
End ***OAECD-Algorithm Procedure***

The time complexity of the modified OAECD is $O(n^2)$ because the sub-time complexity is basically all $O(n^2)$ for Step 1, Step 2, and Step 3. In addition, the time complexity for Step 3 can be reduced to $O(nlogn)$ approximately when the sweep-line method [22,23] is applied.

4. Experimental Results

Two types of experiments were conducted to find the performance of the proposed EDP and modified OAECD algorithm. An additional simulation has been performed to verify the feasibility of the modified OAECD algorithm for a map with curvilinear obstacles. The first experiment is conducted to compare the approximation error and the number of vertices of the approximated polygon made by EDP and DP. In each experiment, twenty maps were chosen in the pre-created one hundred fifty random maps, each of which is assumed to be 20×20 m^2 in size and randomly have the position of the obstacle, position of the vertices of each obstacle, and area of the obstacle by using the random function in the math library of the MS Visual C++ compiler. The number of obstacles and vertices of each obstacle were fixed at fifteen. Bezier curve was used for the curvilinear obstacle representation by interconnecting the vertices of each obstacle and creating naturally curved obstacles. Each map was created by an image in bitmap format.

The second experiment was conducted to compare the performance of the OAECD algorithm and the VCD algorithm. Similar to the first experiment, the size of the map was assumed as 20×20 m^2, and the position of the obstacle, positions of the vertices of each obstacle, number of vertices of each obstacle, and number of obstacles in the map were random variables.

Table 1 shows the experimental results by averaging the results of each experiment. Here, IA is the percentage area of the inner space of the approximated polygons outside the obstacle regions in comparison with the area of the original obstacle region. OA is the percentage area of the outer space of the approximated polygons inside the obstacle regions in comparison with the area of the original obstacle region. SA is the sum of IA and OA. AVG is the average values for various ε.

Table 1. Performance comparison between Expanded Douglas–Peucker (EDP) algorithm and Douglas–Peucker (DP) alogrithm for various ε values (The number of obstacles and vertices of each obstacle were fixed as 15, and θ_{rot} for EDP was fixed as 30°).

ε (m)	EDP (%)			DP (%)		
	IA	OA	SA	IA	OA	SA
0.05	19.05	0.00	19.05	2.45	0.70	3.15
0.08	28.46	0.00	28.46	4.89	1.17	6.06
0.11	37.95	0.00	37.95	7.19	1.54	8.73
0.14	47.63	0.00	47.63	9.26	1.83	11.08
0.17	57.34	0.00	57.34	11.50	2.08	13.58
0.20	67.17	0.00	67.17	13.38	2.48	15.86
0.23	77.11	0.00	77.11	15.52	2.76	18.28
0.26	87.01	0.00	87.01	17.23	3.20	20.43
0.29	96.91	0.00	96.91	19.55	3.56	23.10
0.32	106.7	0.00	106.7	22.00	3.86	25.85
0.35	116.7	0.00	116.7	23.90	4.40	28.30
0.38	127.3	0.00	127.3	26.91	4.83	31.74
0.41	137.9	0.00	137.9	27.99	5.27	33.26
0.44	148.0	0.00	148.0	30.57	5.62	36.19
0.47	158.8	0.00	158.8	32.13	5.86	37.99
0.50	169.5	0.00	169.5	33.11	6.60	39.72
AVG	92.72	0.00	92.72	15.53	2.99	18.52

From Table 1, the average approximation error of the result of the EDP algorithm is approximately 5 times larger than that of DP algorithm to cover the entire area of the original obstacle regions.

Table 2 shows the experimental results by averaging the results of ten experiments with 15 random obstacles and 15 random vertices for each obstacle.

From Table 2, on average, the result of the EDP algorithm has 2.88 times more vertices than the result of the DP algorithm when θ_{rot} for EDP is 30°.

Figure 7 shows a graphical representation of Tables 1 and 2. The approximation performance of EDP is worse than DP in the aspects of IA, SA, and average number of vertices. Nonetheless, OA of

EDP is completely zero and better than DP since the result of the EDP can cover the entire area of the original obstacle regions.

Table 2. Average number of vertices created by EDP and DP (The numbers of obstacles and vertices of each obstacle were fixed as 15, and θ_{rot} for EDP was fixed as 30°).

ε (m)	Average Number of Vertices	
	EDP	DP
0.05	31.08	15.92
0.08	27.87	11.92
0.11	25.89	10.19
0.14	24.68	9.18
0.17	23.67	8.33
0.20	22.85	7.66
0.23	22.13	7.32
0.26	21.55	6.83
0.29	20.97	6.56
0.32	20.15	6.20
0.35	19.32	5.91
0.38	18.97	5.58
0.41	18.63	5.40
0.44	18.06	5.06
0.47	17.65	5.00
0.50	17.16	4.78
AVG	21.91	7.65

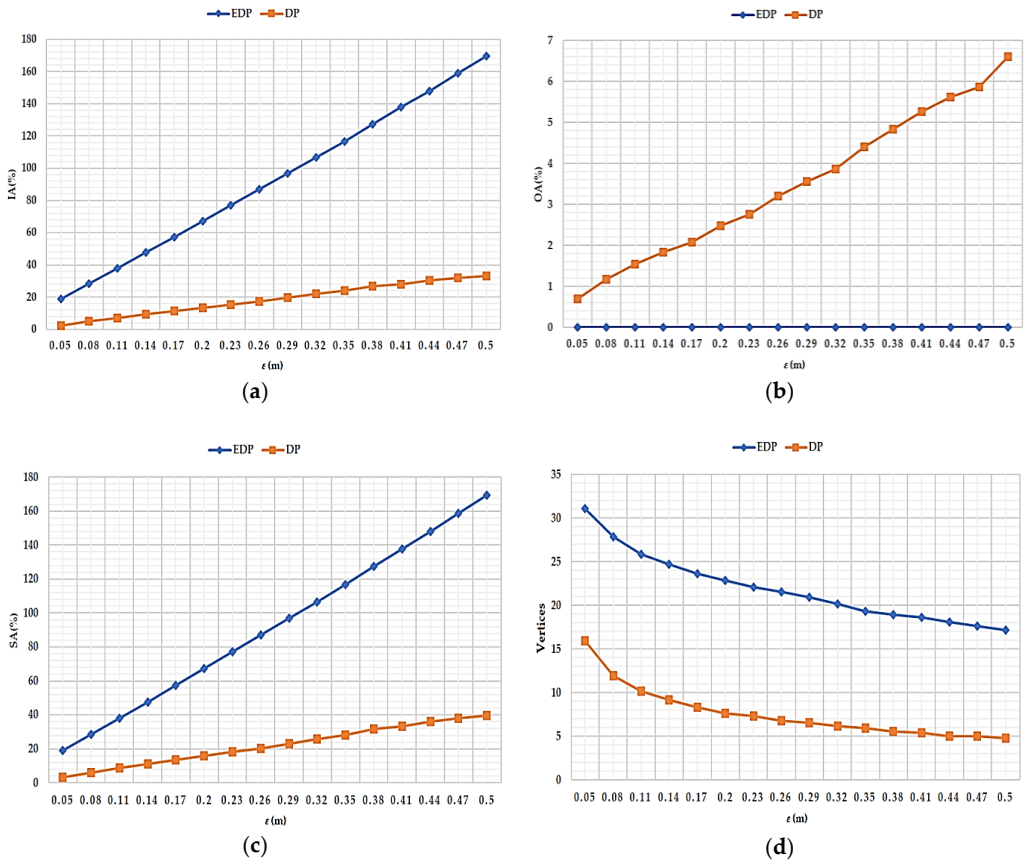


Figure 7. Performance comparison between EDP and DP for various ε values: (**a**) Ratio of the inner space of the approximated polygons outside the obstacle region (IA); (**b**) Ratio of the outer space of the approximated polygons inside the obstacle region (OA); (**c**) Sum of IA and OA (SA); (**d**) Average number of vertices created by EDP and DP (The numbers of obstacles and vertices of each obstacle were fixed as 15, and θ_{rot} for EDP was fixed as 30°).

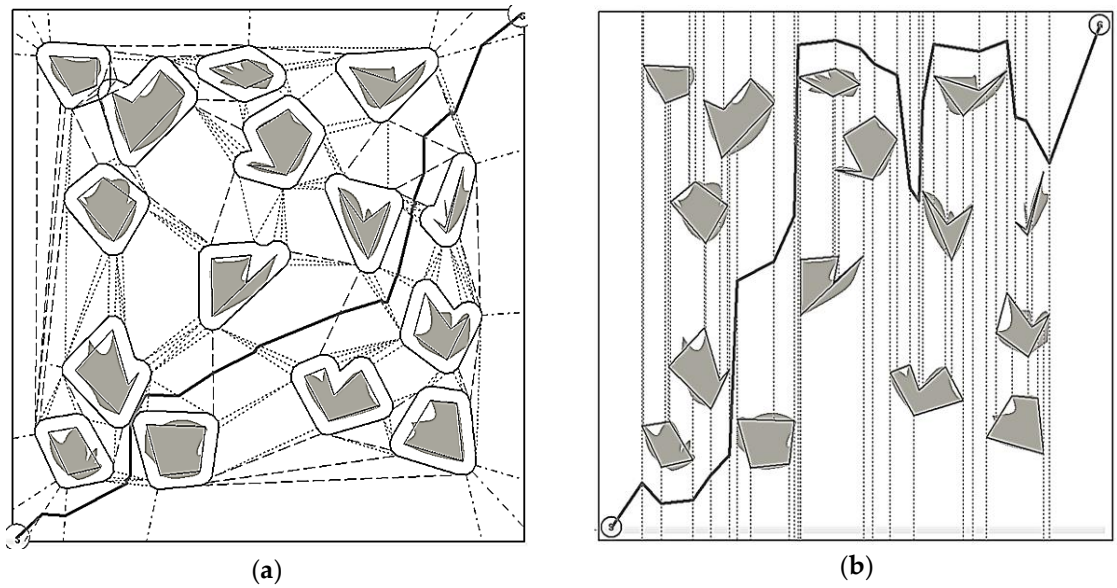
Table 3 compares the performance of the modified OAECD algorithm and VCD algorithm. Here, NV is the total number of vertices in the connectivity graph, VV is the number of vertices that composes the generated path by the A* algorithm, and PL is the path length.

Table 3. Performance comparison between modified Opposite Angle-Based Exact Cell Decomposition (OAECD) and Vertical Cell Decomposition (VCD) (The numbers of obstacles and vertices of each obstacle were fixed as 15).

(NO, NV)	OAECD			VCD		
	NV (ea)	VV (ea)	PL (m)	NV (ea)	VV (ea)	PL (m)
(1,10)	7.60	4.30	28.43	12.00	7.00	29.27
(3,10)	19.80	6.00	29.16	36.60	16.50	33.43
(5,10)	31.70	10.20	29.85	61.80	24.70	37.84
(7,10)	44.10	10.60	30.37	88.20	29.90	45.89
(9,10)	56.80	11.90	29.68	111.80	34.60	47.90
(11,10)	68.50	13.40	30.88	139.00	36.80	53.64
(13,10)	79.20	14.90	30.62	164.70	43.80	56.38
(15,10)	92.50	17.60	30.92	192.00	45.70	48.31
(15,3)	40.70	9.80	28.88	76.60	28.90	57.03
(15,5)	54.90	11.10	29.72	108.00	29.80	51.94
(15,7)	69.20	13.80	29.83	141.20	37.70	59.94
(15,9)	84.70	15.70	30.48	174.60	43.50	54.79
(15,11)	100.60	17.60	30.82	207.80	43.00	53.86
(15,13)	116.10	21.40	31.26	249.50	46.90	58.88
(15,15)	130.60	20.30	31.32	279.20	43.30	55.28
AVG	66.47	13.24	30.15	136.20	34.14	49.63

From Table 3, the path length by the modified OAECD algorithm is 60.7% shorter than the path length by VCD algorithm on average.

Figure 8 shows the comparison of VCD with DP and OAECD with EDP to show the feasibility of the modified OAECD algorithm for maps with static curvilinear obstacles.



(**a**) (**b**)

Figure 8. Comparison between OAECD with EDP and VCD with DP (ε = 0.05 m): (**a**) Path planning using EDP and OAECD; (**b**) Path planning using DP and VCD (The numbers of obstacles and vertices of each obstacle were fixed as 15, and θ_{rot} for EDP was fixed as 30°).

The EDP shows that the polygonal approximation completely covers each obstacle. In addition, the OAECD shows a more natural-looking and efficient path than VCD in Figure 8a. The path from VCD with the DP algorithm may also easily collide with the obstacle, such as Figure 8b.

Even though the path in Figure 8a is more natural looking than that in Figure 8b, there are still some sharp corners which may occur generating an additional problem of motion control for real-world robots due to kinematic constraints. These corners can be smoothed by some additional path smoothing techniques [5,24].

5. Conclusions

In this paper, the Expanded Douglas–Peucker (EDP) polygonal approximation algorithm and its application method for the Opposite Angle-based Exact Cell Decomposition (OAECD) are proposed for the mobile-robot path-planning problem with curvilinear obstacles. In addition, mathematical analysis has been conducted to guarantee the circumscription of obstacle regions by the EDP approximation. Since OAECD is basically focused on the static environment, OAECD is not robust enough to sensor noises in the dynamic environment. Nonetheless, the proposed method is useful with both polygonal and curvilinear obstacles, since the EDP approximation can guarantee the circumscription of obstacle regions, and the modified OAECD algorithm can effectively reduce the number of decomposing cells.

The experimental results show that the path generated by the OAECD algorithm with the EDP approximation looks much more natural and is collision-free. That path is also more efficient than the path generated by the VCD algorithm with DP approximation, although on average, the EDP approximation may induce a larger approximation error and more approximation vertices than DP approximation.

Author Contributions: Idea and conceptualization: J.-W.J. and S.-B.C.; methodology: J.-W.J. and S.-B.C.; software: S.-B.C. and Y.S.; experiment: S.-B.C., J.-.G.K. and D.-W.L.; validation: J.-W.J.; investigation: S.-B.C. and J.-G.K.; resources: J.-W.J.; writing: J.-W.J., S.-B.C., J.-G.K., D.-W.L. and Y.S.; visualization: S.-B.C. and J.-G.K.; project administration: J.-W.J.

Funding: This research was supported by the Ministry of Science and ICT, Korea, under the National Program for Excellence in Software supervised by the Institute for Information & Communications Technology Promotion (2016-0-00017), the KIAT (Korea *Institute* for Advancement of Technology) grant funded by the Korea Government (MOTIE: Ministry of Trade Industry and Energy) (No. N0001884, HRD program for Embedded Software R&D) and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2018R1A5A7023490).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Latombe, J.-C. *Robot Motion Planning*; Kluwer Academic Publishers: Dordrecht, The Netherlands, 1991.
2. Choset, H.; Lynch, K.M.; Hutchinson, S.; Kantor, G.; Burgard, W.; Kavraki, L.E.; Thrun, S. *Principles of Robot Motion: Theory, Algorithms, and Implementations*; MIT Press: Boston, MA, USA, 2005.
3. La Valle, S.M. *Planning Algorithms*; Cambridge University Press: Cambridge, UK, 2006.
4. Yang, L.; Qi, J.; Song, D.; Xiao, J.; Han, J.; Xia, Y. Survey of robot 3D path planning algorithms. *J. Control Sci. Eng.* **2016**, *5*, 22–44. [CrossRef]
5. Gonzalez, D.; Perez, J.; Milanes, V.; Nashashibi, F. A Review of Motion Planning Techniques for Automated Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2016**, *17*, 1135–1145. [CrossRef]
6. Yang, H.; Jia, Q.; Zhang, W. An Environmental Potential Field Based RRT Algorithm for UAV Path Planning. In Proceedings of the 2018 37th Chinese Control Conference (CCC), Wuhan, China, 25–27 July 2018; pp. 9922–9927.
7. Yanyi, Y.; Yingming, Z.; Xingchen, L. An improved artificial potential field algorithm based on nonuniform cell decomposition. In Proceedings of the 2017 6th International Conference on Measurement, Instrumentation and Automation (ICMIA 2017), Zhuhai, China, 29–30 June 2017.

8. Avnaim, F.; Boissonnat, J.D.; Faverjon, B. A practical exact motion planning algorithm for polygonal objects amidst polygonal obstacles. In Proceedings of the IEEE International Conference on Robotics and Automation, Philadelphia, PA, USA, 24–29 April 1988.
9. Brooks, R.A.; Lozano-Perez, T. A subdivision algorithm in configuration space for find path with rotation. *IEEE Trans. Syst.* **1985**, *15*, 224–233.
10. Iswanto, I.; Oyas, W.; Imam, C.A. Quadrotor Path Planning Based on Modified Fuzzy Cell Decomposition Algorithm. *Telecommun. Comput. Electron. Control* **2016**, *14*, 655–664. [CrossRef]
11. Nora, S.; Tschichold-Gurmann, N. *Exact Cell Decomposition of Arrangements Used for Path Planning in Robotics*; Technical Report; ETH Zürich, Department of Computer Science: Zürich, Switzerland, 1999.
12. Zhang, L.; Kim, Y.J.; Manocha, D. A hybrid approach for complete motion planning. In Proceedings of the International Conference on Intelligent Robots and Systems, San Diego, CA, USA, 29 October–2 November 2007.
13. Kim, J.-T.; Kim, D.-J. New Path Planning Algorithm based on the Visibility Checking using a Quad-tree on a Quantized Space, and its improvements. *J. Inst. Control Robot. Syst.* **2010**, *16*, 48. [CrossRef]
14. Arney, T. An efficient solution to autonomous path planning by Approximate Cell Decomposition. In Proceedings of the 3rd International Conference on Information and Automation for Sustainability, Melbourne, Australia, 4–6 December 2007.
15. Rosell, J.; Iniguez, P. Path planning using Harmonic Functions and Probabilistic Cell Decomposition. In Proceedings of the IEEE International Conference on Robotics and Automation, Barcelona, Spain, 18–22 April 2005.
16. Cowlagi, R.V.; Tsiotras, P. Beyond quadtrees: Cell decompositions for path planning using wavelet transforms. In Proceedings of the 46th IEEE Conference on Decision and Control, New Orleans, LA, USA, 12–14 December 2007.
17. Ghita, N.; Kloetzer, M. Cell Decomposition-Based Strategy for Planning and Controlling a Car-like Robot. In Proceedings of the 14th International Conference on System Theory and Control, Sinaia, Romania, 17–19 October 2010.
18. So, B.-C.; Jung, J.-W. Mobile Robot Path Planning with Opposite Angle-Based Exact Cell Decomposition. *Adv. Sci. Lett.* **2012**, *15*, 144–148. [CrossRef]
19. Ramer, U. An iterative procedure for the polygonal approximation of plane curves. *Comput. Graph. Image Process.* **1972**, *1*, 244–256. [CrossRef]
20. Hwang, Y.-S.; Lee, J. Robust 3D map building for a mobile robot moving on the floor. In Proceedings of the 2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), Busan, Korea, 7–11 July 2015; pp. 1388–1393.
21. Gao, H.; Hu, M.; Gao, T. Robust detection of median filtering based on combined features of difference image. *Signal Process. Image Commun.* **2017**, *72*, 126–133. [CrossRef]
22. Douglas, D.; Peucker, T. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Can. Cartogr.* **1973**, *10*, 112–122. [CrossRef]
23. Chazelle, B.; Edelsbrunner, H. An optimal algorithm for intersecting line segments in the plane. In Proceedings of the 29th Annual Symposium on Foundations of Computer Science, White Plains, NY, USA, 24–26 October 1988.
24. Ravankar, A.; Ravankar, A.; Kobauashi, Y.; Hishino, Y.; Peng, C.-C. Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges. *Sensors* **2018**, *18*, 3170. [CrossRef] [PubMed]