COMPUTER-AIDED DESIGN & APPLICATIONS, 2016
VOL. 13, NO. 1, 77–85
http://dx.doi.org/10.1080/16864360.2015.1059198

Simulation of a robot machining system based on heterogeneous-resolution representation

Yonghua Chen and Ying Wei

The University of Hong Kong

ABSTRACT
Collision avoidance is a frequently encountered problem in machining processes, especially in robot-based machining. In a robot machining system, collision may occur between the robot arm, the tool, tool holder and the work piece together with its fixtures. Therefore, a precise collision detection algorithm is critical to ensure that the tool path is collision free, particularly in the area where the tool has contact with the work piece. To verify tool paths, a simulation system is developed. In the proposed system, the robotic system is modeled as a Constructive Solid Geometry (CSG) tree where the Denavit-Hartenberg (D-H) notation is applied to represent the robot arm transformation matrix. A combined CSG representation and STereoLithography (STL) representation, the so called heterogeneous-resolution method is proposed to represent the work piece. By heterogeneous-resolution, the area of the work piece near the current machining contact point is represented by triangular facets at a controlled accuracy whereas other parts of the work piece are described by grid height array (GHA) to save computation time in order for the simulation to have less latency. When a collision is detected for a given cutter contact point, both the cutter location and the robot arm position are modified. The proposed method is implemented into a simulation software where the feasibility of the algorithm is tested and verified.

KEYWORDS
robot machining; path planning; collision detection; simulation

1. Introduction

In modern industries, robotic arms have been widely used not only in their traditional applications areas such as pick and place, welding, etc., but also used in large part machining and grinding [7]. In any robotic applications, path planning and simulation is very important as it determines if an expected task can be done satisfactorily and safely or not [20]. A major part of robotic simulation is the detection and avoidance of collision. Collision can be described as detecting the intersection among objects. The question received substantial attention in the last decade, and various classes of algorithms had been developed. Collision detection at discrete time instance/position is a practical method which had been used in many collision detection systems. The basic idea is to model an object in each position at a discrete time instance/position instance, than calculate the intersection between/among objects. The most popular method is by using B-Rep to describe the objects that need to be checked, and then verifying if the edges of an object piercing into another object at discrete time instance. Some improved methods use Spatial-Occupancy Enumeration (SOE) to address the problem of reducing the number of pairs of objects or primitives that need to be checked. Octree [15], k-d tree [21], C-tree [18], CSG-tree, R-tree and its variant [17], BSP-tree, Boxtree [5], OBBtrees [12], B-Rep indices [19], tetrahedral meshes [13] and regular grids [11] are examples of SOE methods to solving this problem. In the typical Octree, the space is divided into 8 sub-spaces with each one recursively subdivided further when needed. All the faces/objects that are in one sub-space don't intersect with any other faces/objects with the other 7 sub-spaces. Then, faces/objects need to be checked is reduced effectively. OBBtree is more efficient than any other hierarchical trees because of its tightest bounding box. In OBBtree, instead of using the axis-aligned rectangular box, it makes use of statistical techniques to analysis the distribution of vertices in space. Then a tight fitting oriented bounding box is yielded. The tight box can reduce calculation and detect collision more accurate. More recently, Balasubramaniam et al. [4] introduced a new precise method named vision-based collision detection for detecting collision in 5-axis machining. In their work, visibility is used instead of accessibility for verifying tool path for rough machining. At each discrete position along the tool path, if the tool can "see" the point to be machined, that means there isn't collision.

CONTACT Yonghua Chen yhchen@hku.hk

In other methods, Boundary Volumes is also a popular method in collision detection [14–16]. A commercial collision detection software I-COLLIDE [8], with the method of “sweep and prune”, is often applied in collision detection. Sweep volume [1] is another representation for collision detection, the idea is that when an object moves along a given line or curve, the maximum space is the swept volume in a given time interval. Adding time as a dimension, Cameron [6] developed a four-dimension collision detection method based on sweep volume. Based on the four-dimension method, a vertex algorithm was developed by Aliyu [3] for detecting collision between two simple objects.

In general, most of the existing collision detection methods work well for some special case, and the accuracy in representing an object isn’t high enough and it is much lower than the requirement of machining processes. In a machining process, since the tool is moved to a area very close the work piece, a high accuracy such as 0.01 mm is essential in collision detection between a ball end tool and the machined surface. This accuracy requires both a fast and accurate collision detection method. In the proposed system, the work-piece is represented as an STL file (this is nowadays very common as additive manufacturing technologies are in wide spread use). The accuracy of part representation can be controlled by the number of triangles. When a tool approaches the part model, triangular facets that are close enough to it are identified and converted to a Height Grid Array (HGA) representation (That is, the part surfaces under current machining are represented as HGA. This is why the representation is called heterogeneous-resolution). With the CSG representation of the robotic arm, the tool, the tool holder and the heterogeneous-resolution represented work piece, collision detection algorithm is developed.

2. System overview

The collision detection developed in this paper is applied to a robot machining system for large scale object rapid prototyping. The robotic system consists of an ABB IRB1400 articulated robot with six-degree-of-freedom mounted on a two-meter long linear track as shown in Fig. 1. With this configuration, the robot can cover a working envelope of 4M (Length) x 2M (Width) x 2M (Height). Seveal fixtures are installed in the working platform for holding work-pieces, such as, a clamp, a rotary table, etc. In the current system, collision might occur among the robot arm, the tool holder, and the work piece. Accurate collision detection is required especially for the tool and the work piece since they are very close in the machining process.

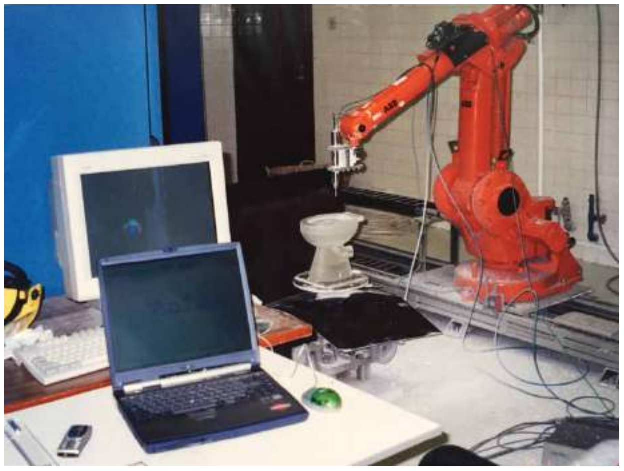
Figure 1. Robot and its cutting tool.

The Robot and its manipulator are active parts in the robot machining system. In the system, a robot, a high-speed spindle and a ball end mill cutter are used. Their shapes are very simple, can be described by cylinders, boxes and spheres in CSG representation.

To describe the robot arm kinematics, many rotation algorithms had been applied and Denavit-Hartenberg [9] notation and its derivations [2] are the most popular. D-H notation is widely used in the transformation of coordinate systems of linkages and robot mechanisms. It can be used to represent the transformation matrix between links. With D-H notation, inverse problem can be solved easily [10] and the robot arm positions can be calculated with a given position and orientation of the manipulator.

The D-H Notation allows the description of a generic robot arm with 4 parameters for each link. With these parameters, a coordinate system is attached to each link. By convention, all coordinate systems (or frames) follow the right hand rule. Fig. 2a shows the D-H coordinate system of the ABB IRB 1400 robot that was used in the machining system. (x_0, y_0, z_0)means the base coordinate system. (x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3), (x_4, y_4, z_4), (x_5, y_5, z_5), (x_6, y_6, z_6) represent the coordinate system each joint of the robot. A complete list of D-H parameters of each joint of the robot and the tool is given in Table 1. In the table, referring to Fig. 2b, θ_i is the joint angle from the x_{i-1} axis to the x_i axis about the z_{i-1} axis (using the right-hand rule), d_i is the distance from the origin of the $(i–1)$th coordinate frame to the intersection of the z_{i-1} axis with the x_i axis along the z_{i-1} axis, a_i is the offset distance from the intersection of the z_{i-1} axis with the x_i axis to the origin of the i th frame along the x_i axis (or the shortest distance between the z_{i-1} axis and z_i axes), α_i is the offset angle from the z_{i-1} axis to the z_i axis about the x_i axis (using the right-hand rule). The maximum and minimum angles show the range of each joint.

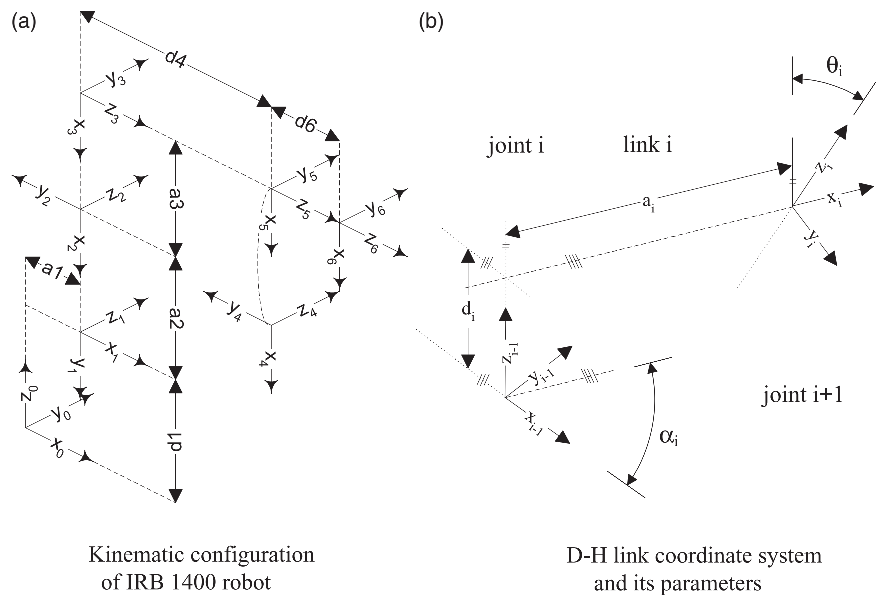
(a)
(b)
d4
d6
a3
a2
a1
d1
joint i
link i
joint i+1
θ_i
a_i
d_i
α_i
Kinematic configuration of IRB 1400 robot
D-H link coordinate system and its parameters

Figure 2. Kinematic configuration of the robot.

Table 1. D-H parameters of the robotic joints.

Joint No. i	α_i	a_i	d_i	θ_i	Min Angle	Max Angle
1	−90	150	475	0	−170	170
2	0	−600	0	0	−70	70
3	90	−120	0	0	−65	70
4	−90	0	720	0	−150	150
5	90	0	0	0	−115	115
6	0	0	85	0	−300	300
Tool	0	100	200	0	–	–

Based on these parameters, a robot link matrix can be generated. The D-H transformation matrix for adjacent coordinate frames is known as

$$^{i-1}A_i(\theta_i)_{i=1,2,3,4,5,6,7} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

where $^{i-1}A_i(\theta_i)_{n=1,2,3,4,5,6}$ means the link matrix of each two adjacent coordinate systems. When $i = 7$, $^6A_7(\theta_7)$ means the link from the sixth joint to the end of the tool and θ_7 is constant after installing the tool.

The homogeneous matrix $^0T_7$ which specifying the location of the end of the tool with respect to the base coordinate system is the chain product of successive coordinate transformation matrices of $^{i-1}A_i(\theta_i)$, and is expressed as

$$^0T_7 = {}^0A_1{}^1A_2\cdots{}^6A_7 = \prod_{i=1}^{7}{}^{i-1}A_i(\theta_i) \tag{2}$$

Given a series of points on the tool path for the manipulators, the positions and orientations of these points are described as

$$T = \begin{bmatrix} \mathbf{n}_j^x & \mathbf{s}_j^x & \mathbf{a}_j^x & P_j^x \\ \mathbf{n}_j^y & \mathbf{s}_j^y & \mathbf{a}_j^y & P_j^y \\ \mathbf{n}_j^z & \mathbf{s}_j^z & \mathbf{a}_j^z & P_j^z \\ 0 & 0 & 0 & 1 \end{bmatrix}_{j=1\cdots m} \tag{3}$$

where P means position, **n, s, a** means orientations and m is the number of positions. This inverse problem can be solved at each position and proper parameters θ_1,θ_2, θ_3, θ_4, θ_5 and θ_6 cold be found. With the solution, the positions of each link (arm) of the robot and the tool can be calculated in R^3.

In the machining process, tool and tool holder will move to a very close area of the work piece, so they must be accurately modeled. Given the tool center point, the ball end of the cutter can be represented as:

$$K_{sphere} = \{\mathbf{p}|\ |\mathbf{p} - \mathbf{p}_o| = R\} \tag{4}$$

where $\mathbf{p_o}$ is vector from the original point to the center of the sphere and $\mathbf{p}, \mathbf{p}_o \in \mathcal{R}^3$

The shaft of the tool, the spindle, and the robot arm are cylinders. A cylinder with base point $\vec{p}_o$, radius R, length L and orientation $\mathbf{n}$ in R^3 can be represented as:

$$K_{sphere} = \{\mathbf{p}|\ |\mathbf{p} - \lambda Ln - \mathbf{p}_o| = R : (\mathbf{p} - \lambda L\mathbf{n} - \mathbf{p}_o) \bullet \mathbf{n} = 0, \lambda \in (0,1)\} \tag{5}$$

where $\vec{\mathbf{p}}_o, \vec{\mathbf{p}} \in \mathcal{R}^3$.

The fixture of the spindle can be described as a box, with a base point vector $\mathbf{p_o}$, length L, width W, height H and orientations **n, a, s** of three adjacent edges, it can be represented by six bounding rectangles as:

$$K_{box} = \left\{\mathbf{p}|\mathbf{p} \in K_i^p, i = 1, 2 \cdots 6\right\} \tag{6}$$

where $\mathbf{p} \in \mathcal{R}^3$,K_i^p are six bounding rectangles of the box.

By the solution of the robot transformation matrix, with the Equation (4), Equation (5) and Equation (6), the robot and the tool positions can be easily described in R^3 by simple CSG primitives for collision detection.

3. STL representation and Grid Height Array (GHA) representation

An STL file is composed of an unordered list of triangles which are described by a set of X, Y and Z coordinates in Cartesian coordinate and a normal vector which points to the outside of the model surface. For object with curved surfaces, the STL file is an approximation of the original model. The quantity and size of the triangular facets dictate how accurately the actual model is approximated.

In STL representation, a given model can be described as:

$$S = \{Tr_i|Tr_i = [V_i^1, V_i^2, V_i^3, \mathbf{n}_i]^T, i = 1 \cdots m\} \tag{7}$$

where $V_i^1, V_i^3, V_i^3 \in \mathcal{R}^3$is the vertices of a triangle, $\mathbf{n}_i$ is the normal vector of this triangle, m is the number of triangles for the model's exterior surface.

Given a CAD model as Fig. 3a, suppose the machining orientation is the inverse Z direction of Cartesian coordinate in Fig. 3, a plane perpendicular to the machining orientation is tessellated into grids based on which a HGA is constructed as in Fig. 3c. With GHA, a complex model is represented as a series of cuboids in a given orientation. Equation (8) shows the GHA representation of a solid model.

$$G = h_{m*n} = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1n} \\ h_{21} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ h_{m1} & \cdots & \cdots & h_{mn} \end{bmatrix} \tag{8}$$

where m is the number of grids in row and n is the number of grids in column, $m \times n$ is the number of grids in this GHA. The accuracy of GHA is based on grid size.

4. Heterogeneous-resolution representation

In STL representation, all geometric features are represented by triangles. Since the description of a triangle is very simple, determining if it has intersection with an object or not is very simple. But a STL file is usually consisted of tens (or hundreds) of thousands of triangles. If every triangle is checked in the calculation, it is very computation expensive. In the presented method, a method called heterogeneous-resolution is applied to increase the efficiency of collision detection.

With GHA, a surface model can be described at a given accuracy using simple CSG primitives. If there is collision between the GHA model and other parts in the working environment, collision might occur. Of course, GHA can be used as the model representation in collision detection, but in the area around the tool contact point, the accuracy of GHA is not high enough for collision detection. To be more precise in collision detection, the original STL representation is used. Since the number of the triangles in a STL file is very large, it is desirable to remove triangles that are unlikely to cause any collision at a given tool position. Here the proposed heterogeneous-resolution representation is used for this purpose. In geometric representation,

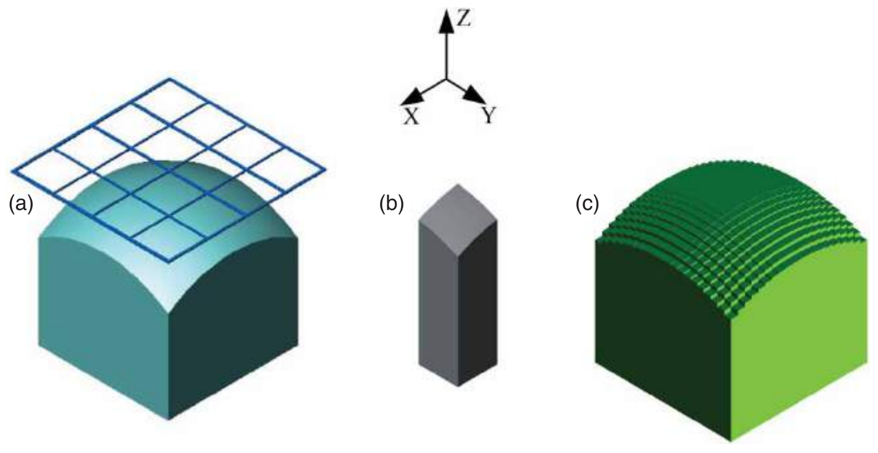

Figure 3. Grid Height Array concept.

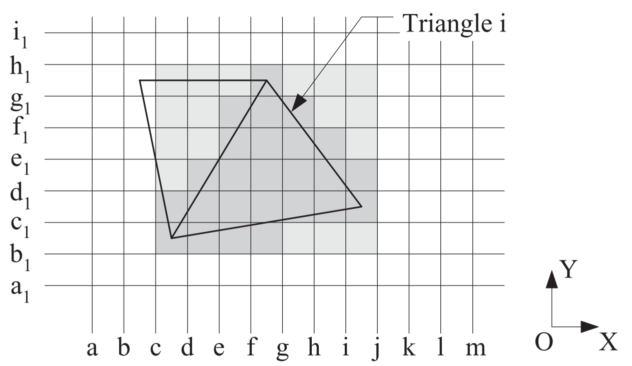

Figure 4. Projecting STL file to the tessellated plane.

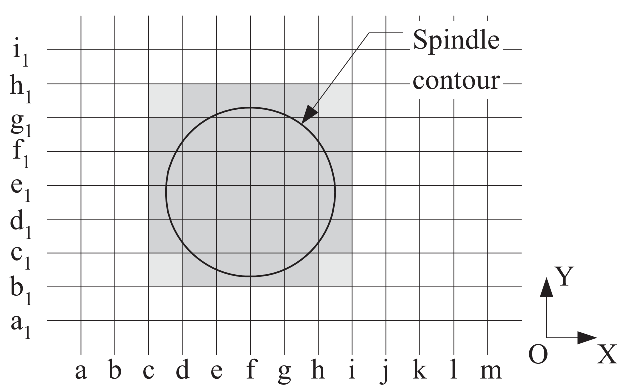

Figure 5. Projecting tool to the tessellated plane.

the heterogeneous-resolution is a hybrid CSG/STL representation. In terms of heterogeneous-resolution representation, the surface of the model close to the tool and tool holders is represented by its original triangular facets from the STL file while the rest is represented by GHA which is a much simpler representation and require much less memory space.

The key problem in heterogeneous-resolution representation is to identify the relevant triangles between the tool and the work piece. In R^3, finding the relations between the triangles and the moving tool is very difficult. In the proposed method, the relationship of the tool and the triangular surfaces is found by projecting them onto the tessellated plane of GHA, then grids on the plane are taken as a media to identify them. The following description focuses on the three steps in finding the relevant triangles:

1. Project triangles of the whole STL file onto the tessellated plane of GHA and identify the occupied grids for each triangle;
2. Project the tool onto the same plane and find the intersected grids which should be represented by triangles;
3. For each grid relevant to the tool on the projection plane, find the relevant triangles.

With the GHA representation in Equation (8), the grids set can be described as

$$G^F = \{G_{fg} \mid [V_{fg}^1,\ V_{fg}^2,\ V_{fg}^3,\ V_{fg}^4\ ] : f = 1, 2, \cdots m, g = 1, 2, \cdots n\} \tag{9}$$

Suppose the machining orientation is along **Z** axis. According to the grids applied in the GHA, projecting triangular mesh S as Equation (7–8) on the tessellated **XOY** plane. Fig. 4 shows two projected triangles in the mesh and the projected triangle set is

$$S^p = \{Tr_i | Tr_i = [V_i^1, V_i^2, V_i^3, \mathbf{n}_i]^T, i = 1 \cdots m_t\} \tag{10}$$

where $V_i \in R^2$, m_t is the number of triangles.

To find the relevant grids, the maximum and minimum **X** and **Y** value of the three vertices of the triangle are found. To triangle i, the gray area cjb_1h_1 in Fig. 5 represents the possible relevant grids as:

$$G_i^{RP} = \{{}^{rp}G_{pq}^i|[V_{pq}^1, V_{pq}^2, V_{pq}^3, V_{pq}^4] : p \le m, q \le n\} \tag{11}$$

All vertices of these possible relevant grids are checked to find if any of them are inside the triangle. Relevant grids are found by the following cases:

$$G_i{}^P = \{{}^pG_{pq}^i|[V_{pq}^1, V_{pq}^2, V_{pq}^3, V_{pq}^4] : p \le m, q \le n\} \tag{12a}$$

while $V_{pq}^1 \in Tr_i$ or $V_{pq}^2 \in Tr_i$ or $V_{pq}^3 \in Tr_i$ or $V_{pq}^4 \in Tr_i$;

$$G_i^P = G_i{}^{RP}, p \le m,\ q \le n \tag{12b}$$

where $V_{pq}^1 \notin Tr_i$ or $V_{pq}^2 \notin Tr_i$ or $V_{pq}^3 \notin Tr_i$ or $V_{pq}^4 \notin Tr_i$.

For instance, the relevant grids of triangle i are found as deep gray area in Fig. 4.

Then, the tool is projected to the same plane. Usually, the fixtures of the tool are much higher than the model surface features, so only the tool, tool shank and spindle are considered here. As Fig. 1, the spindle's radius is the largest one. Because the tool orientation is parallel to the **Z** axis of the coordinate, so the projection of the tool is a circle whose radius is the radius of the spindle as Fig. 5. Using the same algorithm as above, the relevant grids are got as the deep gray area as

$$G^{TP} = \{G_{st}^{tp} \mid [V_{st}^1,\ V_{st}^2,\ V_{st}^3, V_{st}^4\ ] : s \le mt \le n\} \tag{13}$$

With the grids selected by tool, the relevant triangles are selected. To each grid G_{st}^{tp} in G^{TP} and each triangle Tr_i in S^p,

$$K = G_{st}^{tp} \cap G_i^P, s < m, t < n \tag{14}$$

where G_i^P is the grids occupied by Tr_i. If $K \neq \Phi$, this triangles is marked and should appear in the multi-resolution model. Such as in Fig. 4, grid e_1d_1fg relates to

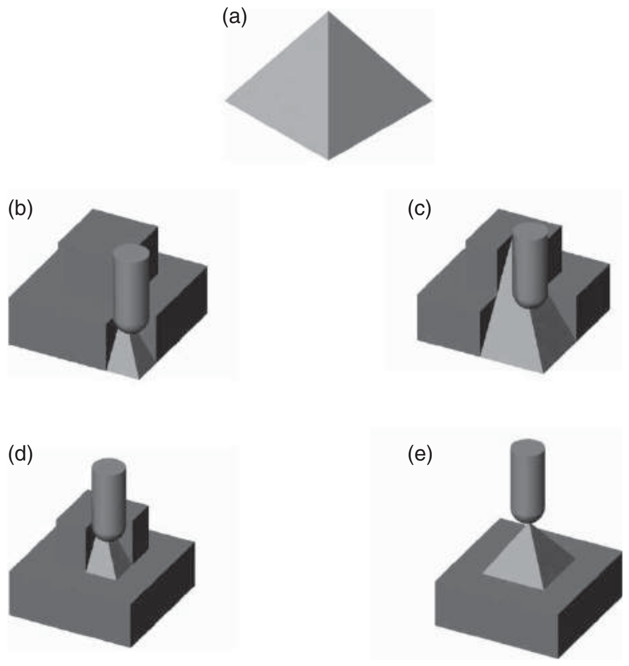

Figure 6. Multi-resolution representation.

the tool and triangle *i* in Fig. 5, so, in this grid, the work piece should be represented by triangle *i*.

By GHA and the identified triangles, the original model is represented at a heterogeneous-resolution accuracy as

$$M = \begin{bmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,n} \\ h_{2,1} & h_{2,2} & \ddots & \vdots \\ \vdots & \ddots & T & \vdots \\ h_{m,1} & \cdots & \cdots & h_{m,n} \end{bmatrix} \tag{15}$$

In this model, the concerned area T, which is composed of grids in G^{TP}, is represented by the original STL file. The rest is described by GHA, which is a series of cuboids.

Fig. 6 shows an example model by heterogeneous-resolution. In Fig. 6a, a simple STL model is shown where in Fig. 6b, c, d, e, the model is described by heterogeneous-resolution according to the tool position. In Fig. 6b, c, d, e, the area under the tool which is supposed to be the area under machining is described by triangles from STL files where the rest are represented by GHA. Thus, high accuracy (about 0.01 mm) can be easily got in the area under the tool. To the rest, detecting collision among CSG features is easy and fast.

5. Collision detection

When the tool moves to a CL/CM point on the tool path, collision may occur between the tool (including spindle and tool holder) and work piece or between the robot and the work piece. Using the above algorithm, the robot and the tool are represented by simple CSG primitives while the work piece is represented with heterogeneous-resolution representation. With these representations, the collision detection problem is simplified to the following operations:

1. Detecting collision between simple CSG primitives;
2. Detecting collision between simple CSG primitives and triangular facets from the heterogene ous-resolution represented object.

5.1. Detecting collision between simple CSG primitives

Collision detection between simple CSG primitives is described in many previous works. There are three kinds of simple CSG primitives in the robot machining system as the following

1. Sphere: The ball end of the tool;
2. Cylinder: Tool shank, spindle and robot arm;
3. Box: Tool fixture and GHA.

Since tool, tool fixture, spindle and robot arm are part of the robot, there can't be any collision among them. The collision detection problem is further simplified to:

1. Detecting collision between sphere and box;
2. Detecting collision between cylinder and box;
3. Detecting collision between box and box;

Detecting collision among simple CSG primitives can be done by solving simple equations. If there is collision between two primitives, the equations representing the primitives will have a solution or one primitive is enclosed by another. For instance, given a sphere as Equation 4 and a box as Equation 6, Equation 16 can be got as:

$$\begin{cases} |\mathbf{p} - \mathbf{p}_o| = R \\ \mathbf{p}_o \in K_i^p, i = 1, 2 \cdots 6 \end{cases} \tag{16}$$

where the first one is the sphere and the second is the six faces of a box. If a solution is got or the sphere encloses the box or the box encloses the sphere, that means there is collision.

5.2. Detecting collision between CSG primitives and triangular facets

Collision detection between simple CSG primitives and triangles can be ascribed to solving simple equations. The calculations can be further simplified by a unique feature of STL file. A STL representation provides not only the vertices of a triangle, but also the unit normal of the triangle. With the unit normal of a triangle, calculating the distance from a point to a triangle is very easy as in Fig. 7.

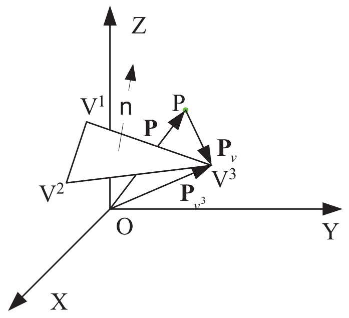

Figure 7. Calculating the distance of a point and a triangle.

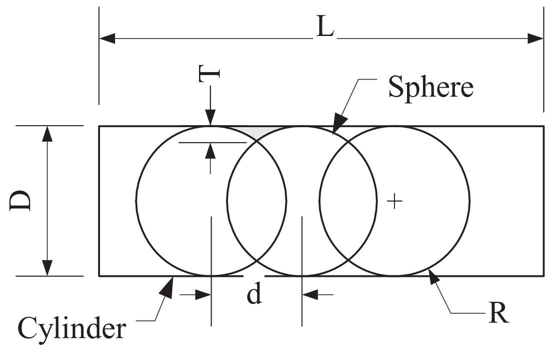

Figure 8. Representing a cylinder using spheres.

Suppose the normal of a triangle is **n**, given a vector **p** in R^3 which point to a point P, a vector $\mathbf{p}_{v^i}$ from point P to a vertex of the triangles can be defined as:

$$\mathbf{p}_v = \mathbf{p}_{v^i} - \mathbf{p} \tag{17}$$

The scalar product of vector $\mathbf{p_v}$ and the normal direction **n** is:

$$M = \mathbf{n} \bullet \mathbf{p}_v \tag{18}$$

If M is less than zero, that means the point is at the side in which **n** points to, otherwise it is at the other side. $|M|$ is the distance from the point to the triangle.

Given a sphere as Equation (4) and a triangle set as Equation (8), collision won't occur when $|M| > R$, where R is the radius of the sphere.

If $|M| < R$, collision may occur between the triangle and the sphere. The question could be summarized by detecting if the ball pierces into the triangle, or if the triangle is enclosed by the box. By solving the equations, collision is easily detected when there is an intersection.

The shape of tool shank and the spindle is cylinder. With a cylinder as Equation (5) and a triangle set as Equation (8), the collision can be detected by dividing the cylinder into a series of spheres along its centerline as Fig. 8. Given a tolerance T, the distance between two adjacent spheres is

$$d \le 2\sqrt{R^2 - (R-T)^2} \tag{19}$$

where $R = \frac{D}{2}$. Then a cylinder could be described by k spheres where

$$k = \mathrm{int}(\frac{L}{d} + 0.5) \tag{20}$$

where L is the length of the cylinder.

With these spheres, collision between the cylinder and the triangles can be easily and quickly detected by detecting collision between the spheres and the triangles.

Collision between a box and a triangle could be summarized by detecting if any edges pierces into the surface of another object or if the triangle is enclosed by the box.

5.3. Collision detection algorithm

With the above description, Algorithm 1 is developed to detecting collision in the system.

If a collision is detected, centered at the CM point, the tool is rotated at a given step of $\pi/36$ along the generation line of a cone whose apex is the CM point as Fig. 9. At this position, the projection of the tool is not a circle. Here a rectangle is defined in the projection plane as in Fig. 10 to find the relevant triangles, the width of the projected rectangle is

$$W = 2R \tag{21}$$

Algorithm 1

```
Given a CL data;
Calculate_position (robot_arm);
Calculate_position(tool);
Calculate_position(tool_fixture);
Multi_resolution(model, tool_position);
Check_collision(tool, tool_ fixture);
Check_collision(tool_fixture, robot_arm);
Check_collision(tool, model);
Check_collision(tool_shank, model);
Check_collision(Spindle, model);
```

and the length is

$$L = L_s \sin\frac{\pi}{6} + R\sin\frac{\pi}{6} \tag{22}$$

where L_s is the length of the spindle and R is the radius of the spindle.

If collision still occurs when the vertex angle of the cone is more than $\pi/4$, to avoid vibrations caused by the cutting force, the tool should retreat a given step (say by

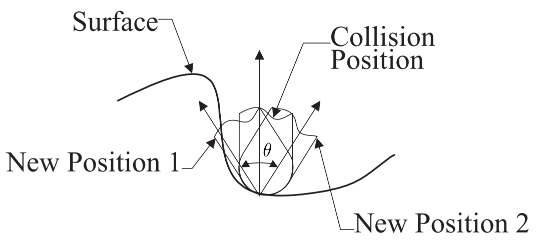

Figure 9. Rotating the tool.

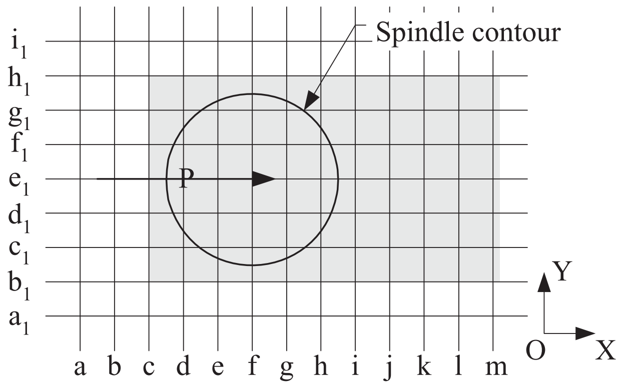

Figure 10. Inclined spindle projection.

0.1 mm), then the algorithm is repeated until no collision is detected.

6. Applications

The proposed collision detection method is based on the above heterogeneous-resolution representation of a work piece model. Compared to tradition methods, it is a simple and precise solution. Simulations are performed before the machining of the two models in the experiment. All simulation is based on a computer with Intel(R) Core(TM)2, Duo CPU E8500 @ 3.16 GHz and 8GB RAM memory. In the machining simulation of the first one, a small boat hull model (300 mm in length, 120 mm in width, and 100 mm in height) is clamped using a vise in front of the robot as in Fig. 11a. The computational time is approximately 395 seconds (without finishing cutter path simulation). In the second sample, a large boat model of 567 (Height) X150 (Width) X 250 (Length) mm is fixed on a rotary table as in Fig. 11b. With such a fixture, the robot can approach the model surface from many possible orientations. The simulation takes approximately 568 seconds (without finishing cutter path simulation). The actual machining process is shown in Fig. 12. The results of the experiments have demonstrated that the simulation and collision detection algorithms are effective for the robot machining system. In future research, GPU cards will be added to increase the graphical processing speed.

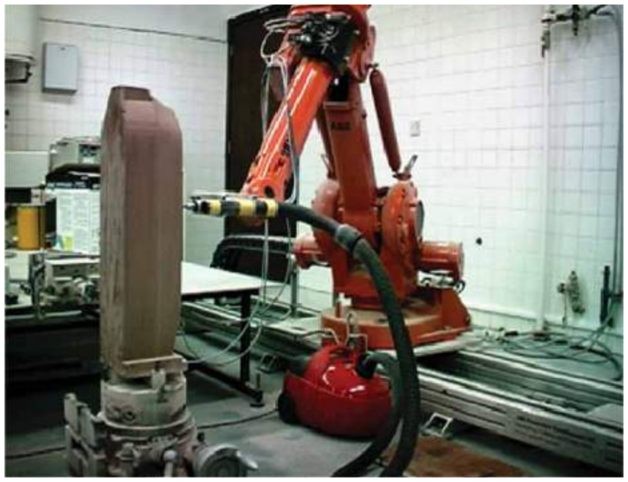

Figure 12. Actual machining process for a large ship model.

7. Conclusions

The multi-resolution based collision algorithm aims at a robot machining system for rapid prototyping. In the presented algorithm, with a given tool position and orientation on the tool path, each components on the robot is modeled with D-H notation. With the presented multi-resolution technology, the concerned area of the work

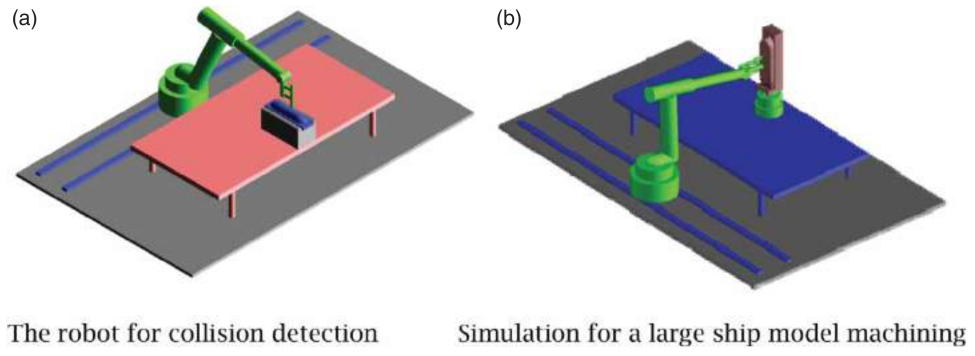
(a)
(b)
The robot for collision detection
Simulation for a large ship model machining

Figure 11. System simulation.

piece is represented by STL format at a given accuracy where the others are described by GHA. When collision is detected for a given contact point, the tool orientation is modified in order to avoid it. Comparing to the tradition method, it can get a fast and precise solution especially for two close objects. With the conducted simulations and experiments, the effectiveness of the system is verified.

Acknowledgements

The authors are grateful to a CRCG small project grant from the University of Hong Kong.

ORCID

Yonghua Chen http://orcid.org/0000-0003-4020-1977

References

[1] Abdel-Malek, K.; Yeh, H.-J.; Othman, S.: Sweep volumes: void and boundary identification, Computer-Aided Design, 30(13), 1998, 1009–1018. http://dx.doi.org/10.10/S0010-4485(98)00054-2.

[2] Abderrahikm, M.; Whittaker, A.-R.: Kinematic model identification of industrial manipulators, Robotics and Computer Integrated Manufacturing, 16, 2000, 1–8. http://dx.doi.org/10.1016/S0736-5845(99)00038-1.

[3] Aliyu, M.-D.-S.; Al-Sultan K.-S.: Fast collision detection in four-dimensional space, European Journal of Operational Research, 114, 1999, 437–445. http://dx.doi.org/10.1109/70.56661.

[4] Balasubramaniam, M.; Laxmiprasad, P.; Sarma, S; Shaikh, Z.: Generating 5-axis NC roughing paths directly from a tessellated representation, Computer-Aided Design, (32), 2000, 261–277. http://dx.doi.org/10.1016/S0010–4485(99)00103-7.

[5] Barequet, G.; Chazelle, B.; Guibas, L.-J., Mitchell, J., Tal, A.: BOXTREE: a hierarchical representation for surfaces in 3D, Computer Graphics Forum, 15(C), 22–26 August, 1996, 387–396. http://dx.doi.org/10.1111/1467-8659.1530387.

[6] Cameron, S.: Collision detection by four-dimensional intersection testing, IEEE transactions on robotics and automation, 6(3), 1990, 291–302. http://dx.doi.org/10.1109/70.56661.

[7] Chen, Y.-H.; Deng, F.-H.: Robot Machining: Recent Development and Future Research Issues, International Journal of Advanced Manufacturing Technology, 66(9–12), 2012, 1489–1497. http://dx.doi.org/10.1007/s00170-012-4433-4.

[8] Cohen, J.; Lin, M.; Manocha, D.; Ponamgi, M.: I-COLLIDE: an interactive and exact collision collision detection system for large-scale enviroment, Proceedings of ACM Interactive 3D Graphics Conference, 1995, 189–196. http://dx.doi.org/10.1145/199404.199437.

[9] Denavit, J.; Hartenberg, R.-S.: A kinematic notation for lower-pair mechanisms based on matrices, ASME Journal of Applied Mechanisms, 22(2), 1965, 215–221.

[10] Fu, K.-S.; Gonzalez, R.-C.; Lee, C.-S.-G.: Robotics: Control, Sensing, Vision, and Intelligence, McGraw-Hill Book Company, Singapore, 1987.

[11] Garcia, A.-A; Serrano, N; Flaquer, J.: Solving the collision detection problem, IEEE Computer Graphics Application, 1(4), 1994, 36–43. http://dx.doi.org/10.1109/38.279041.

[12] Gottschalk, S.; Lin, M.-C.; Manocha, D.: OBBtree: A hierarchical structure for rapid interference detection, Proceedings of SIGGRAPH'96, New Orlean, USA, 4–9 August, 1996, 171–180. http://dx.doi.org/10.1145/237170.237244.

[13] Held, M.; Klosowski, J.-T.; Mitchell, J.-S-.B.: Evaluation of collision detection methods for virtual reality fly-throughs, Proceedings of 7th Can. Conf. Comput. Geom., Quebec, Canada, 27–28 May, 1995, 205–210.

[14] Hubbard, M.: Collision detection for interactive graphics applications, IEEE Transaction of Visual Computer Graphics, 1(3), 1995, 218–230. http://dx.doi.org/10.1109/2945.466717.

[15] Noborio, H.; Fukuda, S.; Arimoto, S.: Fast interference check method using octree representation, Advanced Robotics, 3(3), 1989, 193–212. http:/dx.doi.org/10.1163/156855389X00091.

[16] Palmer, I.-J; Grimsdale, R.-L.: Collision detection for animation using sphere-trees, Computer Graphics Forum, 14(2), 1995, 105–116. http://dx.doi/org/10.1111/1467-8659.1420105.

[17] Samet H.: Spatial Data Structures: Quadtrees, Octrees, and Other Hierarchical Methods, Addison-Wesley, Reading, MA, 1989.

[18] Sellis, T.; Roussopoulos, N.; Faloutsos, C.: The R C-tree: A dynamic index for multidimensional objects, Proceedings 13th VLDB Conference, Brighton, U.K, 12–14 Spetember, 1987, 507–518.

[19] Vanecek, G. Jr.: Brep-index: a multidimensional space partitioning tree, International Journal of Computer Geometry Applications, 1(3), 1991, 243–261. http://dx.doi.org/10.1142/S0218195991000189.

[20] Vosniakos, G.; Matsas, E.: Improving feasibility of robotic milling through robot placement optimization, Robotics and Computer-integrated manufacturing, 26, 2010, 517–525. http://dx.doi.org/10.1016/j.rcim.2010.04.001.

[21] Youn, J.-H.; and Kohn, K.: Realtime collision detection for virtual reality applications, IEEE First Annual Virtual Reality Symposium, Seattle, USA, 18–22 September, 1993, 415–421. http://dx.doi.org/10.1109/VRAIS.1993.380750.