Abdel-Malek, K., (1999), "Collision Detection of Manipulator Arms with Complex Links," ***International Journal of Robotics and Automation*****, Vol. 14, No.2, pp. 78-86.**

Karim Abdel-Malek
Department of Mechanical Engineering and
Center for Computer Aided Design
The University of Iowa
Iowa City, IA 52242
E-mail: amalek@icaen.uiowa.edu

ABSTRACT

A method for detecting collision of manipulator links is presented and used to avoid collision in a robotics environment in which manipulator links and obstacles are represented by complex solids. Surface boundary representation methods are used to envelop each solid with a known surface (ruled, double-curved, surface patches, and planar surfaces). Negative entities are defined on surfaces having discontinuities. Based on this representation scheme, a collision detection method is developed that incorporates complex solids. The collision avoidance algorithm is based on checking the surfaces of solid pairs of bodies for interference. The intersection curve (or branch) is calculated and checked for existence inside the boundary of entities on the two surfaces. Each segment of the intersecting curve is analyzed by introducing a new method for determining the existence of a point inside the boundary of an entity. This method is based upon computing the *circulation* along a closed contour of a surface. Boundary curves using this method need not be convex or simply connected. The theory presented is implemented into a computer program and used to detect collisions of a six-axis manipulator arm.

KEY WORDS: Robotics, collision-detection, manipulators, and interference.

1 INTRODUCTION

In robotic cells and when performing simulation of robot motions, it is possible to unwittingly require one component of a robot to collide with another. Researchers have addressed the collision detection (avoidance) problem in robotics by modeling solids primarily with simple representations. Before addressing the problem of collision avoidance, it is important to introduce the method by which a solid can be represented. Therefore, a convenient method for representing solids is needed. In the past, two distinct methods B-Rep and CSG have been used [21]. Using these methods, numerous collision-detection approaches have been presented varying in complexity, time, and efficiency. A comparative study of the well-known interference algorithms can be found in [15]. Surface intersection problems were addressed using four main categories of methods [9], namely algebraic, lattice evaluation, marching methods, and recursive subdivision methods. Some of the early work using a computational method to check for the interference of edges and surfaces of two defined solids was presented by [6]. Other methods of collision detection include the use of barycentric coordinates [19], detecting collisions using a characterization of an empty space to describe entities [24], and using linear segments [7]. Other general considerations for collision avoidance are reported [5, 13].

Other researchers in the field have addressed the real-time collision-avoidance problem [18,23, 15]. The method of minimizing distances has also been implemented [25, 26, 17, 10]

An efficient method characterized by representing the obstacles as inequality constraints in the workspace such that these inequalities are satisfied throughout the desired trajectory was presented

by Glass *et al.* [11, 12]. An algorithm that is based on maintaining a structure reflecting the urgency or imminence of collision for each pair was presented by Hayward [15].

In this paper, manipulator links are enveloped by a set of parametric surfaces enveloping the solid as presented in Sec. 2. Discontinuities are also modeled using this technique. The mathematical formulation for detecting collision between various entities is presented in Sec. 3. The computational complexity of the various steps in the formulation is also addressed in Sec. 4. Several examples are performed and a report of the use of this method in a robotics environment is presented in Sec. 5.

2 BODY REPRESENTATION

To adequately represent complex solids, a method using a combination of B-reps and CSG is used. Before proceeding with describing this approach, some terms are defined.

Surface: A subset $S \subset R^3$ such that there exists a map $\mathbf{x}: \mathbf{U} \rightarrow \mathbf{V} \cap S$ where **x** is differentiable (e.g., a cylindrical surface with infinite length).

Entity: A bound surface (e.g., the surface of a cylinder bound between two angles and between two planes). Note that a surface may be composed of several entities.

Negative entity: An entity that describes a discontinuity on the surface.

Convex curve: A simple closed regular plane curve whose curvature never changes direction.

Ruled Surfaces: A one parameter family of lines $\{\mathbf{a}(t), \mathbf{w}(t)\}$ is a correspondence that assigns to each $t \in I$ a point $\mathbf{a}(t) \in R^3$ and a vector $\mathbf{w}(t) \in R^3, \mathbf{w}(t) \neq 0,$ so that both $\mathbf{a}(t)$ and $\mathbf{w}(t)$ depend differentially on *t*. The parametrized surface based on the parametrized lines is

$$\mathbf{x}(t,v) = \mathbf{a}(t) + v\mathbf{w}(t), \text{ such that } \quad t \in I \text{ and } v \in R, \text{ where } v_1 \leq v \leq v_2 \qquad (1)$$

Surfaces of Revolution: Let $S \subset R^3$ be the set obtained by rotating a regular plane curve **c** about an axis in the plane. To illustrate, consider the *xz*-plane as the plane of the curve and the z axis as the rotation axis. Thus $x = f(v)$, and $z = g(v)$, a < v < b, with $f(v) > 0$ is a parametrization, where *u* is the rotation angle about the z axis. The position vector is thus

$$\mathbf{x}(u,v) = \begin{bmatrix} f(v)\cos u & f(v)\sin u & g(v) \end{bmatrix}^T \qquad (2)$$

which is a mapping from the open set $U = \{(u,v) \in R^2; u_1 \leq u \leq u_2, v_1 \leq v \leq v_2\}$ into S.

Surface Patches: The algebraic form of a Bi-cubic surface patch [20] shown in Fig. 1 is

$$\mathbf{x}(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij} u^i v^j \qquad (3)$$

from the open set $U = \{(u,v) \in R^2; 0 \leq u \leq 1, \quad 0 \leq v \leq 1\}$ where the a_{ij} are called algebraic coefficients that determine the patch shape and its position in space.

Equation 3 can be written in a compact form suitable for manipulation as

$$\mathbf{x}(u,v) = UM_c BM_c^T V^T \qquad (4)$$

where B is the so-called corner information matrix (each element in the B matrix is a vector),

$$B = \begin{bmatrix} \mathbf{x}(0,0) & \mathbf{x}(1,0) & \mathbf{x}_v(0,0) & \mathbf{x}_v(1,0) \\ \mathbf{x}(0,1) & \mathbf{x}(1,1) & \mathbf{x}_w(0,1) & \mathbf{x}_v(1,1) \\ \mathbf{x}_u(0,0) & \mathbf{x}_u(1,0) & \mathbf{x}_{uv}(0,0) & \mathbf{x}_{uv}(1,0) \\ \mathbf{x}_u(0,1) & \mathbf{x}_u(1,1) & \mathbf{x}_{uv}(0,1) & \mathbf{x}_{uv}(1,1) \end{bmatrix} \qquad (5)$$

$\mathbf{x}_u(u,v) = \partial\mathbf{x}(u,v)/\partial u$, $\mathbf{x}_v(u,v) = \partial\mathbf{x}(u,v)/\partial v$, and the twist vector is $\mathbf{x}_{uv}(u,v) = \partial^2\mathbf{x}(u,v)/\partial u\partial v$. The 4×4 matrix M is called the blending function, and the parameter matrices are $U = \begin{bmatrix}1 & u & u^2 & u^3\end{bmatrix}$, and $V = \begin{bmatrix}1 & v & v^2 & v^3\end{bmatrix}$

Parametrization of Inequality Constraints: The parametric form of an inequality given by

$$u_i^{\min} \le u_i \le u_i^{\max} \tag{6}$$

is obtained by introducing a new variable λ and the transformation [14]

$$u_i = a_i + b_i \sin\lambda \tag{7}$$

where $a_i = \left(u_i^{\max} + u_i^{\min}\right)/2$ and $b_i = \left(u_i^{\max} - u_i^{\min}\right)/2$ are the mid point and half range of (6).

For further discussion, entities are classified into two types:

<u>Type I</u>: *Ruled, curved, and surface patches*

<u>Type II</u>: *Surfaces having closed boundary curve*s. (Curves need not be convex nor simply connected).

These two types of entities are adequate to represent complex parts having holes, chamfers, and fillets. Using this approach, a solid is **enveloped** by entities (positive and negative). To illustrate, consider the solid depicted in Fig. 2. Entity 1 is toroidal, with the mapping

$$\mathbf{x}(u,v) = \begin{bmatrix}(R + r\cos u)\cos v & (R + r\cos u)\sin v & r\sin u\end{bmatrix}^T \tag{8}$$

from the open set $U = \left\{(u,v) \in R^2; -\frac{\pi}{2} \le u \le -\frac{\pi}{2}, 0 \le v \le \pi\right\}$. Entities 2, 3, and 4 are positive cylindrical entities, while entities 5, 7, and 9 are positive planar entities. Negative entities are 6 (associated with entity 5) and 8 (associated with entity 7). The above nine entities completely envelop the solid, thus completely representing it. In addition, in defining holes, the surface of the hole itself is defined as a positive entity enveloping the solid body (cylindrical surface 4), while the discontinuities leading to that hole are the negative entities 6 and 8.

The entities shown in Fig. 2 completely represent the given solid. The collision detection algorithm will be presented in the following section in view of detecting interference between the two type of entities.

3 COLLISION DETECTION

In order to detect collision between two solids, the entities (positive and negative) enveloping the two solids are used to check for interference. The general procedure is to determine the intersecting curve (or disjoint branches of the curve) resulting from one solid penetrating the other, followed by determining whether each segment of the curve coexists on both surfaces (inside the boundary of each positive entity but outside each negative entity). The necessary condition for two solids to collide is that a segment of the curve of intersection between any two entities coexists inside boundaries of positive entities, but outside boundaries of negative entities.

Because two types of entities are used, three sections are presented addressing intersections between (a) entities both from Type I, (b) entities both from type II, and (c) an entity from type I and another from type II.

3.1 Interference Detection Between Type II Entities

An entity is defined by a parametric equation of its surface and a curve defining its boundary. Boundary curves of each entity are checked for interference with the surface of the other entity. The procedure is repeated for each pair of entities associated with each solid. This will result in

intersection points defining the intersection curve. This curve will then be verified for existence inside boundaries of positive and negative entities.

To illustrate the methodology, consider two solids depicted in Fig. 3a. For two of the surfaces in question, solid 1 has positive entity 1 and negative entity 2, while solid 2 has positive entity 3 and negative entity 4. Fig. 3b depicts the two solids at a different instance of time when one is interpenetrating the other.

Entities 1 and 2 are on the surface of the plane Ω_1, and entities 3 and 4 are on the plane Ω_2. Figure 4 depicts the four entities in more detail. The intersection of the boundary curve $\mathbf{c}_1(t)$ of entity 1 with the surface $\mathbf{x}_2(u,v)$ of entity 3, will result in intersection points $\mathbf{p}_2$ and $\mathbf{p}_8$. The intersection of the boundary curve $\mathbf{c}_2(t)$ of entity 2 with the surface $\mathbf{x}_2(u,v)$ of entity 3 results in intersection points $\mathbf{p}_4$ and $\mathbf{p}_7$. Similarly, the intersection of the boundary curve $\mathbf{c}_3(t)$ of entity 3 with the surface $\mathbf{x}_1(u,v)$ of entity 1 results in intersection points $\mathbf{p}_1$ and $\mathbf{p}_6$. Finally, the intersection of the boundary curve $\mathbf{c}_4(t)$ of entity 4 with the surface $\mathbf{x}_1(u,v)$ of entity 1 results in intersection points $\mathbf{p}_3$ and $\mathbf{p}_5$. Thus, the intersection curve is well defined; i.e., all boundary points of intersection are computed. A necessary condition for the two solids in Fig. 3b to collide is that a line segment on the line of intersection between any two entities coexists inside boundaries of positive entities, but outside boundaries of negative entities. Since all points of intersection are computed, it is possible to represent each line segment on the line of intersection by a single point $\mathbf{s} = (\mathbf{p}_{i+1} + \mathbf{p}_i)/2$, for i = 1, ..., 8.

3.1.1 The Existence of a Point Inside the Boundary of a Curve

To study the existence of a point inside a boundary **c**, let $\zeta(\mathbf{s})$ be the indicator for a point **s** which determines its existence inside the boundary $\mathbf{c}(t)$ of a positive entity, such that

$$\zeta(\mathbf{s}) = \frac{1}{2\pi}\oint_{\mathbf{c}} d\theta = \begin{cases} 1 & \text{inside the boundary} \\ 0 & \text{outside the boundary} \end{cases} \tag{9}$$

where $d\theta$ is the change in angle swept by a vector subtended from the point ***s*** to the boundary ***c(t)***, as depicted in Figs. 5a and 5b, inside and outside, respectively. The proof of Eq. (9) is presented in Appendix A. For the 2-dimensional case, the indicator can be written as

$$\zeta(\mathbf{s}) = \frac{1}{2\pi}\oint_{\mathbf{c}} d\tan^{-1}\left(\frac{(\mathbf{s}-\mathbf{c})\cdot\mathbf{j}}{(\mathbf{s}-\mathbf{c})\cdot\mathbf{i}}\right) \tag{10}$$

For two positive entities 1 and 2, interference will occur if and only if the line segments lay inside the boundaries of both entities. The combined indicator for two positive entities is

$$\zeta_{12}(\mathbf{s}) = \zeta_1(\mathbf{s})\zeta_2(\mathbf{s}) \tag{11}$$

where $\zeta_1(\mathbf{s})$ is the indicator for entity 1, $\zeta_2(\mathbf{s})$ is the indicator for entity 2, and $\zeta_{12}(\mathbf{s})$ is the combined indicator. The indicator for a negative entity (e.g., entity 2) is defined as

$$\zeta_{2\ \text{negative}}(\mathbf{s}) = 1 - \frac{1}{2\pi}\oint_{\mathbf{c}} d\theta = 1 - \zeta(\mathbf{s}) \tag{12}$$

The combined indicator for a segment on the curve in Fig. 4, represented by a point **s** is

$$\zeta_{1234}(\mathbf{s}) = \zeta_1(\mathbf{s})\big(1-\zeta_2(\mathbf{s})\big)\zeta_3(\mathbf{s})\big(1-\zeta_4(\mathbf{s})\big) \tag{13}$$

Thus for two surfaces comprising positive and negative entities, the indicator can be written as

$$\zeta_{1\ldots n} = 1 - \prod_{k=1}^{m}\zeta_k \prod_{l=m+1}^{n}(1-\zeta_l) \tag{14}$$

Where *m* is the number of positive entities, and *n-m* is the number of negative entities on the two surfaces. Consider the middle point $\mathbf{s}_1$ on line segment $\overline{\mathbf{p}_1\mathbf{p}_2}$ in Fig. 4. The point is outside boundaries $\mathbf{c}_1(t)$, $\mathbf{c}_2(t)$ and $\mathbf{c}_4(t)$ thus $\zeta_1(\mathbf{s}_1)=\zeta_2(\mathbf{s}_1)=\zeta_4(\mathbf{s}_1)=0$, and inside boundary $\mathbf{c}_3(t)$ thus $\zeta_3(\mathbf{s})=1$. Using Eq. 14, the combined indicator $\zeta_{1234}(\mathbf{s}_1)=0$ indicates *no interference* due to this segment. While the middle point $\mathbf{s}_2$ on line segment $\overline{\mathbf{p}_2\mathbf{p}_3}$ has $\zeta_1(\mathbf{s}_2)=\zeta_3(\mathbf{s}_2)=1$, and $\zeta_2(\mathbf{s}_2)=\zeta_4(\mathbf{s}_2)=0$ which leads to $\zeta_{1234}(\mathbf{s}_2)=1$ indicating *interference* of the two solids.

The problem of collision detection of solids is reduced to one of determining whether a segment of a curve (represented by a point ***s***) is inside entity boundaries. To solve this problem, [6] suggested casting a ray from the point ***s*** to infinity in the plane. The ray intersects the boundaries of the so-called *face* plane (Fig. 6a depicts a face having a discontinuity). If the ray intersects the edges an odd number of times then the point ***s*** is inside the face boundaries (Fig. 6b). If the ray intersects the edges an even number of times then the point is outside. This method has many deficiencies and is difficult to implement on a computer. It necessitates breaking the area into simpler polygonal surfaces and thus eliminating discontinuities on the surface (the so-called *bridging* method).

3.1.2 Multiple Curve Boundaries

The method presented in this paper has no limitations on the complexity of the bounding curve. i.e., the curve **c** need not be convex nor simply connected. If the boundary is composed of a multiple of curves, then the region is segmented into a number of entities. Figure 7 depicts a type I entity comprising a region enclosed by a multiple of curves.

The region may be segmented into a polygon described as $\mathbf{p}_i$, $(i=1,\ldots,5)$ and a sector enclosed by the arc $\mathbf{p}_1\mathbf{p}_5$ and the line segment $\overline{\mathbf{p}_1\mathbf{p}_5}$ on the plane Ω. In the case of the polygonal boundary, the circulation (Eqs. 9 and 10) become a summation of the variation of angle as the vector $\mathbf{p}_i-\mathbf{s}$ is sweeping vertices on the boundary curve. The indicator is

$$\zeta(\mathbf{s})=\frac{1}{2\pi}\sum_{i=1}^{v-1}\tan^{-1}\left(\frac{\left\|(\mathbf{p}_i-\mathbf{s})\times(\mathbf{p}_{i+1}-\mathbf{s})\right\|}{(\mathbf{p}_i-\mathbf{s})\cdot(\mathbf{p}_{i+1}-\mathbf{s})}\right)=\left\{\begin{array}{l}1\ \text{point inside}\\ 0\ \text{point outside}\end{array}\right\} \qquad (15)$$

where *v* is the total number of vertices. To determine whether the point **s** is inside the sector, Eq. 9 may be evaluated. However, it is more efficient to calculate two angles β_1 (the angle between the vector $\mathbf{s}-\mathbf{c}$ and the vector $\mathbf{p}_1-\mathbf{c}$) and the angle β_2 (the angle between the vector $\mathbf{p}_1-\mathbf{c}$ and $\mathbf{p}_5-\mathbf{c}$) such that

$$\beta_1=\tan^{-1}\frac{\left\|(\mathbf{s}-\mathbf{c})\times(\mathbf{p}_1-\mathbf{c})\right\|}{(\mathbf{s}-\mathbf{c})\cdot(\mathbf{p}_1-\mathbf{c})} \quad \text{and} \quad \beta_2=\tan^{-1}\frac{\left\|(\mathbf{p}_5-\mathbf{c})\times(\mathbf{p}_5-\mathbf{c})\right\|}{(\mathbf{p}_5-\mathbf{c})\cdot(\mathbf{p}_5-\mathbf{c})} \qquad (16)$$

For a point **s** to be inside the sector, two inequality constraints have to be satisfied

$$0<\beta_1<\beta_2 \quad \text{and} \quad |\mathbf{s}-\mathbf{c}|<|\mathbf{p}_1-\mathbf{c}| \qquad (17)$$

The computational complexity associated with evaluating Eq. 15 is discussed in Sec. 4.

3.2 Interference Detection Between Type I Entities

Figure 8a depicts entities enveloping a mechanical part shown in Fig. 8b. The mechanical part has collided with a second body B where only entity 11 is shown from solid B . Note that negative entity 3 is a discontinuity of type I entity, and that the surface of the groove is enveloped by entities 4,5,6, and 7. Let ${}^A\mathbf{x}_i$ be the *i*th surface enveloping solid A resolved in the A-coordinate system,

and ${}^{B}\mathbf{x}_j$ be the *j*th entity enveloping solid B resolved in the B-coordinate system, both shown in Fig. 8b.

3.2.1 General Procedure

For a general manipulator comprising *n*-links, the general algorithm for detecting collision between the solids is as follows.

Initiate a time loop, wait for next control interval, time $t = t + \Delta t$, and identify a position vector $\mathbf{h}(t)$ where the manipulator should be at time *t*. The joint solution corresponding to $\mathbf{h}(t)$ is $\mathbf{q}[\mathbf{h}(t)]$ where $\boldsymbol{q}$ is the vector of generalized coordinates.

Consider the set of solids A, numbered 1 through *n* and a second set B, numbered 1 through *m*. The two coordinate systems are related by the (3×3) homogeneous rotation matrix relating ${}^{A}\mathbf{R}_{B}(\mathbf{q})$, and the (3×1) position vector ${}^{A}\mathbf{P}_{Bo}$. Also consider that body 1 is composed of k surfaces, while body 2 is composed of L surfaces.

Let the variable u' be incremented such that $u'_p = u'_1 + \Delta u'$ until $u'_p = u'_2$, where p is a dummy index. Collision occurs if and only if there exists a solution to

$$\Phi = \begin{bmatrix} {}^{A}\mathbf{x}_i(u,v) - {}^{A}\mathbf{R}_{B}(\mathbf{q}[\mathbf{h}(t)])\,{}^{B}\mathbf{x}_j(u'_p, v') + {}^{A}\mathbf{p}_{Bo} \\ u - a_1 - a_2 \sin\lambda_1 \\ v - b_1 - b_2 \sin\lambda_2 \\ v' - c_1 - c_2 \sin\lambda_3 \\ 1 - \prod_{k=1}^{m} \zeta_k \prod_{l=m+1}^{n} (1-\zeta_l) \end{bmatrix} = \mathbf{0} \qquad (18)$$

where λ_1, λ_2, and λ_3 are as defined in Eq. 7. To demonstrate this algorithm, consider the interference between entity 11 and entity 8 of Fig. 8a. Entity 8, a catenoid enveloping solid A, is parametrized as ${}^{A}\mathbf{x}_8 = [a\cosh v\cos u, a\cosh v\sin u, av]^T$ constrained as $U = \{(u,v) \in R^2; 0 < u < 2\pi, 0 < v < d\}$, while entity 10, a cylindrical surface enveloping body B is parametrized as ${}^{B}\mathbf{x}_{11} = [b\cos u', b\sin u', v']^T$ and constrained as $U = \{(u',v') \in R^2; (5\pi/4) < u' < 2\pi, 0 < v' < e\}$.

The transformation of coordinates between solid A and B is described by

$${}^{A}\mathbf{R}_{B} = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \text{ and } {}^{A}\mathbf{P}_{Bo} = \begin{bmatrix} C & 0 & 0 \end{bmatrix}^T \qquad (19)$$

The constraint matrix Φ describing the collision (Eq. 18) is

$$\Phi = \begin{bmatrix} a\cosh v\cos u + v' - C \\ a\cosh v\sin u - b\sin u'_p \\ av - b\cos u'_p - d \\ u - \pi - \pi\sin\lambda_1 \\ v - \frac{d}{2} - \frac{d}{2}\sin\lambda_2 \\ v' - \frac{e}{2} - \frac{e}{2}\sin\lambda_3 \\ \zeta_2\zeta_{10}(\zeta_3 - 1) - 1 \end{bmatrix} = \mathbf{0} \qquad (20)$$

If a solution to Φ *exists* then *interference occurs* and collision is detected.

3.2.2 An Efficient Approach to Detecting Intersection

A more efficient approach to the problem of detection between the pair of entities $\mathbf{x}_1(u,v)$ and $\mathbf{x}_2(s,t)$, bound by curves characterized by parametric inequality constraints imposed on the variables in the form of $u_1 = a_1 + b_1\sin\lambda_1$, is to write the constraint in the form

$$\mathbf{G}(\mathbf{y}) = \begin{bmatrix} \mathbf{x}_1(u,v) - \mathbf{x}_2(s,t) \\ u_i - a_i - b_i\sin\lambda_i \end{bmatrix} = \mathbf{0} \qquad (21)$$

where the vector $\mathbf{y} = \left[u, v, s, t, \lambda_1, \lambda_2, \lambda_3, \lambda_4\right]^T$. The constraint defined by $\mathbf{G}(\mathbf{y})$ represents seven equations with eight unknowns. A solution to this problem was addressed by Abdel-Malek and Yeh [1]. Starting with an initial guess on one of the surfaces $\mathbf{y}^1$, the new generalized coordinates are calculated by evaluating $\Delta\mathbf{y} = \mathbf{G}_\mathbf{y}^*(-\mathbf{G})$ where $\mathbf{G}_\mathbf{y}^* = \mathbf{G}_\mathbf{y}^T\left(\mathbf{G}_\mathbf{y}\mathbf{G}_\mathbf{y}^T\right)^{-1}$ is the Moore-Penrose pseudo inverse of the Jacobian $\mathbf{G}_\mathbf{y} = \left[\partial G_i/\partial y_j\right]$. The new set of extended parameters is computed as $\mathbf{y}^{i+1} = \mathbf{y}^i + \Delta\mathbf{y}^i$.

3.3 Interference Detection of Type I Entities with Type II Entities

Figure 9 depicts an interference between a composite parameter surface Type I entity described by ${}^A\mathbf{x}(u,v)$ and a type II entity described by a set of vertices. The equation governing the plane is

$$(\mathbf{x} - \mathbf{p}_1)\bullet\left[(\mathbf{p}_{i+1} - \mathbf{p}_i)\times(\mathbf{p}_i - \mathbf{p}_{i-1})\right] = ax + by + cz + d = 0 \qquad (22)$$

where *i* is a constant. The equation for a composite parameter surface $\mathbf{x}(u,v)$ is

$$\mathbf{x}(u,v) = \left[UMB_x M^T V^T \quad UMB_y M^T V^T \quad UMB_z M^T V^T\right]^T \qquad (23)$$

To solve the intersection problem, substitute Eqs. (23) into Eq. (22) such that

$$UM(aB_x + bB_y + cB_z)M^T V^T + d = 0 \qquad (24)$$

Alternatively, Eq. (24) can be solved by varying the matrix *V* of parameters *v* along the parameter lines and solving the cubic function.

$$f(1, u, u^2, u^3) = UM(aB_x + bB_y + cB_z)M^T V_j^T + d = 0 \qquad (25)$$

Equation (25) and problems of this type can be solved using two steps:

(1) Compute the possible intersection of each parametric curve on the surface $\mathbf{x}(u,v)$ with $\mathbf{x}(u',v')$ the so-called hunting phase [20]. The system of equations can be expressed as

$$\mathbf{x} = {}^{A}\mathbf{x}(u, v_p) - {}^{A}\mathbf{R}_B \, {}^{B}\mathbf{x}(u', v') + {}^{A}\mathbf{P}_{Bo} = \mathbf{0} \tag{26}$$

where ${}^{A}\mathbf{R}_B$ and ${}^{A}\mathbf{P}_{Bo}$ have been previously defined. Equation 26 can be written as

$$\mathbf{r}_1(u, v_p) - \mathbf{r}_2(u', v') = \mathbf{0} \tag{27}$$

Using the Newton-Raphson method

$$u_{i+1} = u_i - \begin{vmatrix} r_x & \frac{\partial r_x}{\partial u'} & \frac{\partial r_x}{\partial v'} \\ r_y & \frac{\partial r_y}{\partial u'} & \frac{\partial r_x}{\partial v'} \\ r_z & \frac{\partial r_z}{\partial u'} & \frac{\partial r_x}{\partial v'} \end{vmatrix} \Bigg/ \begin{vmatrix} \frac{\partial r_x}{\partial u} & \frac{\partial r_x}{\partial u'} & \frac{\partial r_x}{\partial v'} \\ \frac{\partial r_y}{\partial u} & \frac{\partial r_y}{\partial u'} & \frac{\partial r_x}{\partial v'} \\ \frac{\partial r_z}{\partial u} & \frac{\partial r_z}{\partial u'} & \frac{\partial r_x}{\partial v'} \end{vmatrix} \tag{28}$$

which may be written as $$u_{i+1} = u_i - \frac{(\mathbf{r}_1 - \mathbf{r}_2) \cdot (\mathbf{r}_2^{u'} \times \mathbf{r}_2^{v'})}{r_1^{u} \cdot (\mathbf{r}_2^{u'} \times \mathbf{r}_2^{v'})} \tag{29a}$$

similarly $$u'_{i+1} = u'_i + \frac{(\mathbf{r}_1 - \mathbf{r}_2) \cdot (\mathbf{r}_2^{u} \times \mathbf{r}_2^{v'})}{r_1^{u} \cdot (\mathbf{r}_2^{u'} \times \mathbf{r}_2^{v'})} \quad \text{and} \quad v'_{i+1} = v'_i + \frac{(\mathbf{r}_1 - \mathbf{r}_2) \cdot (\mathbf{r}_2^{u'} \times \mathbf{r}_2^{u})}{r_1^{u} \cdot (\mathbf{r}_2^{u'} \times \mathbf{r}_2^{v'})} \tag{29b,c}$$

Where the derivatives are calculated at (u_i, u'_i, v'_i). The resulting intersection point can be written as ${}^{A}\mathbf{x}(u_o, v_o) = {}^{B}\mathbf{x}(u'_o, v'_o)$. In the case of surface patches, it is necessary to obtain the intersection of the second set of parameters. Intersection points lying on the parameter curves are then traced so that they can be properly interpolated to form the complete intersection curve segments across the interior regions of the parameter curve grid. This can be achieved by fixing the second set of parameters.

(2) Determine whether the set of points is inside the boundary of the type II entity

$$\zeta({}^{B}\mathbf{x}(u'_o, v'_o)) = \frac{1}{2\pi} \oint_{\mathbf{c}} d\theta = \begin{cases} 0 & \text{indicating no interference} \\ 1 & \text{indicating interference} \end{cases} \tag{30}$$

$$= \frac{1}{2\pi} \sum_i \tan^{-1} \left(\frac{\left\| \left(\mathbf{p}_i - {}^{B}\mathbf{x}(u_o, v_o) \right) \times \left(\mathbf{p}_{i+1} - {}^{B}\mathbf{x}(u_o, v_o) \right) \right\|}{\left(\mathbf{p}_i - {}^{B}\mathbf{x}(u_o, v_o) \right) \bullet \left(\mathbf{p}_{i+1} - {}^{B}\mathbf{x}(u_o, v_o) \right)} \right) \tag{31}$$

It is evident that this method, although accurate, does require a significant number of computations. The method is accurate because no approximations in representing the solid were made, and indeed, each entity bounding the solid has been checked. The complexity however, in view of other methods is addressed in the following section.

4 COMPUTATIONAL COMPLEXITY

The efficiency of the presented method has been modified with the use of the fast-converging Moore-Penrose pseudo inverse as an iterative procedure. Although still not the most efficient algorithm, the proposed formulation has proved viable for accurate detection of interference. The computational complexity of each step is addressed in terms of the total number of additions and multiplications performed by each step of the algorithm.

For the most widely used algorithms such as in the distance computation between complex objects in 3D space, the computational times were reported as nearly linear in the total number of vertices *M*, required to specify the two polytopes [10]. Many algorithms are specifically designed to achieve bounds on the form of the asymptotic computational time. For two dimensional problems,

$O(\log^2 M)$ algorithm [22] and $O(\log M)$ algorithm [8] have been reported. For 3D problems, linear computational complexity of M have been presented [10]. More recently, [15] reported an algorithm that requires $O(\log M^2)$ for testing and of order $O(\log M \log\log M)$ for sorting. The authors report that the method can be reduced to $O(\log M)$ if used on-line.

In order to delineate the work performed here and for future comparisons, the various methods presented above will be evaluated in terms of computational complexity. Of course, the number of iterations needed to converge may vary depending on the complexity of the two surfaces. In general, the pseudo inverse when used, converges within 1 to 7 iterations.

Consider two solids, each bound by a set of surfaces and numbered as $1..n$ and $1..m$, respectively. The computational complexity in terms of the total number of additions and multiplications necessary to detect collision between two solids is presented in Tables (1,2). For type II entities and for the purpose of determining the computational complexity, it is assumed that each entity is bound by one positive curve $\mathbf{c}_1(t)$, and contains a negative entity bound by $\mathbf{c}_2(t)$.

Table 1: Computational complexity for detecting interference of Type II entities

Operation	Additions	Multiplications
Intersection of $\mathbf{c}_1(t)$ with $\mathbf{x}_2(s,t)$	$(3^3/3)(n+m)$	$(3^3/3)(n+m)$
Intersection of $\mathbf{c}_2(t)$ with $\mathbf{x}_2(s,t)$	$(3^3/3)(n+m)$	$(3^3/3)(n+m)$
Intersection of $\mathbf{c}_3(t)$ with $\mathbf{x}_1(u,v)$	$(3^3/3)(n+m)$	$(3^3/3)(n+m)$
Intersection of $\mathbf{c}_4(t)$ with $\mathbf{x}_1(u,v)$	$(3^3/3)(n+m)$	$(3^3/3)(n+m)$
Calculating mid-points $(\mathbf{p}_{i+1}+\mathbf{p}_i)/2$	1	1
Computing the indicator $\zeta(\mathbf{s})$	$12v$	$13v$
The combined indicator, where n_1: number of negative entities of *n* m_1: number of negative entities of *m*	(n_1+m_1)	$(n+m)$
Total	$37(n+m)+12v(n+m)+n_1+m_1$	$38(n+m)+13v(n+m)$

For the intersection of Type I entities, we have selected the Moore-Penrose pseudo inverse method as a detection algorithm because of its efficiency. The computational complexity of the various steps can be found in [1]. The computational complexity of the detection of type I colliding with type II entities is presented in Table 2.

Table 2 Computational complexity for the interference detection of Type I and II entities

Operation	Additions	Multiplications
Evaluation the derivatives $\mathbf{r}_2^{u'}$, $\mathbf{r}_2^{v'}$, and $\mathbf{r}_1^{u}$	$15(m+n)$	$20(m+n)$
Eq. (29a)	$9(m+n)$	$16(m+n)$
Eq. (29b)	$9(m+n)$	$16(m+n)$
Eq. (29c)	$9(m+n)$	$16(m+n)$
Computing the indicator $\zeta(\mathbf{s})$ where v is the number of points computed	$12v$	$13v$
The combined indicator, where n_1, m_2 number of negative and positive entities of *n* and *m*.	(n_1+m_2)	$(n+m)$
Total	$42(m+n)+12v$ $+(n_1+m_2)$	$69(m+n)+$ $13v$

A significant difference in computational complexity is noted between the three types of surfaces used to envelope the solid. It is noted that for the Moore-Penrose method, although it is an iterative method, it does exhibit a quadratic convergence and in most cases converges within 4 to 7 iterations. In comparison with other methods that are currently used based on the computation of distances between objects, it is evident that the proposed method is very accurate. The method is accurate because no approximations are made.

However, the proposed method is of lesser performance because it does depend on checking the interference of all pairs of entities on the two links of the manipulator. This gives rise to a high number of computations as shown in Tables (1-3). Other relevant characteristics are presented in Table 3.

Table 3 Characteristics of

Characteristic	The proposed method	The method of distances between entities [7, 10]
Guaranteeing interference if one exists	No	Yes
High Performance	Less performance	Higher performance
Accuracy	Exact	Less accurate
Feasible for real-time implementation	No	Yes
Sensitive to erroneous data	Less sensitive	Yes

5 CONCLUSIONS

A broadly applicable numerical formulation for detecting collision between manipulator links are developed and illustrated. Algorithms for detecting the interference between two types of geometric entities are also presented. The representation of complex manipulator links is performed using a proposed representation scheme. It was shown that the method does not use any approximations and therefore, exhibits a high degree of accuracy, especially in the high fidelity representation of solids. Based on a computational complexity analysis, it was shown that the presented method is intensive, not suitable for real-time applications in its present form. The method, however, is well-suited for off-line planning stages, when high fidelity models of the manipulator and external environment is necessary. One of the three algorithms presented, although iterative, does present a significant efficiency due to the high rate of convergence demonstrated by the Moore-Penrose pseudo inverse. The method is implemented into an experimental computer code and integrated into a manufacturing environment. It was shown that the algorithm has accomplished the detection of collisions of a six-axis manipulator working in a cell with many obstacles represented by complex solids.

Appendix A

Proof of Eq. 9 is presented for a point **s** outside the boundary **c**(t). If there exists a potential function $\mathbf{F}$, continuous for all points inside the closed curve **c**(t), such that $\nabla \times \mathbf{F} = 0$ in the region, then $\mathbf{F}$ is the gradient of a single valued scalar function ϕ in that region $\mathbf{F} = \nabla\phi$, and the circulation of $\mathbf{F}$ around **c**(t) is zero, i.e.,

$$\oint_{\mathbf{c}} \mathbf{F} \cdot d\mathbf{r} = \oint_{\mathbf{c}} d\phi = 0 \tag{A-1}$$

Consider a potential function $\mathbf{F}$,

$$\mathbf{F} = \begin{bmatrix} -\dfrac{(\mathbf{p}-\mathbf{s}) \bullet \mathbf{j}}{|\mathbf{p}-\mathbf{s}|^2} \\ \dfrac{(\mathbf{p}-\mathbf{s}) \bullet \mathbf{i}}{|\mathbf{p}-\mathbf{s}|^2} \end{bmatrix} = \begin{bmatrix} -\dfrac{y-s_y}{(x-s_x)^2+(y-s_y)^2} \\ \dfrac{x-s_x}{(x-s_x)^2+(y-s_y)^2} \end{bmatrix} \tag{A-2}$$

F is continuous in the region bound by **c** (since the point **s** is outside), $\mathbf{p} = \begin{bmatrix} x & y \end{bmatrix}^T$ is any point on the curve **c**, and $\mathbf{s} = \begin{bmatrix} s_x & s_y \end{bmatrix}^T$. Then,

$$\mathbf{F} \bullet d\mathbf{r} = \frac{(x - s_x)dy - (y - s_y)dx}{(x - s_x)^2 + (y - s_y)^2} \tag{A-3}$$

$$= d\tan^{-1}\left(\frac{y - s_y}{x - s_x}\right) = d\varphi \tag{A-4}$$

Thus, for a closed curve ***c***, the *circulation* of **F** around **c** is zero, which indicates that the change of angle for a point outside the closed curve **c** is zero.

To prove that the change of angle for a point inside the closed curve **c** is equal to 2π, Cauchy's residue theorem is used. If the point **s** is inside the region bound by the closed curve **c**, a singularity occurs for the function **F** at the point **s.** The residue theorem [16] states that if a singularity exists, then

$$\oint_{\mathbf{c}} f(z)dz = 2\pi i \sum_{k=1}^{n} \text{Res}(a_k) \tag{A-5}$$

Where a_k are the poles of the function inside the region. Thus for the integral $\oint_c d\theta$, let

$$z - s = e^{i\theta} \tag{A-6}$$

differentiating both sides of Eq. A-6, we get

$$d\theta = \frac{dz}{i(z - s)} \tag{A-7}$$

integrating Eq. A-7,

$$\oint_{\mathbf{c}} d\theta = \oint_{\mathbf{c}} \frac{dz}{i(z - s)} = 2\pi i \left(\lim_{z \to p} \frac{z - p}{i(z - p)} \right) = 2\pi \tag{A-8}$$

i.e., for a point inside a closed region, the change of angle experienced is 2p.

REFERENCES

[1] Abdel-Malek, K. and Yeh, H. J., "On the Determination of Starting Points for Parametric Surface Intersections," 1996, *Computer-Aided Design*, Vol. 29, No. 1, pp.21-35.

[2] Abdel-Malek, K., 1993, *Off-Line Programming using Commercial CAD Systems, and Design Criteria for Intrinsically High Accuracy Robots*, PhD Dissertation, University of Pennsylvania, Philadelphia, PA.

[3] Abdel-Malek, K., 1994, "Six Axis Robot and Robot Interface" **No. 5,511,147**, US Patent and Trademark Office, Washington, D.C.

[4] Allgower, E.L. and Georg, K., 1990, *Numerical Continuation Methods: An Introduction*. Springer-Verlag, Berlin.

[5] Borrel, P. Liegeois, A., and Dombre, E., 1983, "The Robotics Facilities in the CAD-CAm CATIA System," *Developments in Robotics 1983*, IFS Publications Ltd., Kempston, Bedford, England.

[6] Boyse, J.W., 1979, "Interference Detection Among Solids and Surfaces," *Communications of ACM*, v.22, no.1, January 1979, pp. 3-9.

[7] Cameron, S., 1985, "A Study of the Clash Detection Problem in Robotics," *IEEE Proceedings of the International Conference on Robotics and Automation*, pp.488-493.

[8] Edelsbrunner, H., 1985, "On Computing the Extreme Distances Between Two Convex Polygons," *J. Algorithms*, Vol. 6, pp. 515-542.

[9] Faux, I. and Pratt, M.J., 1979, *Computational Geometry for Design and Manufacture*, Ellis Horwood, Chichester.
[10] Gilbert, E.G. 1989, "Computing the Distance Between Smooth Objects in Three-Dimensional Space," Proc. of IEEE Int. Conf. Rob. Autom., Vol. I, pp. 158-163.
[11] Glass, K., Colbauch, R., Lim, D., and Seraji, H., 1993, "On-Line Collision Avoidance for Redundant Manipulators," *Proceedings of the IEEE Internat. Conf. on Robotics and Automation*, v.1, pp. 36-42.
[12] Glass, K., Colbauch, R., Lim, D., and Seraji, H., 1995, "On-Line Collision Avoidance for Redundant Manipulators," *IEEE Transactions on Robotics and Automation, Vol. 11, No.3,* pp. 448-457.
[13] Hasegawa, et al., 1992, "Considerations for Collision Avoidance Problem of Robot Manipulator," *Journal of the Japan Society of Precision Engineering*, 58:7, pp. 1251-1256.
[14] Haug, E.J., Luh, C.M., Adkins, F., and Wang, J.Y., 1994, "Numerical Algorithms for Mapping Boundaries of Manipulator Workspaces," Proceedings of the 24th ASME Mechansims Conference.
[15] Hayward, V., Aubry, S., Foisy, A., and Ghallab, Y., 1995, "Efficient Collision Prediction Among Many Moving Objects," *Int. Journal of Robotics Research*, Vol. 14, No.2, pp. 129-143.
[16] Hildebrand, F.B., 1976, *Advanced Calculus For Applications*, Prentice Hall, Englewood Cliffs, NJ.
[17] Meyer, W., 1986, "Distances Between Boxes: Applications to Collision Detection and Clipping," *IEEE International Conference on Robotics and Automation*, pp.597-602.
[18] Mitsi, S., Bouzakis, K., 1993, "Simulation of Redundant Manipulators for Collision Avoidance in Manufacturing and Assembly Environments," *Mechanism and Machine Theory*, v.28(1), pp. 13-21.
[19] Pasarello, C.E., 1982, "Interference Detection Using Barycentric Coordinates," *Mechanics Research Communications* (UK), v.9, no.6, pp.373-378.
[20] Qiulin, D., and Davies, B.J., 1987, Surface Engineering Geometry for Computer-Aided Design and Manufactureing, Ellis Horwood, Chichester, U.K.
[21] Requicha, A.A., 1980, "Representations for Rigid Solids: Theory, Methods, and Systems," *Computing Surveys*, v.12, no.14, p.437.
[22] Schwartz, J.T., 1981, "Finding the Minimum Distance Between Two Convex Polygons," *Inform. Process. Lett.*, Vol. 13, pp. 168-170.
[23] Seitz, B. and Cipra, R.J., 1992, "Real-time Collision Avoidance of a Planar Manipulator with an Interfering Single Link Arm," *Proceedings of the IEEE International Conference on Robotics and Automation*, v.2, pp. 1494-1499.
[24] Udupa, S.M., 1977, "Collision Detection and Avoidance in Computer Controlled Manipulators," *Proceedings of the 5th IJCAI-Robotics*, pp. 737-748.
[25] Wang, D., and Hamam, Y., 1992, "Optimal Trajectory Planning of Manipulators with Collision Detection and Avoidance," *Int. J. of Robotics Research*, v. 11, n. 5, pp. 460-468.
[26] Zghal, H., et al., 1992, "Collision Avoidance of a Multiple Degree of Redundandancy Manipulator Operating Through a Window," *J. of Dynamic Systems*, v.114, no.4, pp. 717-721.

Figures

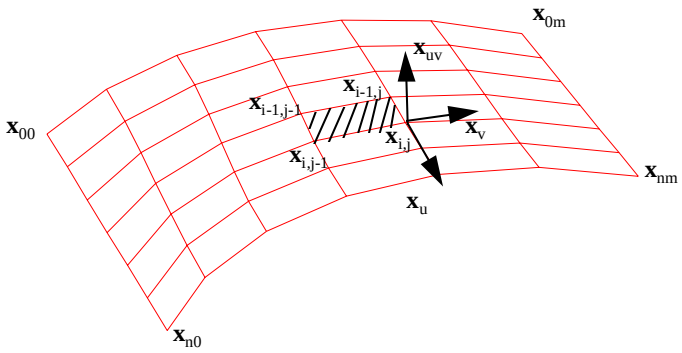

Figure 1 A surface patch

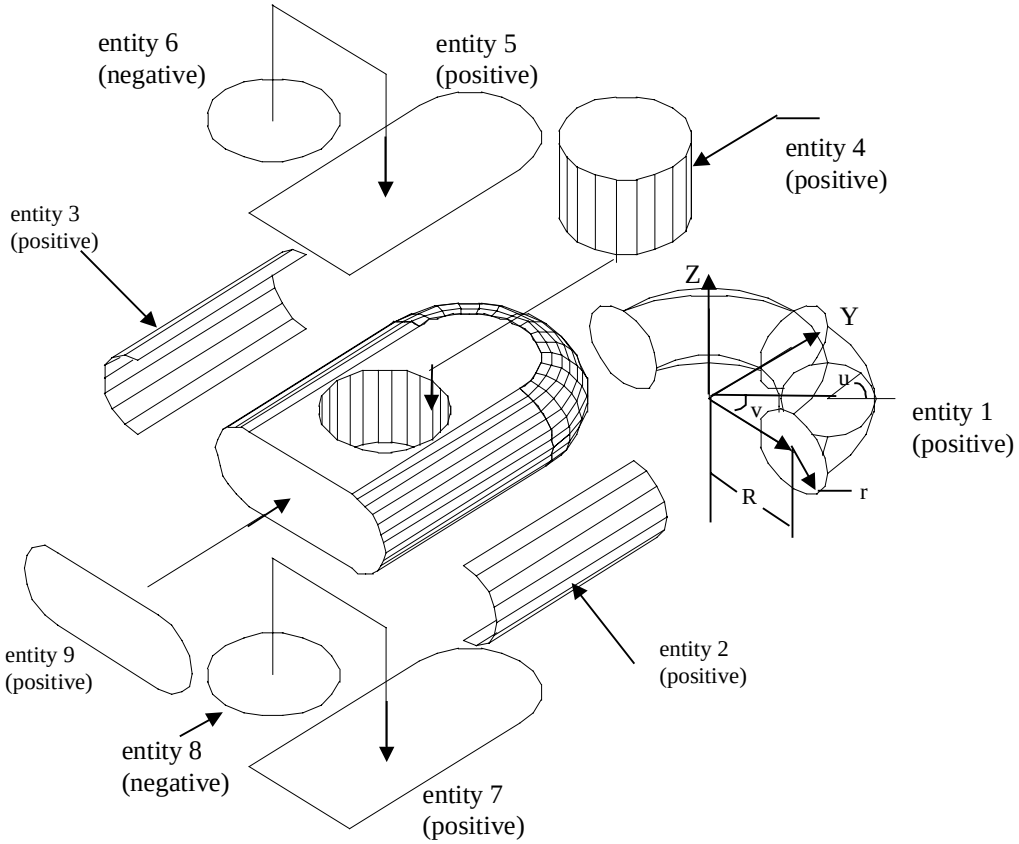

Figure 2 Indicating entities on a mechanical part

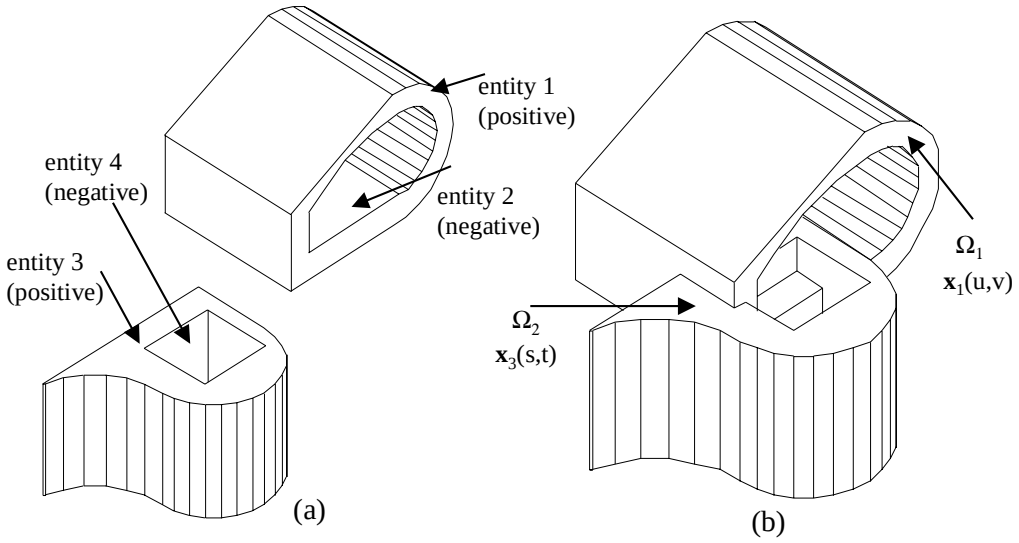

Figure 3 (a) Two solids having positive and negative entities
(b) Two solids in interference

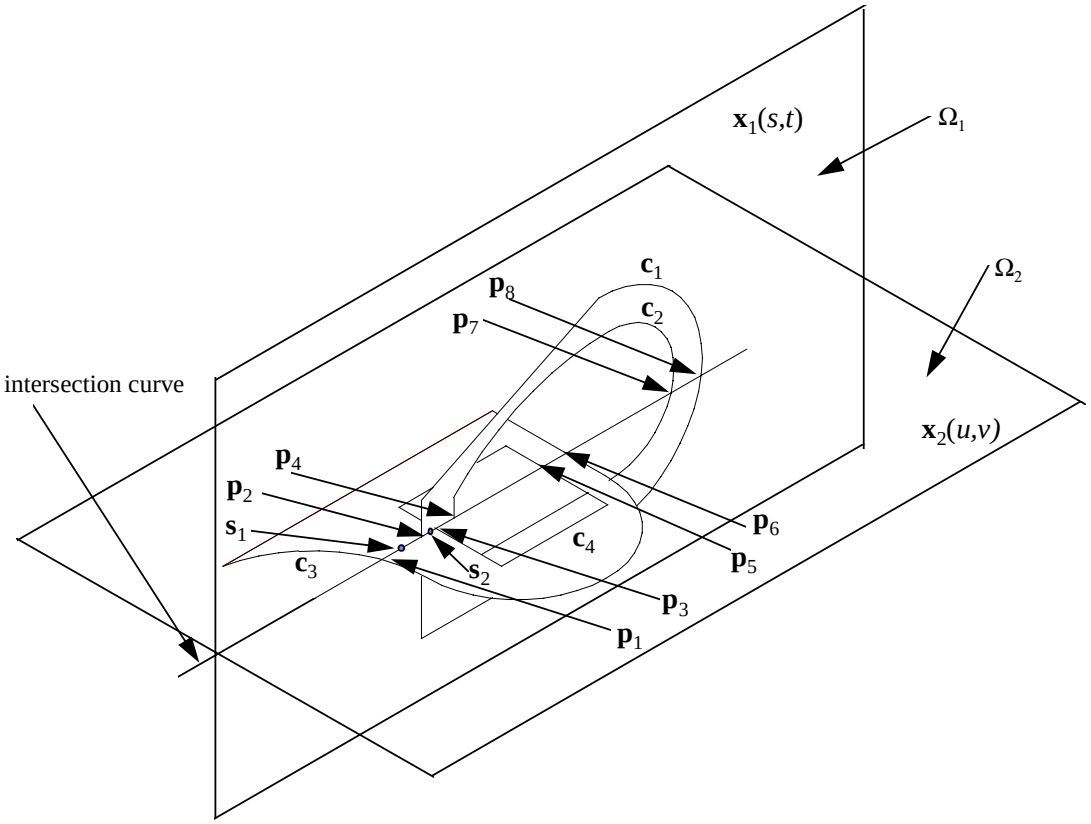

Figure 4 Determining intersection points

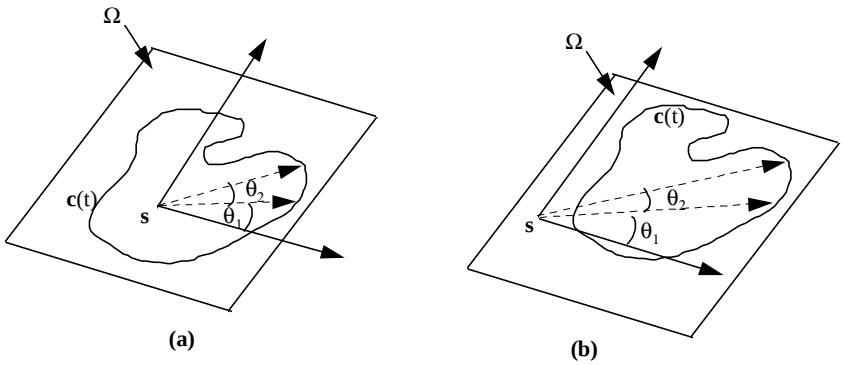

Figure 5 A vector sweeping the boundary with center at ***s ;*** (a) inside, and (b) outside the boundary

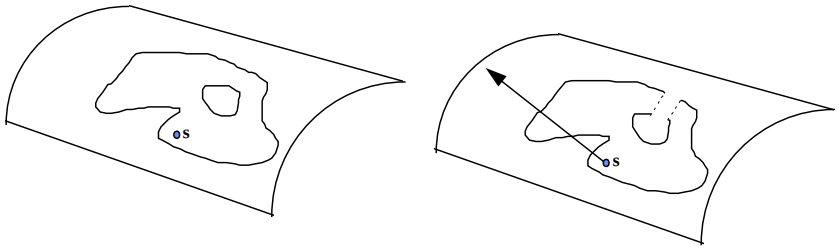

Figure 6 (a) an entity having a discontinuity, (b) the bridging method and casting a ray

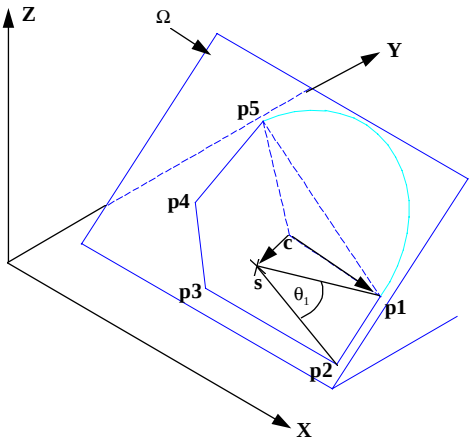

Figure 7 An entity comprising a polygon and a sector

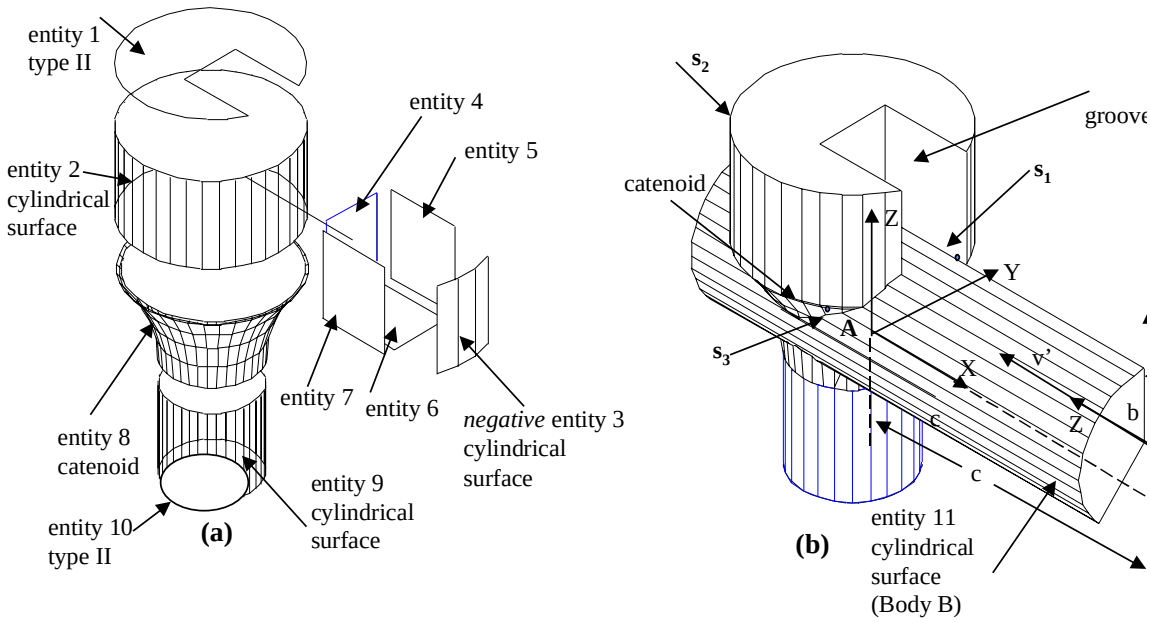

Figure 8 (a) A mechanical part enveloped by entities (b) Collision detection of type I entities

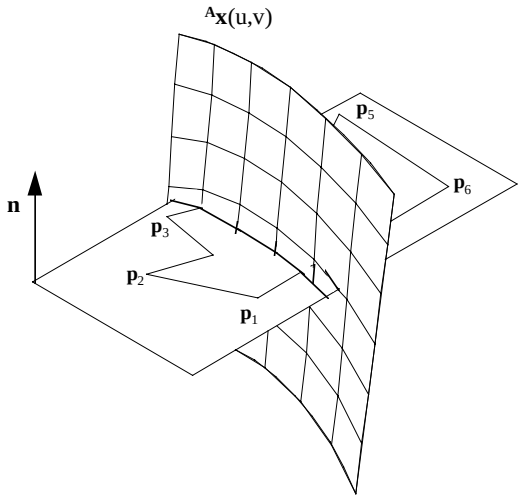

Figure 9 Type I entity in interference with type II entity.

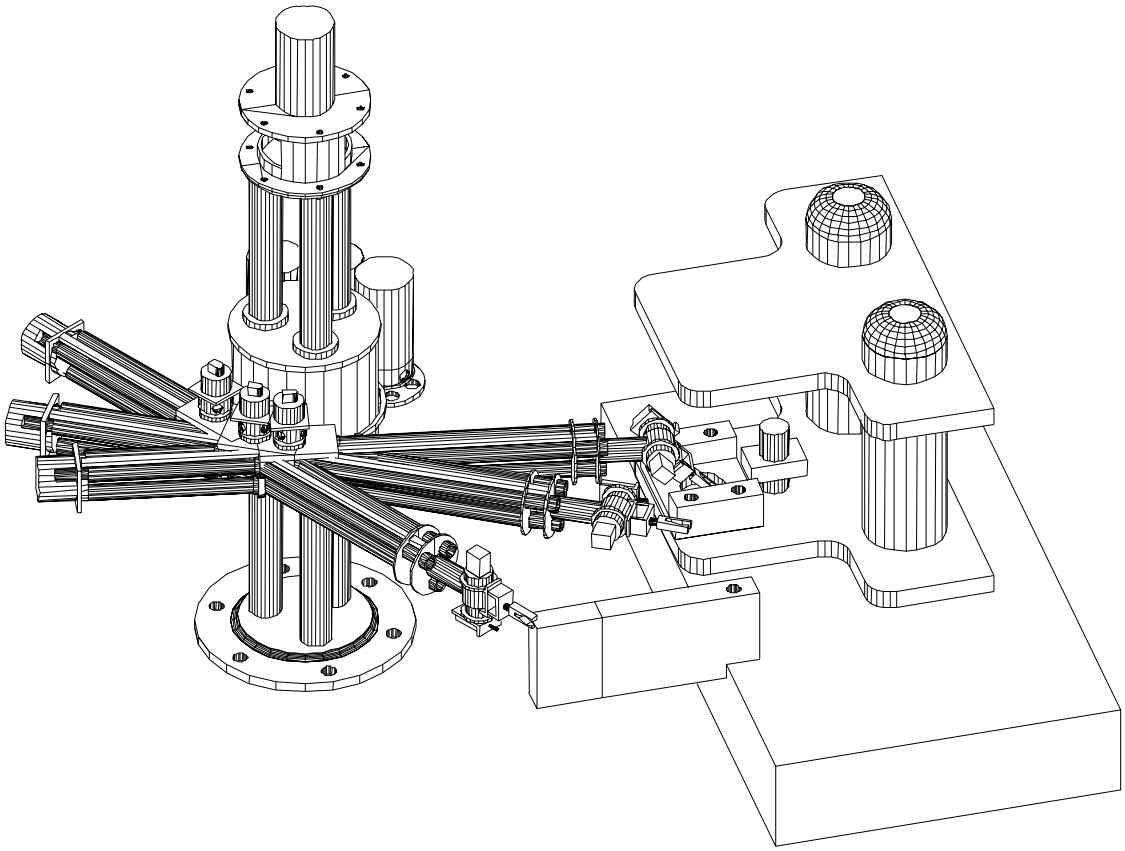

Figure 10 Detecting collision of a manipulator arm working in close proximity of a dieset

Biography

Karim Abdel-Malek is an assistant professor of Mechanical engineering at the University of Iowa. He received his BS degree in ME from the University of Jordan, his MS and PhD from the University of Pennsylvania, both in mechanical engineering. He was a Fulbright scholar for two years at the University of Pennsylvania. Dr. Abdel-Malek consulted for the US manufacturing industry for a number of years before joining the faculty at Iowa. He was on leave for two years from Iowa 96-98. His research interests include mathematical formulations for CAD/CAM systems, robotics, dynamics, and machine design.

Acknowledgements
This research was funded by a grant from the Good Samaritan Hospital in Kearney, Nebraska.