Robotica (2017) volume 35, pp. 1585–1597. © Cambridge University Press 2016
doi:10.1017/S0263574716000321

Path planning in distorted configuration space

Chao Chen*

Department of Mechanical and Aerospace Engineering, Monash University, Melbourne, Victoria, Australia

(Accepted May 2, 2016. First published online: May 30, 2016)

SUMMARY

An effective algorithm for path planning is introduced based on a novel concept, the distorted configuration space (DC-space), where all obstacles deform into dimensionless geometric objects. Path planning in this space can be conducted by simply connecting the starting position and ending position with a straight line. This linear path in the DC-space is then mapped back into a feasible in the original C-space. The advantage of this approach is that no trial-and-error or iteration is needed while a feasible path can be found directly if it exists. An algorithm with general formulas is derived. Examples in 2D and 3D are provided to validate this concept and algorithm.

KEYWORDS: Path planning, Automation, Motion planning, Navigation, Mobile robots.

1. Introduction

Path planning is one fundamental problem in robotics research, with applications in industrial robots,[1] domestic robots,[2] UAV,[3] autonomous vehicles,[4,5] and computational biology.[6] Many different algorithms were developed to tackle this problem. Typically, the algorithms can be classified into deterministic approaches, probabilistic approaches, and hybrid approaches.[7]

Deterministic approaches typically construct the exact roadmaps, before searching a feasible or optimum path. The methods for generating roadmaps include visibility graphs,[8] Voronoi diagrams,[9] Delaunay triangulation,[10] and cell decomposition.[11] Search methods include Dijkstra,[12] A*,[13] Fuzzy Logic Control,[14,15] Genetic Algorithms,[16] and Ant Colony Optimization.[17] However, the explicit construction of obstacles in the C-space is intractable, especially in high dimensional space.[18] The potential field method[19] utilizes artificial potential fields to push and pull a robot towards the destination by avoiding obstacles. The challenge of this method is the local minima and narrow channels.[20] To meet real-time requirements and avoid local minimums, the fuzzy virtual force[21,22] was developed by means of the Bayesian network and fuzzy logic reasoning.

Probabilistic approaches are mainly sampling based and known to successfully handle many challenging problems. Typically, the approaches can be classified into two categories: multi-query and single-query. The former uses a constructed roadmap for multiple path generations, while the latter constructs a tree data structure upon every query. The probabilistic roadmap method (PRM) approach[23,24] and the rapidly exploring random tree (RRT)[25] belong to these two categories, respectively. Other probabilistic planners such as the one based on expansive spaces,[26] also exist. The probabilistic and sample based approaches are usually applied to the problems when the exact free space is unknown.

Hybrid approaches combine the advantages of both deterministic and probabilistic algorithms. The Randomized Potential Planner (RPP)[27] is an example, which utilizes random walks to escape local minima in the potential fields. There are excellent reviews on this topic.[7,18,28] The main challenges this work focuses on are (1) narrow channels in C-space, and (2) high variance of running time of path planner.

The narrow channels in C-space is a major challenge to sampling-based approaches.[29] The running time increases exponentially as the width of the channel reduces. In order to overcome this challenge,

* Corresponding author. E-mail: chao.chen@monash.edu

https://doi.org/10.1017/S0263574716000321 Published online by Cambridge University Press

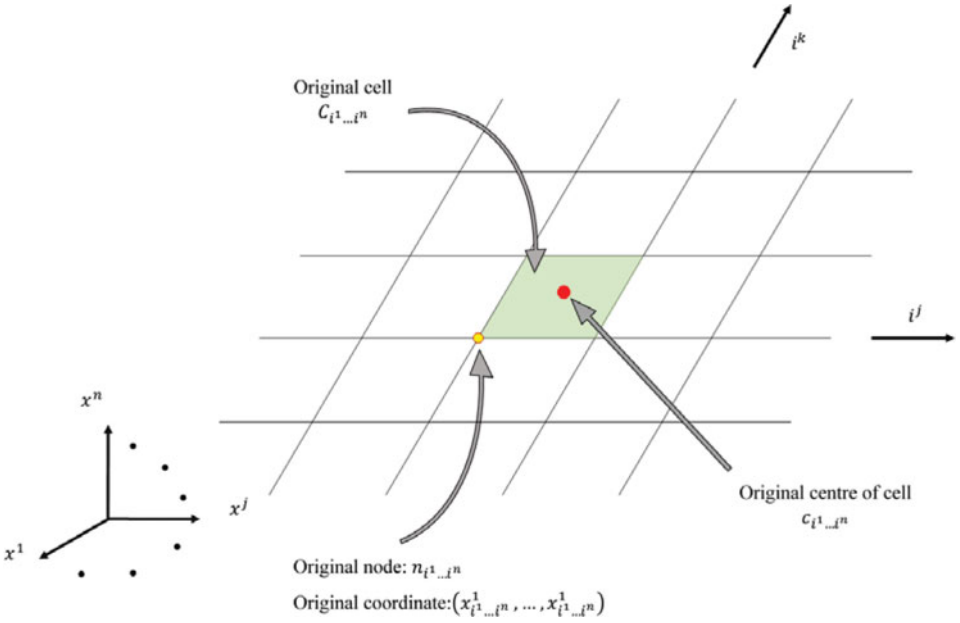


Fig. 1. Notations for C-space.

a number of sampling strategies were developed: boundary,[30] Gaussian,[31] bridge-test,[32] hybrid,[33] and visibility PRM.[34] However, the problem still remains. It was reported in ref. [35] that one feasible path was found after 6000 iterations and 4700 explored nodes by means of RRT* Tree. One more important issue is the relatively high variance of the running time of PRM planners.[29] The high variance often requires setting a large time limit for the planner to terminate without a path.[29] This may not be acceptable in some applications where many planning queries have to be processed sequentially.[36]

This works attempt to solve these challenges by a new concept called the distorted configuration space (DC-space), where all obstacles disappear due to the distortion of the C-space. The DC-space is achieved by artificially compressing all obstacles in the C- space into dimensionless geometric objects such as points, so that all obstacles virtually vanish in the distorted space. In the DC-space, the chance of intersection between a path with a lower dimensional object is zero, and hence, a path can be planned arbitrarily, i.e., a straight path. This path in the DC-space can be then mapped into a feasible path in the original C-space. This DC-space approach gives the consistent running time for a new path because the start and goal can be directly connected by a straight line, which is always a valid path in DC-space. The only computation required is to transform this line segment in the DC-space to the path in the C-space since the DC-space is pre-computed. This involves the calculations of a number of intersections between the line and the corresponding faces of the cells on the path.

The rest of this work is organized as follows. Section 2 derives the DC-space and the mapping algorithm. Examples in 2D and 3D are illustrated in Sections 4 and 5, respectively.

2. Generation of Distorted Configuration Space

A n-dimensional C-space with obstacles and a convex boundary is assumed known. In the case of a concave boundary, a convex boundary can be constructed such that parts of the boundary turn into obstacles. Cell decomposition is applied to the C-space with evenly spanned grids. The grid coordinates are defined by i^j along jth axis, where $j = 1, 2, \ldots, n$. The position coordinates are defined by x^j along the j th axis, with $j = 1, 2, \ldots, n$. A node with grid coordinates $(i^1, \ldots, i^n)$ is noted by $N_{i^1\ldots i^n}$, whose position is given by, as shown in Fig. 1,

$$n_{i^1\ldots i^n} = \left[x^1_{i^1\ldots i^n} \cdots x^n_{i^1\ldots i^n}\right]^T. \tag{1}$$

https://doi.org/10.1017/S0263574716000321 Published online by Cambridge University Press

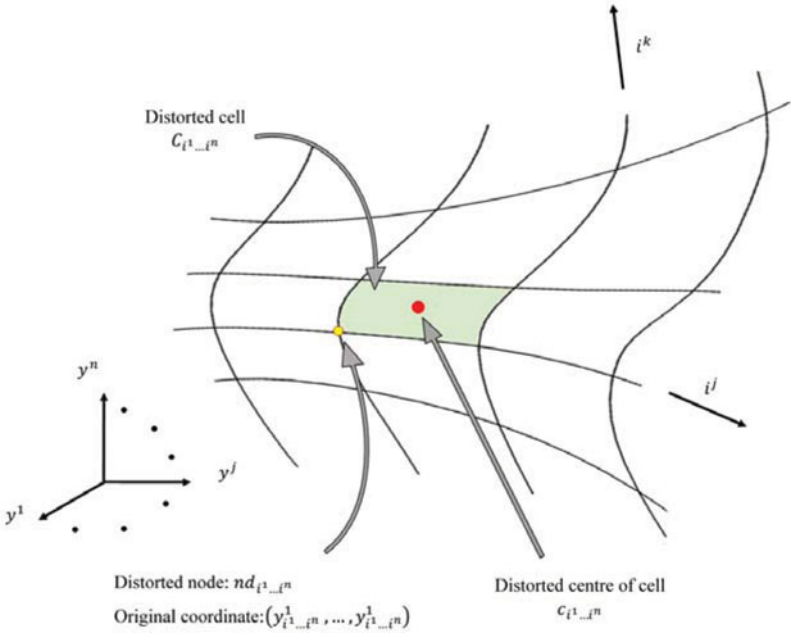


Fig. 2. Notations for distorted C-space.

Assume that the entire bounded space is divided by grids into a finite number of regular n-cubes with unit edge-length. Hence, we have

$$n_{i^1\ldots i^n} = \left[i^1 \cdots i^n\right]^T.$$

Consider some obstacles occupying a number of cells, which are called occupied cells. All occupied cells can be grouped into separated sets. All occupied cells within one set must be connected. A node belonging to any occupied cell is called an occupied node, while a node not belonging to any occupied cell is called a free node.

The key point is to force each set of occupied cells into an object in the subspace R^m, where $m < n - 1$, such that this object has theoretically zero chance to be encountered by another point. For example, the object can be a point in case $n = 2$, and a point or a line in case $n = 3$. Such objects occupy no space geometrically in the n-dimensional space. The deformation of the occupied cells yields the distortion of the whole grid, which yields the DC-space. The positions of nodes in the DC-space are labeled as, as shown in Fig. 2,

$$nd_{i^1\ldots i^n} = \left[y^1_{i^1\ldots i^n} \cdots y^n_{i^1\ldots i^n}\right]^T, \tag{2}$$

where $y^j_{i^1\ldots i^n}$ is the coordinate of Node, $N_{i^1\ldots i^n}$, along the jth axis after distortion.

The distortion is computed by assuming that the connection between any two neighboring nodes is a spring with natural length of zero and stiffness of K. A spring between $N_{i^1\ldots i^j\ldots i^n}$ and $N_{i^1\ldots(i^j+1)\ldots i^n}$ is noted as $S^j_{i^1\ldots i^n}$, where j indicates that this spring is along the jth axis. A stable distorted grid is reached when the total potential energy is minimum, which yields the following theorem.

Theorem 1. *In a stable distorted grid, a free node sits at the geometric center of its neighboring nodes.*

https://doi.org/10.1017/S0263574716000321 Published online by Cambridge University Press

Proof. the potential energy stored in $\text{Spring}^j_{i^1\ldots i^n}$ is given by

$$\begin{aligned} lV^j_{i^1\ldots i^n} &= \frac{1}{2}K\, nd_{i^1\ldots i^j\ldots i^n} - nd_{i^1\ldots(i^j+1)\ldots i^n} \\ &= \frac{1}{2}K\sum_{k=1}^{n}(y^k_{i^1\ldots i^j\ldots i^n}\cdots y^k_{i^1\ldots(i^j+1)\ldots i^n})^2. \end{aligned} \tag{3}$$

The overall potential energy is given by

$$V=\sum_{i^1=0}^{n-1}\cdots\sum_{i^j=0}^{n}\cdots\sum_{i^n=0}^{n}V^1_{i^1\ldots i^n}+\cdots+\sum_{i^1=0}^{n}\cdots\sum_{i^j=0}^{n-1}\cdots\sum_{i^n=0}^{n}V^j_{i^1\ldots i^n}+\cdots+\sum_{i^1=0}^{n}\cdots\sum_{i^j=0}^{n}\cdots\sum_{i^n=0}^{n-1}V^n_{i^1\ldots i^n}.$$

A stable state is achieved when the overall potential energy is minimized. Equivalently, the partial differentiation of the overall potential energy with respect to an arbitrary free node, $N_{i^1i^2\ldots i^n}$, must be zero, i.e.,

$$\frac{\partial V}{\partial \text{nd}_{i^1i^2\ldots i^n}}=0. \tag{4}$$

Note that only the potential energy stored in the springs connected to $N_{i^1i^2\ldots i^n}$ is dependent on $nd_{i^1i^2\ldots i^n}$. It is not difficult to see that these springs are $S^1_{i^1i^2\ldots i^n}$, $S^2_{i^1i^2\ldots i^n}$, $\cdots$, $S^n_{i^1i^2\ldots i^n}$, and $S^1_{(i^1-1)i^2\ldots i^n}$, $S^2_{i^1(i^2-1)\ldots i^n}$, $\cdots$, $S^n_{i^1i^2\ldots(i^n-1)}$. Therefore, Eq. (4) can be written as

$$\begin{aligned} &\frac{\partial V^1_{i^1i^2\ldots i^n}}{\partial \text{nd}_{i^1i^2\ldots i^n}}+\frac{\partial V^2_{i^1i^2\ldots i^n}}{\partial \text{nd}_{i^1i^2\ldots i^n}}+\cdots+\frac{\partial V^n_{i^1i^2\ldots i^n}}{\partial \text{nd}_{i^1i^2\ldots i^n}}+\frac{\partial V^1_{(i^1-1)i^2\ldots i^n}}{\partial \text{nd}_{i^1i^2\ldots i^n}}+\frac{\partial V^2_{i^1(i^2-1)\ldots i^n}}{\partial \text{nd}_{i^1i^2\ldots i^n}}+\cdots. \\ &\qquad+\frac{\partial V^n_{i^1i^2\ldots(i^n-1)}}{\partial \text{nd}_{i^1i^2\ldots i^n}}=0 \end{aligned} \tag{5}$$

Substituting Eq. (1) and Eq. (4) into Eq. (5) yields

$$\begin{aligned} y^j_{i^1i^2\ldots i^n}=\frac{1}{2n}\Big(&y^j_{(i^1-1)i^2\ldots i^n}+y^j_{(i^1+1)i^2\ldots i^n}+y^j_{i^1(i^2-1)\ldots i^n}+y^j_{i^1(i^2+1)\ldots i^n}+\cdots, \\ &\cdots+y^j_{i^1i^2\ldots(i^n-1)}+y^j_{i^1i^2\ldots(i^n+1)}\Big) \end{aligned} \tag{6}$$

with $j=1,2,\ldots,n$. This solution indicates that a free node must be the geometric center of all its surrounding nodes to have the minimum potential energy. □

To keep a convex structure, any free node on a boundary is restricted to move on the boundary. A boundary with dimension of $n-m$ is called a $n-m$ boundary, where m is positive integer no greater than n. A $n-m$ boundary can be considered to the intersection of m hyperplanes with dimension of $n-1$. It is not difficult to see that the number of $n-m$ boundaries in n-cubic workspace is given by $2mC(n,m)=2mn!/(m!(n-m)!)$. Due to the convex structure, any two points inside the boundaries can be connected by a straight line without passing any boundary.

Solving the Eq. (6) with respect to all free nodes and the boundary conditions yields the coordinates of the distorted grid. In the original grid, the geometric center of a cell, $C_{i^1\ldots i^n}$ is given by

$$c_{i^1\ldots i^n}=\frac{1}{2^n}\left(\sum_{k^1=0}^{1}\cdots\sum_{k^n=0}^{1}n_{(i^1+k^1)\cdots(i^n+k^n)}\right). \tag{7}$$

https://doi.org/10.1017/S0263574716000321 Published online by Cambridge University Press

Similarly, the geometric center of the distorted cell, $C_{i^1 \ldots i^n}$, is given by

$$c_{i^1 \ldots i^n} = \frac{1}{2^n} \left(\sum_{k^1=0}^{1} \cdots \sum_{k^n=0}^{1} nd_{(i^1+k^1)\cdots(i^n+k^n)} \right). \tag{8}$$

3. Path Planning

The problem of path planning is to find a path connecting the given start and goal. The start and goal are represented by the geometric centers, p and q, of the start cell and goal cell, respectively. p and q are calculated by Eq. (8) in the DC-space. The path in the DC-space can be as simple as a straight line, which is adopted here. This straight-line path in R^n is formulated as

$$s = (1 - \lambda)\, p + \lambda q, \tag{9}$$

where $\lambda \in [0, 1]$.

The mapping of this path back into the original space is done by recording the sequence of the cells which this line passes through. Therefore, the intersection of the line with a cell is required. More specifically, the intersection between the line and the faces of a cell is required to identify the next cell on the path.

A face of a distorted cell in R^n can be modeled as a $(n-1)$ -ruled surface. The ruled surfaces with different orders were discussed in ref. [37]. Assume the face towards the i^j -axis in Cell $C_{i^1 \ldots i^n}$ is of interest. The face towards the $(i^j + 1)$ -axis can be treated similarly. All vertices on this face are $C_{(i^1+k^1)\cdots(i^j)\cdots(i^n+k^n)}$, where $k_1, \ldots, k_n$ are 0 or 1 with k_j being excluded. For the simplicity of expression, the following notations are introduced: $(w^1, \ldots, w^{n-1})$ is the same as $(i^1, \ldots, i^n)$ with i^j removed; $(u^1, \ldots, u^{n-1})$ is the same as $(k^1, \ldots, k^n)$ with k^j removed. Therefore, a vertex on the face can be represented as

$$N^{i^j}_{(w^1+u^1)\cdots(w^{n-1}+u^{n-1})} \equiv N_{(i^1+k^1)\cdots(i^j)\cdots(i^n+k^n)}. \tag{10}$$

Theorem 2. *Any point on the (n-1)-ruled surface matching the face towards i j on a distorted n-cubic cell is expressed as*

$$r = \sum_{u^{n-1}=0}^{1} \cdots \sum_{u^1=0}^{1} ((-1)^{\sum_{l=1}^{n-1} u^l} \prod_{l=1}^{n-1} (\lambda_l - u^l)\, nd^{i^j}_{(w^1+u^1)\cdots(w^{n-1}+u^{n-1})}), \tag{11}$$

where $nd^{i^j}_{(w^1+u^1)\cdots(w^{n-1}+u^{n-1})}$ is the position of the node in the distorted grid and $\lambda_l \in [0,\ 1]$.

Proof. Formula (11) can be proved in an inductive way, In the case of distorted 2-cubic cell, the face toward on axis i^j is a line segment with vertices $N^{i^j}_{w^1}$ and $N^{i^j}_{w^1+1}$. This line segment can be expressed as

$$r = \lambda_1 nb^{i^j}_{w^1} + (1 - \lambda_1)\, nb^{i^j}_{w^1+1} = \sum_{u^1=0}^{1} ((-1)^{u^1} (\lambda_1 - u^1)\, d^{i^j}_{(w^1+u^1)}),$$

which agrees with Eq. (11).

Assuming (11) holds for the case of a distorted n -cubic cell, we want to show that Eq. (11) holds for a distorted $(n+1)$ -cubic cell as well. Considering the face towards the i^j -axis again, the n -ruled surface can be constructed by a line segment along the additional axis, i^{n+1} or w^n. The two ends of

https://doi.org/10.1017/S0263574716000321 Published online by Cambridge University Press

this line segment are given by, respectively,

$$r_1 = \sum_{u^{n-1}=0}^{1} \cdots \sum_{u^1=0}^{1} ((-1)^{\sum_{l=1}^{n-1} u^l} \prod_{l=1}^{n-1} (\lambda_l - u^l)\, nd^{i^j}_{(w^1+u^1)\cdots(w^{n-1}+u^{n-1})w^n}),$$

$$r_2 = \sum_{u^{n-1}=0}^{1} \cdots \sum_{u^1=0}^{1} ((-1)^{\sum_{l=1}^{n-1} u^l} \prod_{l=1}^{n-1} (\lambda_l - u^l)\, nd^{i^j}_{(w^1+u^1)\cdots(w^{n-1}+u^{n-1})(w^n+1)}).$$

Hence, the point on the n -ruled surface is a point on this line segment, i.e.,

$$\begin{aligned} r &= \lambda_n r_1 + (1-\lambda_n) r_2 \\ &= \lambda_n \sum_{u^{n-1}=0}^{1} \cdots \sum_{u^1=0}^{1} ((-1)^{\sum_{l=1}^{n-1} u^l} \prod_{l=1}^{n-1} (\lambda_l - u^l)\, nd^{i^j}_{(w^1+u^1)\cdots(w^{n-1}+u^{n-1})w^n}) \\ &\quad + (1-\lambda_n) \sum_{u^{n-1}=0}^{1} \cdots \sum_{u^1=0}^{1} ((-1)^{\sum_{l=1}^{n-1} u^l} \prod_{l=1}^{n-1} (\lambda_l - u^l)\, nd^{i^j}_{(w^1+u^1)\cdots(w^{n-1}+u^{n-1})(w^n+1)}) \\ &= \sum_{u^n=0}^{1} \sum_{u^{n-1}=0}^{1} \cdots \sum_{u^1=0}^{1} ((-1)^{\sum_{l=1}^{n-1} u^l} \prod_{l=1}^{n-1} (\lambda_l - u^l)\, nd^{i^j}_{(w^1+u^1)\cdots(w^{n-1}+u^{n-1})(w^n+u^n)}), \end{aligned}$$

which agrees with Eq. (11). The proof is complete. □

Given a cell $C_{i^1\cdots i^n}$ on the path, the intersection between each face of the cell and the straight-line path is computed by solving $s = r$ for λ and λ_l, where s and r are given by Eqs. (9) and (11), respectively. The corresponding intersecting face in $C_{i^1\cdots i^n}$ determines the next cell on the path. This procedure starts from the start cell and finishes at the goal cell. The pseudo-code for this procedure is given in Algorithm 1.

Algorithm 1 Mapping path from the DC-space to the original C-space.

```
Calculate the center of the start cell, p; Calculate the center of the goal cell, q;
Formulate the line s connecting p and q according to Eq. (9);
step = 1;
path[step] = cellstart;
pace = 0;
while path[step] ≠ cellgoal do
  for i from 1 to 2n do
    Calculate the intersection between line s and Face i and obtain λ;
    if λ > pace then
      pace = λ, and f = i
    end
  end
  path[step + 1] =the cell adjacent to the current cell by Face f;
  step ++
end
return path
```

https://doi.org/10.1017/S0263574716000321 Published online by Cambridge University Press

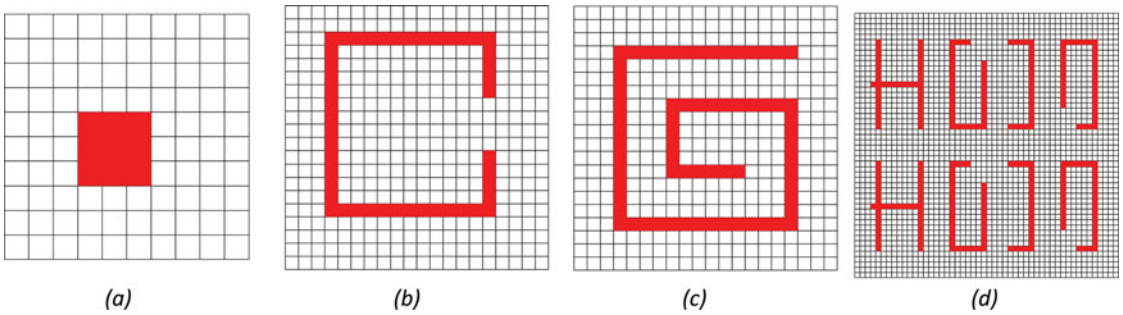


Fig. 3. Examples.

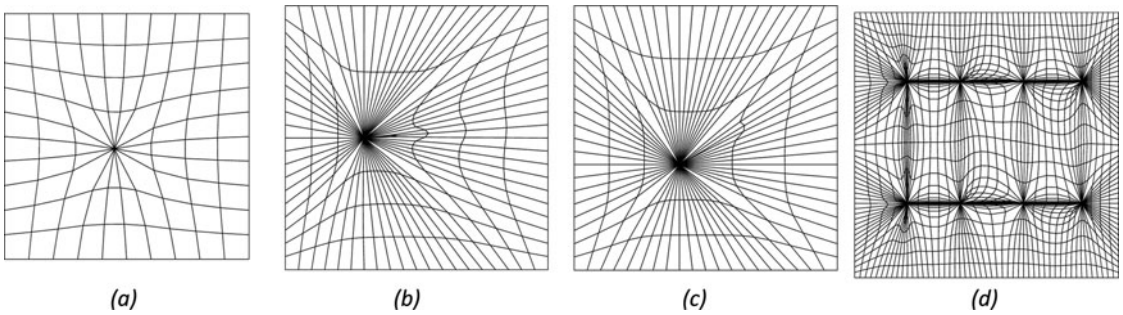


Fig. 4. DC-space maps of the examples: (a) the obstacle shrinks into a point at (4.5, 4.5), (b) the C-shaped obstacle shrinks into a point at (6, 10), (c) the spiral obstacle shrinks into a point at (8, 8), and (d) all obstacles are reduced into sporadic points.

4. 2D Examples

Consider a 2-dimensional space. The grid coordinates are defined by i^1 and i^2. A node with grid coordinates (i^1, i^2) is noted by $N_{i^1 i^2}$, whose positions before and after distortion are given by $n_{i^1 i^2} = [x^1_{i^1 i^2} \quad x^2_{i^1 i^2}]^T$, $nd_{i^1 i^2} = [y^1_{i^1 i^2} \quad y^2_{i^1 i^2}]^T$, respectively. According to Theorem 1, the position of a free $\text{Node}_{i^1 i^2}$ in the DC-space satisfies the following constraint:

$$y^j_{i^1 i^2} = \frac{1}{4}\left[y^j_{(i^1-1)i^2} + y^j_{(i^1+1)i^2} + y^j_{i^1(i^2-1)} + y^j_{i^1(i^2+1)}\right], \tag{12}$$

with $j = 1, 2$.

For simplicity, the nodes on the boundary are treated as fixed nodes. As for obstacle nodes, their positions can be determined arbitrarily, as long as the obstacles are forced to shrink into sporadic geometric points, which occupy no space in the DC-space map. Solving all Eq. (12) on all free nodes with boundary constraints yields the coordinates of the distorted grid. Four examples are shown in Fig. 3, where solid blocks represent the obstacles in the C-space. Their DC-space maps are shown in Fig. 4, respectively.

Straight-line paths are planned in the DC-space maps, as shown in Fig. 5. According to Algorithm 1, the paths are recorded in sequences of cells, which are mapped back to the original C-space, as shown in Fig. 6.

5. 3D Examples

Consider a 3-dimensional space. The grid coordinates are defined by i^1, i^2, i^3. A node with grid coordinates (i^1, i^2, i^3) is noted by $N_{i^1 i^2 i^3}$, whose position before and after distortion are given by $n_{i^1 i^2 i^3} = [x^1_{i^1 i^2 i^3} \quad x^2_{i^1 i^2 i^3} \quad x^3_{i^1 i^2 i^3}]^T$ and $nd_{i^1 i^2 i^3} = [y^1_{i^1 i^2 i^3} \quad y^2_{i^1 i^2 i^3} \quad y^3_{i^1 i^2 i^3}]^T$, respectively. According to

https://doi.org/10.1017/S0263574716000321 Published online by Cambridge University Press

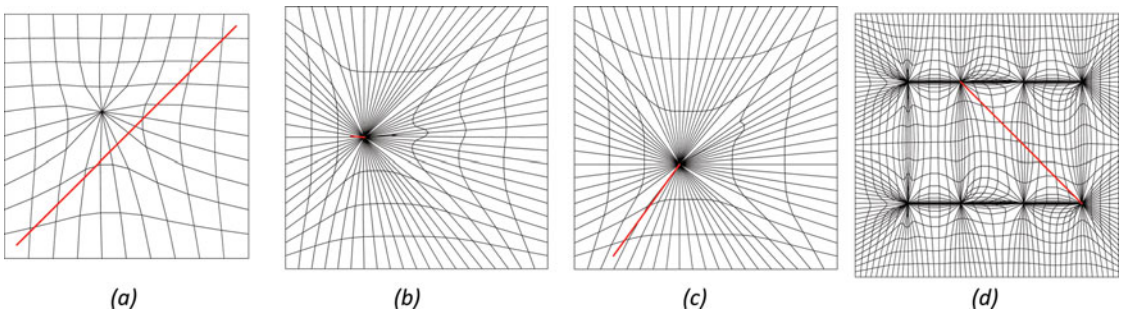


Fig. 5. Path planned in the DC-space maps: (a) from Cell(0,0) to Cell(9,9), (b) from Cell(2,10) to Cell(5,10), (a) from Cell(2,0) to Cell(5,5), (a) from Cell(6,12) to Cell(30,29).

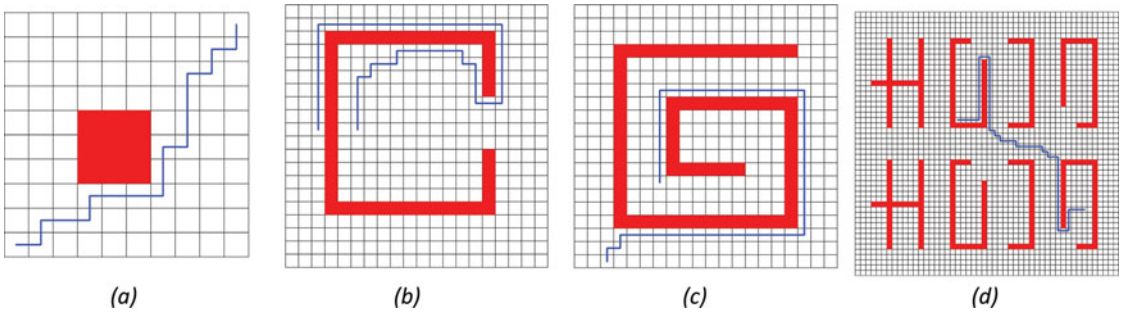


Fig. 6. Path mapped back to the original maps: (a) from Cell(0,0) to Cell(9,9), (b) from Cell(2,10) to Cell(5,10), (a) from Cell(2,0) to Cell(5,5), (a) from Cell(6,12) to Cell(30,29).

Theorem 1, the position of node $N_{i^1 i^2 i^3}$ in the DC-space satisfies the following constraint:

$$y^j_{i^1 i^2 i^3} = \frac{1}{6}\left[y^j_{(i^1-1)i^2 i^3} + y^j_{(i^1+1)i^2 i^3} + y^j_{i^1(i^2-1)i^3} + y^j_{i^1(i^2+1)i^3} + y^j_{i^1 i^2(i^3-1)} + y^j_{i^1 i^2(i^3+1)}\right], \quad (13)$$

with $j = 1, 2, 3$.

The nodes on the boundary-planes can slide within the plane. Solving all Eq. (13) on all free nodes yields the coordinates of the distorted grid. Two examples are shown in Fig. 7. In Fig. 7(c), the two block obstacles in the first example are forced to deformed into two separate geometric points, (1.5, 1.5, 0) and (4, 3, 2), respectively. In Fig. 7(d), all cave-shaped obstacles deform into geometric line segments with the same heights. The horizontal positions of these six line segments are (11, 10), (11, 25), (28, 10), (28, 25), (45, 10), and (45, 25), respectively. Note that only one portion of the DC-space map is shown in Fig. 5(d) for visibility.

The straight-line path in the DC-space is mapped back to the original C-space by Algorithm 1. Four paths generated in the second example are illustrated in Figs. 8-11, respectively. The straight-line paths in the DC-space are shown in Figs. 8(a)-11(a), where only the surrounding distorted cells around the cells at the start and at the end are displayed for visibility. The paths are mapped back to the original C-space, as shown in Figs. 8(b)-11(b), and Figs. 8(c)-11(c), where the obstacles are displayed.

The examples show that this approach is capable to generate collision-free paths in an environment with complicated obstacles, and avoid traps induced by concave obstacles. It can be observed that paths are attracted to the obstacles in the original space when they are close.

6. 3R-Robot

A 3R-robot is shown in Fig. 12, where the obstacles are the black, green and red circles. The geometric parameters are L1 = L2 = L3 = 90 cm. Each link is modeled as five overlapping circles with the radii of 22.5 cm. Each obstacle circle has a radius of 10 cm. The joint angles, $\theta 1$, $\theta 2$ and $\theta 3$ are restricted to $[-95^\circ, 95^\circ]$. According to the algorithm described in Section 2, the C-space and DC-space are

https://doi.org/10.1017/S0263574716000321 Published online by Cambridge University Press

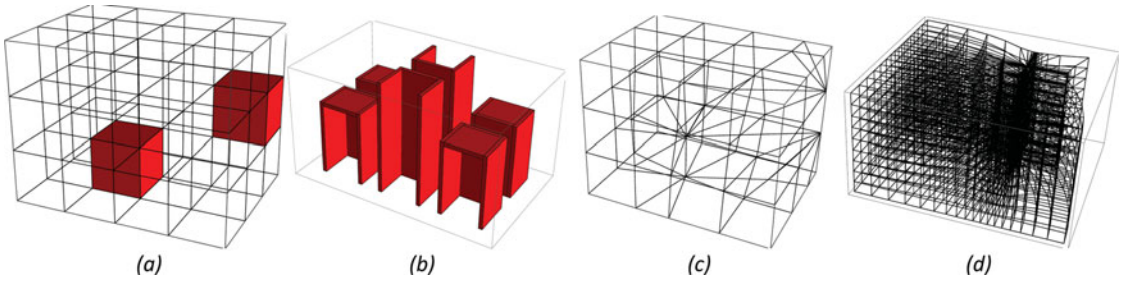


Fig. 7. 3D examples: (a) and (b) are the original maps with obstacles of 3D Examples 1 and 2, respectively; (c) and (d) are the DC-space maps free of obstacles of 3D Examples 1 and 2, respectively.

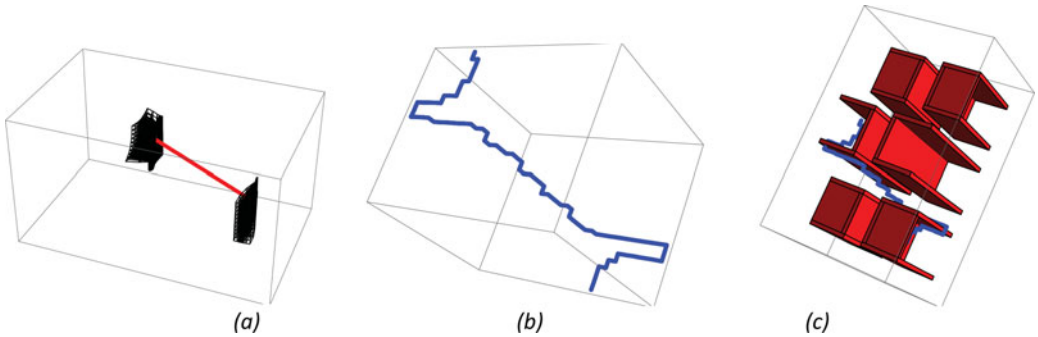


Fig. 8. Path 1 generated from Cell(25,10,26) to Cell(45,22,5) in the 3D example: (a) the straight-line path in the DC-space, (b) the path mapped into the original space, and (c) the same path with obstacles displaced.

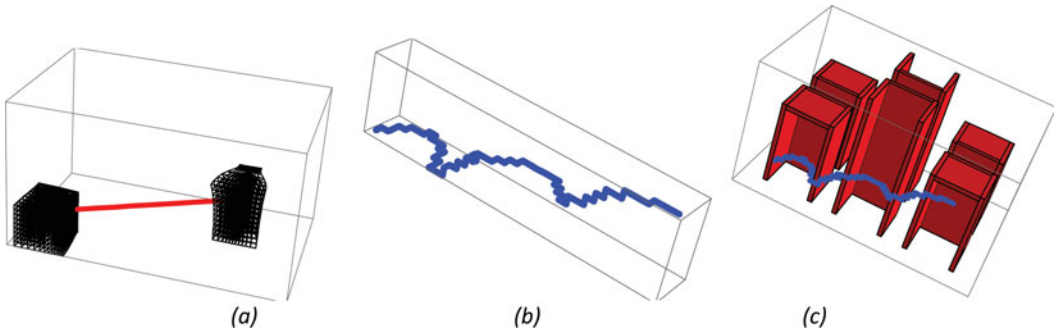


Fig. 9. Path 2 generated from Cell(6,5,5) to Cell(45,5,15) in the 3D example: (a) the straight-line path in the DC-space, (b) the path mapped into the original space, and (c) the same path with obstacles displaced.

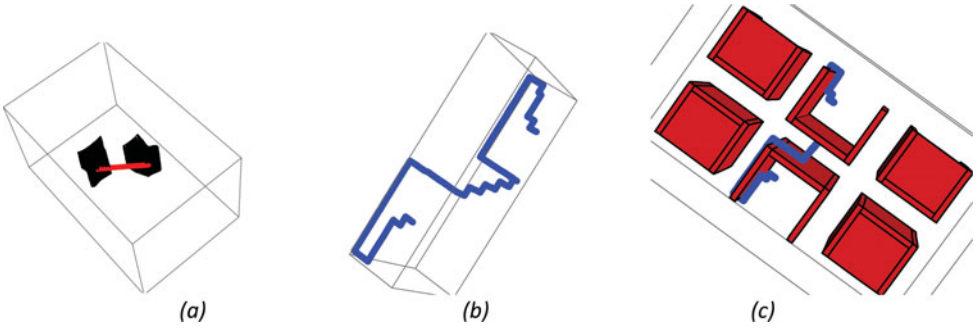


Fig. 10. Path 3 generated from Cell(25,10,20) to Cell(25,25,5) in the 3D example: (a) the straight-line path in the DC-space, (b) the path mapped into the original space, and (c) the same path with obstacles displaced.

https://doi.org/10.1017/S0263574716000321 Published online by Cambridge University Press

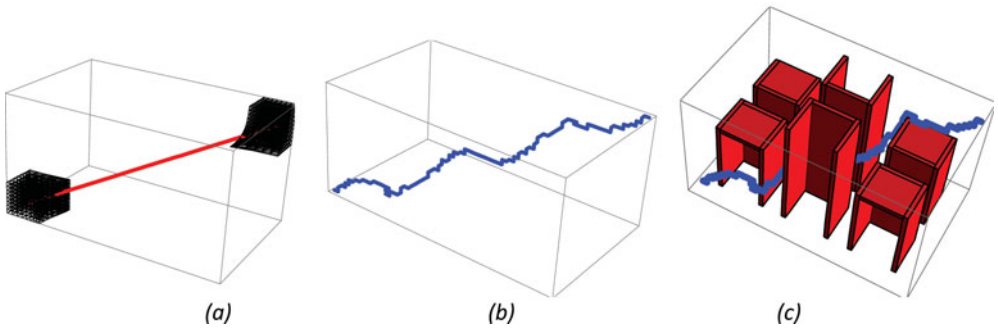


Fig. 11. Path 4 generated from Cell2,2,2 to Cell54,32,25 in the 3D example: (a) the straight-line path in the DC-space, (b) the path mapped into the original space, and (c) the same path with obstacles displaced.

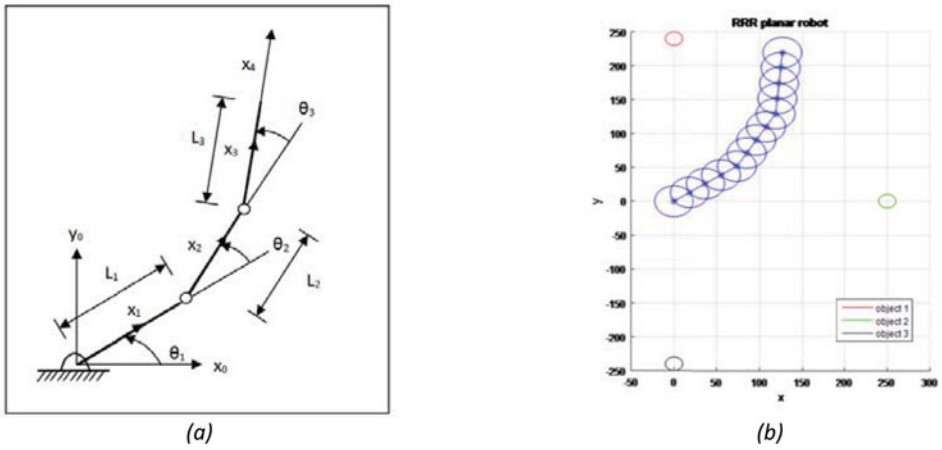


Fig. 12. A 3R-robot: (a) the geometric description and (b) the obstacles and volumes of links.

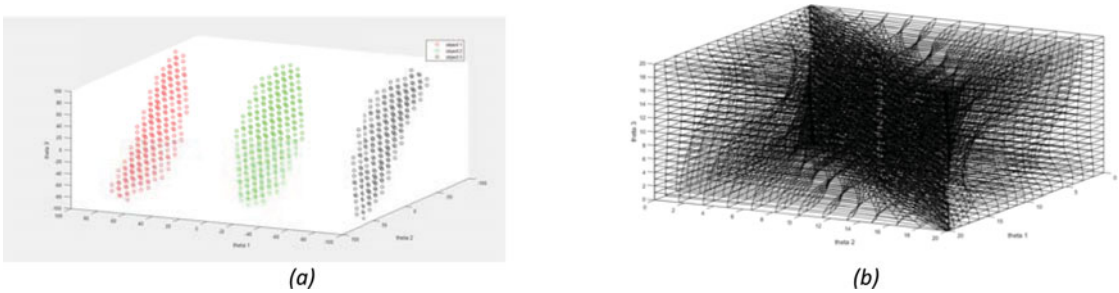


Fig. 13. (a) The C-space and (b) DC-space of the 3R-robot.

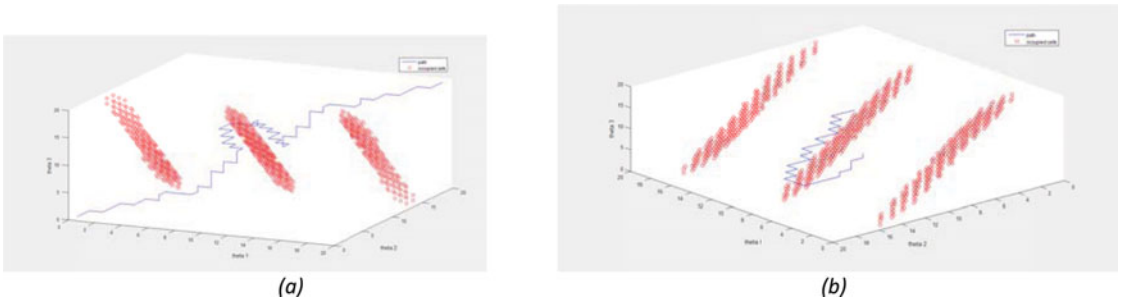
generated, as shown in Fig. 13. Two example paths in the C-space mapped from the straight path in the DC space are shown in Fig. 14.

7. Discussion and Conclusion

An efficient algorithm based on the DC-space was introduced to tackle the path planning in a known C-space. Once the DC-space map is constructed, a feasible plan can be found quickly because the algorithm only evaluates the cells on the path. The main advantage of this algorithm is that the path can be obtained immediately in the DC-space if a feasible path exists, without applying trial-and-error or iterations in typical heuristic and sample-based approaches. The general formulas were derived while the examples in 2D and 3D were provided to validate this concept.

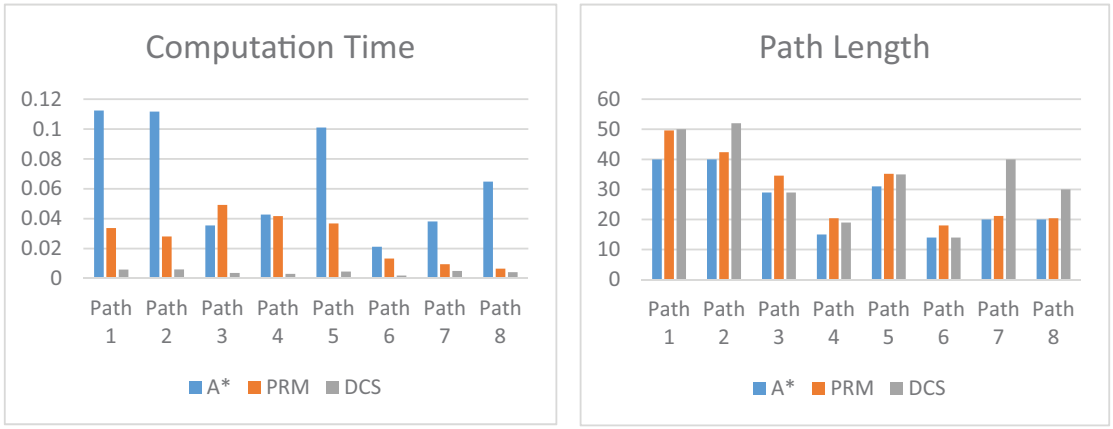
A comparison of DC-Space with two common algorithms, A* search and PRM, are conducted. With all three algorithms, the same grid size as shown in Fig. 3(b) is used. The cost function in A* is

https://doi.org/10.1017/S0263574716000321 Published online by Cambridge University Press



(a) (b)

Fig. 14. Two example paths generated for (a) from Cell(0.5,0.5,0.5) to Cell(19.5,19.5,19.5) and (b) from Cell(4.5,13.5,12.5) to Cell(14.5,5.5,8.5).



(a) (b)

Fig. 15. Comparison of (a) computation time and (b) path length of path finding by A*, PRM, and DCS (DC-Space Method) in the example shown in Fig. 3(b).

defined as the sum of the travel distance from the start cell to the current cell and the heuristic defined as the straight distance from the current cell to the end cell. The best-first search is applied, i.e., the cell with minimum cost is chosen for every step. In PRM, a path tree is grown from the start cell and end cell. A collision detection algorithm attempts to connect any new sampled cell with the samples in the tree, by means of a straight line. If there is no collision with obstacles, a collision-free path between the samples is added to the path tree. When the tree includes both the start and end cells, a shortest path between them is found. The maximum number of iterations is set to be 50 in PRM.

Eight different experiments are run with respect to these three algorithms. Each algorithm is run 10 times to get the average value. The A* and DCS algorithms are quite consistent in each experiment while the deviation among the results produced by PRM is high.

As shown in Fig. 15(a), the computation time by DCS is much less than the time required by A* and PRM. The main reason is that DCS does not need to explore the whole workspace to find the path. Instead, it only explores the cells on the path. The price of this efficiency is that DCS requires pre-processing time to create the DC-space. In the case that path planning needs to be performed continuously in a given C-space, DCS can significantly reduce the computation cost.

As shown in Fig. 15(b), A* produces the shortest path since its heuristic function ensures that the final result is the optimum solution. As a newly developed algorithm, DCS in this work can only perform feasible path finding.

The comparison of the number of iterations is shown in Fig. 16. A* requires more iterations than PRM and DCS, which is consistent with the computation time associated with these methods. The number of iterations in DCS always equals to the travel distance, because DCS only checks the cells on the path.

https://doi.org/10.1017/S0263574716000321 Published online by Cambridge University Press

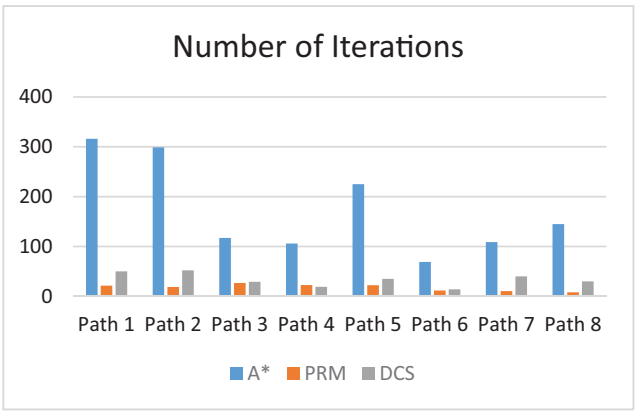


Fig. 16. Comparison of the number of iterations for path finding by A*, PRM, and DCS in the example shown in Fig. 3(b).

More works need to be done to improve the performance of this algorithm. First, an obtained path is by no means close to an optimum path, even though it is straight-line solution in the DC-space. This is because neither the DC-space nor the C-space is not flat. There can be two possible approaches to optimize the path generated by the DCS technique: (1) pre-process: using correct metric to distort the C-Space into a quasi-flat space so that a straight line in the DC-Space is approximately an optimum path in C-Space, and (2) post-process: shaping the obtained feasible path by the DCS into an optimum path. If the pre-process is achieved, the DCS can directly provide an optimum solution without searching the space, which can significantly reduce the computational time as compared to existing methods such as A* and PRM in a static and known environment. The post-process may be achieved by taking random samples around the feasible path found initially. This procedure should also be more efficient because the searching space is narrowed down to the area around the feasible path only.

Second, singularity may occur when the distance between a straight-line path and a fixed node into which an obstacle deforms cannot be distinguished by a computer because of the max digital precision. This problem can be readily solved by introducing one or more arbitrary via points in the DC-space to take detours. Third, in the case of a trap-shaped obstacle such as the one shown in Fig. 3(c), the free cells deep in the trap can deform into cells that are too compact to be recognized by a computer due to the digital precision. Therefore, these cells cannot be reached by Algorithm 1. Further investigations are required to tackle these issues so that this concept and algorithm can be applied to general and real robotic systems in partially known C-space with high dimensions. Potential solutions to the "curse of dimensionality" may include using a relatively coarse resolution in a high-dimensional C-Space by sacrificing some free space under the constraint of computation power and developing the analytical mapping between the C-Space and the DC-Space so that discretization is not required for distortion.

References

1. L. P. Ellekilde and H. G. Petersen, "Motion planning efficient trajectories for industrial bin-picking," *Int. J. Robot. Res.* **32**(9-10), 991–1004 (2013).
2. S. S. Srinivasa, D. Berenson, M. Cakmak, A. Collet, M. R. Dogar, A. D. Dragan, R. A. Knepper, T. Niemueller, K. Strabala, J. M. Vande Weghe and J. Ziegler, "Herb 2.0: Lessons learned from developing a mobile manipulator for the home," *Proc. Ieee* **100**(8), 2410–2428 (2012).
3. K. Yang, S. K. Gan and S. Sukkarieh, "A gaussian process-based rrt planner for the exploration of an unknown and cluttered environment with a uav," *Adv. Robot.* **27**(6), 431–443 (2013).
4. J. Yoon and C. D. Crane, "Path planning for unmanned ground vehicle in urban parking area," *Proceedings of the 11th International Conference on Control, Automation and Systems (Iccas)*, Gyeonggi-do, Republic of Korea (2011) pp. 887–892.
5. Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli and J. P. How, "Real-time motion planning with applications to autonomous urban driving," *Ieee Trans. Control Syst. Technol.* **17**(5), 1105–1118 (2009).
6. J. C. Latombe, "Motion planning: A journey of robots, molecules, digital actors, and other artifacts," *Int. J. Robot. Res.* **18**(11), 1119–1128 (1999).

https://doi.org/10.1017/S0263574716000321 Published online by Cambridge University Press

7. M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access* 2, 56–77 (2014).
8. T. Maekawa, T. Noda, S. Tamura, T. Ozaki and K. Machida, "Curvature continuous path generation for autonomous vehicle using b-spline curves," *Comput.-Aided Des.* **42**(4), 350–359 (2010).
9. O. Takahashi and R. J. Schilling, "Motion planning in a plane using generalized voronoi diagrams," *Ieee Trans. Robot. Autom.* **5**(2), 143–150 (1989).
10. G. E. Jan, C. C. Sun, W. C. Tsai and T. H. Lin, "An o(n log n) shortest path algorithm based on delaunay triangulation," *Ieee-Asme Trans. Mechatronics* **19**(2), 660–666 (2014).
11. R. A. Brooks and T. Lozanoperez, "A subdivision algorithm in configuration space for findpath with rotation," *Ieee Trans. Syst. Man and Cybern.* **15**(2), 224–233 (1985).
12. E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik* **1**, 269–271 (1959).
13. P. E. Hart, N. J. Nilsson and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.* **4**, 100–107 (1968).
14. H. Seraji and A. Howard, "Behavior-based robot navigation on challenging terrain: A fuzzy logic approach," *Ieee Trans. Robot. Autom.* **18**(3), 308–321 (2002).
15. H. R. Beom and H. S. Cho, "A sensor-based navigation for a mobile robot using fuzzy-logic and reinforcement learning," *Ieee Trans. Syst. Man Cybern.* **25**(3), 464–477 (1995).
16. Y. R. Hu, S. X. Yang, L. Z. Xu and M. Q. H. Meng, "A knowledge based genetic algorithm for path planning in unstructured mobile robot environments," *Proceedings of the IEEE International Conference on Robotics and Biomimetics*, *IEEE ROBIO*, Shenyang, China (2004) pp. 767–772.
17. M. A. P. Garcia, O. Montiel, O. Castillo, R. Sepulveda and P. Melin, "Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation," *Appl. Soft Comput.* **9**(3), 1102–1110 (2009).
18. S. M. LaValle, *Planning Algorithms* (Cambridge Univ. Press, Cambridge, 2006).
19. O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.* **5**(1), 90–98 (1986).
20. Y. Koren and J. Borenstein, "Potential-field methods and their inherent limitations for mobile robot navigation," *Ieee International Conference on Robotics and Automation*, Sacramento, CA (vols. 1–3, 1991) pp. 1398–1404.
21. Z. J. Chen Z. N. Dong, R. L. Zhang and R. Zhou, "Study on uav path planning approach based on fuzzy virtual force," *Chin. J. Aeronautics* **23**, 341–350 (2010).
22. L. R. Sun *et al*., "Mobile robot real-time path planning based on virtual targets method," *3rd International Conference on Measuring Technology and Mechatronics Automation*, Shangshai, China (2011) pp. 568–572.
23. L. E. Kavraki, P. Svestka, J. C. Latombe and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Ieee Trans. Robot. Autom.* **12**(4), 566–580 (1996).
24. L. Kavraki and J. C. Latombe, "Randomized preprocessing of configuration space for path planning - articulated robots," *Iros '94 - Intelligent Robots and Systems: Advanced Robotic Systems and the Real World*, Munich, Germany (vols. 1–3, 1994) pp. 1764–1771.
25. S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," *Algorithmic and Computational Robotics: New Directions*, Natick, Massachusetts (2001) pp. 293–308.
26. D. Hsu, R. Kindel, J. C. Latombe and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *Int. J. Robot. Res.* **21**(3), 233–255 (2002).
27. J. Barraquand and J. C. Latombe, "Robot motion planning-a distributed representation approach," *Int. J. Robot. Res.* **10**(6), 628–649 (1991).
28. J. T. Schwartz and M. Sharir, "A survey of motion planning and related geometric algorithms," *Artificial Intell*. **37**(1–3), 157–169 (1988).
29. D. Hsu, J. C. Latombe and H. Kurniawati, "On the probabilistic foundations of probabilistic roadmap planning," *Int. J. Robot. Res.* **25**, 627–643 (Jul. 2006).
30. N. M. Amato and Y. Wu, "A randomized roadmap method for path and manipulation planning," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Minneapolis, MN (vol. 1, Apr. 1996) pp. 113–120.
31. V. Boor, M. H. Overmars and A. F. van der Stappen, "The gaussian sampling strategy for probabilistic roadmap planners," *Proceedings of the 1996, IEEE International Conference on Robotics and Automation (ICRA)*, Detroit, MI (vol. 2, May, 1999) pp. 1018–1023.
32. Z. Sun, D. Hsu, T. T. Jiang, H. Kurniawati and J. H. Reif, "Narrow passage sampling for probabilistic roadmap planning," *IEEE Trans. Robot.* **21**, 1105–1115 (Dec. 2005).
33. S. Thomas, M. Morales, X. Y. Tang and N. M. Amato, "Biasing samplers to improve motion planning performance," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Roma, Italy (Apr. 2007) pp. 1625–1630.
34. T. Simeon, J. P. Laumond and C. Nissoux, "Visibility-based probabilistic roadmaps for motion planning," *Adv. Robot.* **14**, 477–493 (2000).
35. E. Mohamed and S. Milan, "Sampling-based robot motion planning: A review," *IEEE Access* **2**, 56–77 (Jan. 2014).
36. K. Hauser, T. Bretl and J. C. Latombe, "Learning-assisted multi-step planning," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Barcelona, Spain (Apr. 2005) pp. 4575–4580.
37. W. L. Edge, *The Theory of Ruled Surfaces* (Cambridge Univ. Press, London, 1931).

https://doi.org/10.1017/S0263574716000321 Published online by Cambridge University Press