Proceedings of the
American Control Conference
Seattle, Washington • June 1995

AN APPROACH TO ON-LINE PATH PLANNING FOR MULTIPLE ROBOTS

Wei LI
National Laboratory for Intelligent Technology and Systems
Department of Computer Science, Tsinghua University
Beijing, 100084, China
Fax: (861) 256-2768
dcsjpf@tsinghua.edu.cn

ABSTRACT

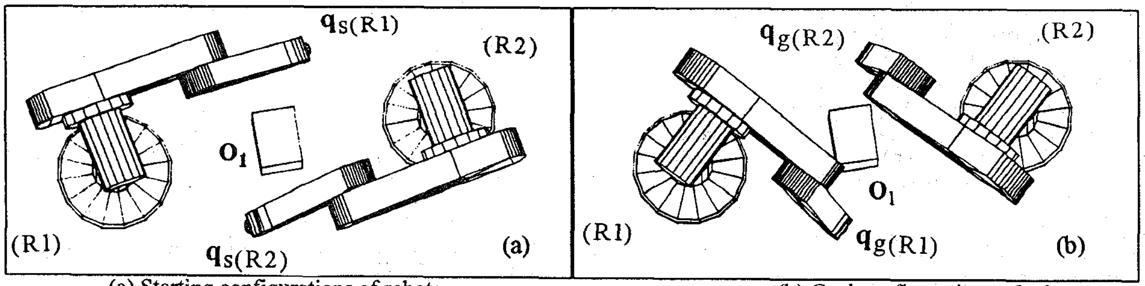
This paper presents an approach to on-line path planning for multiple robots in configuration space. On the basis of defining some points in workspace as fundamental obstacles, we develop an efficient algorithm for mapping a complex Cartesian obstacle by selecting obstacle's critical points. Its computational time for mapping a two-dimensional obstacle is approximately 5ms with a 33Mhz 80486 CPU. To plan a collision-free path for a master robot, we use slice configuration space obstacles for building a free subspace. To plan a collision-free path for a slave robot, swept volumes of the master robot are taken as additional moving obstacles into consideration. Finally, we report graphical simulation results with respect to two PUMA robots sharing a common workspace.

1. INTRODUCTION

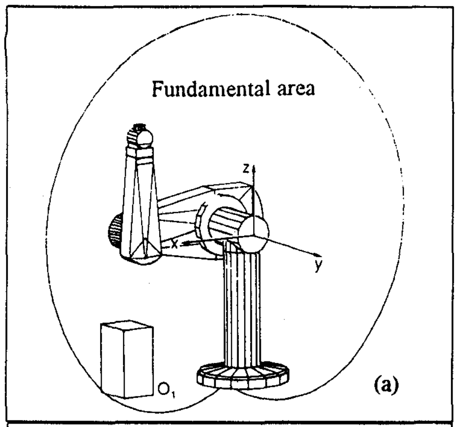
In recent years, a great deal of research has been devoted to the problems of using multiple robots, such as coordinated motion, control architecture, and collision avoidance. This paper presents a practical approach to on-line path planning for multiple robots in configuration space (C-space), which is further extension to our works [1]-[3]. Its basis is to use the analytical models in [3] to precompute C-space obstacles of some points in workspace (W-space), defined as fundamental obstacles. A key step of mapping a complex Cartesian obstacle is superimposing the images of point obstacles. Usually, a cell decomposition approach is used to represent point obstacles' images, and their superposition is performed by activating all irregular cells [4]. It is noticed, however, that many irregular cells need to be activated repeatedly more times since the majority of the point obstacles' images for mapping a same Cartesian obstacle completely or partially overlap with each other (see Section 3). To improve mapping performance, the idea of superimposing point obstacles' images used here is to compute their contours by the critical points of the Cartesian obstacle. Its computational cost for mapping a two-dimensional (2D) obstacle is approximately 5ms with a 33 Mhz 80486 CPU. To plan a collision-free path for a master robot, slice C-space obstacles are computed for building a free subspace that has more computation accuracy than one represented by slice projections [5]. Since this approach, including mapping and searching, is suitable for real-time motion planning, swept volumes, produced during master robot motion, can be considered as additional moving obstacles in planning paths for a slave robot. To demonstrate the effectiveness of the approach, we report graphical simulation results with respect to two PUMA robots sharing a common W-space, as shown in Fig. 1.

2. REVIEW OF PREVIOUS WORK

Methods for computing a C-space can be roughly divided into two groups. One of the most widely used methods is to decompose the C-space into regular and irregular cells [5]-[7]. However, these C-space algorithms are not efficient enough to be practical in real-time path planning for multiple robots because large amounts of computational time are needed to deal with robot's kinematics and geometry as well as the obstacles' geometry before searching for a path, and their computational cost grows exponentially with the number of degrees of freedom. Besides, the efficiency of path searching using cell decomposition approach depends largely upon the size of cells generated. If the decomposed cells are large, the searching process is fast but path searching may fail due to the loss of accuracy (e.g., Lozano's slice projections [5]); if the



(a) Starting configurations of robots (b) Goal configurations of robots
Fig. 1. Two PUMA robots sharing a common W-space



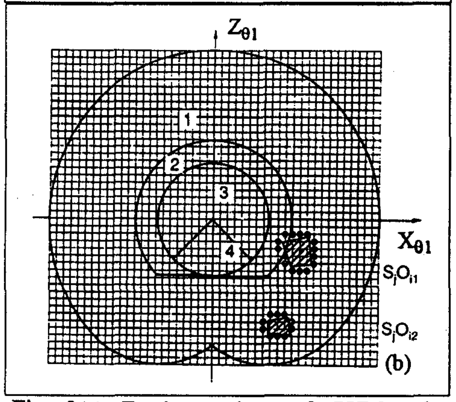


Fig. 2(a). Fundamental area of a PUMA robot
(b). Fundamental obstacles of a PUMA robot

decomposed cells are small enough, collision-free paths can be found, but the search process requires more execution time. Another method for obstacle mapping is to compute collision boundaries between a robot and obstacles in terms of the robot's kinematics on the assumption that the robotic arms and obstacles are all polyhedra. For instance, Lumelsky in [8] has studied different kinematic configurations corresponding to robotic arms with revolute or sliding joints, and Ge and McCarthy have presented the algebraic equations of C-space obstacles [9]. Since solving these algebraic equations is a time consuming work before path searching, this method is usually used for a single robot system.

Local approach is also frequently used for path planning [10]. Its principle is to hypothesize a path and test it for potential collisions. Its advantage is suitable for multiple robot systems. However, this approach only provides little information on how a path may be modified to avoid these or other collisions. To overcome this deficiency, powerful heuristics has been used to guide the search in local regions with the help of the C-space [11]. However, this approach requires considerable computational time for collision detection in terms of robot's kinematics and geometry as well as obstacles' geometry.

3. MAPPING MORE COMPLEX OBSTACLES BY CRITICAL POINTS

Before we discuss the algorithm for obstacle mapping, we briefly present the concepts of fundamental area and fundamental obstacles, as introduced in [1]. Since most robot's W-spaces are symmetric, they can be formed by moving an area round symmetric axes of the W-spaces. Such an area is known as a fundamental area of a robot. A grid is used to discretize the fundamental area. Intersection points of verticals and horizons on the grid are defined as fundamental obstacles $\mathbf{FO}_i = (x_{\theta_1}, z_{\theta_1})$ shown in Fig. 2 [3], where θ_1 represents the angle between the fundamental area and the symmetric axis of the W-space. Since $\mathbf{FO}_i$ are independent of a real obstacle in a dynamic environment, their C-space obstacles, denoted by $\mathbf{CO_R}(\mathbf{FO}_i)$, are precomputed by the analytical models in [3]. For computing complex C-space obstacles, we only need to

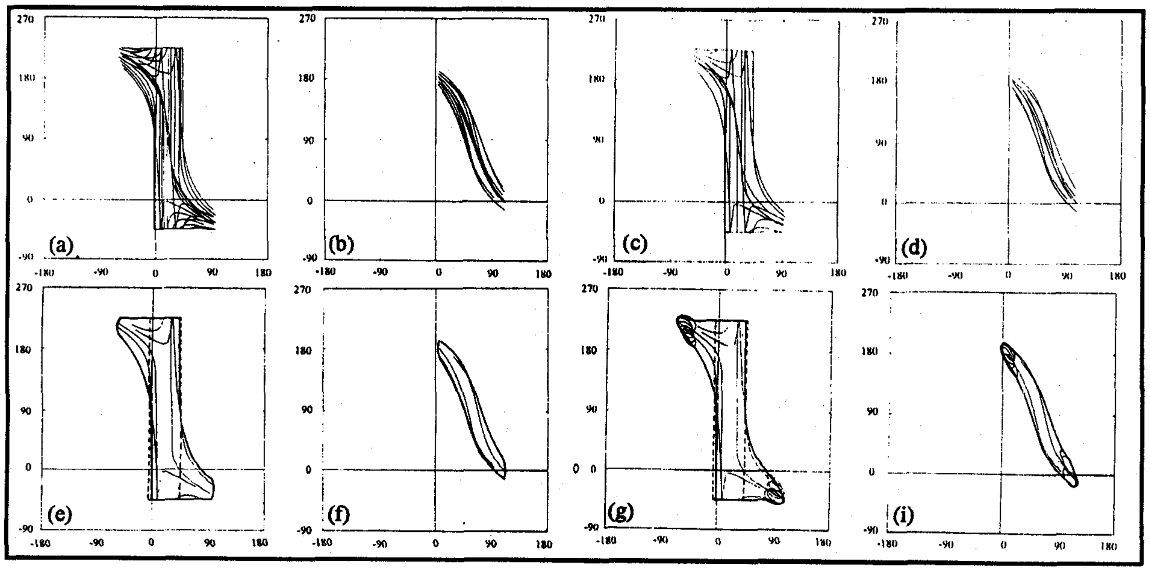
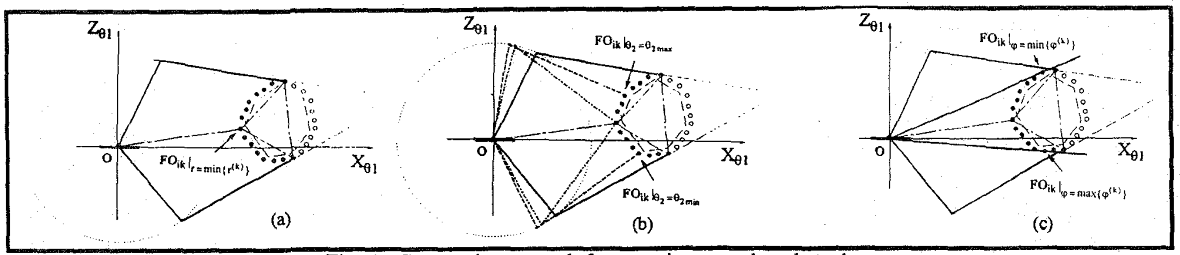


Fig. 3. Mapping $\mathbf{S_jO_{i1}}$ and $\mathbf{S_jO_{i2}}$ by their critical fundamental obstacles



$Z_{\theta 1}$
O
$X_{\theta 1}$
$FO_{ik}|_{r=\min\{r^{(k)}\}}$
(a)
$FO_{ik}|_{\theta_2=\theta_{2\max}}$
$FO_{ik}|_{\theta_2=\theta_{2\min}}$
(b)
$FO_{ik}|_{\varphi=\min\{\varphi^{(k)}\}}$
$FO_{ik}|_{\varphi=\max\{\varphi^{(k)}\}}$
(c)

Fig. 4. Geometric approach for mapping complex obstacles

save the images of $\mathbf{FO}_i$, that are located along the positive half of the horizontal axis on the fundamental area, since on their basis all $\mathbf{CO_R}(\mathbf{FO}_i)$ can be computed by the use of the distance, r, between $\mathbf{FO}_i$ and the original point, and the angle of $\mathbf{FO}_i$, φ [2].

In order to map a three-dimensional (3D) obstacle $\mathbf{O}_i$ with complex geometry, $\mathbf{O}_i$ is processed as follows [2]: 1. Decompose $\mathbf{O}_i$ into a finite set of 2D obstacles $\mathbf{S}_j\mathbf{O}_i$ by moving the fundamental area in a collision interval $[\theta_{1\min}, \theta_{1\max}]$ with a step length $\Delta\theta_1$; 2. Compute all $\mathbf{FO}_i$ on 2D obstacles' borders. Fig. 2(b) shows two 2D obstacles $\mathbf{S}_j\mathbf{O}_{i1}$ and $\mathbf{S}_j\mathbf{O}_{i2}$ as well as their $\mathbf{FO}_{ik}$ on borders. Since $\mathbf{FO}_i$ and $\mathbf{CO_R}(\mathbf{FO}_i)$ describe the key relationship between the W-space and the C-space, the C-space obstacle $\mathbf{CO_R}(\mathbf{S}_j\mathbf{O}_i)$ can be computed by:

$$\mathbf{CO_R}(\mathbf{S}_j\mathbf{O}_i) = \mathbf{CO_R}(\mathbf{FO}_{i1}) \cup \cdots \cup \mathbf{CO_R}(\mathbf{FO}_{ik}) \cup \cdots \quad (1)$$

Since upper and lower boundaries of $\mathbf{CO_R}(\mathbf{S}_j\mathbf{O}_i)$, $\mathbf{CO_R}(\mathbf{S}_j\mathbf{O}_i)_{\text{upper}}$ and $\mathbf{CO_R}(\mathbf{S}_j\mathbf{O}_i)_{\text{lower}}$, consist of upper and lower boundaries of $\mathbf{CO_R}(\mathbf{FO}_{ik})$, respectively, computing $\mathbf{CO_R}(\mathbf{S}_j\mathbf{O}_i)$ is determining boundaries of all $\mathbf{CO_R}(\mathbf{FO}_{ik})$ that are functions of the joints θ_2 and θ_3.

All $\mathbf{CO_R}(\mathbf{FO}_{ik})$ related to $\mathbf{S}_j\mathbf{O}_{i1}$ and $\mathbf{S}_j\mathbf{O}_{i2}$ are plotted, respectively, in Fig. 3(a-b). It can be observed that some of $\mathbf{CO_R}(\mathbf{FO}_{ik})$ completely or partially overlap with others, so many irregular cells must be activated repeatedly more times using cell decomposition approach for superimposing $\mathbf{CO_R}(\mathbf{FO}_{ik})$. Fig. 4 shows some $\mathbf{FO}_{ik}$ represented by the o points. After such $\mathbf{FO}_{ik}$ are removed from Eq. (1), $\mathbf{CO_R}(\mathbf{S}_j\mathbf{O}_i)$ keeps unchanged (see $\mathbf{CO_R}(\mathbf{S}_j\mathbf{O}_{i1})$ and $\mathbf{CO_R}(\mathbf{S}_j\mathbf{O}_{i2})$ in Fig. 3(c-d)). The reason is that their images completely overlap with the images of $\mathbf{FO}_{ik}$ represented by the • points in Fig. 4.

The boundaries of $\mathbf{CO_R}(\mathbf{S}_j\mathbf{O}_i)$ for the joints θ_2 and θ_3 are formed when the robot touches the boundary of $\mathbf{S}_j\mathbf{O}_i$ from the exterior in each of both cases [12]: 1. The robot arms contact a vertex of $\mathbf{S}_j\mathbf{O}_i$; 2. The robot end-effector contacts an edge of $\mathbf{S}_j\mathbf{O}_i$. In [2], we report that $\mathbf{CO_R}(\mathbf{S}_j\mathbf{O}_i)$ is most mainly formed when the robot arms contact $\mathbf{S}_j\mathbf{O}_i$, so we select some $\mathbf{FO}_i$ from those $\mathbf{FO}_{ik}$ that can be contacted by the robot arms to improve mapping performance. Such $\mathbf{FO}_i$ are known as critical $\mathbf{FO}_i$. Firstly, the nearest $\mathbf{FO}_{ik}$ to the original point, denoted by $\mathbf{FO}_{ik}|_{r=\min\{r^{(k)}\}}$, is defined as a critical $\mathbf{FO}_{ik}$ shown in Fig. 4(a). Secondly, $\mathbf{FO}_{ik}$ with $\theta_{2\min}$ and $\theta_{2\max}$, denoted by $\mathbf{FO}_{ik}|_{\theta_2=\theta_{2\min}}$ and $\mathbf{FO}_{ik}|_{\theta_2=\theta_{2\max}}$, are defined as critical $\mathbf{FO}_{ik}$ shown in Fig. 4(b). Finally, $\mathbf{FO}_{ik}$ with the maximum and minimum φ, denoted by $\mathbf{FO}_{ik}|_{\varphi=\min\{\varphi^{(k)}\}}$ and $\mathbf{FO}_{ik}|_{\varphi=\max\{\varphi^{(k)}\}}$, are also considered as critical $\mathbf{FO}_{ik}$ of $\mathbf{S}_j\mathbf{O}_i$, as shown in Fig. 4(c). The images of the critical $\mathbf{FO}_{ik}$ govern $\mathbf{CO_R}(\mathbf{S}_j\mathbf{O}_i)$, because: 1. $\mathbf{FO}_{ik}|_{r=\min\{r^{(k)}\}}$ contributes the largest collision area in the C-space among all $\mathbf{FO}_{ik}$; 2. $\mathbf{FO}_{ik}|_{\theta_2=\theta_{2\min}}$ and $\mathbf{FO}_{ik}|_{\theta_2=\theta_{2\max}}$ determine the forbidden region $[\theta_{2\min}, \theta_{2\max}]$ for the joint θ_2 and can be contacted by the robot arms; 3. $\mathbf{FO}_{ik}|_{\varphi=\min\{\varphi^{(k)}\}}$ and $\mathbf{FO}_{ik}|_{\varphi=\max\{\varphi^{(k)}\}}$ can be contacted by the robotic arms when the arms stretch up, as shown in Fig. 4(c).

In fact, determining $\mathbf{CO_R}(\mathbf{S}_j\mathbf{O}_i)_{\text{upper}}$ is computing the upper boundary of $\mathbf{CO_R}(\mathbf{FO}_i)_{\text{upper}}|_{r=\min\{r^{(k)}\}}$, $\mathbf{CO_R}(\mathbf{FO}_i)_{\text{upper}}|_{\theta_2=\theta_{2\max}}$, and $\mathbf{CO_R}(\mathbf{FO}_i)_{\text{upper}}|_{\varphi=\max\{\varphi^{(k)}\}}$, while determining $\mathbf{CO_R}(\mathbf{S}_j\mathbf{O}_i)_{\text{lower}}$ is computing the lower boundary of $\mathbf{CO_R}(\mathbf{FO}_i)_{\text{upper}}|_{r=\min\{r^{(k)}\}}$, $\mathbf{CO_R}(\mathbf{FO}_i)_{\text{upper}}|_{\theta_2=\theta_{2\min}}$, and $\mathbf{CO_R}(\mathbf{FO}_i)_{\text{upper}}|_{\varphi=\min\{\varphi^{(k)}\}}$. 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:

$$\begin{aligned} CO_R(S_jO_i)_{upper} = {} & CO_R(FO_i)_{upper}\big|_{r=\min\{r^{(k)}\}} \\ & \cup CO_R(FO_i)_{upper}\big|_{\theta_2=\theta_{2max}} \\ & \cup CO_R(FO_i)_{upper}\big|_{\varphi=\max\{\varphi^{(k)}\}} \end{aligned} \quad (2)$$

$$\begin{aligned} CO_R(S_jO_i)_{lower} = {} & CO_R(FO_i)_{lower}\big|_{r=\min\{r^{(k)}\}} \\ & \cup CO_R(FO_i)_{lower}\big|_{\theta_2=\theta_{2min}} \\ & \cup CO_R(FO_i)_{lower}\big|_{\varphi=\min\{\varphi^{(k)}\}} \end{aligned} \quad (3)$$

Since r and φ of each FO_{ik} can be computed [3], and $\theta^{(k)}_{2\min}$ and $\theta^{(k)}_{2\max}$ of each $CO_R(FO_{ik})$ can be obtained from the database of $CO_R(FO_{ik})$, the critical FO_i can be easily found among all FO_{ik}. For the critical FO_i, their image $CO_R(FO_i)$ can be obtained on the basis of the database [3] instead of computing robot's kinematics and geometry as well as obstacles' geometry. Since the number of the criticalFO_i in Eqs. (2)-(3) are constant, the computational time for mapping S_jO_i is about constant, e.g., the time for mapping S_jO_{i1} and S_jO_{i2} is approximately 5ms with a 33 Mhz 80486 CPU. To close the left-upper and right-lower parts of $CO_R(S_jO_i)$ without considering a robot's gripper and its payload, we connect the limits' points of its left-upper and right-lower parts, as shown by the thick curves in Fig. 3(e-f). If the robot's gripper and its payload are considered, these parts of $CO_R(S_jO_i)$ are closed by the maximum and minimum values ofθ_2 of payload C-space obstacles, as shown in Fig. 3(g-i).

For mapping the 3D obstacle O_1 in Fig. 2(a), O_1 is decomposed by the fundamental area into an infinite set of S_jO_i, and, at a given θ_1, mapping S_jO_i can be quickly performed by the algorithm discussed above. By using the union operation again:

$$CO_R(O_i) = CO_R(S_jO_1) \cup \cdots \cup CO_R(S_jO_i) \cup \cdots \quad (4)$$

O_1 is mapped into the C-space and its computational time is approximately 35ms. It should be noticed that obstacle mapping by critical FO_i usually is "exact" enough for robot transfer motion since, first, fundamental obstacles that describe S_jO_i keep certain distances to the border of S_jO_i, and second, a path to be generated in practice will not be too close to the boundaries of C-space obstacles.

4. MOTION PLANNING FOR A MASTER ROBOT BY SLICE CONFIGURATION OBSTACLES

The conditions for building a free space are to avoid dealing with unimportant parts of the C-space and to keep the connectivity between starting and goal configurations. For building such a free subspace, slice C-space obstacles on a cross-section plane, which is determined by starting and goal configurations, is computed [3]. The problem is that an infinite number of cross-section planes can be built through the starting configuration and the goal configuration. In order to find a short collision-free path, a cross-section plane should be chosen so that its slice C-space obstacles have minimal area. Here, a cross-section plane for the PUMA 560 robot:

$$a\theta_1 + b\theta_2 + c\theta_3 + d = 0 \quad (5)$$

is chosen to be parallel to the axis θ_1 because the joints θ_2 and θ_3 are coupled. The parameters (a, b, c, d) in Eq. (5) can be determined [3]:

$$\begin{cases} a = 0 \\ b = \theta_{3s} - \theta_{3g} \\ c = \theta_{2g} - \theta_{2s} \\ d = -a\theta_{1s} - b\theta_{2s} - c\theta_{3s} \end{cases} \quad (6)$$

where θ_{1s}, θ_{2s}, θ_{3s} and θ_{1g}, θ_{2g}, θ_{3g} are the position vectors of the starting configuration and the goal configuration, respectively.

Using slice C-space obstacles, a free subspace can be represented by a graph. The shortest path in the graph can be searched by the well known algorithm A*. To plan robot orientation movements, a linear interpolation is used [3]. Fig. 5 shows a slice C-space obstacle related to the starting and goal configurations $\{q_{s(R1)}, q_{g(R1)}\}$ of the robot R_1 (left) in Fig. 7 that is defined as a master robot. Fig. 7 shows its motion on the path in Fig. 5 from $q_{s(R1)}$ to $q_{g(R1)}$.

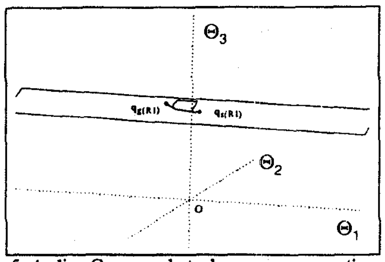


Fig. 5. A slice C-space obstacle on a cross-section plane

5. COMPUTING SWEPT VOLUMES FOR A SLAVE ROBOT

The difference between planning collision-free paths for a slave robot and the master robot is that the slave robot not only avoid collisions with obstacles in the W-space but also the master robot. Since the proposed approach is fast enough to plan paths in real-time, swept volumes, produced during master robot motion, can be considered as additional moving obstacles for the slave robot. Since a collision-free path for the master robot consists of a set of safe configurations $\{q_1,\ldots,q_i,\ldots,q_n\}$, the swept volume, V_i, of the master robot over the range

$[q_i,\ q_{i+1}]$ can be approximately computed in terms of robot's kinematics. Since $\mathbf{V}_i$ can be expressed as:

$$\mathbf{V}_i = \bigcup_k \mathbf{V}_i^k \tag{7}$$

where $\mathbf{V}_i^k$ is the swept volume for link *k* and is computed as follows:

1. Determine the original positions p_i^k and p_{i+1}^k of the coordinate frame of link *k* in configurations q_i and q_{i+1};
2. Compute swept area S_i^k of link *k* over range $[q_i,\ q_{i+1}]$ by connecting p_i^k, p_{i+1}^k, p_{i+1}^{k+1}, and p_i^{k+1} based on the link length L^k, as shown in Fig. 6;
3. Obtain swept volume V_i^k for link *k* by expanding S_i^k according to the geometry of robot's arms.

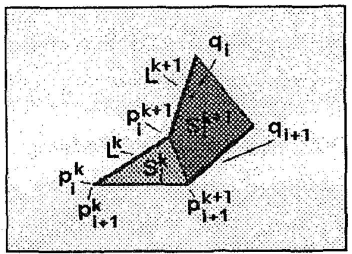


Fig. 6. Approximate computing swept volumes

Fig. 1 shows, when the master robot moves from its starting configuration $q_{s(R1)}$ to its goal configuration $q_{g(R1)}$, there exist potential collisions with the slave robot. According to the swept volumes V_i of the master robot, the slave robot R_2 (right) first moves away from the collision zone caused by master robot motion, as shown in Fig. 7(a). When the master robot reaches its goal configuration $q_{g(R1)}$ shown in Fig. 7(b-d), the slave robot moves to its goal configuration $q_{g(R2)}$, as shown in Fig. 7(e-f).

6. CONCLUSIONS

This paper presents an approach to real-time path planning for multiple robots. By selecting critical fundamental obstacles, we can be quickly computed the approximation of the C-space obstacle. Usually, such an approximation provides sufficient information to plan a collision-free path for a master robot. A free subspace constructed by slice C-space obstacles makes path searching quicker and simpler. To plan slave robot motion, swept volumes of the master robot are taken as additional moving obstacles into consideration. The computational time for motion planning of the given example, including mapping and searching, is approximately 50ms for the master robot and 180ms for the slave robot with a 33Mhz 80486 CPU. However, motion planning for the case of a common task like carrying an object does not reported in this paper. This problem will be dealt with in our further works.

REFERENCES

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

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

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

[4] M. S. Branicky and W. S. Newmann, "Rapid computation of configuration space obstacles", Proc. IEEE Int. Conf. Robotics and Automation, pp.304-310, 1990

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

[6] B. Faverjon, "Obstacle avoidance using an octree in the configuration space of a manipulator", Proc. IEEE Int. Conf. Robotics, pp.504-512, 1984.

[7] C. W. Waren, J. C. Danos, and B. W. Mooring, "An approach to manipulator path planning", Int. J. Robotics Res., vol.8, no.5, pp.87-95, 1989

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

[9] Q. Ge and J. M. McCarthy, "An algebraic formulation of configuration-space obstacles for spatial robots", Proc. IEEE Int. Conf. Robotics and Automation, pp.1542-1547, 1990.

[10] E. Freund and H. Hoyer, "A method for automatic collision avoidance for robots", Robotersysteme, vol.1, pp.67-73, 1990

[11] K. Kondo, "Motion planning with six degrees of freedom by multistrategic bidirecional heuristic free-space enumeration", IEEE Trans. Robotics and Automation, RA-7, pp.267-277, 1991

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

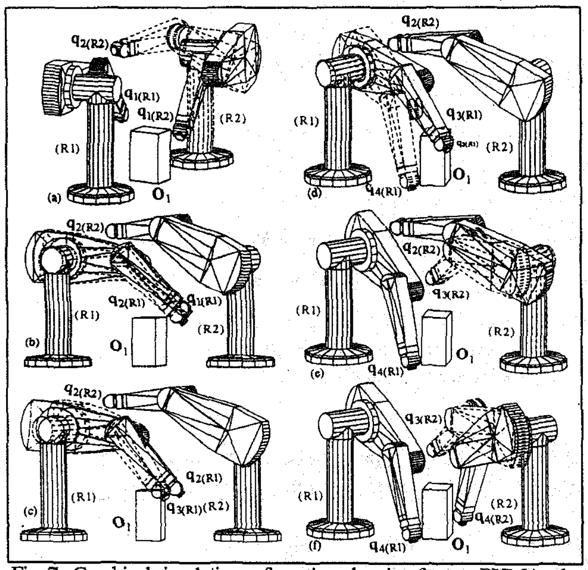


Fig. 7. Graphical simulations of motion planning for two PUMA robots