Robotica (2007) volume 25, pp. 1–11. © 2006 Cambridge University Press
doi:10.1017/S0263574706002943 Printed in the United Kingdom

On the construction of discretized configuration space of manipulators

X. J. Wu[1], J. Tang[2,*] and K. H. Heng[1]

[1]*School of Mechanical and Aerospace Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798*
[2]*Department of Mechanical Engineering, University of Connecticut, 191 Auditorium Road, Unit 3139, Storrs, CT 06269, USA*

(Received in Final Form: June 20, 2006, First published online: August 8, 2006)

SUMMARY
In this research, we study the construction of configuration space (C-space) of manipulators. The proposed approach is based upon precomputing the global topology of a robot's free space, and consists of an offline phase and an online phase. In the offline phase, a C-space obstacle database (COD) for a given robot is developed in which the C-space obstacle (C-obstacle) maps are stored and indexed by the cells of the workspace; in the online phase when the same robot is operated in a real environment, those maps whose indices match the real obstacle cells are identified and then extracted from the COD. The superposition of these maps forms forbidden region in operation. This proposed approach is a generic one and can be applied to manipulators with arbitrary kinematic structures and geometries. The construction of the COD, which is generally the most time-consuming step, is implemented in the offline phase, and the online computing only involves the identification of the components matching the COD indices. Therefore, this proposed approach for C-space construction can be realized in a real-time online fashion and is especially suitable for robot manipulation under dynamic operations. We carry out analyses on several types of manipulators to verify and demonstrate the feasibility and efficiency of the proposed approach.

KEYWORDS: Robot manipulators; Configuration space obstacle; Bit maps; Motion planning; Dynamic environment.

1. Introduction

The notion of Configuration Space (C-space), which is defined in terms of the parameters that specify the position and posture of a robot where each dimension represents a joint coordinate,[1] plays an important role in robot motion planning. In a C-space, the obstacles in the workspace are mapped as forbidden regions (referred to as C-obstacle), and the complement of the C-obstacle constitutes the free space C_{free}. Path planning for a robot with n degrees-of-freedom (DOF) can thus be converted to the problem of planning a path for a particle in an n-dimensional C-space. A series of methods have been proposed for the construction of C-space, some of which involve computing the boundary of the C-obstacle analytically. Lozano-Perez considered the case where both the robot and the obstacles were convex polygons or polyhedra, and the C-obstacle boundary for an n-DOF manipulator was approximated by sets of $(n-1)$-dimensional slices recursively built up from one-dimensional slices.[2] Donald characterized the five-dimensional C-space boundaries of a six-dimensional C-obstacle using an algebraic format of constraints based on the contact condition.[3] Maciejewski and Fox also studied the analytical description of the boundaries of C-obstacle and derived the connectivity of C-space for revolute manipulators.[4] Zhao *et al*. developed an analytical representation of C-obstacle using a set of parametric equations.[5] These parametric equations were resulted from mapping the boundaries of the obstacles from workspace into the C-space through using the inverse pseudo kinematics. The aforementioned analytical approaches have been demonstrated on robots with small number of DOFs. Nevertheless, the research on analytical approach for C-space construction with large number of DOFs under practical environment has been rare, due to the much increased complexity in analytically representing the kinematic structures in such applications.

An alternative route is to obtain an approximation of the C-obstacle by using a discretization of the configuration space, such as a bitmap or a gray-scale representation. A bitmap can explicitly represent the free part of the C-space with 0, and represent the part that gives rise to collisions with an obstacle with 1. Newman and Branicky identified the elemental building blocks that can be easily transformed from the workspace to the C-space.[6] They stored the C-space transforms of shapes as bitmaps first, and then used the superposition of these primitive maps to construct the configuration space maps for industrial robots. Lozano-Perez and O'Donnell implemented a similar approach to the parallel robot motion planning.[7] These primitive bitmaps, however, were limited to 2-DOF RR-type manipulators or puma-like robots with consecutive parallel rotational joints. Kavraki[8] and Curto and Moreno[9] explored the general method of bitmap calculation for C-obstacle without heuristics, where the C-space maps of obstacles were computed using the Fast Fourier Transform (FFT). This algorithm is based on the observation that the configuration space is a convolution of the workspace and the robot.

* Corresponding author. E-mail: jtang@engr.uconn.edu

Kyatkin and Chirikjian[10] and Chirikjian and Ebert-Uphoff[11] have considered gray-scale descriptions of C-obstacle. They used the discrete-motion group as an approximation to the Euclidean group, and also applied FFT techniques to calculate the C-obstacles of mobile robots, which move among static obstacles. The obvious advantage of discretized representation of C-obstacle is that it provides a simple means for superposing precomputed maps, and it can take advantage of the well-established FFT technique including computing hardware implementation. The main drawback of this technique, however, is that it is computationally costly. While it has been recognized that the discretized representation of C-space (or C-obstacle) could potentially have wide applications in robot motion planning involving large number of DOFs especially in path searching, currently the existent practical motion planners based on explicit representation of the C-space are still hampered by the curse of dimensionality.[12]

In this paper, a new two-phase approach for the construction of C-space is proposed. The proposed approach is based upon precomputing the global topology of a robot's free space, and consists of an offline phase and an online phase. In the offline phase, a C-obstacle database (COD) for a given robot is developed in which the C-obstacle maps are stored and indexed by the cells of the workspace; in the online phase when the same robot is operated in a real environment, those maps whose indices match the real obstacle cells are identified and then extracted from the COD. The superposition of these maps forms the forbidden region in operation or the C-obstacle. This proposed approach is a generic one and can be applied to manipulators with arbitrary kinematic structures and geometries. The construction of the COD, which is generally the most time-consuming step, is implemented in the offline phase, and the online computing only involves the identification of the components matching the COD indices. This leads to a significant reduction of computational time required for the online C-space construction. This new approach for the C-space construction can be realized in a real-time online fashion and is especially suitable for manipulator motion planning in dynamic environments. We use a series of simulation cases involving a 3-DOF manipulator and a 5-DOF manipulator to demonstrate the performance of the proposed scheme.

2. Basic strategy for C-obstacle database construction and matching

Let $q = (q_1, \ldots, q_n)$ denote a configuration of a manipulator with n DOFs, where each element of q is a joint parameter, measuring either the angular displacement or the linear displacement depending on the type of the joint. The proposed approach for C-space construction consists of two phases, i.e., an offline phase and an online phase.

2.1. Offline C-obstacle database creation

In the offline phase, at first, the workspace, denoted as W, is decomposed into cells under certain resolution defined as $W = \cup w_i$. Then, for each cell of the workspace, w_i, we find all configurations of the robot, denoted as Q_{w_i}, under which we assume robot A passes position w_i in the workspace. Mathematically, we define $Q_{w_i} = \{q | A_q \cap w_i \neq 0\}$, where A_q refers to robot A in configuration q. Q_{w_i} constitutes the C-obstacle in the C-space corresponding to a particle-type obstacle located at position w_i in workspace, and it is referred to as the C-obstacle map. All of the C-obstacle maps can be organized into a database, called C-obstacle database (COD), and indexed by w_i.

Here we use a planar manipulator to illustrate the procedure for offline COD creation. As shown in Fig. 1(a), in the x–y plane, there exists an RR manipulator A and a rectangular-type obstacle B. Figure 1(b) shows the grid of the workspace W after its decomposition, $W = \cup w_i$. As the first step of the proposed approach, the C-obstacle database for robot A will be built in the offline phase, i.e., for each cell w_i in W, we set up the C-obstacle map Q_{w_i}. Based on the rationale that the robot will collide with an obstacle whenever this obstacle overlaps spatially with any part of the robot, we may develop a procedure for setting up the C-obstacle database containing all the C-obstacle maps for robot A shown in Fig. 1(a). As an illustrative example, for an arbitrary cell w^* selected from W, those configurations constituting Q_{w^*} are given in the workspace (Fig. 1(c)) and in the C-space (Fig. 1(d)), respectively. The offline C-obstacle database can be expressed as

$$\begin{aligned} \mathrm{COD} &= \{q | A_q \cap W \neq 0\} = \{q | A_q \cap (\cup w_i) \neq 0\} \\ &= \cup(\{q | A_q \cap w_i \neq 0\}) = \cup(Q_{w_i}) \end{aligned} \tag{1}$$

Let N_A be the total number of cells of robot A, and let $\theta_{i\max}$ and $\theta_{i\min}$ be the maximum and minimum value of each joint variable, respectively. We may have the following pseudo code.

```
for θ1 = θ1 min : θ1 max
  for θ2 = θ2 min : θ2 max
    {
      Decompose robot A under configuration (θ1, θ2),
        so that A(θ1,θ2) = ⋃_{i=1}^{NA} ai;
      for k = 1 : NA
        add configuration (θ1, θ2) to Qai
    }
```

(1)

2.2. Online C-space construction

In the online phase, for a real-time scenario containing the same robot A and an obstacle B, the C-obstacle transformed from B relative to A, $CO_A(B)$, can be obtained based on the offline C-obstacle database. Indeed, the obstacle B is firstly decomposed under the same grid resolution defined in the offline phase, i.e., $B = \cup b_i$. Then the offline C-obstacle database is searched and those maps Q_{b_i} with indexes matching b_i are extracted. According to the union property for C-space,[1] the superposition of Q_{b_i} is thus the real-time C-obstacle during manipulator operation, i.e., $CO_A(B) = \cup b_i$. For the illustrative case shown in Fig. 1(a), we may assume that the decomposition result of the obstacle B is based on the same resolution used for W, i.e., $B = \cup b_i = \{b_1, b_2, b_3, b_4\}$.

https://doi.org/10.1017/S0263574706002943 Published online by Cambridge University Press

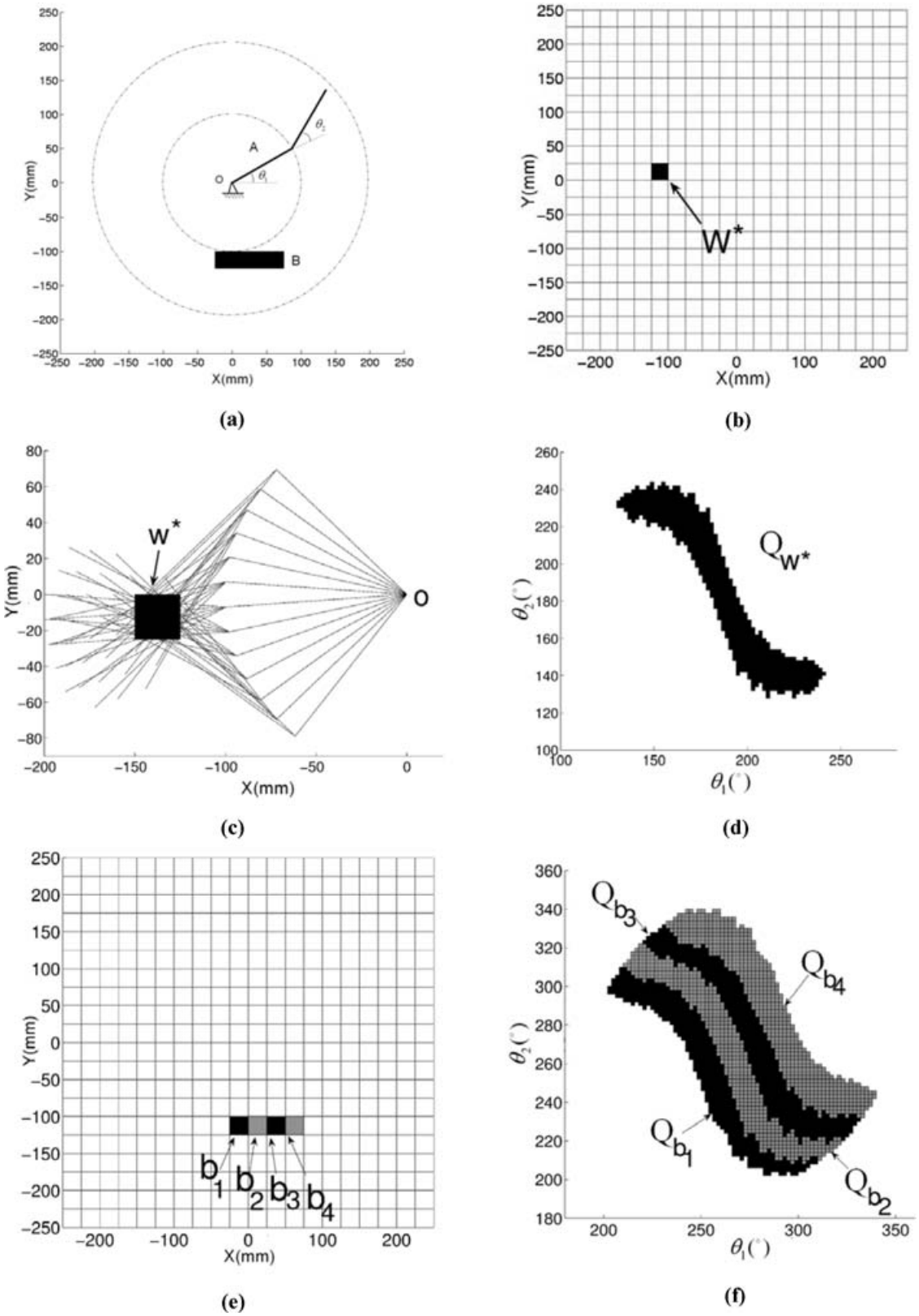


Fig. 1. Simulation results of planar RR manipulator. (a) Scenario, (b) Grid of W, (c) Q_{w^*} shown in workspace, (d) Q_{w^*} shown in configuration space, (e) decomposition of B, (f) $\cup Q_{b_i}$ shown in configuration space.

We then obtain the resultant C-obstacle $CO_A(B)$ as follows, which is also shown in Fig. 1(f),

$$CO_A(B) = \{q|A_q \cap B \neq 0\} = \{q|A_q \cap (\cup b_i) \neq 0\}$$
$$= \cup(\{q|A_q \cap b_i \neq 0\}) = \cup(Q_{b_i}) \quad (2)$$

As can be seen from the procedure described above, the construction of the C-obstacle database in the offline phase does not require any knowledge of the obstacles in the real workspace. Once the COD for a given robot is constructed, it can be utilized as a data source in

https://doi.org/10.1017/S0263574706002943 Published online by Cambridge University Press

different application scenarios. Since the time-consuming setup of a COD is realized in the offline phase and only the identification of maps in the COD is needed in the online phase, the proposed approach is thus expected to be able to deliver real-time result for C-space construction. In Section 4, we will examine the running time of this proposed approach in some typical cases. The real-time property also makes the approach especially suitable for manipulator motion planning in dynamic environment. In such type of planning, the robot does not have *a priori* information about the environment, thus scenario detection via sensors is required, as described in [13, 14]. The proposed approach can then be applied in such a sequence: At each sampling moment, the obstacles are detected by sensors and decomposed into cells, then the corresponding C-space is constructed by C-obstacle map superposition; after that, a suitable search algorithm such as depth first, Dijkstra's algorithm, A*, or best first[15] can be employed to search for collision-free path within the constructed C-space.

3. Representation scheme

Since the amount of space required for storing the COD increases drastically as the dimension of the robot and the resolution in the workspace and C-space increase, an efficient representation scheme for an object in the workspace and C-space is critical, which is detailed as follows.

3.1. Representations of the robot and the obstacle in the workspace

When the robot and the obstacle are decomposed in the workspace, both of them can be represented by either the boundary or the entity fashion. We may design four patterns involving different representation modes of robot A and obstacle B in a planar case, as shown in Figs. 2(a)–(d). The following observations can be made.

- If robot A is represented by the boundary, the calculated result of superposition $\cup Q_{b_i}$ contains the complete boundary and part of internal cells of the C-obstacle $CO_A(B)$, as shown in Fig. 2(e).
- If the robot is represented by the entity, i.e., the boundary plus all the internal cells, then no matter the obstacle is represented by boundary or entity, the result of $\cup Q_{b_i}$ contains not only the boundary, but also all the internal cells of $CO_A(B)$, which is shown in Fig. 2(f). This can be explained by the fact that when A coincides with B at one of B's internal cell, e.g., b_{in}, then A must at the same time coincide with B at one boundary cell of B, e.g., b_{bound}, and therefore $\forall b_{\text{in}} \in B$, $Q_{b_{\text{in}}} \subset (\cup Q_{b_{\text{bound}}})$.

Based on the above observations, the following representation strategies will be used in this paper. If only the boundary of the C-obstacle is sought, we choose boundary representation for both the robot and the obstacle; if the whole entity of the C-obstacle is sought, we use boundary representation for the obstacle and use entity representation for the robot.

3.2. C-space representation

In general, the 2^m tree representation of a space or an object Ω is a hierarchical data structure of degree 2^m, where m is the degree of Ω. Each node of the tree is a cuboid cell which is labeled as Black, White, or Gray, respectively. Only those nodes that are Gray may have children, and each of those nodes has 2^m children.[16] The 2^m tree structure can save memory storage by merging a set of neighboring White cells or neighboring Black cells into a single White or Black cell at the upper level. In this research, we employ the approach of 2^m tree data structure, outlined in [2, 17], to represent a robot C-space.

In real applications, if a robot C-space is represented by a uniform 2^m tree, i.e., along each coordinate, the space will be divided into the same number of intervals, which yields a huge number of cells in the C-space. For example, for a manipulator with 6 DOF, if a maximum depth of 5 is assigned to each coordinate, then the total number of nodes in C-space will be $2^{5\times 6} \approx 10^9$. We may, actually, reduce the size of the required data set. Firstly, for a manipulator, under the same amount of joint change, the smaller the distance between the joint and the base, the larger the end-effector's maximum movement will be in the workspace. This leads to one category of criterion to determine the joint resolution, that is, the step length along each joint coordinate should result in almost equal displacement of the robot end-effector in the workspace. Considering Fig. 3(b), we may determine the resolution as

$$\Delta q_i = 2 \arcsin\left(\frac{\text{MaxMove}}{2l_i}\right) \tag{3}$$

where MaxMove is the required moving precision of the end-effector in the workspace. This selection of resolution may avoid abrupt motions and yield smooth paths.[18,19]

Secondly, if the resolution of each joint is sufficiently small, no additional collision check has to be made for adjacent link positions. Furthermore, the swept volume between adjacent link positions is zero, and we have reasonable workspace coverage. For a two-dimensional link, if the swept area between two adjacent link positions is zero, no collision check for intermediate positions is needed, as shown in Fig. 4, where the maximum joint change is defined as[19]

$$\Delta\phi = \arctan(r/l) \tag{4}$$

where r and l are the link height and length, respectively. We may then have the second category criterion, i.e., the chosen maximal step length should lead to a reasonable workspace coverage.

For either criterion, the C-space should be divided into different number of cells along each joint coordinate, which leads to a nonuniform 2^m tree representation. As an example, the depth of a 2^m tree to represent the C-space of an educational robot ESHED SCORBOT ER4pc with 5 DOF is obtained according to category 1 and 2 criteria, respectively, which is shown in Table I. In this example, joint 5 is an exception whose rotation does not change the position of the end-effector, thus, $l_5 = 0$, and a small-depth value is

https://doi.org/10.1017/S0263574706002943 Published online by Cambridge University Press

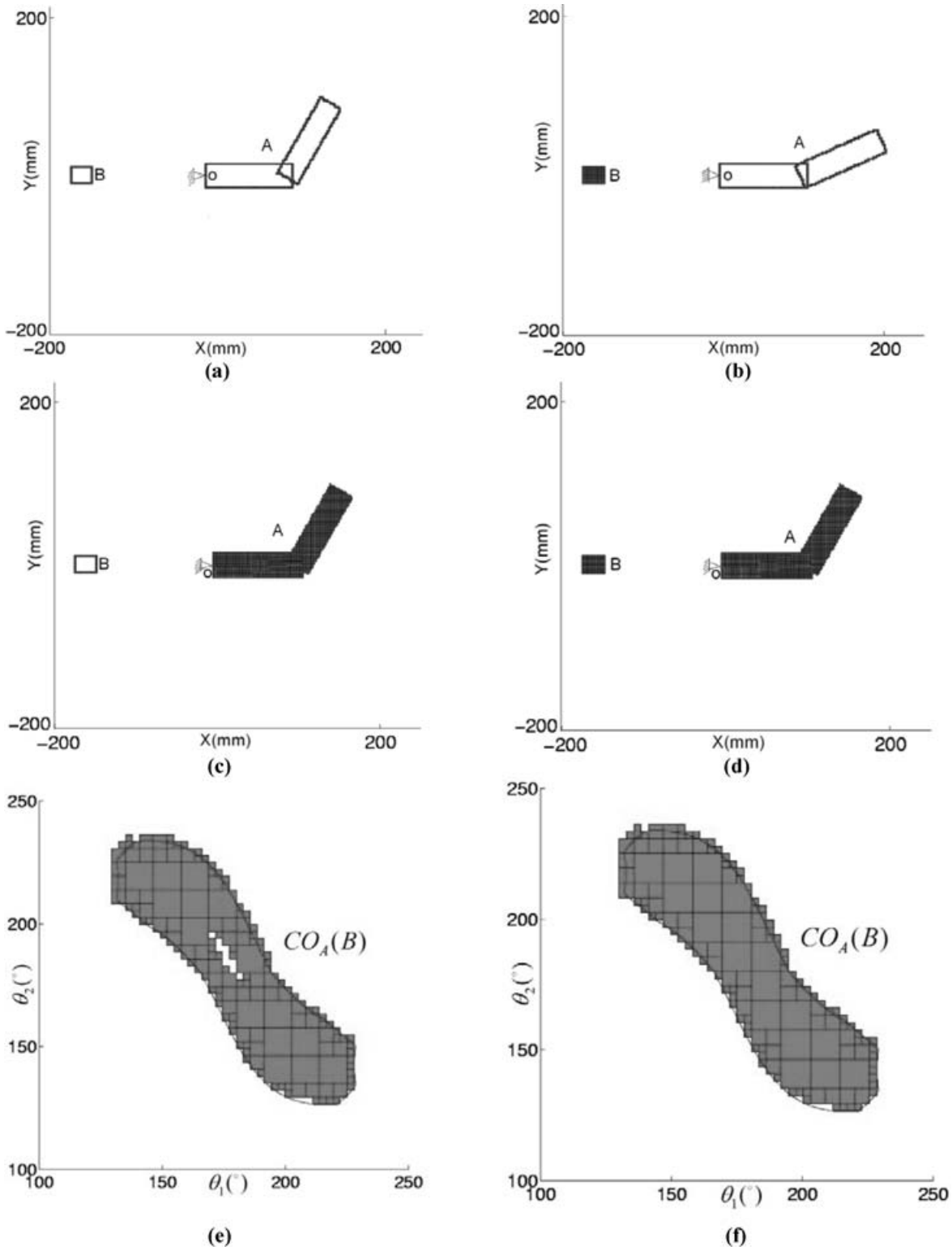


Fig. 2. Different representation modes in workspace, and the corresponding C-obstacle. (a) W-space (A: boundary, B: boundary), (b) W-space (A: boundary, B: entity), (c) W-space (A: entity, B: boundary), (d) W-space (A: entity, B: entity), (e) C-space calculated from (a) or (b) [Resolution: 128×128 in W, 128×128 in C], (f) C-space calculated from (c) or (d) [Resolution: 128×128 in W, 128×128 in C].

chosen for joint 5. In later simulations in this paper, we choose the depth value as (5,5,4,3,3) based on the category 1 criterion.

Figure 5 shows a nonuniform quadtree sample (where $m = 2$). As can be seen from Fig. 5(c), each node of the quadtree is equally divided into four children at both depth 1 and 2; but at depth 3, the map is divided only along the latitudinal coordinate, i.e., each node is divided into two sub-nodes only. As also shown in Fig. 5(c), each quadtree node is labeled with an index value, referred to as a Morton code,[20] which maps an n-dimensional vector to a one-dimensional scalar. A Morton code has one or several digits, each of which

https://doi.org/10.1017/S0263574706002943 Published online by Cambridge University Press

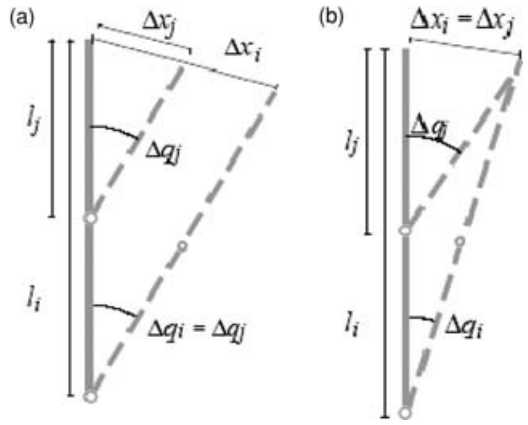


Fig. 3. The uniform and nonuniform discretization of joints (from [18]). (a) Uniform discretization ($\Delta q_i = \Delta q_j$) results in different Cartesian movements $\Delta x_i \neq \Delta x_j$ when different joints i, j are moved, (b) Reasonable discretization ($\Delta q_i \neq \Delta q_j$) results in equal maximum Cartesian movement $\Delta x_i = \Delta x_j$ when different joints i, j with distance l_i, l_j to the end-effector are moved.

Fig. 4. Adjacent link positions with swept area 0 between them (from [19]).

Table I. Process to determine the resolution leading to nonuniform C-space discretization.

		Joint				
		1	2	3	4	5
r (mm)		150	70	70	40	15
l (mm)		595	580	360	140	0
Cat. 1	Resolution ratio	$\frac{1}{595}$	$\frac{1}{580}$	$\frac{1}{360}$	$\frac{1}{140}$	NA
	Rounded resolution ratio	$\frac{1}{2^5}$	$\frac{1}{2^5}$	$\frac{1}{2^4}$	$\frac{1}{2^3}$	$\frac{1}{2^3}$
Cat. 2	$\Delta\phi_i = \arctan(r/l)$	14.3°	7°	11°	16.4°	NA

is called a directional digit. The Morton code specifies the node position in the tree and its position with respect to its siblings in the following manner.

- A node can be tracked by starting from the root, traversing downwards and following its directional digit at each depth. Shown in Fig. 5 as an example, the black node with Morton code 211 can be tracked along the dotted line.
- At the longitudinal direction, counting from the left-most cell, the decimal value of the Morton code denotes the sequential position of the corresponding cell in the current depth.

The Morton code can be obtained by binary interleaving of the cell's coordinates.[16] Alternatively, the sample space in Fig. 5(a) can be divided only along the latitudinal coordinate at depth 1 and be divided along both coordinates at both depth 2 and 3, which results in a different but equivalent nonuniform quadtree, as shown in Fig. 6. In the remainder of this paper, the first convention will be employed.

In Fig. 5(d), two arrays with bit encoding are adopted to represent this quadtree. In the Black/White array, a single binary bit is used to denote whether the corresponding tree node contains black element, i.e., bit value 0 represents that the node is White; and value 1 means the node is Black or Gray. Similarly, in the Leaf array, bit value 1 represents nonleaf node and value 0 for leaf node. Under this strategy, only 8 bytes are needed to store all the information of the quadtree shown in Fig. 5(c), which reduces the memory requirement significantly than the common array or pointer-type representations.

It is worth noting that certain trade-off exists for the 2^m tree representation. Before any operation about the 2^m tree proceeds, the data stored in the tree representation need to be decoded from the compressed bit encodings, which could to certain extent slow down the speed for tree traversing and operations such as union and subtraction, etc.

3.3. Optimal resolution selection

For the proposed approach of C-space construction, the selection of resolution in the workspace and the selection of resolution in the C-space have a significant impact on both the computational cost and the accuracy. Several sets of resolutions are tested for the aforementioned illustrative case. The calculated C-obstacles are shown in Figs. 2 and 7, which leads to the following observations.

- Under a fixed resolution in workspace, as the resolution in C-space gets finer, the calculated C-obstacle boundary gets smoother and converges continuously to theoretical C-obstacle boundary, as illustrated in Figs. 2(f) and 7(a), (b).
- Under a fixed resolution in C-space, the increase of resolution in workspace leads to the same trend, which is shown in Figs. 2(f) and 7(c), (d).

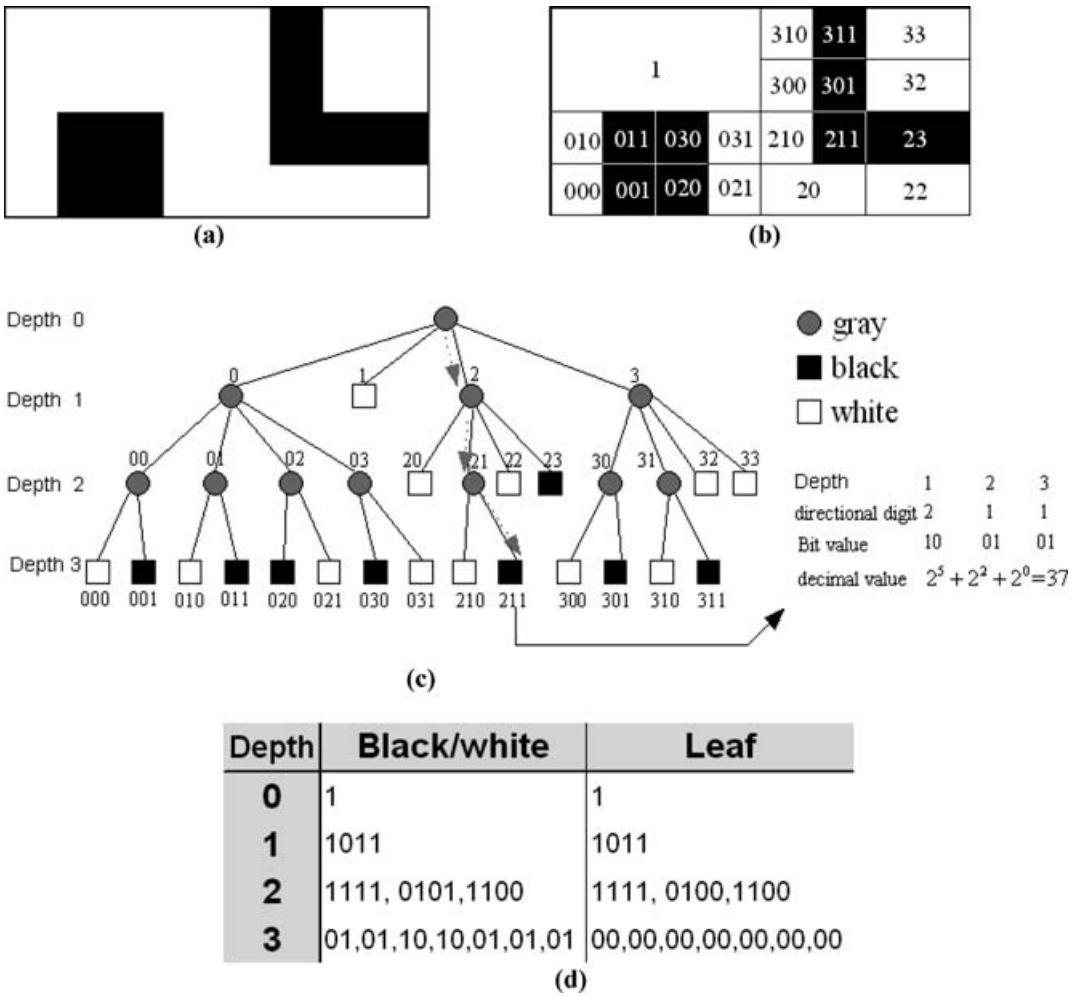
Normally, the precision requirement is determined by a real robot application, hence under the premise that the precision is met, low resolution can be chosen to reduce the computational cost.

3.4. Encoding the C-obstacle database over neighborhoods in workspace

Usually, there are only minor variations in the C-obstacle maps over local neighborhoods of the workspace, and the C-obstacle database can be encoded for further storage space reduction.

3.4.1. Reference C-obstacle maps. As an approximation, the C-obstacle map corresponding to an arbitrary cell in the workspace may be represented by the bitmap corresponding to the nearest neighboring reference cell in workspace. The neighborhoods of cells in the workspace can be obtained by tree decomposition. The density of reference cells in the workspace is determined by the kinematic structure

https://doi.org/10.1017/S0263574706002943 Published online by Cambridge University Press



Depth	Black/white	Leaf
0	1	1
1	1011	1011
2	1111, 0101,1100	1111, 0100,1100
3	01,01,10,10,01,01,01	00,00,00,00,00,00,00

(d)

Fig. 5. Quadtree demo for nonuniform 2^m tree. (a) A 2D sample space, (b) non-uniform quadtree decomposition and node labeling, (c) Quadtree representation of the sample space, (d) The bit encoding result of the quadtree in (c).

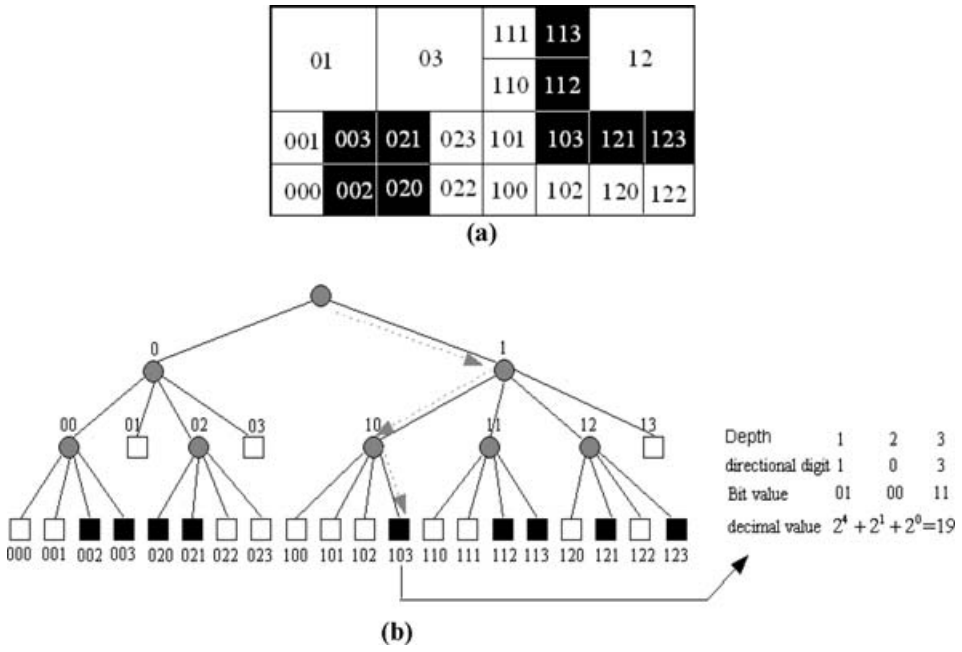


Fig. 6. Nonuniform space decomposition under another convention. (a) Another format of non-uniform decomposition, (b) quadtree demo under another decomposition convention.

https://doi.org/10.1017/S0263574706002943 Published online by Cambridge University Press

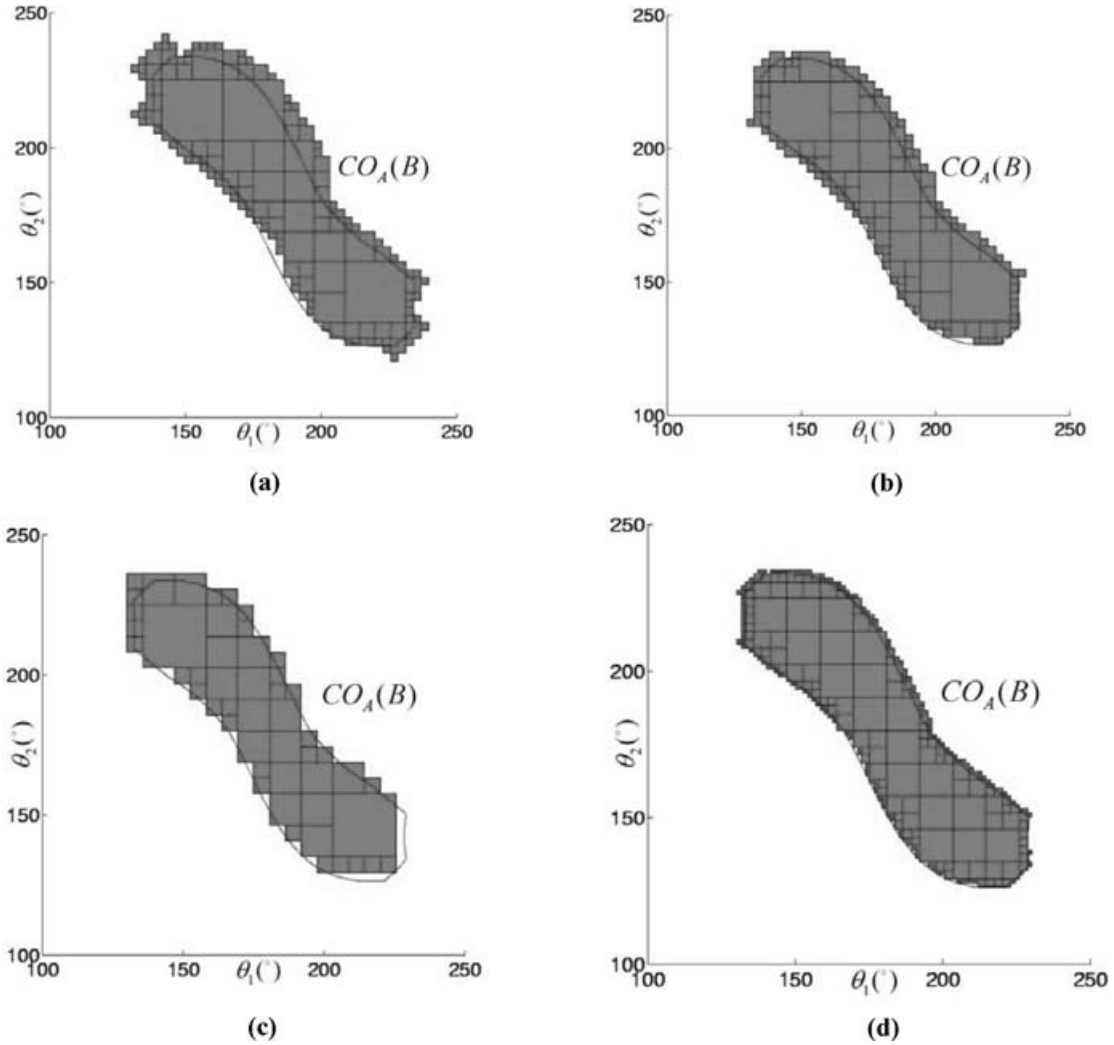


Fig. 7. Calculated C-obstacle under different resolution settings. (a) C-space (Resolution: 32 × 32 in W and 128×128 in C), (b) C-space [Resolution: 64 × 64 in W and 128 × 128 in C], (c) C-space [Resolution: 128×128 in W, 64 × 64 in C], (d) C-space (Resolution: 128×128 in W, 256×256 in C). (The real line in each sub-figure denotes the theoretical C-obstacle boundary, which is obtained from analytic derivation.)

and geometric dimensions of the robot. Figure 8 shows an example C-obstacle map corresponding to the case given in Fig. 2(c), which is calculated based on this approximation. Under the adopted resolution, the deviation between the approximate C-obstacle and the original one is noticeable but small.

3.4.2. Interpolation of the reference C-obstacle maps. Another reasonable approximation of the C-obstacle map corresponding to an arbitrary cell in the workspace can be made by using the linear interpolation between the reference C-obstacle maps. The principal of this linear interpolation is illustrated in Fig. 9(a) and (b). Such a scheme is analogous to the morphing procedure in image processing that employs all nodes as the so-called key points. Figure 9(c) shows the C-obstacle calculated by using the linear interpolation that also corresponds to the case given in Fig. 2(c). Compared with the result from purely neighborhood encoding, the C-obstacle calculated by the linear interpolation technique matches the original boundary better.

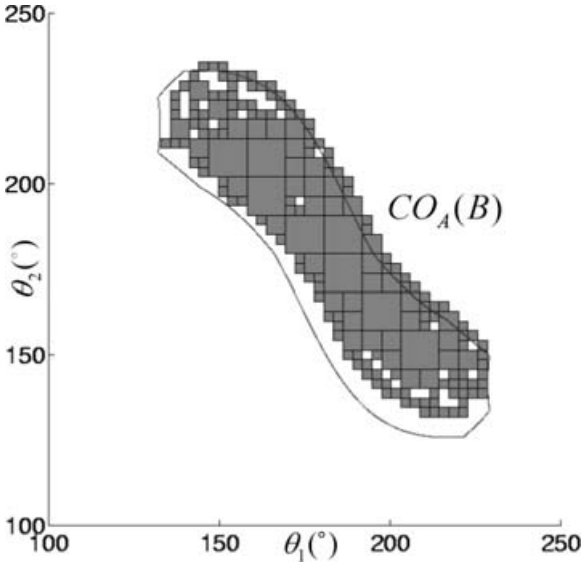


Fig. 8. C-space calculated based on neighboring encoding. (Resolution: 128×128 in W, 128×128 in C. Density of reference cells in W: 32×32).

https://doi.org/10.1017/S0263574706002943 Published online by Cambridge University Press

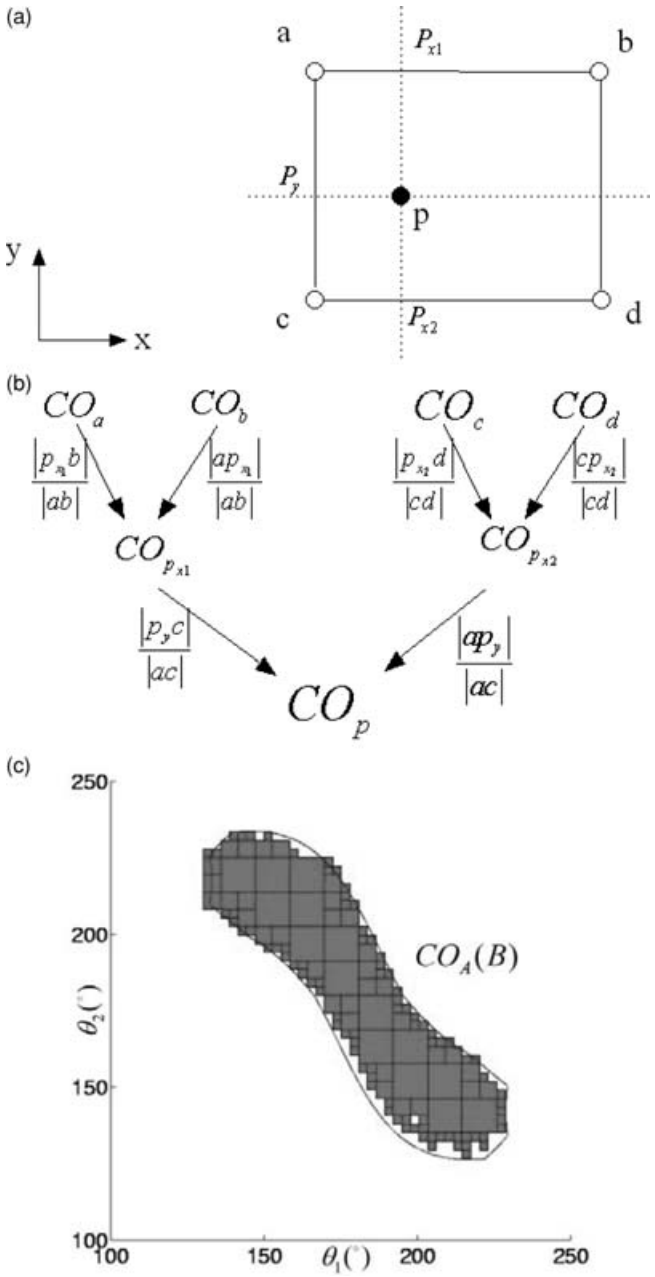


Fig. 9. Demonstration of C-obstacle interpolation in a 2D case. (a) Reference grid in workspace, (b) obtain CO_p by linear interpolation (interpolation weights are marked on the arrows), (c) calculated C-obstacle over neighborhoods in workspace (resolution: 128×128 in W and 128×128 in C; density of reference cells in W: 32×32).

4. Case studies and complexity analysis

In this section, we perform case studies to verify and demonstrate the proposed approach. Some of the simulations have already been presented in the previous sections as illustrative examples. Here in this section, we focus on the computational efficiency and complexity analysis. The computer programs for simulation are written in C++ language. All simulations are run on a personal computer with 1.5 GHz Pentium-M CPU and 512 MB memory under Windows XP operating system.

Table II. Settings and calculation result of the 3-DOF robot.

Settings in workspace	Range of space	[−250, 250] × [−250, 250] × [−250, 250]
	Resolution	3.9 mm, 3.9 mm, 3.9 mm
	No. of cells	$128\times128\times128 = 2.1\times10^6$
	No. of reference cells	$16\times16\times16 = 4096$
Settings in C-space	Range of space	[0, 360°] × [0, 360°] × [0, 360°]
	Resolution	2.81°, 2.81°, 2.81°
	No. of cells	$128\times128\times128 = 2.1\times10^6$
Storage scheme	2^mtree + bit encoding + Neighborhood encoding	
Running time	COD setup	≈ 2 h
	C-space construction for scenario in Fig. 10(a)	0.26 s
Storage space	24 MB	

4.1. C-space construction for RRR robot

In this first demonstration case, a 3-DOF spatial manipulator is employed as the prototype. Its three links are modeled as proximate cylinders with the same dimension (radius = 25 mm, length = 100 mm). Each joint is allowed to move from 0 to 360°. A cuboid obstacle is placed within the reachable range of the robot.

The workspace and the calculated C-obstacle are shown in Fig. 10, and the simulation parameters and the computational performance are listed in Table II. In this simulation, the neighborhood of workspace cells is utilized and the density of reference cells is $128\times128\times128 = 2.1\times10^6$. In [7], the time to build a $64\times64\times64$ configuration space for the first 3 DOFs of the Puma robot on a Thinking Machines' Connection Machine with 8 k processors is approximately 2 s. It should be noted that a rigorous comparison of computational efficiency is difficult to achieve, because of the difference in computing platform, programing language, and operating system, etc. However, with the current result we may conclude that the online running time of the proposed approach is much shorter and warrants the real-time application under dynamic operations.

4.2. C-space construction for a 5-DOF robot

Currently, manipulators with 5 or more DOFs are widely used in industry. Clearly, constructing the explicit C-space for these types of manipulators is quite useful for practical applications. It is worth emphasizing that due to the system complexity in kinematic structure and geometric shape, very few successful studies on C-space construction for such manipulators have been reported in literature. In this simulation case, ESHED SCORBORT ER4pc, a 5-DOF educational robot is used as the prototype to evaluate the proposed approach. The simulation scenario is shown in Fig. 11, where a wall is arranged as the obstacle. During the simulation, the CAD model of the robot links is loaded into the program to facilitate the construction of the COD. The simulation parameters and the computational performance are listed in Table III. While the offline running time based on the

https://doi.org/10.1017/S0263574706002943 Published online by Cambridge University Press

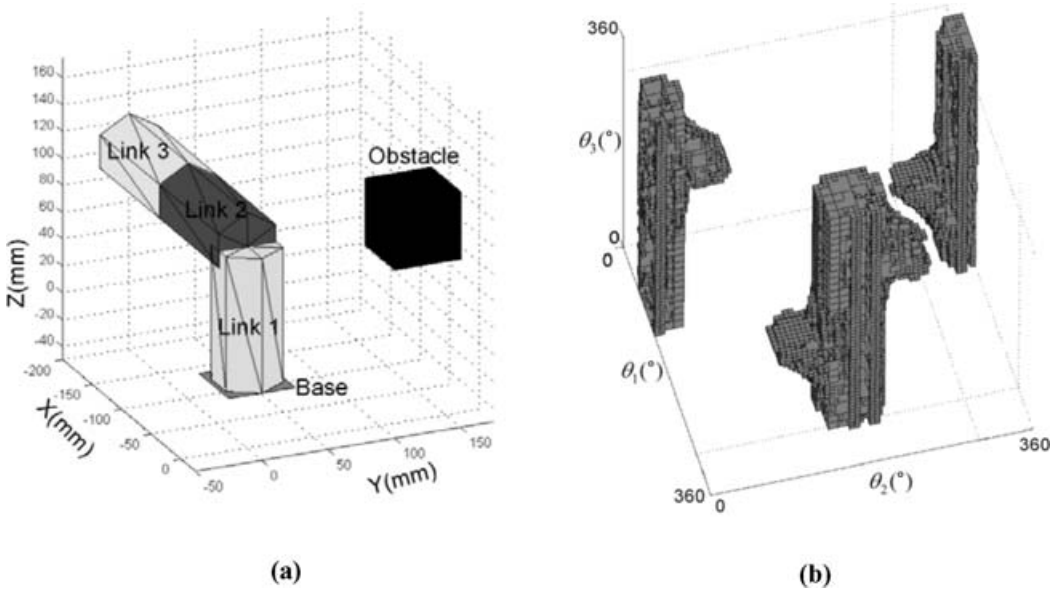


Fig. 10. Simulation results of a spatial RRR manipulator. (a) RRR manipulator and the obstacle in workspace, (b) calculation result of C-obstacle.

proposed approach is comparable to the results presented in [19], the extremely short online running time again shows that this approach can lead to the real-time motion planning under dynamic operations.

4.3. Computational complexity analysis

The worst-case time complexity of the proposed algorithm can be analyzed as follows.

- In the offline phase, the complexity of creating a COD is $O(C_{\mathrm{W}} \cdot C_{\mathrm{C}})$, where C_{W} denotes the complexity of the robot workspace and it can be expressed as $\prod_{d=x,y,z}(N_d)$ with N_d being the resolution in each dimension of the workspace, C_{C} denotes the complexity of a C-obstacle map and it can be expressed as $\prod_{d=1}^{n}(N_d)$ with N_d being the resolution in each dimension of the C-space, and n is the number of DOF. Therefore, the complexity of the COD can be denoted as $\prod_{d=x,y,z,1,\ldots,n}(N_d)$.
- In the online phase, the complexity for C-obstacle assembling can be expressed as $O(C_{\mathrm{O}} \cdot C_{\mathrm{C}})$, where C_{O} is the complexity of the obstacles, C_{C} is still the complexity of a C-obstacle map.

In normal robot applications, the obstacles occupy only a small part of the robot workspace, thus the complexity of obstacles C_{O} is far less than that of the workspace C_{W}. Furthermore, the hierarchical representation scheme of the C-obstacle maps can greatly reduce the time complexity in the

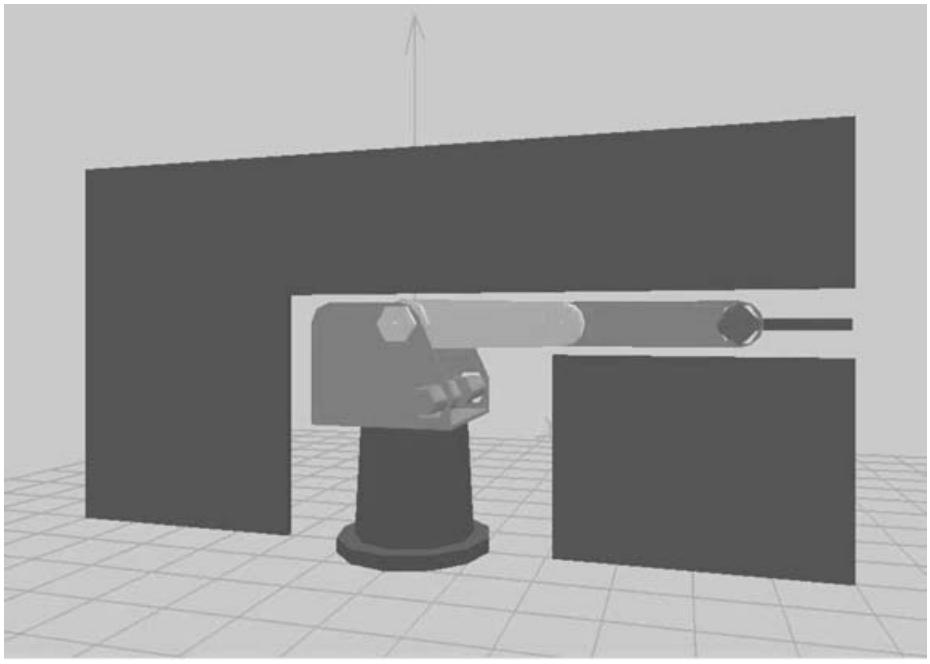


Fig. 11. ESHED SCORBOT-ER4PC robot and the obstacle.

https://doi.org/10.1017/S0263574706002943 Published online by Cambridge University Press

Table III. Settings and calculation result of the SCORBOT robot.

Settings in workspace	Range of space	$[-600, 600] \times [-600, 600] \times [-600, 600]$
	Resolution	18.75 mm, 18.75 mm, 15.625 mm
	No. of cells	$64 \times 64 \times 64 = 2.62 \times 10^5$
	No. of obstacle cells after the wall is decomposed	1840
Settings in C-space	Range of space	$[-130.0, 180^\circ] \times [-125.0, 40.0^\circ] \times [-130.0, 130^\circ] \times [-220.0, 40.0^\circ] \times [0, 360^\circ]$
	Resolution	4.84°, 5.15°, 16.25°, 32.5°, 42.0°
	Total no. of cells	10^6
Storage scheme	2^m tree + bit encoding	
Running time	COD setup	≈ 4 h
	C-space construction for scenario in Fig. 11	2.015 s
Storage space	125 MB	

online phase. As an example, in the aforementioned second case study, the offline phase of COD creation costs nearly 4 h while the online C-space construction corresponding to a wall-type obstacle only needs 2.015 s. It should be noted that with the further increase of the number of DOFs, the storage space requirement could increase drastically. For those applications, the inherent representation of the C-space connectivity, such as the dynamic roadmap[21] could be promising.

5. Concluding remarks

In this paper, a new two-phase approach for C-space construction of manipulators is proposed. In the offline phase, a C-obstacle database which stores the C-obstacle maps indexed by the cells of the workspace is created. Based on the offline C-obstacle database, one can obtain the C-space under real-time operation. The proposed approach is generic in nature and its online running time is extremely short, which makes it especially suitable for manipulator motion planning under dynamic environments. Simulations on 3-DOF and 5-DOF manipulators have verified the improved performance, and suggested the potential for real-time application under dynamic operating conditions.

References

1. T. Lozano-Perez, "Spatial planning: A configuration space approach," *IEEE Trans. Comput.* **C-32**, 108–120 (1983).
2. T. Lozano-Perez, "A simple motion planning algorithm for general robot manipulators,"*IEEE Trans. Robot. Autom.* **3**, 224–238 (1987).
3. B. Donald, "A search algorithm for motion planning with 6-degrees of freedom," *Artif. Intell.* **31**, 295–353 (1987).
4. A. Maciejewski and J. Fox, "Path planning and the topology of configuration space," *IEEE Trans. Robot. Autom.* **9**, 444–456 (1993).
5. C. Zhao, M. Farooq and M. Bayoumi, "Analytical solution for configuration space obstacle computation and representation," *Proceedings of the 21st International Conference on Industrial Electronics, Control, and Instrumentation (IECON)* (1995) pp. 1278–1283.
6. W. S. Newman and M. S. Branicky, "Real-time configuration space transforms for obstacle avoidance," *Int. J. Robot. Res.* **10**, 650–667 (1991).
7. T. Lozano-Perez and P. O'Donnell, "Parallel robot motion planning," *Proceedings of IEEE International Conference on Robotics and Automation* (1991).
8. L. Kavraki, "Computation of configuration space obstacles using the fast fourier transform," *IEEE Trans. Robot. Autom.* **11**, 408–413 (1995).
9. B. Curto and V. Moreno, "Mathematical formalism for the fast evaluation of the configuration space," *IEEE International Symposium on Computational Intelligence in Robotics and Automation* (1997) pp. 194–199.
10. A. B. Kyatkin and G. S. Chirikjian, "Computation of robot configuration and workspaces via the Fourier transform on the discrete-motion group," *Int. J. Robot. Res.* **10**, 650–667 (1999).
11. G. S. Chirikjian and I. Ebert-Uphoff, "Numerical convolution on the Euclidean group with applications to workspace generation," *IEEE Trans. Robot. Autom.* **14**, 123–136 (1998).
12. K. Gupta and A. Pobil, *Practical Motion Planning in Robotics: Current Approaches and Future Directions* (Wiley, New York, 1998).
13. H. Choset and J. Burdick, "Sensor-based exploration: The hierarchical generalized Voronoi graph," *Int. J. Robot. Res.* **19**, 96–125 (2000).
14. H. Choset, S. Walker, K. Eiamsa-Ard and J. Burdick, "Sensor based exploration: Incremental construction of the hierarchical generalized Voronoi graph," *Int. J. Robot. Res.* **19**, 126–148 (2000).
15. S. M. LaValle, *Planning Algorithms* (Cambridge University Press, Cambridge, 2006).
16. H. Samet, *The Design and Analysis of Spatial Data Structures* (Addison-Wesley, Massachusets, 1990).
17. B. Faverjon, "Obstacle avoidance using an octree in the configuration space of a manipulator," *Proceedings of IEEE International Conference on Robotics and Automation* (1984) pp. 504–512.
18. D. Henrich, C. Wurll and H. Worn, "On-line path planning with optimal C-space discretization," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (1998).
19. E. Ralli and G. Hirzinger, "A global and resolution complete path planner for up to 6 dof robot manipulators," *Proceedings of IEEE International Conference on Robotics and Automation* (1996) pp. 3295–3302.
20. A. K. Hanan Samet, "Octree approximation and compression methods," *Proceedings of the 1st International Symposium on 3D Data Processing Visualization and Transmission (3DPVT'02)* (2002).
21. P. Leven and S. Hutchinson, "Toward real-time path planning in changing environments," *Proceedings of Workshop Algorithmic Foundations Robotics* (2000) pp. 363–376.

https://doi.org/10.1017/S0263574706002943 Published online by Cambridge University Press