Proceedings of the 1996 IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems

Sensor-Based Obstacle Modeling in Configuration Space for Manipulator Motion Planning

Wei LI; Chenyu MA
Zushun CHEN; Qi CAO

Department of Computer Science and Technology
Tsinghua University
Beijing, 100084, P. R. China
email: lw@s1000e.cs.tsinghua.edu.cn

S.K. TSO

Center for Intelligent Design, Automation and Manufacturing
Department of Manufacturing Engineering
City University of Hong Kong, HK
email: sktso@hkueee.hku.hk

Abstract

This paper presents an approach to sensor-based obstacle modeling in a configuration space for manipulator motion planning in unknown environments. In order to achieve this objective, an efficient algorithm is used to fast map obstacles based on defined fundamental obstacles in the workspace and their images in the configuration space. A robotic manipulator is assumed to be equipped with "distance" sensors to detect obstacles in the local region. By computation of the critical points of an obstacle based on information acquired by the "distance" sensors, an obstacle model in the configuration space is constructed. By using this sensor-based configuration space modeling, robot motion planning in unknown environments can be performed in realistic time frames.

1. Introduction

It is well known that motion planning based on sensors is a key issue of manipulator application in the real world. One of the most widely used approaches to motion planning, including obstacle mapping and path searching, is based on a configuration space (C-space) modeling. The algorithms reported in [1][2][3] show that motion planning in the C-space is accurate and efficient in static environments. However, these C-space algorithms, such as cell decomposition, etc., are not suitable for sensor-based path planning in unknown environments because there is lack of a model for connection between the C-space algorithms and information from sensors. One of their deficiencies is that large amounts of computational time are needed to deal with a robot's kinematics and geometry as well as the obstacles' geometry before searching for a path.

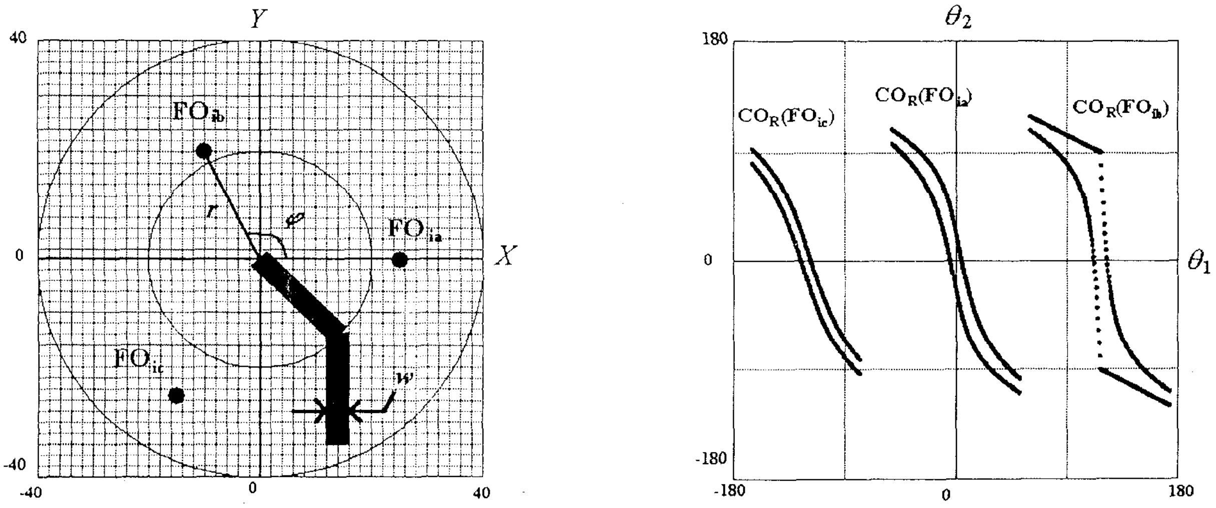
In [4], Lumelsky presents an interesting algorithm for motion planning in dynamic environments. For a manipulator, its obstacle modeling in the C-space serves to compute the collision boundaries between a robot and the obstacles. Because this modeling approach has to solve the algebraic equations of the C-space obstacles in terms of the robot's kinematics based upon a simplified geometric model of the robotic arm, it is also a time consuming work.

In [5][6][7], we present approaches for fast mapping an obstacle from a workspace (W-space) into a C-space. Its basis is to define some points in the W-space as fundamental obstacles and to precompute their C-space obstacles according to a robot's kinematics and geometry. Using the fundamental obstacles and their images in the C-space, we propose an efficient algorithm for a C-space modeling based on "distance" sensors. Its idea is to compute their approximate contours from the critical points of an obstacle based on information acquired by the sensors. On the basis of this C-space modeling, we adopt the algorithms proposed in [8] to plan a collision-free path.

This paper is organized as follows. First, considering a planar robot, Section II briefly presents the concept of fundamental obstacles and gives the algebraic computation for mapping the fundamental obstacles to the C-space. Section III proposes the method for mapping complex obstacles by using the critical points. Section IV presents an obstacle modeling in the C-space based on sensor information for motion planning. Section V extends this method for motion planning in 3D space. Finally, Section VI summarises the work presented in this paper.

2. Fundamental Obstacles and Their Images in C-Space

Before we discuss the proposed approach, we introduce *fundamental obstacles* and *their images* in the C-space. Since a two-link planar manipulator is the fundamental part of a real manipulator, such as a PUMA 560 robot, we will use it to describe our basic approach. Fig. 1a shows the W-space of the manipulator. A grid is used to discretize this W-



(a) (b)

Fig. 1. (a) Fundamental obstacles $\mathbf{FO}_{ia}$, $\mathbf{FO}_{ib}$, $\mathbf{FO}_{ic}$ in the W-space, and (b) images $\mathbf{CO_R}(\mathbf{FO}_{ia})$, $\mathbf{CO_R}(\mathbf{FO}_{ib})$, $\mathbf{CO_R}(\mathbf{FO}_{ic})$ in the C-space.

space. Intersection points of verticals and horizontals on the grid are defined as fundamental obstacles $\mathbf{FO}_i$=(x, y) shown in Fig. 1a. Each $\mathbf{FO}_i$ has two important parameters:

$$r = \sqrt{x^2 + y^2} \tag{1}$$

$$\varphi = \arctan\left(\frac{y}{x}\right) \tag{2}$$

where r is the distance between $\mathbf{FO}_i$ and the original point, and φ is the angle between r and the X axis. For example, in Fig. 1a, φ and r are two parameters of $\mathbf{FO}_{ib}$. In [7], we have in detail discussed how to choose fundamental obstacles and to locate them in W-space.

Since $\{\mathbf{FO}_i\}$ are independent of a real obstacle in an unknown environment, their C-space obstacles, denoted by $\mathbf{CO_R}(\mathbf{FO}_i)$, can be precomputed in terms of the kinematics and geometry of the robot. We first define l_1 and l_2 as the length of the first and second link of the manipulator, respectively. When $r < l_1$, the model for computing $\mathbf{CO_R}(\mathbf{FO}_i)$ is very simple. Hence here we only show the case when $l_1 \le r \le l_2$.

The analytical model for computing $\mathbf{CO_R}(\mathbf{FO}_i)$ without considering the robot's geometry can be written as follows:

$$\theta_1 = \varphi \pm \arccos\left(\frac{r^2 + l_1^2 - s^2}{2l_1 r}\right) \tag{3}$$

$$\theta_2 = \pm \arccos\left(\frac{r^2 - l_1^2 - s^2}{2l_1}\right) \tag{4}$$

where $0 \le s \le l_2$. In order to avoid collision, the image, $\mathbf{CO_R}(\mathbf{FO}_i)$, has to be modified by taking the robot's geometry into consideration. The forbidden region in the C-space must be enlarged by the upper boundary θ_{2u} and the lower boundary θ_{2l} of the second joint as follows:

$$\theta_{2u} = \theta_2 + \arcsin\left(\frac{w}{s}\right) \tag{5}$$

Table 1. A sample of the image of a fundamental obstacle $\mathbf{MFO}_{24}$ = (24, 0).

$\mathbf{MFO}_{24}$ = (24, 0)						
k (24)	$\theta_{1\min}^{(k)}$	$\theta_{2\min}^{(k)}$	$\theta_{2l}^{(k)}(1)$	$\theta_{2l}^{(k)}(2)$	$\cdots$	$\theta_{2l}^{(k)}(t_v^{(k)})$
$t_v^{(k)}$	$\theta_{1\max}^{(k)}$	$\theta_{2\max}^{(k)}$	$\theta_{2u}^{(k)}(1)$	$\theta_{2u}^{(k)}(2)$	$\cdots$	$\theta_{2u}^{(k)}(t_v^{(k)})$

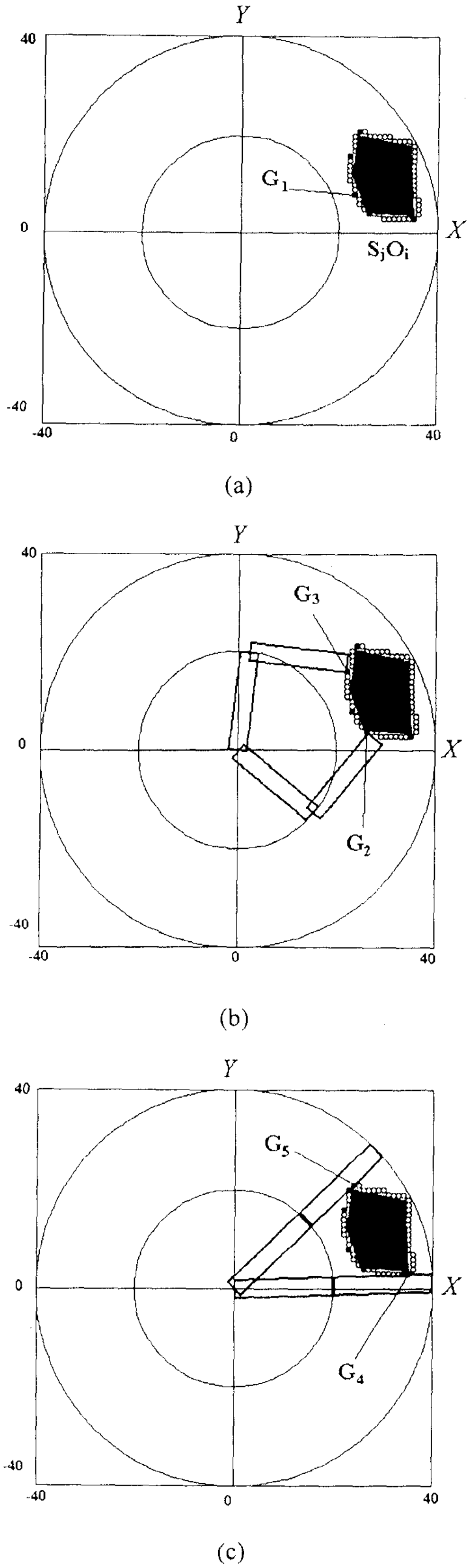


Fig. 2. Critical points of an obstacles

$$\theta_{2l} = \theta_2 - \arcsin\left(\frac{w}{s}\right) \tag{6}$$

where w is the width of the robot's arm. When s changes from 0 to l_2, the C-space obstacle is generated by formula (3)-(6). Fig. 1a shows fundamental obstacles $\mathbf{FO}_{ia}$, $\mathbf{FO}_{ib}$ and $\mathbf{FO}_{ic}$; Fig. 1b shows their corresponding images $\mathbf{CO_R}(\mathbf{FO}_{ia})$, $\mathbf{CO_R}(\mathbf{FO}_{ib})$ and $\mathbf{CO_R}(\mathbf{FO}_{ic})$ regarding the robot's geometry.

To compute complex C-space obstacles, we only need to save the images of $\mathbf{FO}_i$, which are located along the positive half of the horizontal axis, denoted by $\mathbf{MFO}_k$. In this paper, we choose forty $\mathbf{MFO}_k$ (k = 1, 2,..., 40). Based on $\mathbf{CO_R}(\mathbf{MFO}_k)$, all $\mathbf{CO_R}(\mathbf{FO}_i)$ can be computed by the use of r and φ in [7]. As an example, Table 1 gives a sample of $\mathbf{CO_R}(\mathbf{MFO}_k)$ saved in a database, where $\theta_{1\min}^{[k]}$, $\theta_{1\max}^{[k]}$, $\theta_{2\min}^{[k]}$ and $\theta_{2\max}^{[k]}$ are the minimal and maximal values of θ_1 and θ_2 for $\mathbf{CO_R}(\mathbf{MFO}_k)$.

3. Mapping Complex Obstacles by Critical Points

Since $\mathbf{FO}_i$ and $\mathbf{CO_R}(\mathbf{FO}_i)$ describe the key relationship between the W-space and the C-space, for a complex obstacle $\mathrm{S_j}\mathbf{O}_i$ in two dimensions, we can compute its C-space obstacle $\mathbf{CO_R}(\mathrm{S_j}\mathbf{O}_i)$ according to

$$\mathbf{CO_R}(\mathrm{S_j}\mathbf{O}_i) = \mathbf{CO_R}(\mathbf{FO}_1) \cup \cdots \cup \mathbf{CO_R}(\mathbf{FO}_k) \cup \cdots \tag{7}$$

where $\mathbf{FO}_k$ are the fundamental obstacles on borders of $\mathrm{S_j}\mathbf{O}_i$. Since the upper and lower boundaries of $\mathbf{CO_R}(\mathrm{S_j}\mathbf{O}_i)$, denoted by $\mathbf{CO_R}(\mathrm{S_j}\mathbf{O}_i)_{\mathrm{upper}}$ and $\mathbf{CO_R}(\mathrm{S_j}\mathbf{O}_i)_{\mathrm{lower}}$, consist of upper and lower boundaries of $\mathbf{CO_R}(\mathbf{FO}_i)$, respectively, the computation of $\mathbf{CO_R}(\mathrm{S_j}\mathbf{O}_i)$ determines the boundaries of all $\mathbf{CO_R}(\mathbf{FO}_i)$. A 2D obstacle $\mathrm{S_j}\mathbf{O}_i$ is shown in Fig. 2a and all $\mathbf{FO}_k$ related to $\mathrm{S_j}\mathbf{O}_i$ are shown as 'o'. All $\mathbf{CO_R}(\mathbf{FO}_k)$ should be computed together to form $\mathbf{CO_R}(\mathrm{S_j}\mathbf{O}_i)$. It can be noted that some of $\mathbf{CO_R}(\mathbf{FO}_k)$ completely or partially overlap with each other, and hence many irregular cells must be activated repeatedly by using the cell decomposition approach for superimposing $\mathbf{CO_R}(\mathrm{S_j}\mathbf{O}_i)$.

In our approach, $\mathbf{CO_R}(\mathrm{S_j}\mathbf{O}_i)$ is represented by their boundaries rather than their irregular cells, and we propose an algorithm for obstacle mapping using the critical points of an obstacle. The boundaries of $\mathbf{CO_R}(\mathrm{S_j}\mathbf{O}_i)$ for the joints θ_1 and θ_2 are formed when the robot touches the boundary of $\mathrm{S_j}\mathbf{O}_i$ from the exterior in each of the two cases [9]: 1. The robot links contact a vertex of $\mathrm{S_j}\mathbf{O}_i$; 2. The robot end-effector contacts an edge of $\mathrm{S_j}\mathbf{O}_i$. It has been reported that $\mathbf{CO_R}(\mathrm{S_j}\mathbf{O}_i)$ is most often formed whenever the robot arm contacts $\mathrm{S_j}\mathbf{O}_i$ [4]. Hence we select such $\mathbf{FO}_i$ from equation (7) that can be contacted by the robot links to improve mapping performance. According to these principles, we define the following $\mathbf{FO}_i$ as the critical points, shown as '•' in Fig. 2a. First, the fundamental obstacle $\mathbf{FO}_i$ with the minimum r, denoted by $\mathbf{G}_1$, is defined as a critical $\mathbf{FO}_i$, since it is the nearest fundamental obstacle to the original point, shown in Fig. 2a. Secondly, the fundamental obstacle $\mathbf{FO}_i$ with $\Theta_{1\min}$ and $\Theta_{1\max}$, denoted by $\mathbf{G}_2$ and $\mathbf{G}_3$ are defined as critical points shown in Fig. 2b. Finally, the fundamental obstacle $\mathbf{FO}_i$ with the minimum and maximum φ, denoted by $\mathbf{G}_4$ and $\mathbf{G}_5$, are also considered as critical points of $\mathbf{S}_j\mathbf{O}_i$, as shown in Fig. 2c. The critical points' images govern $\mathbf{CO_R}(\mathbf{S}_j\mathbf{O}_i)$, because: 1. The critical point $\mathbf{G}_1$ contributes the largest collision area in the C-space among all $\mathbf{FO}_i$; 2. The critical points $\mathbf{G}_2$ and $\mathbf{G}_3$, which determine the forbidden region $[\Theta_{1\min}, \Theta_{1\max}]$ for the joint θ_1, can be contacted by the robot links; 3. The critical points $\mathbf{G}_4$ and $\mathbf{G}_5$ also can be contacted by the robotic arm when the arm stretches up, as shown in Fig. 2c. On the assumption that the number of $\mathbf{FO}_i$ for modeling a 2D obstacle $\mathbf{S}_j\mathbf{O}_i$ is J, we propose the following *Algorithm* 1 to compute the critical points for $\mathbf{S}_j\mathbf{O}_i$.

Algorithm 1:
find_critical_points():
Step 1. $r_{\min} = \infty$, $\Theta_{1\min} = \infty$, $\Theta_{1\max} = -\infty$, $\varphi_{\min} = \infty$, and $\varphi_{\max} = -\infty$;
Step 2. <u>for</u> j = 1 <u>to</u> J <u>do</u>

Step 2.1 $r = \sqrt{x^2 + y^2}$;
<u>if</u> $r < r_{\min}$ <u>then</u> $r_{\min} = r$; $g_1 = j$ <u>end if</u>;
Step 2.2 $\varphi = \arctan\left(\dfrac{y}{x}\right)$;
<u>if</u> $\varphi < \varphi_{\min}$ <u>then</u> $\varphi_{\min} = \varphi$; $g_2 = j$ <u>end if</u>;
<u>if</u> $\varphi > \varphi_{\max}$ <u>then</u> $\varphi_{\max} = \varphi$; $g_3 = j$ <u>end if</u>;

Step 2.3 $n = \left\lfloor \dfrac{r}{\delta} \right\rfloor$;

$\theta_{1\min}^{(j)} = \theta_{1\min}^{[n]} + \varphi$;
$\theta_{1\max}^{(j)} = \theta_{1\max}^{[n]} + \varphi$;
<u>if</u> $\theta_{1\min}^{(j)} < \Theta_{1\min}$ <u>then</u> $\Theta_{1\min} = \theta_{1\min}^{(j)}$; $g_4 = j$ <u>end if</u>;
<u>if</u> $\theta_{1\max}^{(j)} > \Theta_{1\max}$ <u>then</u> $\Theta_{1\max} = \theta_{1\max}^{(j)}$; $g_5 = j$ <u>end if</u>;
<u>end for</u> j

where the symbol $\lfloor\ \rfloor$ takes the maximum integer that is smaller than the quotient, and g_1, g_2, g_3, g_4 and g_5 are the sequence numbers of the critical points $\mathbf{G}_1$, $\mathbf{G}_2$, $\mathbf{G}_3$, $\mathbf{G}_4$ and $\mathbf{G}_5$. The computational complexity of *Step* 1 in *Algorithm* 1 is $O(1)$. Since equations (1) and (2) can be performed by the finite number of fundamental operations K_0, the computational complexity of *Step* 2.1-2.3 are $O(1)$. Hence the complexity of *Algorithm* 1 is $O(1)+O(J)$. Since, obviously, J is much smaller than the total number of fundamental obstacles, the time complexity of *Algorithm* 1 can be expressed by $O(1)$.

For the critical points, their images can be obtained on the basis of the database [5] instead of by computing the robot's kinematics and geometry as well as the obstacles' geometry. In fact, determining the upper boundary of $\mathbf{CO_R}(\mathbf{S}_j\mathbf{O}_i)_{\text{upper}}$ serves to calculate the upper boundary of $\mathbf{CO_R}(\mathbf{G}_1)_{\text{upper}}$, $\mathbf{CO_R}(\mathbf{G}_3)_{\text{upper}}$ and $\mathbf{CO_R}(\mathbf{G}_5)_{\text{upper}}$, while determining $\mathbf{CO_R}(\mathbf{S}_j\mathbf{O}_i)_{\text{lower}}$ serves to compute the lower boundary of $\mathbf{CO_R}(\mathbf{G}_1)_{\text{lower}}$, $\mathbf{CO_R}(\mathbf{G}_2)_{\text{lower}}$ and $\mathbf{CO_R}(\mathbf{G}_4)_{\text{lower}}$. Therefore, $\mathbf{CO_R}(\mathbf{S}_j\mathbf{O}_i)_{\text{upper}}$ and $\mathbf{CO_R}(\mathbf{S}_j\mathbf{O}_i)_{\text{lower}}$ can be expressed as:

$$\mathbf{CO_R}(\mathbf{S}_j\mathbf{O}_i)_{\text{upper}} = \mathbf{CO_R}(\mathbf{G}_1)_{\text{upper}} \cup \mathbf{CO_R}(\mathbf{G}_3)_{\text{upper}} \cup \mathbf{CO_R}(\mathbf{G}_5)_{\text{upper}} \tag{8}$$

$$\mathbf{CO_R}(\mathbf{S}_j\mathbf{O}_i)_{\text{lower}} = \mathbf{CO_R}(\mathbf{G}_1)_{\text{lower}} \cup \mathbf{CO_R}(\mathbf{G}_2)_{\text{lower}} \cup \mathbf{CO_R}(\mathbf{G}_4)_{\text{lower}} \tag{9}$$

We propose the following algorithm to compute $\mathbf{CO_R}(\mathbf{S}_j\mathbf{O}_i)_{\text{upper}}$ and $\mathbf{CO_R}(\mathbf{S}_j\mathbf{O}_i)_{\text{lower}}$:

Algorithm 2:
generate_C_space_obstacle():
Step 1. $P_{\min} = \left\lceil \dfrac{\Theta_{1\min}}{\Delta\Theta_1} \right\rceil$;

$P_{\max} = \left\lfloor \dfrac{\Theta_{1\max}}{\Delta\Theta_1} \right\rfloor$;

$M = P_{\max} - P_{\min} + 1$;
Step 2. <u>for</u> $m = 1$ <u>to</u> M <u>do</u>
$\Theta_{2l}(m) = \infty$;
$\Theta_{2u}(m) = -\infty$;
<u>end for</u> m
Step 3. <u>for</u> $i = g_1, g_3, g_5$ <u>do</u>

Step 3.1 $\sigma_{\min} = \left\lceil \dfrac{\theta_{1\min}^{(i)}}{\Delta\Theta_1} \right\rceil$;

$\sigma_{\max} = \left\lfloor \dfrac{\theta_{1\max}^{(i)}}{\Delta\Theta_1} \right\rfloor$;

$s = \theta_{1\min}^{(i)} \bmod \Delta\Theta_1$;
$\omega = \sigma_{\max} - \sigma_{\min} + 1$;
$\tau = \sigma_{\min} - P_{\min}$;

Step 3.2 <u>for</u> $k = 1$ <u>to</u> ω <u>do</u>
$v_2 = (1-s)*\theta_{2u}(k) + s*\theta_{2u}(k+1)$;
<u>if</u> $v_2 > \Theta_{2u}(\tau+k)$ <u>then</u> $\Theta_{2u}(\tau+k) = v_2$ <u>end if</u>;
<u>end for</u> k;
<u>end for</u> i
Step 4. <u>for</u> $i = g_1, g_2, g_4$ <u>do</u>

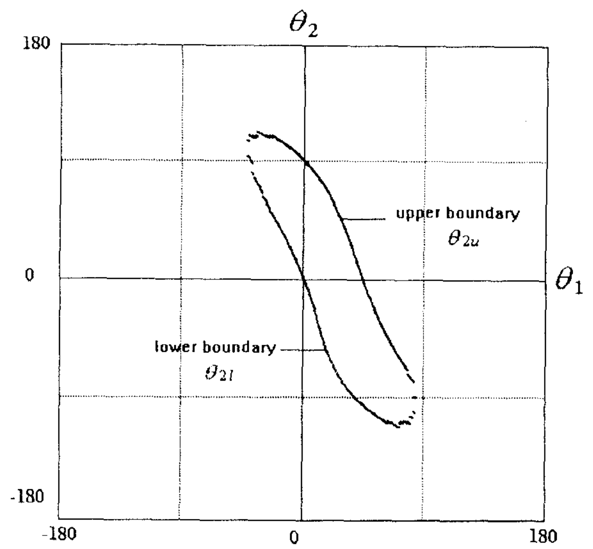


Fig. 3. Generation of C-space from critical points

Step 4.1 $\sigma_{min} = \left\lceil \frac{\theta_{1min}^{(i)}}{\Delta\Theta_1} \right\rceil$;

$\sigma_{max} = \left\lfloor \frac{\theta_{1max}^{(i)}}{\Delta\Theta_1} \right\rfloor$;

$s = \theta_{1min}^{(i)} \bmod \Delta\Theta_1$;

$\omega = \sigma_{max} - \sigma_{min} + 1$;

$\tau = \sigma_{min} - P_{min}$;

Step 4.2 <u>for</u> $k = 1$ <u>to</u> ω <u>do</u>

$v_2 = (1-s)*\theta_{2l}(k)+s*\theta_{2l}(k+1)$;

<u>if</u> $v_1 < \Theta_{2l}(\tau+k)$ <u>then</u> $\Theta_{2l}(\tau+k)=v_1$ <u>end</u> <u>if</u> ;

<u>end</u> <u>for</u> k ;

<u>end</u> <u>for</u> i

where the symbol $\lceil\ \rceil$ takes the minimum integer that is larger than the quotient, and the symbol *mod* takes the remainder. In all algorithms, the small letter θ is used to represent the image of a point obstacle; while the capital-letter Θ is used to represent the image of an obstacle except at the point obstacles. Therefore, $\mathbf{CO_R}(\mathbf{S_jO_i})_{upper}$ and $\mathbf{CO_R}(\mathbf{S_jO_i})_{lower}$ are represented by $\Theta_{2u}(m)$ and $\Theta_{2l}(m)$ ($m = 1, ..., M$), and $\Theta_{2u}(1)$ and $\Theta_{2l}(1)$ as well as $\Theta_{2u}(M)$ and $\Theta_{2l}(M)$ are the functions of Θ_{1min} as well as Θ_{1max}. The computational amount of *Step* 1 and *Step* 2 in *Algorithm* 2 is $O(1)$, and that of *Step* 3 and *Step* 4 is also $O(1)$ since the number of $\mathbf{CO_R}(\mathbf{MFO}_k)$ in Table 1 is smaller than a constant. Therefore, the total complexity of *Algorithm* 2 can be expressed by $O(1)$. Its computation can be competed in about 5ms using a PC386 computer. The mapping result using the critical points can be seen in Fig. 3.

4. Sensor Based Obstacle Modeling in C-Space for Motion Planning

One of the most important steps for motion planning in an uncertain world is obstacle modeling based on sonar data. Using the mapping method given in the last section, we present an approach to C-space obstacle modeling based on information obtained from "distance" sensors are assumed to be attached to the second link of the robot. The approach will be described through an example as shown in Fig. 4a-4h.

The aim of motion planning is to find a collision free path from a start position to a goal position. Building a C-space using the critical points of the obstacles in the W-space is fast enough for a planner to give the path in real time. In an unknown environment, however, we cannot get the entire knowledge of the environment in advance. Hence it is very important to acquire information on obstacles from sensors.

In our study, "distance" sensors are distributed regularly on both sides of the second link of the planar robot, as shown in Fig. 4a. Each sensor can return the vertical distance (straight-up to the second link) between it and the obstacle. Thus the boundary points of the obstacle in a local region can be found according to distance data. They are approximated by $\mathbf{FO}_i$ and can be mapped into the C-space according to the precomputed result of $\mathbf{FO}_i$. These $\mathbf{FO}_i$ are shown by 'o' in Fig. 4a and specify all the possible collisions in the local region. In other word, if the robot does not contact these $\mathbf{FO}_i$, it will not collide with any obstacles when it moves in small steps. Here we consider only those $\mathbf{FO}_i$ situated on one side of the robot's second link when we try to let the link move toward them.

Once distance information on the local region is acquired, a C-space obstacle can be formed according to the critical points of all $\mathbf{FO}_i$. Through computation using algorithm 1, the critical points can be acquired and they are represented by '•' in Fig. 4a (In some cases, the critical points may number be 4, 3, or even 2). The image of the obstacle in the local region in the C-space is displayed in Fig. 4b. Based on the C-space modeling, then, the planner generates a local path to the goal point for robot motion in small steps. If the robot arrives at the goal point, the planning is finished; Alternatively, if it does not reach the goal point, it should rebuild all $\mathbf{FO}_i$ and critical points according to updated information from the sensors, and once it is found that the critical points are different from the last ones, the planner should re-plan a new route starting from this point to the goal point. It should be remarked that only the current $\mathbf{FO}_i$, computed from the new information, are to be considered. We do not accumulate historical $\mathbf{FO}_i$ since

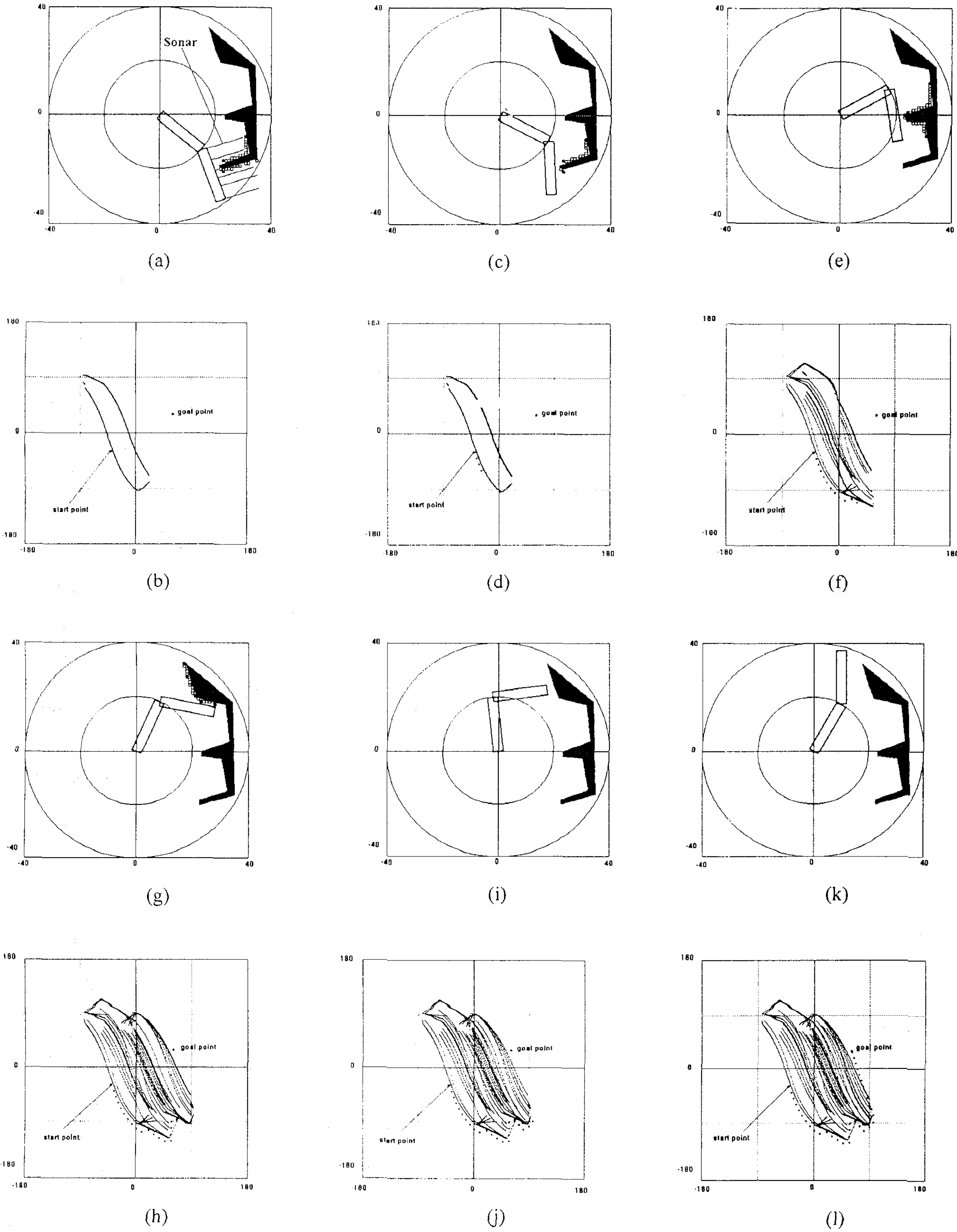


Fig. 4. An example of motion planning for a planar robot

these $\mathbf{FO}_i$ would be far from robot and they do not affect the robot's current planning.

In our example, the robot starts from the start point and finds that there is no difference between the current critical points and the old ones when it moves to the second and the third configurations. When it gets to the fourth position (as shown in Fig. 4c and 4d), it finds a different situation. Then, the C-space is rebuilt according to the new critical points and the planner will generate a new path. Thus the work continues until the goal is reached. Fig. 4a, 4c, 4e, 4g, 4i and 4k describe some consequent configurations and the critical points' positions, while Fig. 4b, 4d, 4f, 4h, 4j and 4l give the configuration spaces corresponding to Fig. 4a, 4c, 4e, 4g, 4i and 4k, respectively. The thick solid line in each one of these C-space figures represents the current C-space obstacle, while the other lines describe the old C-space obstacles. In Fig. 4i and 4k, no critical points can be found since the robot has passed the obstacle to the goal point. In the other hand in Fig. 4j and 4l, no thick solid line is found, which means that the current local C-space is an obstacle free space.

The approach above can be genealised by means of following algorithm.

Algorithm 3:

Step 1. old_$g_1 = \infty$, old_$g_2 = \infty$, old_$g_3 = \infty$, old_$g_4 = \infty$, and old_$g_5 = \infty$; *current* = *start*; *path*[0] = *start*; $i = 0$;

Step 2. while *current* $\neq$ *goal* do

Step 2.1 *find_fundamental_obstacles*();

Step 2.2 g_1, g_2, g_3, g_4, g_5 = *find_critical_point*();

Step 2.3 if old_ $g_1 \neq g_1$ or old_ $g_2 \neq g_2$ or old_ $g_3 \neq g_3$ or old_ $g_4 \neq g_4$ or old_ $g_5 \neq g_5$
then
generate_C_space_obstacle();
g_1 = old_ g_1; g_2 = old_ g_2; g_3 = old_ g_3;
g_4 = old_ g_4; g_5 = old_ g_5;
end if

Step 2.4 *next* = *motion-planning* (*current*, *goal*);
current = *next*;

Step 2.5 *path*[*i*] = *next*; *i*=*i*+1;

end while

where *start* and *goal* are separately the start position and goal position of the robot, g_1, g_2, g_3, g_4 and g_5 have the same meaning as those in *Algorithm* 1 and 2, while old_g_1, old_g_2, old_g_3, old_g_4 and old_g_5 are used to keep old values of them, respectively. The aim of the function *find_fundamental_obstacles* is to acquire all $\mathbf{FO}_i$ approximately standing for real obstacles according to distance data, and that of the function *motion-planning* is to generate the next position to which the robot should move according to a certain method, respectively. The result of the motion planning is to be recorded in *path*.

5. Extension to 3D Motion Planning

Let us consider motion planning of a 3D robot like a PUMA 560, whose first three joint angles are defined as θ_0, θ_1, and θ_2 from the base, respectively. A global or local 3D C-space must be built. What we should consider is the first joint's mapping. K. Sun and V. Lumelsky addresses the problem of collision-free motion planning of a 3D robot manipulator with sliding joints in an unknown environment in [10]. In their paper, sensors are installed on the arm to detect a contact with an obstacle. However, this approach is not suitable for a robot PUMA 560 with revolute joints. In our simulation, we furnish "distance" sensors on all four sides of the third link of the revolute robot. Thus, they can receive not only information considering motion of the second and third link, just like the case of a planar robot discussed above, but also information about the first link's motion. Some boundary points in a 3D obstacle can be found and every point must be selected on a proper θ_0 plane and $\mathbf{FO}_i$ near them are to be mapped to generate a 2D C-space. We can also use the critical points in every 2D space to form a 2D C-space. That is , a partial 3D C-space can be formed by generating several 2D C-spaces. One of our simulations on robot PUMA 560 can be seen from Fig. 5.

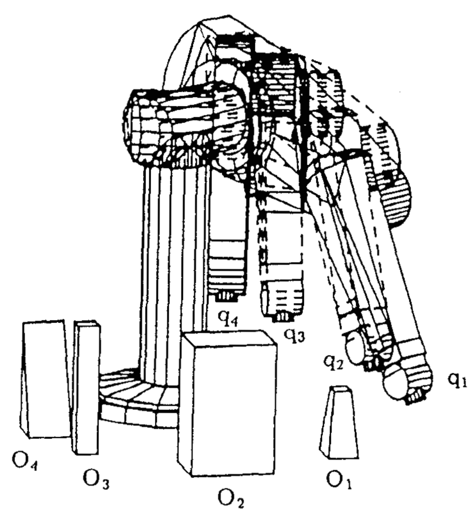


Fig. 5. Motion planning for a PUMA 560 robot

Using the above method, the robot can sense the environment information once it starts to move. When the robot moves to the next position according to the last planning, it should decide whether the critical points in each θ_0 plane are changed. This is the same as for a 2D space. If they are changed, the robot regenerates the C-space obstacle and replans a path; otherwise it continues to the next position and again decides if the C-space is changed. No more than the images of 40 fundamental obstacles are stored even in 3D motion case.

6. Conclusions

In this paper, we present an approximate approach to fast mapping obstacle from the W-space into the C-space based on selecting critical fundamental obstacles, and we analyze its computational complexity as $O(1)$. Usually, the approximation adopted provides sufficient information for the manipulator to plan a realistic collision-free path in the unknown environment. We discuss sensor-based obstacle modeling in the C-space for a planar manipulator and extend it to 3D operation. This C-space obstacle modeling makes path searching quicker and simpler for practical use. In our further research, we will implement this approach on a real robot system, and especially we will study an effect of sensors on planning performance.

Acknowledgments

This work is funded by National Science Foundation of China under grant 69585004 and by Award Foundation of Chinese Education Committee for Outstanding Young Teacher.

References

[1] B. Faverjon, "Obstacle avoidance using an octree in the configuration space of a manipulator", *Proceedings IEEE International Conference on Robotics*, pp. 504-512, 1984

[2] T. Lozano-Perez, "A simple motion - planning algorithm for general robot manipulators", *IEEE Journal of Robotics and Automation*, RA-3, pp. 224-238, 1987

[3] C. W. Warren, J. C. Danos, and B. W. Mooring, "An approach to manipulator path planning", *International Journal of Robotics Research*, vol.8, no.5, pp. 87-95, 1989

[4] V. Lumelsky, "Effect of kinematics on motion planning for planar robot arms moving amidst unknown obstacles", *IEEE Journal of Robotics and Automation*, RA-3, pp. 207-223, 1987

[5] W. Li, "Automatic determination of collision-free paths for general robots", *Robotersysteme*, vol.6, pp. 218-244, 1990

[6] W. Li, "Fast mapping obstacles in the configuration space", *Robotersysteme*, vol.7, pp. 148-154, 1991

[7] W. Li and B. Zhang, "Solving the robotic 'pick-and-place' pathfind problem", *ASME Journal, Manufacturing Review*, vol.6, pp. 114-129, 1993

[8] V. Lumelsky, A. A. Stepanov, "Path planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape", *Algorithmica*, vol.2, pp. 403-430, 1990

[9] Brady M et al., *Robot Motion: Planning and Control.* MIT-Press, Cambridge, pp.13, 1982

[10] K. Sun, V. Lumelsky, "Path planning among unknown obstacles: the case of a three-dimensional Cartesian arm", *IEEE Transactions on Robotics and Automation*, vol.8, no.6, pp. 776-786, 1992