Finding the Maximally Inscribed Rectangle in a Robot's Workspace

Jonghyun Baek, Cornel-Constantin Iurascu, Frank Chongwoo Park*
School of Mechanical and Aerospace Engineering, Seoul National University, Seoul 151-742, Korea

In this paper we formulate an optimization based approach to determining the maximally inscribed rectangle in a robot's workspace. The size and location of the maximally inscribed rectangle is an effective index for evaluating the size and quality of a robot's workspace. Such information is useful for, e. g., optimal worktable placement, and the placement of cooperating robots. For general robot workspaces we show how the problem can be formulated as a constrained nonlinear optimization problem possessing a special structure, to which standard numerical algorithms can be applied. Key to the rapid convergence of these algorithms is the choice of a starting point; in this paper we develop an efficient computational geometric algorithm for rapidly obtaining an approximate solution suitable as an initial starting point. We also develop an improved version of the algorithm of Haug et al. for calculating a robot's workspace boundary.

Key Words : Robot Workspace, Maximally Inscribed Rectangle, Workspace Boundary

1. Introduction

A fundamental criterion to be considered when designing a robot for general-purpose tasks is its workspace. A robot's workspace is typically defined as the set of all reachable configurations of its end-effector. Under the usual practical constraints such as limits on link lengths and joint ranges, and assuming the robot is designed for general tasks, one would like the robot workspace not only to be as large as possible, but topologically well-connected and homogeneous, i. e., the workspace should be simply connected with a minimal number of holes and voids, and should extend uniformly in all directions.

Because of its fundamental nature, the problem of robot workspace analysis has received significant attention in the literature. Of the works that focus on characterizing workspace quality, Gupta (1986) examines the topological properties of the workspace of serial robots, while Bajpai and Roth (1986) do the same for a serial robot containing a single closed loop. Particularly relevant to this paper is the work of Haug et al. (1996), who present an efficient numerical algorithm for determining the boundary of a serial manipulator' s workspace.

One means of simultaneously characterizing the size and homogeneity (of the outer boundary and not taking into account any unreachable end-effector points) of a robot's workspace is in terms of the maximally inscribed rectangle. There are several advantages to this characterization. First, it is quite natural and practical to program a robot for a given task in terms of a Cartesian reference frame. For many robots often the physical worktable itself is rectangular in shape. Hence, by knowing the maximally inscribed rectangle in advance, one can optimally place the robot relative to the worktable to ensure that all points on the worktable are accessible by the robot. The same is true for multiple cooperating robots, where the workspace is the intersection of

* Corresponding Author,
E-mail : fcp@plaza.snu.ac.kr
TEL : +82-2-880-7133; **FAX** : +82-2-883-1513
School of Mechanical and Aerospace Engineering, Seoul National University, Seoul 151-742, Korea.
(Manuscript **Received** October 20, 2000; **Revised** May 21, 2001)

the workspaces of individual robots. Because the workspace of a single robot typically has a complex shape, working in terms of maximally inscribed rectangles considerably simplifies the analysis for multiple robots. Finally, the maximally inscribed rectangle is basic preliminary step toward determining convex polyhedral approximations to the workspace. For these and other reasons, the maximally inscribed rectangle has become a popular method in industry for prescribing the useful region of a robot's workspace (1999).

The maximally inscribed rectangle problem is well-known in computational geometry, and arises naturally in applications where an internal approximation to a polygon is desirable (e. g., laying out apparel and shoe pattern pieces with minimal waste (Daniels et al., 1997). It also arises in collision detection and other mobile robot planning problems (Agarwal, 1996). For example, if the approximated inscribed rectangle to a polygonal robot collides with other objects in the environment, then it is clear that the original polygon also collides with the same objects. Most previous studies in the computational geometry literature focus on convex polygons and polyhedra, usually with respect to fixed orientation and aspect ratio of the rectangle. If concave polygons, variable orientations and aspect ratios are considered, then the problem becomes nonlinear and quite complicated.

In this paper we present an optimization-based algorithm for determining the maximally inscribed rectangle in a robot's workspace. Simpler variations of this problem have been addressed in the computational geometry literature (Daniels et al., 1997; de Berg et al., 1997; O' Rouke, 1998), as well as the literature on mobile robot path planning (Agarwal, 1996), multiple robot manipulators (You and Jeong, 1998), convex programming and optimization (Vandenberghe et al., 1998), and data mining (Hong et al., 1997; Liu et al., 1997). However, previous studies make several restrictive assumptions that limit their usefulness for robot workspace analysis, e. g., assuming fixed aspect ratios and/or orientations for the rectangle, and assuming from the outset that the robot's workspace is a convex polygon. Nevertheless the problem itself is a very general one with a wide range of application beyond robot workspace analysis.

In this paper we frame the problem of determining the maximally inscribed rectangle in a robot's workspace as a multistage constrained optimization problem with a convex objective function. For the case of convex workspaces, in which the boundary can be approximated by a convex polygon, it can be shown that the problem reduces to a convex programming problem subject to multilinear constraints. For the general nonconvex case we formulate the problem as a constrained nonlinear optimization problem with a special structure, to which standard numerical optimization algorithms can be applied. Key to the convergence properties of these numerical algorithms is the choice of initial starting point. In this paper, we propose a computational geometric algorithm for rapidly obtaining an approximate solution that is suitable as a starting point for the numerical optimization procedure. Finally, we also refine the numerical algorithm of Haug et al. (1996) for efficiently determining the workspace boundary, by exploiting any symmetries present in the robot workspace.

The paper is organized as follows. In Sec. 2 we describe the modified cutting plane-based algorithm for determining the workspace boundary of a robot. Section 3 describes the formulation of the maximally inscribed rectangle problem as an optimization problem, together with the computational geometric algorithm for rapidly determining an approximate solution. Analysis results for a class of industrial robots are presented in Sec. 4. We conclude with a summary and discussion of future research topics in Sec. 5.

2. Robot Workspace Generation

In this section we first present an algorithm for determining the reachable workspace, which is the set of all Cartesian positions that can be reached by the end-effector. Our algorithm can be viewed as a refined version of that developed by Haug et al. (1996). To simplify the computation,

we propose two modifications that reduce the problem dimension: the introduction of the notion of cutting plane to symmetries in the robot structure, and explicitly separating the variables into joint angles and output coordinates in the computation.

2.1 Preliminaries

For an n-D. O. F. robot, the relationship between the end-effector Cartesian position $P=[P_1,\cdots, P_m]\in R^m(m\leq 3)$ and the joint space variables $\theta=[\theta_1, \cdots, \theta_n]\in R^n$ is defined by

$$P=f(\theta)$$

where f denotes the Cartesian position portion of the forward kinematic map. By differentiating Eq. (1) with respect to time, the transformation between the end-effector Cartesian velocity and joint velocity is derived as

$$\dot{P}=\frac{\partial f}{\partial \theta}\dot{\theta}=J(\theta)\dot{\theta} \tag{2}$$

where $J(\theta)$ is the $m\times n$ Jacobian matrix. Configurations at which the Jacobian $J(\theta)$ drops rank are referred to as singularities of the robot. At such configurations the robot end-effector loses the ability to move along certain directions in Cartesian space. For our purposes singularities are meaningful because they constitute a part of the workspace boundary together with joint limits. Specifically, let S denote the set of all singular configurations of the robot. Then assuming no joint limits, the boundary of the reachable workspace, denoted ∂W_r, is given by

$$\partial W_r\subset S\equiv\{f(\theta):\theta\in Q\&\text{Rank}(J(\theta)) \text{ is not maximal}\}$$

where f is the forward kinematic map and Q is the space of all allowable joint configurations.

In practice joint limit inequality constraints are often imposed on joints of the form $\theta_i^{min}\leq\theta_i\leq\theta_i^{max}$. By defining a new set of joint variables through a reparametrization via a sinusoidal function, the inequality constraints can be automatically satisfied without explicitly specifying the joint limits:

$$\theta_i=a_i+b_i\sin\lambda_i \tag{3}$$

Here a_i denotes the midpoint of the joint limit $(\theta_i^{max}+\theta_i^{min})/2$, while b_i is half the amplitude $(\theta_i^{max}+\theta_i^{min})/2$. Through the reparametrization, the original Jacobian is changed into the following form:

$$\dot{P}=\frac{\partial f}{\partial\theta}\frac{\partial\theta}{\partial\lambda}\dot{\lambda}=J(\cdot)\begin{bmatrix} b_1\cos(\lambda_1) & 0 & 0\\ 0 & \cdots & 0\\ 0 & 0 & b_n\cos(\lambda_n)\end{bmatrix}\dot{\lambda}=\hat{J}(\lambda)\dot{\lambda} \tag{4}$$

In Eq. (4), $\frac{\partial f}{\partial\theta}$ represents the original Jacobian, while the second term $\frac{\partial\theta}{\partial\lambda}$ describes the reparametrization Jacobian, whose singularities represent the joint limits. We now have a means of completely characterizing the workspace boundary as the singularities of a certain Jacobian.

2.2 Workspace boundary algorithm

The Cartesian workspace boundary for spatial robots can be represented by a collection of two-dimensional surfaces. For open chains these surfaces can be obtained by either rotating or translating a certain cross-section, called the cutting plane, about the first joint axis. A straightforward way to demonstrate this is to consider a 3 joint manipulator, e. g., the RRP one. First, we set aside the first joint rotation, and consider only motion in a vertical plane. The area of reach (assimilated to the workspace cross-section) is obtained as the difference between two pie wedges, given by the minimum and maximum extent of the prismatic joint. Then, the border and volume of the manipulator result from rotating this area about the first joint axis by the distance the centroid of the cross-section moves. In a similar way, the workspace boundary and volume of a 3R manipulator is given as the union of points swept by the cross-section of a revolving torus (due to the last two revolute joints) about the axis of the first joint. The result can be generalized to n-revolute manipulators and the interested reader is referred to (Ceccarelli, 1996) for details. Similar considerations apply to the case when the first joint is prismatic, and the rotation about first joint axis is simply replaced

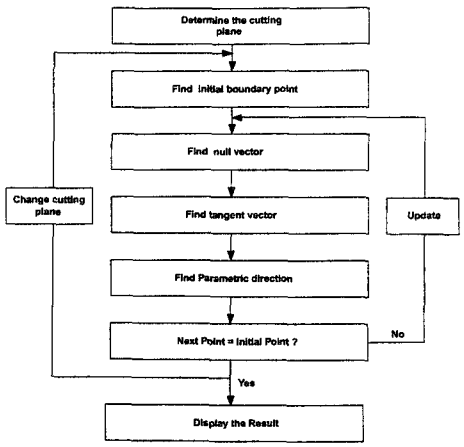

Fig. 1 Block diagram of the workspace generation procedure

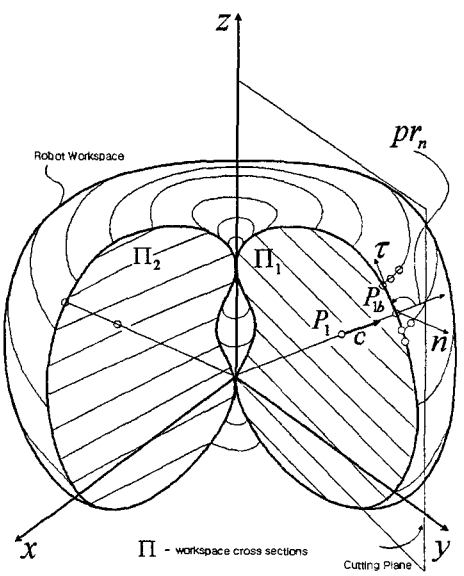

Fig. 2 Schematic of the workspace generation procedure

by translation on the axis. Like this, depending on the the type of the first joint, the cutting plane can be chosen as follows:

- Revolute joint: Any plane that includes the joint axis.
- Prismatic joint: Any plane normal to the joint axis.

Our algorithm is based on Haug's algorithm (1996)(see Appendix for a brief of the algorithm), which suggests the use of the continuation method in the joint and workspace region. At first we separate the variables into joint angles and output coordinates (given by the end-effector cartesian positions). From these separated variables and the cutting plane, we simplify the workspace generation and enrich the physical interpretation by directly relating to the geometric charateristcs of the workspace boundary and to the robot's manipulability index. A brief algorithm is as follows (see also Fig.1 and Fig.2):

(1) Determine the direction and number of the cutting plane. The direction is given by the first joint axis (as already specified above) and the number by the desired accuracy of the workspace outer boundary. The range of the cutting plane (accounting for the first joint limits) can be determined by the inverse kinematics based on a Newton-Rapson iteration.

$$\theta_i = \theta_{i-1} + J^{-1}(P_{desired} - P_{i-1})$$

where i is the iteration number, and $P_{desired}$ is the desirable position.

(2) Find the initial boundary point using a discrete movement on an arbitrary direction of a ray within the cutting plane. If the inverse kinematics solution about some position exists, the position is inside the workspace. Otherwise the point is outside the workspace. The discrete movement along the ray is defined by

$$P^i = P^{i-1} + hc$$

where h is the step size, and c is the ray direction. If the point is outside the workspace, the step size h is halved and the process is repeated until the criteria is met. The step size h is within a prescribed solution tolerance.

(3) Find the normal vector to the boundary surface using the Jacobian rank deficiency.

(4) Find the tangent vector at the boundary point. First, we project the normal vector onto the cutting plane and obtain the tangent vector by rotating the projected normal vector 90 degrees (since the normal vector is orthogonal to the tangent vector for regular curves).

(5) Find the joint space searching direction using the manipulability measure as defined in Yoshikowa(1985). By Taylor's expansion, this

direction is the joint space direction such that det (JJ^T) is maintained to be zero:

$$\det(JJ^T)_\theta = \det(JJ^T)_{\theta_0} + \frac{\partial \det(JJ^T)}{\partial \theta}\zeta + \frac{1}{2}\zeta^T \frac{\partial^2 \det(JJ^T)}{\partial^2 \theta}\zeta$$

where ζ is the joint space direction, θ_0 is the joint value of the present boundary point, and θ is the joint value of the next boundary point. Since at singularities the manipulability index becomes zero, the joint space searching direction results in the null space of Jacobian and Hessian of det (JJ^T) and it does not leave of the cutting plane.

(6) Find the next point using the joint space direction that results in the same direction as the tangent direction. The above process is repeated until the closed workspace is generated for the entire cutting plane. Then, rotate the cutting plane and repeat the above steps.

(7) For determining internal holes, a bifurcation point is traced. At a bifurcation there exist two or more joint space search directions: one corresponding to the boundary, the other corresponding to internal singularities. If we want to find only the outer boundary, the internal directions can be excluded at bifurcation points.

3. Robot Workspace Analysis

This section examines the maximum inscribed rectangle as one means of evaluating the workspace quality. Our primary focus will be on planar workspaces, in which case the rectangle will be two-dimensional. This problem is of practical significance for planar robots such as the SCARA. Moreover, most industrial robots have a symmetric structure which makes the planar results meaningful in a workspace analysis. In any event a careful understanding of the planar case is essential to developing a more complete analysis of three-dimensional workspace, such as finding the maximally inscribed polyhedron. In this section, we show how to formulate the problem as a constrained optimization problem, to which standard numerical optimization algorithms can be applied to obtain a solution. As is well known, the convergence characteristics of any numerical optimization algorithm is determined to a large extent by the choice of initial starting point. We show how to rapidly obtain an approximate solution to the problem, which serves as a good choice of starting point for the optimization.

3.1 Problem formulation

Following standard practice, in what follows we assume as given a polygonal approximation to a robot's workspace, and attempt to find the largest inscribed rectangle inside the polygon with variable orientation and aspect ratio. The problem can be stated as the following optimization problem:

$$\begin{aligned} &Maximize\ Area(R(x)) \\ &Subject\ to\ R(x) \subseteq P \end{aligned}$$

where R is the rectangle, x is the rectangle parameter, and P is the polygonal approximation to a robot's workspace.

In the event that the polygon is convex, the constraint $R(x) \subseteq P$ can be expressed as an inequality of the following form:

$$AR(x)_i \leq B$$

Here, $R(x)_i$ $(i=1, \cdots, 4)$ denotes the vertexes of the rectangle, and A and B are coefficients of the convex polygon $P=\{x \in R^n \mid Ax \leq B\}$ with the inequality holding component-wise. This problem was formulated as a convex programming problem subject to multilinear constraints by Vandenberghe et al. (1998). Observe that the requirement that the vertices of the rectangle lie inside the polygon is assured by the convexity condition.

For workspaces approximated by non-convex polygons, the constraints are more complicated. A visibility query (de Berg et al., 1997) only shows whether edges of the rectangle lie inside the polygon. To formulate the optimization problem, we must quantify how far the vertices and edges of the rectangle lie inside the polygon. That is, we seek a problem formulation of the form

$$\begin{aligned} &Maximize\ Area(R(x)) \\ &Subject\ to\ dist(R(x)_i, P) \leq 0 \\ &\qquad A(x) \leq 0 \end{aligned} \tag{5}$$

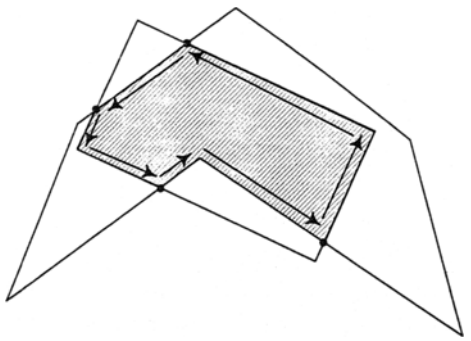

Fig. 3 Finding the intersection polygon

In Eq. (5), $dist(R(x)_i, P)$ is the distance between the point $R(x)_i$ and the simple polygon P, while $A(x)$ denotes the area of the region that extrudes from the polygon. Although the objective function is simple, the constraint is a nonlinear one that must in general be evaluated numerically. In the next section we explain in more detail the procedure for evaluating the constraints.

3.2 Constraint evaluation

The first constraint of Eq. (5) corresponds to the condition that every vertex of the rectangle must lie inside the polygon. The polygon is implemented as a boundary vertex list structure. We define the distance between a vertex and the polygon as follows:

$$dist(R(x)_i, P) = \begin{cases} -min(dist(R(x)_i, \text{each edge of } P)) & \text{if } R(x)_i \in \text{Interior of } P \\ min(dist(R(x)_i, \text{each edge of } P)) & \text{if } R(x)_i \in \text{Exterior of } P \end{cases}$$

Determining whether a point is in the interior or exterior of a polygon can be determined from the winding number, and is a standard procedure in computational geometry(O'Rouke, 1998). The final distance is taken to be the minimum value among the distances between each edge of the polygon and the vertex.

The second constraint of Eq. (5) can be interpreted as the area of the rectangle that lies outside the interior of the polygon; clearly an outer area of zero implies that the rectangle lies completely inside the polygon. The outer area can be evaluated by a combination of an intersection query (O'Rouke, 1998) and an interior query (i.e., determining whether a point lies inside or outside a polygon). To determine the area of the outer polygon($P-R$), we first find the intersection polygon $P \cap R$ as follows (see Fig. 3):

1. Calculate the number of intersection points (n_i) between the edges of the polygon and the rectangle by an intersection query, the number of the rectangle interior points (n_r) (vertexes lying inside the polygon) and the number of polygon interior points (n_p) (vertexes lying inside the rectangle) by an interior query.

2. Determine the first vertex p_i of the intersection polygon $P \cap R$ and the search direction V_s as follows:

• Case $n_r \neq 0$: p_i=a rectangle interior point, V_s=the rectangle direction (see Figure 3);

• Case $n_i=0$, $n_p \neq 0$: p_i=a polygon interior point, V_s=the polygon direction (see Figure 3);

• Case n_r, $n_p=0$, $n_i \neq 0$:p_i=an intersection point, V_s=the direction determined by the interior query of the mid-point between the intersection points on the same edge;

• Case n_r, n_p, $n_i=0$: the rectangle and the polygon are disjoint.

3. Determine the next vertex of $P \cap R$; that is, the next vertex of the search direction until an intersection point is detected. If the intersection point is met, the next vertex is the intersection point and V_s is changed. Perform the above operation until the closed intersection polygon is created.

4. Check if the new intersection polygon is separated by comparing($n_i+n_r+n_p$) with the number of vertexes of the intersection polygon. If the two numbers are not the same, perform the same procedure for the other part.

3.3 The initial guess determination

In the previous section, we formulated the maximally inscribed rectangle problem as a nonlinear constrained optimization problem, for which several local minima may exist. Also, as is well known the convergence behavior of any numerical optimization algorithm depends to a great extent on the choice of the starting point. In this section we present a method for rapidly obtaining an approximate solution to the prob-

lem, that can be used as a starting point for the nonlinear optimization procedure of the original problem. We achieve this by simplifying the original problem to the largest empty rectangle problem as first proposed by Liu et al. (1997).

This approach applies the concept of the maximum hyper rectangle (MHR). MHR refers to any rectangle that contains no discrete workspace boundary points within its interior, and has at least one boundary point on each bounding edge. The main idea of this algorithm is as follows. Given 2-dimensional bounded space S and n points in S, we first start with one MHR, using the maximal point of each dimension, which covers the entire space S and ignoring the interior points. Then each interior point is incrementally inserted in the initial MHR. At each insertion, we update the set of MHRs that have been created. The update process is done as follows (see also Fig. 4). When a new point is added, we first find all existing MHRs that contain this point. These rectangles are no longer MHRs, since they contain the point within their interiors. Using the newly added point, a new lower and upper bound for each dimension are formed to result in new MHRs. If these new MHRs are greater than a certain criteria, they are inserted into the list of existing MHRs.

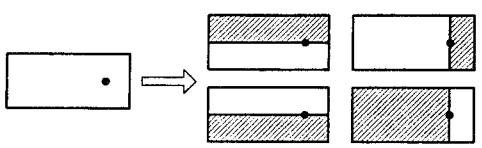

Fig. 4 Illustration of the computational geometry algorithm

We use a simple example to illustrate the algorithm (see Fig. 5). Suppose that the polygonal workspace is changed into a discrete set of data points. First, construct the bounded space S (ABCD). Second, add the point (P_1) inside S. Through an update process, ABCD is deleted from the MHR list, and new MHRs are generated (ABFE, EFCD, AGHD, GBCH). Third, consider the next point(P_2). Because MHR ABFE and AGHD contain P_2, these are changed into new MHRs. ABFE is separated into KBFM, AKME,

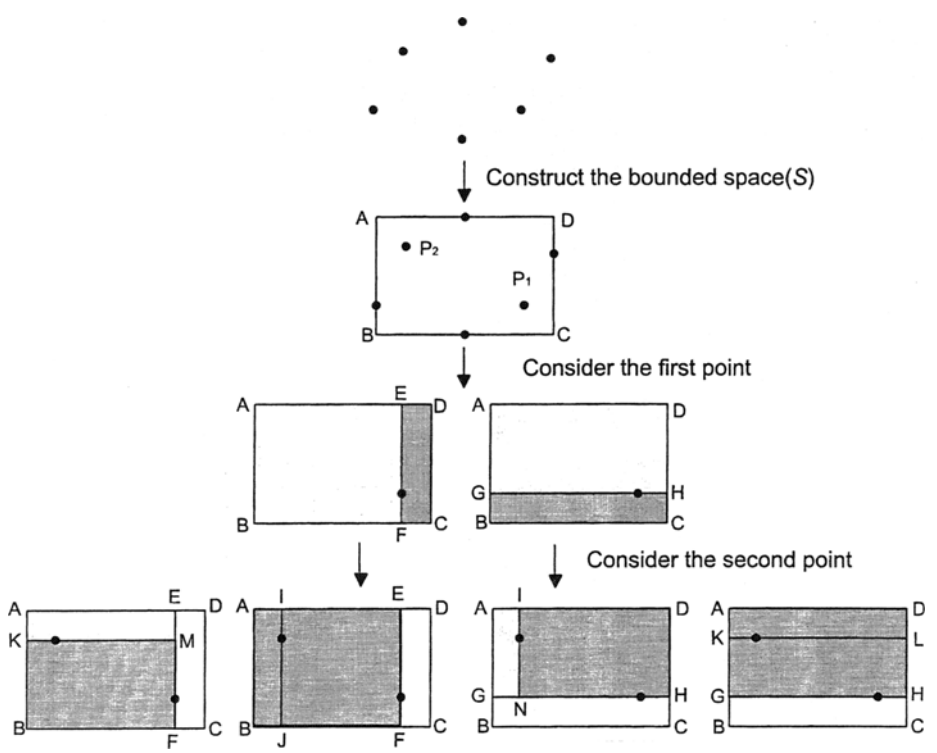

Fig. 5 Procedure of the MHR algorithm

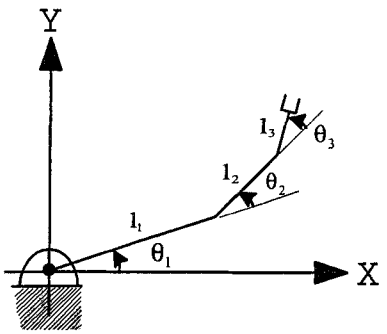

Fig. 6 Planar robot with redundant input

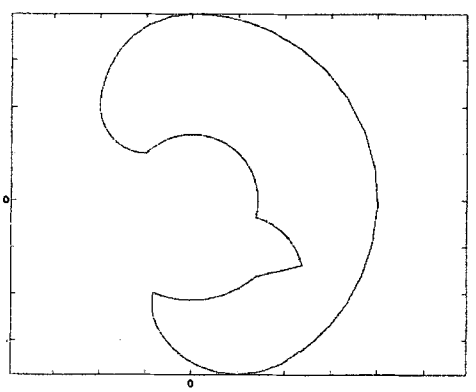

Fig. 7 Workspace of planar robot with redundant input

ABJI, and IJFE. But, because the right side of AKME (ME) is not bounded anymore, AKME is not an MHR. Hence only KBFM, ABJI, IJFE are inserted into the list of existing MHRs. The same process is applied to AGHD, and new 3 MHRs are also generated. Repeating this process until all the points of the workspace boundary are accounted for, we arrive at the MHR of the bounded space S, i. e. IJFE, which represents the initial guess for the nonlinear problem. If more workspace boundary points are considered the approximation of the initial guess increases accordingly.

4. Examples

4.1 A redundant planar open chain

In this section we evaluate the workspace of a three d.o.f. planar open chain using the algorithm of the previous section. The input-output characteristics of the three-revolute joint chain shown in Fig. 6 are the three link angles θ_1, θ_2, θ_3, and the x and y coordinates of the tip. The joint value ranges are defined as $\left[-\frac{\pi}{3}\leq\theta_1\leq\frac{\pi}{2},\ -\frac{\pi}{2}\leq\theta_2\leq\frac{\pi}{2},\ -\frac{\pi}{4}\leq\theta_3\leq\frac{\pi}{2}\right]$. The lengths of the links are $l_1=4$, $l_2=2$, and $l_3=1$. The joint variables θ are first reparameterized in terms of the new variable λ. That is, $\theta_1=\frac{\pi}{12}+\frac{5\pi}{12}\sin(\lambda_1)$, etc., and the forward kinematics and velocity relation are written accordingly. Since we deal with a planar mechanism there is obviously no need to apply any cutting planes. The specifics of the analysis are shown in Table 1, while the pictorial view of the workspace is presented in Fig.7.

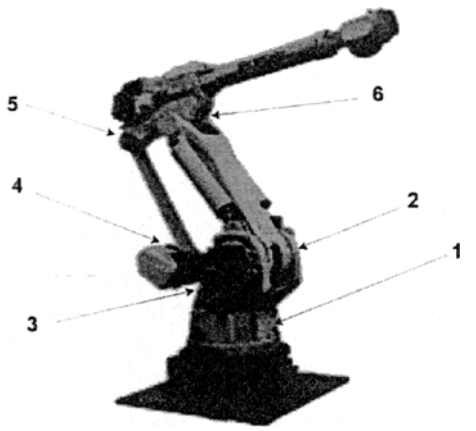

Fig. 8 H120 model

4.2 An industrial robot containing a closed loop

In this section we consider the workspace of the H120 industrial robot manufactured by Hyundai. This robot contains a single closed loop in the center of its structure (see Fig. 8), to allow the actuators to be placed at the base and thereby reduce the inertia of the robot while increasing its payload. The mechanism contains both active and passive joints.

We consider the generation of the workspace with respect to the wrist center point. The Cartesian positioning workspace of the H120 is

Table 1 Analysis conditions and results

	Value		Result
Initial λ	0, 0, 0	Num of Vertex	98
Initial position(mm)	6588.9, 2161.7	Max X(mm)	6968.4
Searching direction(mm)	0, 1	Min X(mm)	-3000.0
Initial boundary point(mm)	6588.9, 2363.5	Max Y(mm)	7000.0
Tangent vector(mm)	-337.6, 941.3	Min Y(mm)	-6460.4
Joint space direction	1, 0, 0	Area(mm^2)	61.6059e6

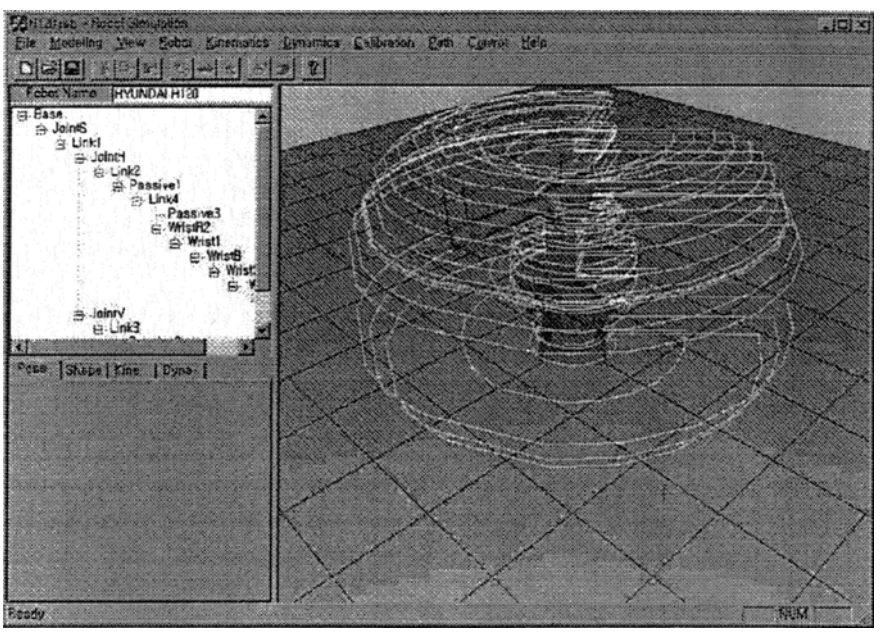

Fig. 9 Workspace of H120 model

hown in Fig. 9. Table 2 lists the data on the workspace of the H120; the wrist point coordinates at the home position are assumed to be (in mm) (1535, 0, 2090). For the H120 the cutting plane can be taken to be any plane containing the z-axis of the fixed frame, or equivalently the first joint axis. Note that the robot workspace can be obtained by sweeping the cross-section in any given cutting plane about the first revolute joint axis. After performing this sweeping operation, the workspace volume is evaluated to be 4, 874, 740 mm^3.

We first obtain an approximate solution for a given fixed orientation of the rectangle, by discretizing the workspace boundary and applying the computational geometric algorithm of the previous section. The results are presented in Fig. 10 and Table 3. The initial conditions of the analysis are given by the x and y-range, as presented in the first column of Table 3. The overall computation time is approximately 7 seconds on a Pentium II 266MHz PC, with the algorithm implemented in C++.

The approximate solution for the largest empty rectangle(x and y-range, displayed in the last column of 3) is then used as the initial starting point in the optimization procedure for arbitrary orientations of the rectangle. We apply the BFGS update algorithm using Cholesky factorization as outlined in Gill et al. subject to the numerical constraint Eq. (5). The optimized largest rectangle is shown in Fig. 11 and its details are

Table 2 Specifics and workspace of H120 model

Joint	Screw	Limits	active	Max Reach	Result
1	0, 0, 1, 0, 0, 0	$-\frac{5\pi}{6}, \frac{5\pi}{6}$	○	Max X(mm)	2,524
2	0, 1, 0, -840, 0, 200	$-\frac{5\pi}{12}, \frac{5\pi}{12}$	○	Min X(mm)	-2,185.8
3	0, 1, 0, -840, 0, 200	$-\frac{\pi}{6}, \frac{11\pi}{18}$	○	Max Y(mm)	2,524
4	0, 1, 0, -840, 0, -300	···	×	Min Y(mm)	-2,524
5	0, 1, 0, -1840, 0, -300	···	×	Max Z(mm)	2,723.2
6	0, 1, 0, -1840, 0, 200	···	×	Min Z(mm)	-259.4

Table 3 Empty rectangle in workspace of H120

Analysis Condition	Value	Largest Rectangle	Value
X-range(mm)	[0, 2524]	X-range(mm)	[589,2180]
Z-range(mm)	[-260 2730]	Z-range(mm)	[255,2170]
Nb. of Points	227	Nb. of Rectangles	1591
Criteria(mm^2)	2.5e6	Area(mm^2)	3,044,060

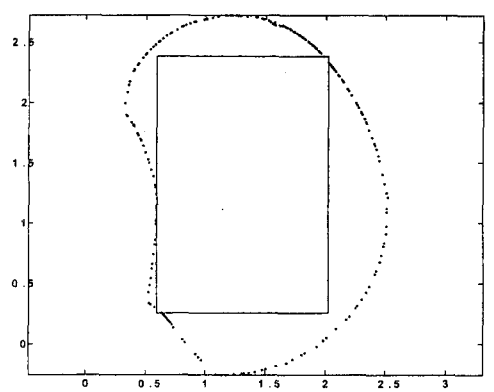

Fig. 10 Empty Rectangle in workspace of H120

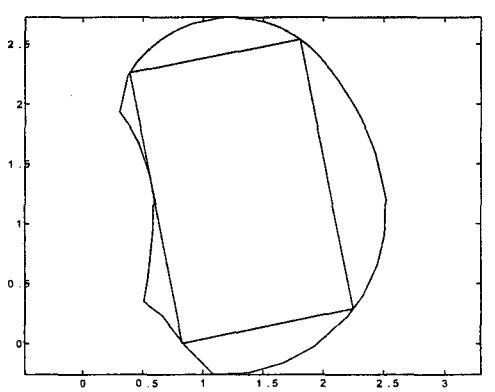

Fig. 11 Inscribed largest rectangle within H120 workspace

presented in Table 4. The area of the largest inscribed rectangle with variable orientation is larger than the initial rectangle of a fixed orientation by more than 10%. The overall computation time takes approximately 27 seconds for this phase of the procedure.

5. Conclusions

In this paper we suggest a method for robot workspace analysis in terms of the maximally inscribed rectangle in the workspace. This approach provides information about the size and homogeneity of the robot's workspace, and is useful for applications ranging from optimal work-table placement to the placement of cooperating robots. We formulate the problem as a constrained nonlinear optimization problem for both convex and concave workspace environments. For the actual optimization procedure, we construct the appropriate constraints and quantify the distance and the outer area between the rectangle and the workspace. Additionally, because the initial starting point is all-important in determining the convergence properties of the

Table 4 Inscribed largest rectangle characteristics within H120 workspace

Inscribed Largest Rectangle	Value
Center(mm)	[1320.03, 127.46]
Width(mm)	1,444.73
Height(mm)	2,289.1
Orientation	0.1929(radian)
Area(mm^2)	3,307,130

optimization, we use an efficient computational geometric algorithm for rapidly obtaining an initial starting point.

Another contribution of this paper is an improved version of the algorithm of Haug et al. (1996) for calculating a robot's workspace based on a singularity analysis. We propose the concept of a cutting plane, and separate the input and output variables for simplifying the determination of the robot workspace. The algorithm is applied to evaluate the workspace of a redundant planar robot and an industrial robot containing a closed loop.

Our algorithm suggests further study on the mechanism workspace optimization. Specifically, the proposed algorithm proposed in this paper can address not only the outer workspace boundary of the mechanism but also the internal directions (holes and voids). Another possible direction for future research is to extend the presented singularity based algorithm to parallel mechanisms.

Acknowledgements

This research was supported by the CAD/CAM NRL and the BK21 Program in Mechanical Engineering at Seoul National University.

Reference

Agarwal, P. K., Amenta, N., Aronov, B., and Sharir, M., 1996, "Largest Placements and Motion Planning of a Convex Polygon," 2nd International Workshop on Algorithmic Foundation of Robotics.

Allgower, E. L., Georg, K., 1990, *Numerical Continuation Methods*, Springer-Verlag, Berlin Heidelberg.

Bajpai, A., Roth, B., 1986, "Workspace and Mobility of a Closed-Loop Manipulator," The Int. J. of Robotics Research

Ceccarelli, M., 1996, "A Formulation for the Workspace Boundary of General N-Revolute Manipulators," *Mech. Mach. Theory*, Vol. 31, No. 5, pp. 637~646.

Daniels, K. M., Milenkovic, V. J., Roth, D., 1997, "Finding the Maximum Area Axis-Parallel Rectangle in a Simple Polygon," In Computational Geometry: Theory and Applications, Vol. 7, pp. 125~148.

Gupta, K. C., 1986, "On the Nature of Robot Workspace," *The Int. J. of Robotics Research*, Vol. 5, No. 2.

Haug, E. J., Luh, C. M., Adkins, F. A., Wang, J. Y., 1996, "Numerical Algorithms for Mapping Boundaries of Manipulator Workspaces," *J. of Mechanical Design*, Vol. 118, No. 2, pp. 228~234.

Hong, K. S., Kim, Y. M., Choi, C., 1997, "Inverse Kinematics of a Reclaimer: Closed-Form Solution by Exploiting Geometric Constraints," *KSME Int. J.*, Vol. 11, No. 6, pp. 629~638.

Hyundai Industrial Robot Brochure, 1999, Hyundai Heavy Insustries Co.

Liu, B., Ku L. P., Hsu, W., 1997, "Discovering Interesting Holes in Data," *Proceedings of Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, Nagoya, Japan, pp. 930~935.

Luenberger, D. G., 1989, *Linear and Nonlinear Programming*, Addison Wesley.

Murray, R. M., Li, Z., Sastry, S. S., 1994, *A Mathematical Introduction to Robotic Manipulation*, Boca Raton, CRC Press.

O'Rouke, J., 1998, *Computational Geometry in C*, Cambridge University Press.

Strang, G., 1986, *Linear Algebra and its Applications*, Harcourt Brace & Company.

Vandenberghe, L., Boyd, S. P., Wu, S. P., 1998, "Determinant maximization with linear matrix inequality constraints," *SIAM Journal on Matrix*

Analysis and Applications, April.

Yoshikawa, T., 1985. "Manipulability of Robotic Mechanisms," *The Int. J. of Robotics Research*, Vol. 4, No. 2, pp. 3~9.

You, S. S., and Jeong, S. K., 1998, "Kinematics and Dynamics Modeling for Holonomic Constrained Multiple Robot Systems through Principle of Workspace Orthogonalization," *KSME Int. J.*, Vol. 12, No. 2, pp. 170~180.

de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O., 1997, *Computational Geometry: Algorithms and Applications*, Springer-Verlag, Berlin Heidelberg.

Appendix

The position and orientation of each body of a mechanism is characterized by the generalized coordinates $q=[q_1, \cdots, q_{nq}]^T \in R^{n_q}$, subject to the independent kinematic constraint equations of the form:

$$\Phi(q)=0 \tag{6}$$

where $\Phi: R^{n_q} \rightarrow R^m$ is a smooth function. To describe the accessible output set of a manipulator, the generalized coordinates are first partitioned into $q=[u^T, v^T, w^T]^T \equiv [u^T, z^T]^T$, where u, v and w are the input, output and intermediate coordinates respectively. Then, upon changing accordingly the constraint equation, the accessible output set A is:

$$A \equiv \{u \in R^{n_u}: \Phi(u, z)=0, \text{ for some } z\} \tag{7}$$

and the boundary of the accessible output set is given by:

$$\partial A \subset \{u \in A: \text{Rank } \Phi_z(u, z) < m, \text{ for some } z \text{ with } \Phi(u, z)=0\} \tag{8}$$

This condition is reduced to an analytical form by noting that a matrix is row rank deficient iff the columns of its transpose are linearly dependent, i. e., :

$$\partial A \subset \{u \in A: \Phi_z^T(u, z)\xi=0,\ \xi^T\xi=1,\ \Phi(u, z)=0\} \tag{9}$$

with $\xi \in R^m$. In the enlarged space of $x=[u^T, z^T, \zeta^T]^T \in R^n$, the workspace boundary as given by the above equation is the projection of solutions of the following equations onto the u-space:

$$G(x)=\begin{bmatrix} \Phi(u, z) \\ \Phi_z^T(u, z)\xi \\ \xi^T\xi-1 \end{bmatrix}=0 \tag{10}$$

Like this, the numerical algorithm for mapping the boundaries of the manipulator workspace is summarized as follows:

1. Find an initial point on the workspace boundary starting from an assembled configuration of the manipulator. For this, a unit vector in the output space is first selected and discrete movements along a ray defined by the starting configuration and the unit vector are performed. The associated values of the intermediate coordinates z that determine the steps along this ray are determined by a numerical procedure from the modified kinematic constraint equations. The next sub step is to find a vector ζ that satisfies the conditions of Eqs. (8) and (9). This is accomplished by a least square procedure while checking on the numerical stability of the solution. This completes the routine for finding the desired starting point on the manipulator boundary.

2. Map one-dimensional solution curves in the manipulator boundary. The desired one-dimensional curve on the boundary is a solution set of (10). If $G_x(x)$ has full rank, a unit tangent vector $h(x) \in R^n$ is uniquely defined by:

$$\begin{aligned} &G_x(x)h(x)=0 \\ &h(x)^Th(x)=1 \\ &\text{Det}\begin{bmatrix} G_x(x) \\ h(x)^T \end{bmatrix}>0 \end{aligned} \tag{11}$$

As long as G_x has full row rank, a unique unit tangent vector can be computed at each point along the solution of (10). At bifurcation points, G_x is row rank deficient and (12) fails to determine a unique tangent. This issue is addressed in the next step.

3. Find tangents to continuation curves at bifurcation points. Tangents to two continuation curves are obtained as solutions of a quadratic equation in two variable, which finally yields a pair of tangents. For more than two continuation curves another method has to be applied.

4. Find barriers to output normal to the boundary. This step assists in identifying curves that comprise the exterior boundary of the accessible output set. It also helps in determining local mobility restrictions along selected curves in the interior of the accessible output.