Interference-Free Polyhedral Configurations for Stacking

Venkateswara R. Ayyadevara, David A. Bourne, Kenji Shimada, and Robert H. Sturges, Jr.

***Abstract*—This paper uses a configuration space (c-space) based method to compute interference-free configuration for stacking polyhedral sheet metal parts. This work forms the interference analysis module of a stacking planner developed by us. Parts in a stack should not interfere with each other and should also satisfy stability, grasping, and stacking plan feasibility related constraints. We present two techniques to speed up the expensive step of c-space obstacle computation. The first technique identifies orientation intervals (for a convex pair of solids) within which the topology of face-edge-vertex graph of an obstacle stays the same. Within this interval, c-space obstacle geometry for one orientation can be extrapolated from obstacle geometry for another orientation. Our experiments show that extrapolation takes an order of magnitude less than the time taken to compute an obstacle from scratch. The second technique computes near optimal interference-free positions for a discrete orientation without having to compute the complete c-space obstacle. Our experiments show that, for complex sheet metal parts, less than 0.1% of the convex component pairs are evaluated in order to compute an interference-free configuration. We describe a configuration space-based method to compute a list of interference-free configurations that can be tested to see if they satisfy the above mentioned constraints. The cost function is a weighted sum of components that penalize floor space utilization and height of center of gravity of parts. The algorithm is able to pick nested stacks that tend to be stable and compact without having to explicitly enumerate features that can be nested. It is also able to accommodate flanges in holes to reduce the value of the user specified cost function. We use three test parts to illustrate the effect of the two techniques to speed up c-space obstacle computation. We also show the stacking plans generated for three different values of the weighting parameter in the cost function used by the stacking planner.**

***Index Terms*—Computer-aided process planning, configuration space, interference, sheet metal, stacking.**

I. Introduction

THERE are many applications that require the precise relative placement of pairs of complex polyhedral parts: e.g., packing, nesting, and stacking. Part stacking is especially difficult, because it combines the problem of final configuration (from packing and two-dimensional (2-D) layout domains) with stability concerns (from assembly planning). The user can provide a cost function that addresses stability and/or space utilization concerns. It is required to compute an interference-free configuration that minimizes the user-specified cost function.

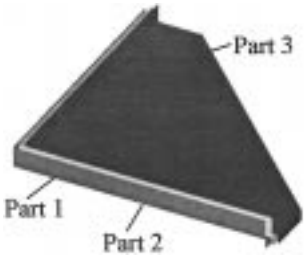


Fig. 1. A compact stack with 3-D nesting leading to compact yet stable stacks.

Optimal stacking of polyhedral parts is difficult. Even the 2-D form, i.e., optimal 2-D layout of blanks on a sheet, with no stability concerns, has been shown to be NP-hard by [1]. Hence, determination of a globally optimal part stack is virtually impossible. We have developed a stacking planner [2] that uses a "generate and test" approach to generate near optimal stacking plans. Such a planner needs tools for interference analysis and stack stability analysis. We have presented tools to evaluate stack stability in [2]. This paper focuses on another aspect of stacking: generating interference-free part configurations that minimize a user-specified cost function. This problem is also of interest to other areas such as part nesting, assembly planning, and packing.

For the purpose of stacking, there are certain qualities that have to be captured by the method we choose to compute interference-free configurations. One is the ability to perform three-dimensional (3-D) nesting of parts (see Fig. 1). This often leads to stable and compact stacks. Designers often build in features to facilitate part nesting. Trying to enumerate all features that enable nesting can be troublesome. It is preferable that the planner seeks out nested configurations. The second quality is the ability to accommodate protruding flanges in holes. This is useful for sheet metal parts. Finally, we need a mechanism to ensure clearance in the plane between parts to prevent robot positioning errors from resulting in collisions.

A. Problem Overview

This paper focuses on one aspect of planning for stacking: computation of interference-free positions and orientation for parts in a stack. The global problem we are addressing is to stack n identical parts while attempting to minimize a cost function that accounts for both part stability and floor space utilization.

Manuscript received August 7, 2000; revised September 1, 2001. This paper was approved for publication by Associate Editor M. Overmars and Editor S. Hutchinson upon evaluation of the reviewers' comments. This work was supported by Amada, Japan Inc., and Amada America.

V. R. Ayyadevara was with the Mechanical Engineering Department, Carnegie Mellon University, Pittsburgh, PA 15213 USA. He is now with Evoxis Inc., Pittsburgh, PA 15222 USA (e-mail: venkat.ayyadevara@evoxis.com).

D. A. Bourne is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: db@ri.cmu.edu).

K. Shimada is with the Mechanical Engineering Department, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: shimada@cmu.edu).

R. H. Sturges, Jr. is with the Industrial and Systems Engineering and Mechanical Engineering Departments, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061 USA (e-mail: sturges@vt.edu).

Publisher Item Identifier S 1042-296X(02)03656-X.

Of course, a stack of parts is physically realizable only when none of the parts interfere with each other.

We describe a method to apply configuration space (c-space) based techniques used for robot path planning to compute a near optimal interference-free configuration for a polyhedral part while adding it to an existing stack (Appendix A contains an introduction to c-space terminology). The final configuration has the following qualities:

1) minimizes a user-specified cost function;
2) lies inside a given space of configurations;
3) avoids interference with parts already in the stack.

The transfer of c-space based techniques from robot path planning domain to stacking domain is not straightforward. Robot path planning involves computation of an interference-free path between start and goal configurations of a robot. Both start and goal configurations are interference-free. The focus is on generating information about connectivity of the set of interference-free configurations and not the exact surfaces bounding this set. In the case of stacking, the desired configuration is often not realizable due to interference. Further, since the cost function is to be minimized, connectivity information alone for the set of interference-free configurations is insufficient. Only an exact description of the bounding surfaces of the configurations that result in interference enable the computation of the optimal interference-free configuration. The requirement in robot path planning of guaranteeing no interference at all configurations along the path does not apply to optimal part placement.

The main contributions of this paper can be summarized as follows.

1) A technique to speed up computation of the set of configurations of a rigid convex polyhedral body that result in geometric interference with another rigid convex polyhedral body. The technique involves extrapolating the set of positions that result in interference for one orientation to obtain the corresponding obstacle for another orientation. We show how to compute the interval of orientations within which this extrapolation is valid.
2) A technique to speed up computation of an optimal position (constant orientation) for a rigid concave polyhedral body such that there is no geometric interference with other rigid concave polyhedral bodies. We are able to compute the optimal interference-free position (minimizing a quadratic cost function) by only partially constructing the set of configurations that lead to geometric interference.
3) Application of c-space based techniques to planning for stacking such that nested stacks are preferred (as in Fig. 1), protruding flanges can automatically be accommodated in holes, robot positioning error is accounted for by the planner.

We first briefly describe the stacking planner. We then formulate the more specific problem of computing near-optimal interference-free configurations for a part being added to a stack. We present two techniques to speed up this computation. This is followed by a discussion of the results from using algorithms developed in this paper for interference analysis by the stacking planner. We then review previous work on computing set of configurations that result in interference between two polyhedrons and their use in domains other than stacking. Finally, we present conclusions and recommendations for future work.

II. Stacking Planner

A. Global Problem Statement

The problem addressed by the planner involves stacking n identical parts. A detailed description of the planner is given in [2]. Here, we describe a few relevant features of the planner. The cost function for the stack is the sum of costs computed for each part in the stack. The first two parts of the function penalize floor space utilization. They measure the planar distance between the center of the floor space and the vertices of the bounding box of the part. This component encourages parts to be pushed toward the point (x_0, y_0, z_0). The third component of the cost function is a measure of stability. It penalizes the height of the center of gravity of each part. For any orientation, this component encourages all parts to be placed on the floor to minimize the c.g. height. The cost function that has to be minimized is given as follows:

$$\begin{aligned} f(w) = &\sum_{i=1}^{n} \left[w(\max(X_i) - x_0)^2 + w(\min(X_i) - x_0)^2\right] \\ &+ \sum_{i=1}^{n} \left[w(\max(Y_i) - y_0)^2 + w(\min(Y_i) - y_0)^2\right] \\ &+ \sum_{i=1}^{n} \left[4(1-w)(\overline{z}_i - z_0)^2\right], \qquad w \in [0, 1] \end{aligned} \tag{1}$$

where

w	parameter that varies between 0 and 1 and is set by the user specifying the importance of space utilization relative to stability; when $w = 0$, there is no cost for space utilization and when $w = 1$, there is no cost for increase in z coordinate of c.g. of the parts constituting the stack;
n	number of parts in the stack;
(X_i, Y_i, Z_i)	set of vertices for part i;
$\overline{z}_i$	z-coordinate of center of gravity of part i;
(x_0, y_0, z_0)	center point or a corner of the designated floor space area.

Fig. 2 shows stacks of 121 cubes for different values of w. The effect of increasing the cost of space usage relative to cost of increase in stack height can be seen from the transition of the stack from a flat pattern to a single column. For intermediate values of w, the stack is a pyramid. While we have shown the whole gamut of the cost function variation, most users will opt for a value of $w = 0.9$ or less.

B. Global Problem: Approach

The planner builds the stack part by part (see Fig. 3 for an example stack with fifteen parts). Every time a new part is added to an existing stack, parts already in the stack are considered stationary. In order to generate compact stacks, the planner uses

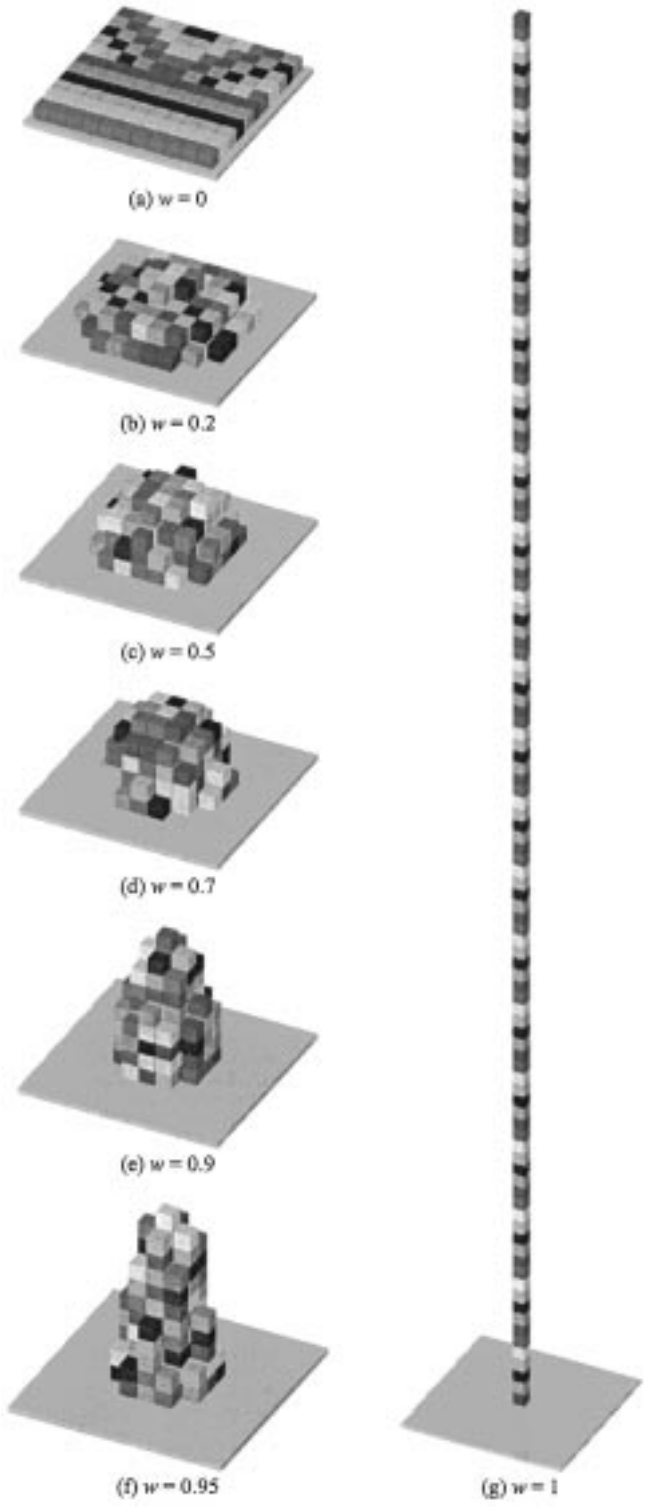


Fig. 2. Effect of increasing importance of floor space utilization relative to stack stability.

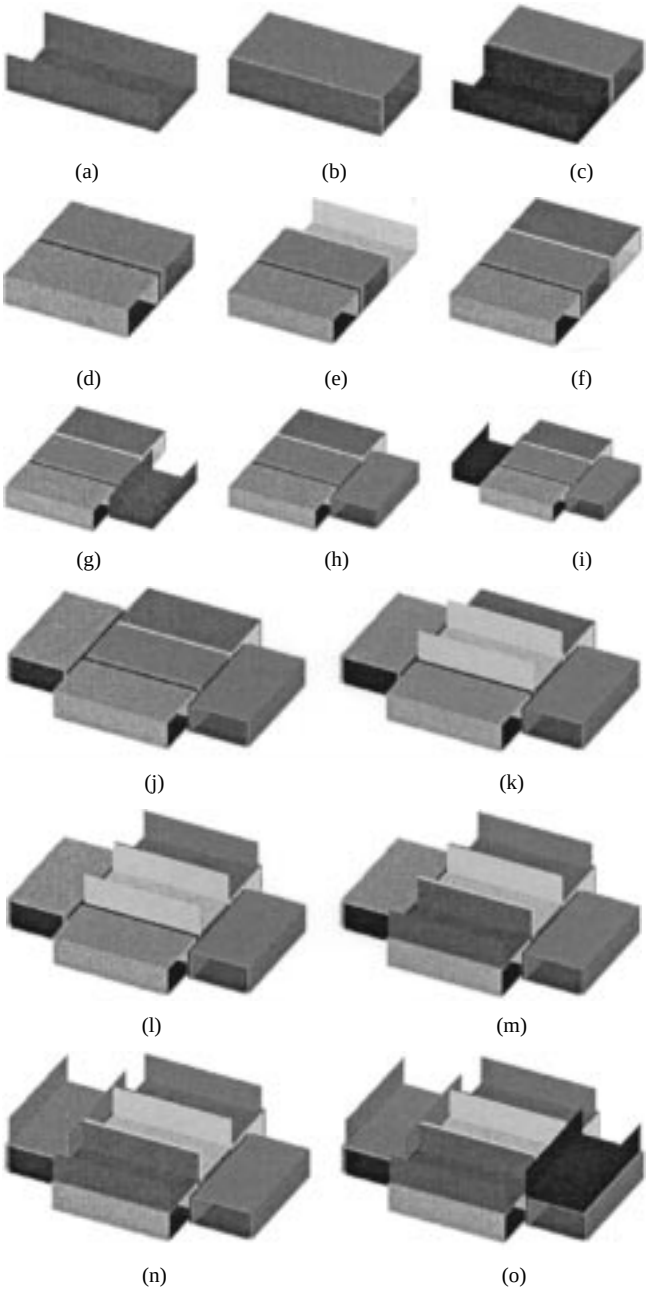


Fig. 3. Incrementally building a stack with fifteen parts: (a) 1 part, (b) 2 parts, (c) 3 parts, (d) 4 parts, (e) 5 parts, (f) 6 parts, (g) 7 parts, (h) 8 parts, (i) 9 parts, (j) 10 parts, (k) 11 parts, (l) 12 parts, (m) 13 parts, (n) 14 parts, and (o) 15 parts.

a two-stage approach to add a new part to an existing stack. In the first stage, it tries to locate the part such that the bounding box of the stack is not enlarged or enlarged by a small amount to accommodate a nested configuration. In Fig. 3, the planner is successfully able to add parts #2, 4, 6, 8, 10, and 12–15 without expanding the bounding box of the stack. If no configuration is found in the first stage, the planner tries to locate the new part without considering the bounding box of the existing stack. In Fig. 3, the second stage is required for parts #1, 3, 5, 7, 9, and 11.

Every part configuration has to satisfy the following conditions.

1) There should be no geometric interference.
2) The part should be supported by other parts and the floor such that it is stable.
3) The part should have a horizontal face that can be grasped using suction cups.
4) It is possible to add the part to the stack by translating it along $-z$ direction.

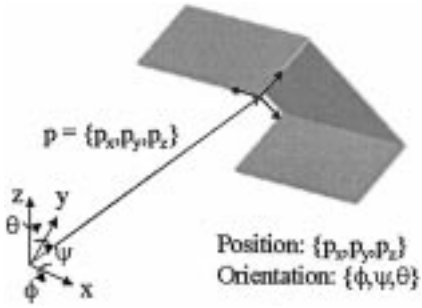


Fig. 4. Polyhedral configuration parameters.

Clearly, the part configurations have to satisfy a number of concerns other than interference avoidance. The focus of this paper is development of algorithms to generate a list of promising interference-free configurations [satisfying condition 1)] that can then be evaluated to see if they satisfy stability, grasping, and stacking plan feasibility concerns [satisfying conditions [2)–4)].

Consider adding a new part to an existing stack. We wish to compute configuration of the new part that minimizes a cost function that is convex with respect to the position of the new part. The new part once added to the stack should not interfere with other parts in the stack and should lie inside the volume obtained by extruding the allocated floor space with the maximum permissible stack height.

As shown in Fig. 4, for every new part that has to be added to an existing stack, we have to determine three position parameters $\{p_x, p_y, p_z\}$ and three orientation parameters $\{\phi, \psi, \theta\}$. Our stacking planner enumerates promising stable and graspable orientations of a part, thus choosing orientation parameters $\{\phi, \psi\}$. Let us assume for the rest of this paper that these two parameters are fixed. That leaves four configuration parameters to be determined: $\{p_x, p_y, p_z, \theta\}$.

The problem can be formulated as follows. Given

1) a new part Λ with position $\mathbf{p} \equiv \{p_x, p_y, p_z\}$ and orientation $\Phi \equiv \{\phi, \psi, \theta\}$, where ϕ and ψ are constant;
2) an existing part stack Γ;
3) a user-defined cost function $f(\mathbf{p}, \theta)$ that is convex in position parameters $\{p_x, p_y, p_z\}$. The present planner uses the cost function from (1), and
4) volume obtained by extruding the floor space by the maximum permissible stack height: Σ;

compute the configuration $Q^* \equiv \{\mathbf{p}^*, \theta^*\}$ as a solution to the problem

$$\begin{aligned} &\min f(p, \theta) \\ &\text{subject to} \\ &\quad \Lambda(p, \theta) \bigcap \Gamma = \varnothing, \\ &\quad \Lambda(p, \theta) \subset \Sigma \Rightarrow p \in \Omega(\theta), \\ &\quad \theta \in [0, 2\pi). \end{aligned} \tag{2}$$

where Ω is the space of positions such that the polyhedron Λ lies inside the volume Σ. A 2-D example is shown in Fig. 5 for two values of θ. We can see that Ω is a function of θ and is given by the following relation:

$$p \in \Omega(\theta) \Rightarrow \Lambda(p, \theta) \subseteq \Sigma. \tag{3}$$

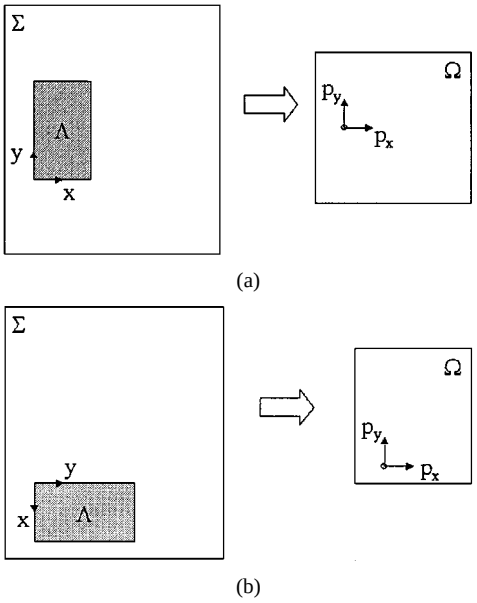


Fig. 5. Obtaining the set of permissible positions Ω from volume Σ. (a) $\Omega = 0°$. (b) $\Omega = 270°$.

III. Approach

Optimal interference-free configuration computation of polyhedral parts is difficult. Even the 2-D form of (2), i.e., optimal nesting of blanks on a sheet ($\mathbf{p} = \{p_x, p_y\}$, $\Phi = \theta$) has been shown to be NP-hard by [1]. Hence, determination of a globally optimal part configuration is impossible in polynomial time. However, computing a closed-form description of c-space obstacle in $\{p_x, p_y, p_z, \theta\}$-space is more difficult than computing the corresponding obstacle in lower dimensional $\{p_x, p_y, p_z\}$-space for a discrete orientations of the new part.

We seek to compute near optimal solutions without exhaustively covering the search space. We consider only discrete values of θ. Only the position parameters now remain to be computed. The main advantage of dealing with $\{p_x, p_y, p_z\}$-space is that c-space obstacles and free space regions corresponding to polyhedrons are polyhedral. Hence, the constraints for the problem in (2) are linear. Computation of a c-space obstacle in $\{p_x, p_y, p_z\}$-space is described in Appendix B. If the cost function is convex, it is easy to compute the optimal interference-free position using a closed form method. By considering discrete orientations, we have traded optimality for the ability to produce near optimal configurations in reasonable time.

A. Techniques to Speed Up Interference-Free Configuration Computation

A near optimal interference-free configuration can be computed by first determining the best interference-free position for each discrete orientation from (2) and then choosing the best configuration of the lot. Determination of the best interference-free position for a discrete orientation requires the computation of the c-space obstacle for the existing stack with respect to the new part. The best interference-free position in this orientation is the position that lies in the region obtained by subtracting the c-space obstacle from the region of permissible positions Ω and minimizes the cost function from (2).

For every discrete orientation, we require the computationally expensive step of computing a c-space obstacle. In order to speed up the computation, we address the following questions.

1) Is it possible to reuse c-space information computed for one discrete orientation for other discrete orientations?
2) Is it possible to compute the interference-free position for a discrete orientation by constructing only a portion of the corresponding c-space obstacle?

We will show in Sections IV and V that the answer to both these questions is yes. This results in a significant saving in computation time.

IV. Extrapolation of a C-Space Obstacle in $\{p_x, p_y, p_z\}$-Space From One Orientation to Another Orientation

This section describes the first of two techniques used by us to speed up c-space obstacle construction. First we characterize the effect of changing orientation parameter θ on c-space obstacle geometry and topology of its face-edge-vertex graph. Next we identify orientation intervals within which the topology stays constant and we can easily extrapolate geometry of the obstacle from one orientation to another.

A. Effect of Change in Orientation on C-Space Obstacle Topology in $\{p_x, p_y, p_z\}$-Space

Appendix B shows us how to compute a c-space obstacle $C(A, B)$ for convex polyhedron B with respect to a convex polyhedron A in $\{p_x, p_y, p_z\}$-space. We can see from (12) that the c-space obstacle is convex. Let us examine the effect on this obstacle of allowing A to rotate about the z axis of world coordinate frame. The points P_{ij} used to compute the obstacle can be divided into two mutually exclusive sets: black extreme points and white interior points.

As A is rotated, the points P_{ij} are transformed as follows:

$$P_{ij}(\theta) = V_j^B - R_{z,\theta} V_{0i}^A, \qquad i = 1, 2, \ldots, n_A, \quad j = 1, 2, \ldots, n_B \tag{4}$$

where $R_{z,\theta}$ is the rotation matrix and V_{0i}^A, $i = 1, 2, \ldots, n_A$ are the vertices of A at $\theta = 0$. As A is rotated, the connectivity graph topology of $C(A, B)$ remains the same as long as the black points remain extreme points and the white points remain interior.

If one of the interior points becomes an extreme point or vice-versa (see Fig. 6), the topology of the obstacle changes. As the value of θ changes, a necessary condition for an interior point to become an extreme point is that it lies inside one of the faces of the convex c-space obstacle. The critical orientation θ_c when this condition occurs is given by the following equation:

$$[P_{ij}(\theta_c) - F_0(\theta_c)] \bullet n(\theta_c) = 0 \tag{5}$$

where F_0 is a vertex on one of the faces of $C(A, B)$ and n is the normal of the same face. Checking for this condition is a conservative method of evaluating if an interior point is about to become an extreme point. Appendix C shows that we can compute critical orientation θ_c by solving a quadratic equation for $\tan(\theta_c/2)$ and then computing the inverse tangent.

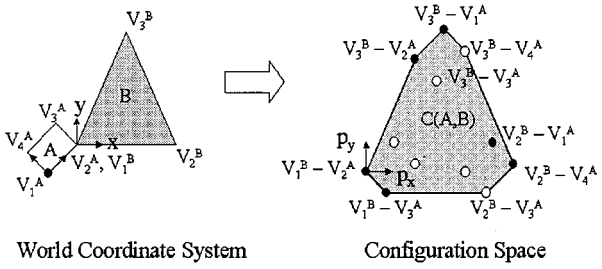


Fig. 6. Effect of rotation of polyhedron A on c-space obstacle $C(A, B)$.

We need to perform this step for every point P_{ij} for every face of the convex c-space obstacle. Therefore, this requires the computation of the c-space obstacle from scratch using the procedure in Appendix B for one orientation in every interval within which the topology of the obstacle stays the same.

B. Extrapolation of C-Space Obstacle Using Constant Topology Orientation Interval

Let us consider computing a c-space obstacle for discrete values of θ starting with $\theta = 0$. The obstacle for $\theta = 0$ can be computed using results in Appendix B in $O(n_A \cdot n_B \cdot \log(n_A \cdot n_B))$ time. The nearest critical orientation where the c-space obstacle topology changes can be computed by solving (5) for all points P_{ij} using the procedure from Appendix C. This gives us the range of θ within which the topology of the obstacle stays the same and we refer to this range as constant topology orientation interval. For all subsequent discrete orientations in this interval, the c-space obstacle can be computed in $O(n_A \cdot n_B)$ time by just determining the new location of the black extreme points in Fig. 6 using (4). For the first discrete orientation that lies outside the constant topology orientation interval, we compute the c-space obstacle from scratch as in the case of $\theta = 0$. For the new c-space obstacle, we can once again compute the new constant topology orientation interval and repeat the process.

The variation of obstacle topology for the polygons from Fig. 6 is shown in Fig. 7. The orientation parameter θ is varied from 0° to 90°. The obstacle in Fig. 7(a) is computed from scratch using the procedure in Appendix B. The nearest critical orientation is computed to be 45° and the constant topology orientation interval is [0°, 45°). Therefore, the c-space obstacles for 15° and 30° in Fig. 7(b) and (c) can be extrapolated from the obstacle in Fig. 7(a). Similarly, we can compute the obstacle for 45° [Fig. 7(d)] is computed from scratch, the new constant topology orientation interval is computed to be [45°, 90°), and the obstacles for 60° and 75° in Fig. 7(e) and (f) respectively are extrapolated from obstacle in Fig. 7(g). The c-space obstacle [Fig. 7(h)] for 90° is then computed from scratch. This example is not trivial as sheet metal parts are decomposed into convex components before computing the c-space obstacles. Hence, interactions between triangular and rectangular polygons that have been extruded by the sheet

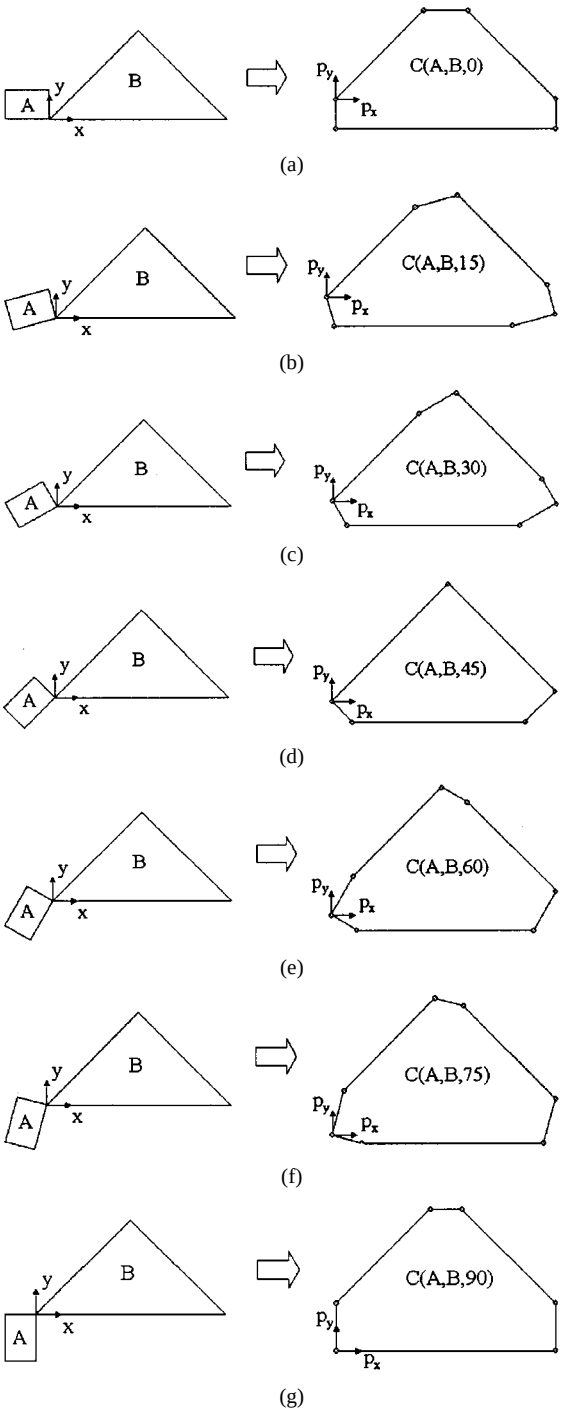


Fig. 7. Variation of C-space obstacle topology as Θ changes. (a) $\Theta = 0°$. (b) $\Theta = 15°$. (c) $\Theta = 30°$. (d) $\Theta = 45°$. (e) $\Theta = 60°$. (f) $\Theta = 75°$. (g) $\Theta = 90°$.

metal thickness occur often in practice and the discretization step for θ is 1°–5°.

V. Interference-Free Configuration Computation Using Partial C-Space Obstacles

In this section we discuss the second technique used by the authors to speed up c-space obstacle construction. This technique results in significant savings in computation time when applied to concave solids that consist of a large number of convex components. This technique is applied to computation of interference-free position for a concave solid with respect to concave obstacles for a discrete orientation.

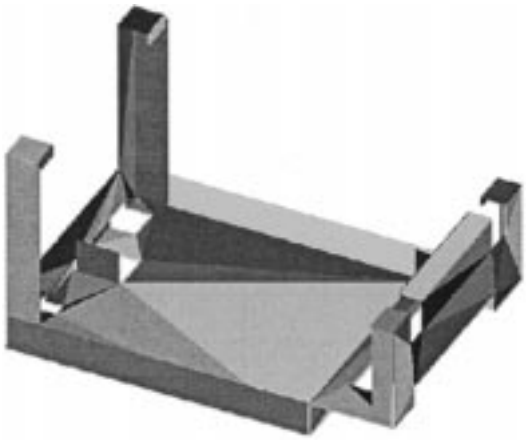


Fig. 8. A sheet metal part with 79 convex components.

A. Interference-Free Configuration Computation for One Discrete Orientation

Consider the part design shown in Fig. 8. This sheet metal part has 79 convex components. Let us assume that the goal is to compute an optimal interference-free position for such a part with respect to an existing stack. For simplicity, let us assume that the existing stack has only one part. Computing the c-space obstacle for the part in Fig. 8 requires computing the obstacle first for 6241 (79^2) convex component pairs. Therefore, computation of the complete c-space obstacle for a concave polyhedral pair for even a single orientation can be time-consuming.

It is, however, possible to compute a near-optimal interference- free position (see Fig. 9) without having to compute the full c-space obstacle for every discrete orientation, i.e., for each discrete orientation, we compute the c-space obstacles for only a few convex component pairs. It should be noted that the solution we obtain by looking at only a portion of the obstacle is as good as the one that would be obtained by looking at the complete obstacle for that discrete orientation. We are not compromising on the solution quality by ignoring portions of the c-space obstacle.

Fig. 9(a) shows the new part, Fig. 9(b) shows the convex decomposition of the part, and Fig. 9(c) shows the set of permissible positions for the new part. Here are the steps for computing a near optimal interference-free position for this orientation.

1) Choose a candidate optimal position $\mathbf{p_d}$ for the new part [Fig. 9(d)] that minimizes the cost function from (2).
2) Construct the list of interfering convex pairs L shown in Fig. 9(e).
3) Construct the c-space obstacle for these interfering convex pairs [Fig. 9(f)].
4) Subtract this c-space obstacle from the set of permissible positions.
5) Choose a new candidate position $\mathbf{p^*_{Lk}}$ that minimizes cost function and go to step 2).

We repeat this process until no additional interference is detected in step 1). In the example shown in Fig. 9, this occurs after just one iteration and the final position is shown in Fig. 9(i).

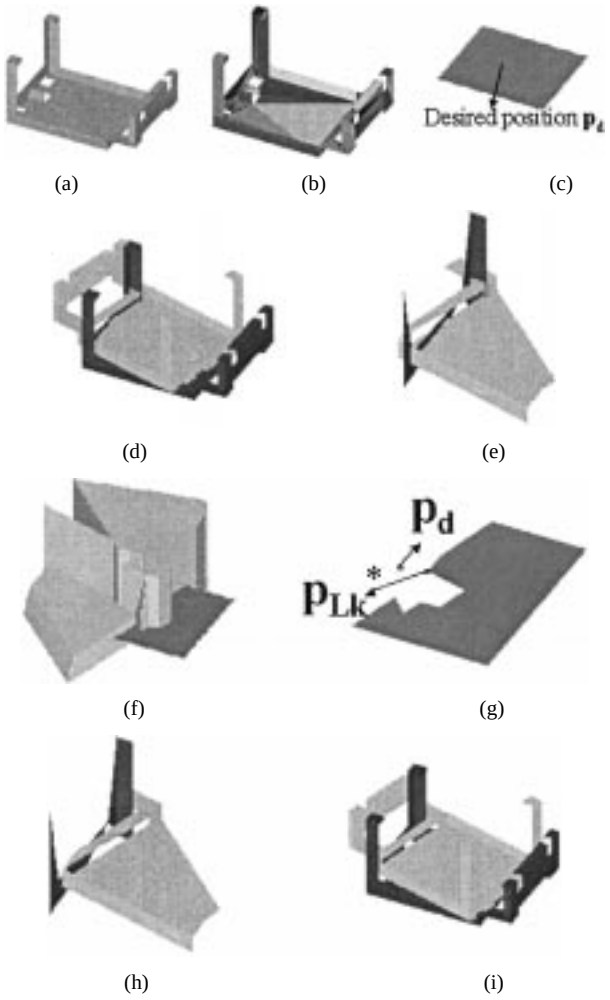


Fig. 9. Computation of an interference-free position using partial C-space obstacle. (a) Part. (b) Convex decomposition. (c) Space of candidate positions. (d) Initial position. (e) Interference convex pairs. (f) C-space obstacle for iinterfering convex pairs. (g) Free space after substracting C-space obstacle from (f). (h) Interference pairs in new candidate position. (i) No interference detected in new candidate position.

During each iteration, the set of permissible positions shrinks in step 2) as a c-space obstacle is subtracted from it. Therefore, there are only two possible outcomes to the iterative process described above. One is that an interference-free position is found and the other outcome is that the set of permissible positions shrinks to a null set as a c-space obstacle gets subtracted from it in step 2) of each iteration.

1) Decomposition of a Concave Part Into Convex Components: We decompose a sheet metal part into convex components by looking at a zero-thickness model of the part. Every convex face of the part is extruded by the sheet metal thickness to form a convex component. Every concave face is tessellated into triangular faces and each triangular face is then extruded by the sheet metal thickness to form a convex component. The part shown in Fig. 8 is decomposed into 79 convex components. It is also possible to merge some triangles into larger convex pieces [3].

2) C-Space Obstacle for a Concave Part With Respect to a Concave Obstacle: Appendix B discussed c-space obstacle computation for convex polyhedral pairs. Let us assume that the

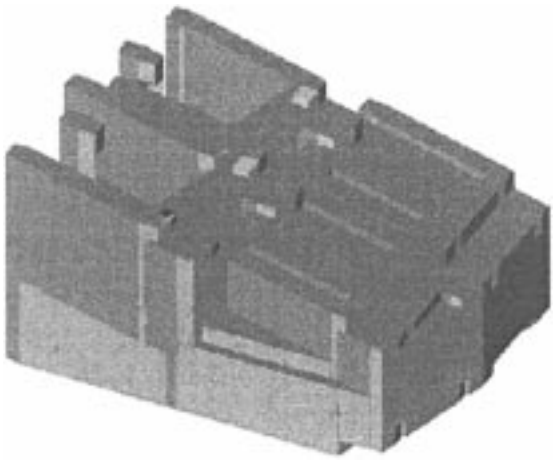


Fig. 10. Complete C-space obstacle $C(\Lambda, \Gamma)$ for existing stack Γ with respect to the new part Λ.

new part Λ and the existing stack Γ from (2) are concave and are decomposed into m_Λ and m_Γ convex components, respectively

$$\Lambda = \bigcup_{i=1}^{m_\Lambda} \Lambda_i, \qquad \Gamma = \bigcup_{j=1}^{m_\Gamma} \Gamma_j \tag{6}$$

where Λ_i and Γ_j are convex components of the new part Λ and existing stack Γ, respectively.

The c-space obstacle $C(\Lambda, \Gamma)$ is the union of the obstacles for all the convex component pairs (Λ_i, Γ_j). Hence, we have the relation

$$C(\Lambda, \Gamma) = \bigcup C(\Lambda_i, \Gamma_j), \qquad i = 1, 2, \ldots, m_\Lambda, \quad j = 1, 2, \ldots, m_\Gamma. \tag{7}$$

3) Interference Pair Lists: Interference detection between a polyhedral pair can be performed much faster than c-space obstacle computation for the same pair. We use an interference detection tool called RAPID [4]. RAPID models a polyhedron using a hierarchical data structure called an oriented bounding box. At the lowest level, the model surface is represented as a set of triangles. RAPID considers polyhedral pairs that are just touching each other as interfering. We have supplemented RAPID with some reasoning about triangle pairs to differentiate between triangle pairs that are touching and those that are penetrating. Thus, we can test all the component pairs (Λ_i, Γ_j) for interference and compile a list L of *interference pairs* as follows:

$$L \equiv \left\{ (\Lambda_i, \Gamma_j) \middle| \Lambda_i \bigcap \Gamma_j \neq \varnothing, \quad i = 1, 2, \ldots, m_\Lambda, \; j = 1, 2, \ldots, m_\Gamma \right\}. \tag{8}$$

The c-space obstacle $C(L)$ corresponding to this list of interference pairs is the union of the obstacles for the individual convex pairs. This obstacle [see Fig. 9(f)] c-space obstacle $C(\Lambda, \Gamma)$ (see Fig. 10) for the existing stack corresponding to the new part. Consider the free space region obtained by subtracting this obstacle from the set of permissible positions for the new part. For any position of the new part in this free

space region, there is no interference between convex pairs of list L.

B. Quality of Interference-Free Position Computed Using Partial C-Space Obstacles

The cost of the interference-free configuration computed using this procedure is as low as the cost of an interference-free configuration computed by a method that constructs the complete obstacle for every discrete orientation θ_k. For the polyhedral pair (Λ, Γ), there are $m_\Lambda m_\Gamma$ possible interference pairs [see (6)]. Consider a single discrete orientation θ_k. The c-space obstacle $C(\Lambda, \Gamma, \theta_k)$ is the set of positions that result in interference between one or more of these $m_\Lambda m_\Gamma$ possible interference pairs. The c-space obstacle $C(L_k)$ is the set of positions that result in interference only between one or more interference pairs that are present in list L_k. Therefore, $C(L_k)$ is a subset of $C(\Lambda, \Gamma, \theta_k)$. From this, we can infer that the set of positions $F(L_k)$, where no interference exists between the interference pairs of L_k, is a superset of free region $F(\Lambda, \Gamma, \theta_k)$, obtained by subtracting c-space obstacle $C(\Lambda, \Gamma, \theta_k)$ from the set of permissible positions $\Omega(\theta_k)$. Hence we have the relation

$$\begin{aligned} F(L_k) &= \Omega(\theta_k) - C(L_k) \\ F(\Lambda, \Gamma, \theta_k) &= \Omega(\theta_k) - C(\Lambda, \Gamma, \theta_k) \\ C(L_k) \subseteq C(\Lambda, \Gamma, \theta_k) &\Leftrightarrow F(L_k) \supseteq F(\Lambda, \Gamma, \theta_k). \end{aligned} \quad (9)$$

The cost of the optimal configuration in region $F(L_k)$ is a lower bound of the cost of the optimal configuration in the region $F(\Lambda, \Gamma, \theta_k)$. This shows that for every discrete orientation θ_k, the cost of the interference-free configuration obtained by considering only interference pairs of L_k is as low as the cost of the interference-free configuration computed by constructing the complete c-space obstacle $C(\Lambda, \Gamma, \theta_k)$ and free region $F(\Lambda, \Gamma, \theta_k)$.

C. Interference-Free Configuration Computation Over All Discrete Orientations

One way to compute the best interference-free configuration for a new part is to compute the interference-free position for every discrete orientation $\theta_k = 0, \Delta\theta, 2\Delta\theta, \ldots, 2\pi - \Delta\theta$ using the iterative procedure from Section V-A and select the best of the computed positions. In this section, we show how to further speed up the process.

The best configuration is the one minimizing the cost function $f(\mathbf{p}, \theta)$ from (2). Let us assume we have computed an interference-free position $\mathbf{p}^*$ for one candidate orientation θ^* such that there is no interference between the new part and the existing stack. The value of the cost function for this configuration is $f(\mathbf{p}^*, \theta^*)$. For certain orientations, even accounting for a few interference pairs can move Λ so far, and hence increase the value of the cost function beyond $f(\mathbf{p}^*, \theta^*)$, that the current orientation is not going to yield the optimal interference-free configuration. We can therefore discard such orientations after analyzing only a few interference pairs and completing iterations till no interference is detected between the new part and the existing stack.

We use a two-stage approach. In the first stage, for every discrete orientation.

1) The new part is positioned at position $\mathbf{p_d}$ that minimizes the cost function $f(\mathbf{p}, \theta)$.
2) We compute interference-free position accounting for interference pairs encountered only at position $\mathbf{p_d}$.
3) The interference-free position and orientation are entered into a priority queue along with the value of the cost function at this configuration.

The second stage of the algorithm consists of the following steps.

1) Extract the cheapest configuration from the priority queue
2) If this configuration for new part Λ is interference-free with respect to existing stack Γ, choose this as the optimal configuration and stop.
3) Else, for current orientation, move to a new position accounting for additional interference pairs encountered in current position. Add new configuration to priority queue with value of cost function at this configuration.
4) If priority queue is not empty, go to step 1).
5) Else, no interference-free configuration exists. Stop the computation.

Thus, orientations are considered for the expensive step of interference-free position computation only if they appear promising. If a certain orientation has a high cost after processing only a few interference pairs, the corresponding configuration is pushed to the bottom of the priority queue and will not be considered again before ruling out other cheaper orientations.

The complexity of this procedure is $O(N_\theta \cdot n_L \cdot (n_\Lambda \cdot n_\Gamma)\log(n_\Lambda \cdot n_\Gamma))$ where N_θ is the number of discrete orientations considered and n_Λ and n_Γ are the number of vertices of Λ and Γ, respectively. The cost of computing a convex hull using all vertices of Λ and Γ is $O(n_\Lambda \cdot n_\Gamma \log(n_\Lambda \cdot n_\Gamma))$. The parameter n_L is the total number of interference pairs considered and is the sum of the number of interference pairs in all L_k. In the worst case, the value of n_L is still the total number of interference pairs $m_\Lambda m_\Gamma$. However, we have seen through tests that for complex parts, $n_L \ll 0.01\, m_\Lambda m_\Gamma$.

VI. Interference Analysis by the Stacking Planner

In this section, we describe how our stacking planner uses the algorithms from the previous section for interference analysis. Interference avoidance is one of many constraints part configurations have to satisfy while computing a stacking plan (see Section II). We discuss some modeling and representation issues that have to be resolved in order to use c-space based analysis by a stacking planner.

A. Interference Analysis

One of the key components of the planner is an Interference Analysis module. For a fixed orientation, it supplies a list of interference-free positions for a new part to be added to an existing stack. This list of positions is sorted in increasing order of cost. So far, we have assumed that the orientation parameters ϕ and ψ are constant. While adding a part to a stack, the planner looks at a discrete orientation sets $\Phi_\mathbf{k} \equiv \{\phi_k, \psi_k, \theta_k\}$.

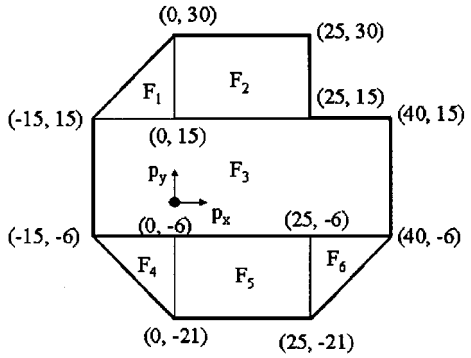


Fig. 11. Representation of concave free space using convex components.

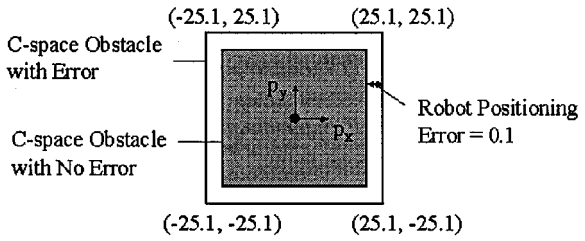


Fig. 12. C-space obstacles with and without accounting for robot positioning error.

The planner using stability and grasping related constraints decides these orientation sets. We wish to allow external modules to rule out orientations for reasons such as ability of the robot to grasp a part. Such concerns may not be addressed by the cost function. Therefore, as recommended by Section V, we use a two-stage approach to compute interference-free positions for candidate orientations individually. Extrapolation (explained in Section IV) from one orientation to another is still performed for θ-intervals.

Consider a new part that has to be added to a stack. The part orientation is constant. The interference analysis module positions the new part on the floor at the center of the floor space. RAPID collision detection library [4] is used to identify the convex polyhedral pairs (one convex polyhedron from the new part and other convex polyhedron from a stack part) that interfere. A list of interference pairs is constructed and we use a two-stage approach, which computes an interference-free position using this list.

1) Modeling of Free Space: One issue is how best to represent the free space information. This will help us decide how to compute a position minimizing the user defined cost function $f(\mathbf{p}, \theta_k)$ over a free region in $\{p_x, p_y, p_z\}$-space. For a fixed orientation, the cost function used currently by the planner is a convex quadratic function of $\{p_x, p_y, p_z\}$. There are closed-form methods available to solve for optimal solutions in a convex space. With the orientation fixed, both c-space obstacles and free regions are polyhedral. Hence, it is convenient to represent the free space as a set of disjoint convex polyhedrons. Fig. 11 shows a concave free space split up into six disjoint components. The position in the concave free space that minimizes the quadratic cost function can be computed as follows. Use a closed-form method to compute the optimal position for each of the six convex components and pick the best of the six positions as the optimal position.

Each convex component of free space is represented as a set of linear inequalities that describe the bounding faces of the component. These linear inequalities act as constraints while solving for an optimal position in this component. Once an interference-free position is computed for every convex component, we pick the one with the least cost among them as the interference-free position for the free region as a whole.

2) Modeling of C-Space Obstacles: C-space obstacles in our planner have two representations. One representation is the same as the one used for convex components of the free space, that is, modeling the obstacle as a intersection of half spaces described by linear inequalities. This is possible because obstacles in our interference analysis module are computed only convex interference pairs. This ensures that the obstacle in $\{p_x, p_y, p_z\}$-space is also convex. The second representation for an obstacle is a face-based representation that describes the boundary faces of the obstacle. The boundary faces and their vertices are used for extrapolation as discussed in Section IV-B.

The linear inequality based representation is useful for incorporating robot positioning errors. Let us assume that all the inequalities have $\geq$ sign. Increasing the right hand side of the inequality by the required amount enforces a separation distance between parts. Consider the representation of the two-dimensional c-space obstacle from Fig. 12 without accounting for robot positioning error. This is shown as the following equation:

$$\begin{aligned} p_x &\geq -25, \quad -p_x \geq -25 \\ p_y &\geq -25, \quad -p_y \geq -25. \end{aligned} \tag{10}$$

Accounting for robot positioning error of, say, 0.1 changes the representation to the following equation:

$$\begin{aligned} p_x &\geq -25.1, \quad -p_x \geq -25.1 \\ p_y &\geq -25.1, \quad -p_y \geq -25.1. \end{aligned} \tag{11}$$

If a solid model had represented the obstacle, growing it by the separation distance would have been messier. The faces of the model have to be moved out and additional faces would have to be generated to fill the gaps between the faces in their new position.

3) Subtraction of a C-Space Obstacle From a Free-Space Component: Since we always represent the free space as a set of disjoint convex polyhedrons, subtraction of an obstacle involves the following steps.

1) Eliminate the free-space components that are completely inside the obstacle.
2) If the free space component and the obstacle intersect, subtract obstacle from the free space component. If the modified free space component is concave, decompose it into smaller convex components.

These tasks can be accomplished easily by manipulating the sign of the inequalities describing the obstacle. There is no need for the more expensive and nonrobust option of constructing solid models of the free space components and obstacles and performing Boolean operations on them.

In Fig. 13, we show the subtraction of an obstacle from the free space components shown in Fig. 11. Component F_1 lies

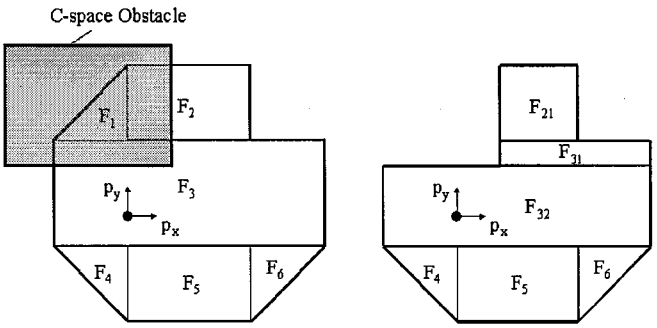


Fig. 13. Subtraction of a C-space obstacle from free space.

completely inside the obstacle and hence is eliminated. Component F_2 overlaps with the obstacle, but need not be decomposed further as the subtracted portion is still convex. The smaller component is labeled F_{21}. Component F_3 becomes concave after subtraction and is decomposed into convex components F_{31} and F_{32}. Components F_4–F_6 are not affected by the subtraction operation.

4) Computation of Multiple Interference-Free Positions for a Single Orientation: Computing an optimal interference-free position requires determining the optimal position in every convex component of the free space and then choosing the cheapest among them as the final answer. Often, the new part in this optimal interference-free position may be unstable or might render other stack parts unstable. Since a position in every convex component (F_1 through F_6 in Fig. 11) has already been computed, the interference analysis module of our stacking planner sorts them in increasing order of cost and then returns the list. Other modules in the planner search this list for a position that satisfies stability, grasping and other user-specified concerns that render the stacking plan feasible. For a detailed description, please see [2].

Computation of one solution for every convex component increases the chance of finding a stable position. Every face of a c-space obstacle reflects a distinct contact state between the interfering convex pairs [5], [6]. Initially, the free space consists of only one convex component. The faces of this component represent the constraints of the floor space and maximum permissible stack height. When an obstacle is subtracted from a free-space component, it is decomposed into smaller convex pieces using the obstacle faces. Hence every face of the free-space component represents a contact state or a space constraint. For a quadratic cost function, such as the one used by our planner, the optimal position in a free-space component always lies on one of its faces. Since different free-space components have faces representing different contact states. Therefore, a list of positions drawn from all the convex components, are likely to result in a different contact state for the new part when added to a stack. We hope that at least one of the contact states renders the new part stable.

VII. Results and Discussion

In this section we first demonstrate the effect of the two techniques described in Sections IV and V on computing interference-free configurations. Next we show the result of using interference-free configuration computation algorithms in our stacking planner along with other analysis modules that evaluate stability, stacking plan feasibility, ability to grasp the part in its final configuration, etc.

(a) (b) (c)

Fig. 14. Test parts. (a) Channel part. (b) Triangle part. (c) Complex part.

We use three test parts. The first part, called *Channel Part*, shown in Fig. 14(a), is the simplest of the parts considered and has three flanges. Flipping alternate parts in a stack upside down can produce nested stacks. The second part (called *Triangle Part*) has six flanges [see Fig. 14(b)] and can be repeated in the same orientation to generate stable yet compact nested stacks. The last part, called *Complex Part*, [see Fig. 14(c)] is a complex part with 79 convex components and hence takes more time to analyze. We use this part to show that the benefits of the speed-up techniques, especially the one described in Section V, increase significantly as the part becomes more complex.

A. Interference-Free Configuration Computation

In this section we show the effect of c-space obstacle computation speed-up techniques from Sections IV and V on the computation of a near-optimal interference-free configuration for a test part when added to an existing stack. Three test parts are considered. The examples in this section involve add a new part to an existing column stack. The optimal configuration minimizes the Euclidean distance from a user-specified desired position $\mathbf{p_d}$ for the part to be added to the stack. Position $\mathbf{p_d}$ places the new part c.g. at centroid of the faces or edge/face combination that supports the whole stack. For example, the stack base in Fig. 15(b) is the large base face of Part 1. For all the test parts, the stack orientation is chosen randomly and the initial new part orientation is chosen as 0°. The only constraints that are enforced are that the part position should lie inside a user provided set of positions [Ω from (2)] and there should be no interference with other parts in the stack. These results might be of interest to people interested in generating c-space obstacles for any application in four-dimensional $\{p_x, p_y, p_z, \theta\}$-space. One area where such c-space obstacles are useful is in path planning, in a workspace with polyhedral obstacles, for a 3-D polyhedral mobile robot with three degrees of freedom: $\{p_x, p_y, \theta\}$. The c-space obstacle in $\{p_x, p_y, \theta\}$-space can be obtained by first computing the c-space obstacle in $\{p_x, p_y, p_z, \theta\}$-space and then intersecting it with the $p_z = 0$ plane.

1) Implementation: We have implemented the two-stage procedure from Section V-C using C++ on a PC with a 266-MHz Pentium Pro processor. The discretization step $\Delta\theta$ (see Section V-C) is chosen as 1°. RAPID library [4] is used for interference detection, *qhull* library [7] is used for convex hull computation, and ACIS geometric kernel[1] is used to model the parts and the c-space obstacles. Triangulating all the concave

[1]Refer to http://www.spatial.com for Acis Geometric Modeling Library.

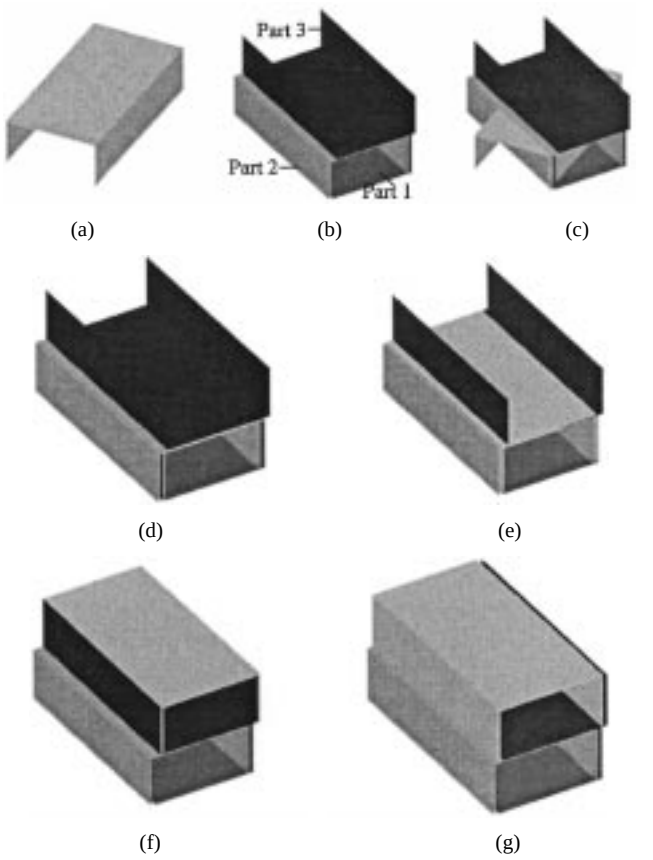


Fig. 15. Addition of a new part to an existing three-part stack for a channel part. (a) New part. (b) Three-part stack. (c) Initial configuration. (d) Intermediate configuration #1. (e) Intermediate configuration #2. (f) Intermediate configuration #3. (g) Final configuration.

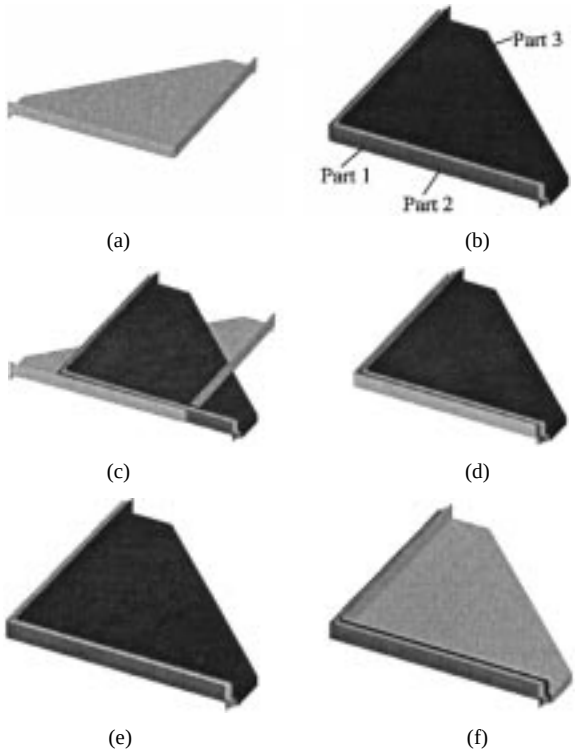


Fig. 16. Addition of a new part to an existing three-part stack for triangle part. (a) New part. (b) Three-part stack. (c) Initial configuration. (d) Intermediate configuration #1. (e) Intermediate configuration #2. (f) Final configuration.

faces of a zero sheet thickness model of the part and extruding the resulting polygons by the sheet metal thickness generate convex components of the sheet metal part.

2) Set of Permissible Positions Ω: For channel and triangle parts, the set of permissible positions Ω is a subset of 3-D $\{p_x, p_y, p_z\}$-space and for Part #3, the set of allowable positions Ω is a subset of 2-D $\{p_x, p_y\}$-space. A 3-D position set Ω is useful for parts that have a large number of stable orientations (Parts #1 and #2) and the 2-D position set Ω is preferred for parts like Complex Part that have few stable orientations. In the 2-D case, the position parameter p_z for a new part can be chosen by identifying a set of edges or faces of parts already in the stack that support the new part rendering it stable. For the 3-D case, position parameter for the new part $\mathbf{p_z}$ is chosen such that the part is placed at the bottom of the stack. Please note that the user does not specify orientation and the planner (in Section VII-A alone) chooses a discrete orientation that enables it to position (with no interference) the new part as close as possible to the desired position.

3) Interference-Free Configurations for the Test Parts: Addition of a new part to an existing stack is shown for the channel part in Fig. 15. The new part to be added is shown in Fig. 15(a) and the three-part stack it has to be added to is shown in Fig. 15(b). The desired position $\mathbf{p_d}$ for the new part [shown in Fig. 15(c)] is such that it lies at the bottom of the stack and its c.g. lies at the centroid of the base face of Part 1. The initial orientation for the new part is 0°. This orientation is not valid as the new part interferes with parts already in the stack. Hence, the planner tries out different discrete orientations and for each discrete orientation, it computes the distance the new part has to move to avoid interference between the interference pairs (see Section V-A3) it encountered at position $\mathbf{p_d}$. For the orientation shown in Fig. 15(d), the new part is almost overlapping with Part 1. In this configuration,[2] the new part still interferes with parts 1 and 2, but the interference pairs are different. The algorithm moves the part up to avoid interference first with parts 1 and 2, and then finally with part 3. The final part configuration is such that the parts are nested.

The algorithm works similarly for Triangle Part as shown in Fig. 16 to produce a nested stack shown in Fig. 16(f). This is one of the main advantages of using c-space representation. It helps us obtain nested stacks without explicitly looking for part features conducive to nesting. For the examples in this section, the initial stack orientation is chosen at random. Hence, we need the discretization step $\Delta\theta$ to be small enough such that at least one orientation that facilitates nesting is not considered. The extrapolation algorithm described in Section IV helps us analyze c-space information for a large number of candidate orientations without having to construct the c-space obstacles for every orientation from scratch.

While the benefits of the extrapolation technique from Section IV apply uniformly to all parts irrespective of their complexity, the benefits of the partial c-space obstacle based technique from Section V are more pronounced for stacks with com-

[2]Only a few candidate configurations considered by the two-stage procedure from Section V-C are shown for the sake of brevity as $\Delta\theta = 1°$. This is true for the stacks shown for triangle and complex parts in Section VII-A.

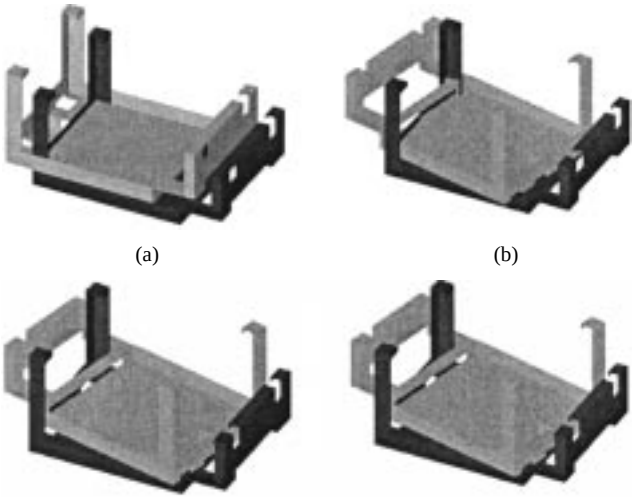


Fig. 17. Computing an interference-free configuration for a new complex part with respect to one stationary part. (a) Initial configuration. (b) Intermediate configuration #1. (c) Intermediate configuration #2. (d) Final configuration.

plex parts. This even applies to stacks with a small number of parts. We show this by presenting the results for a two-part "stack" for Complex Part from Fig. 14(c). Stacks with larger number of parts for Complex Part are dealt with in the next section. The set of permissible positions Ω for this example is two-dimensional and is a convex region in the $\mathbf{p_z} = 0$ plane.

Fig. 17 shows the computation of a near-optimal interference-free configuration for Complex Part. The initial, final and a couple of intermediate configurations are shown in Fig. 17(a), (d), (b), and (c), respectively. This part poses two difficulties:

1) It has a large number of convex components.
2) Choosing a near-optimal configuration sometimes requires the accommodation of protruding flanges in holes.

Partial construction of c-space obstacles for even this two-part case significantly speeds up interference-free configuration computation. The reason is that only a few interference pairs are considered for the expensive step of c-space obstacle computation. Searching in c-space automatically enables us to accommodate flanges in holes without explicitly looking for such features. However, the final configuration in Fig. 17 might not be amenable to automated stacking as the flange is very close to the edge of the hole. Robot positioning error can cause parts to get tangled. One way of avoiding tangle is to prescribe an upper bound on robot positioning error and the final part configuration chosen should be at least that far from the c-space obstacle. This has already been implemented in the planner (see Section VI-A-2 for details).

Fig. 8 shows the convex decomposition of the part into 79 convex components. One problem with the present triangulation scheme is evident from part configurations in Fig. 17(c) and (d). In the configuration in Fig. 17(c), there is a small triangular component of the moving part that is interfering with a component of the stationary part. An additional step is required to compute the small translation to the interference- free configuration in Fig. 17(d). Such steps can be avoided by combining narrow and small triangles with neighboring triangles to create larger convex components.

4) Computation Times: Table I[3] shows the time spent by RAPID for interference detection, time spent for c-space obstacle construction from scratch, and time spent for c-space extrapolation. The time taken per function call is also presented for each of these tasks. We can see from Table I that the time taken per function call for extrapolation of a c-space obstacle is an order of magnitude less than the time taken per function call to compute an obstacle from scratch. We can also see that the total time spent for extrapolation is larger than the total time spent in obstacle construction. This indicates that the constant topology orientation intervals (see Section IV-B) are large enough for the extrapolation function to be valid for a large number of candidate orientations. This shows us that the extrapolation technique described in Section IV is faster than construction from scratch and it is valid for many candidate orientations.

As explained in Section V-A3, we use interference detection to avoid unnecessary computation of c-space obstacles. This is justified by the fact that time taken per call for interference detection is about an order of magnitude less than the time it takes to construct the c-space obstacle from scratch. One interesting observation from Table I is that the time taken per function call for interference detection, c-space obstacle construction, or extrapolation is the same for all the three test parts, in spite of the varying degree of complexity. This is the result of convex decomposition of the parts. Hence, the three routines work on triangles or simple convex shapes most of the time. As part complexity increases, the number of times these routines are called increases. The total time taken for the complex part is only twice as large as the time taken for the other two parts because of two reasons. The first reason is that there is only one stationary part in the stack and the set of permissible positions is 2-D. The second reason is that the algorithm was lucky enough to find an orientation that enabled it to position the new part close to the desired position by accommodating one of the tall flanges in a hole [see Fig. 17(d)]. This prevented the algorithm from considering candidate orientations where such accommodation was not possible and hence the new part had to be moved far from the desired position.

Table II shows the effect of partial c-space obstacle computation on the performance of the two-stage approach from Section V-C. In column 3, the table shows the number of interference pairs (see Section V-A3) that have to be analyzed to compute the complete c-space obstacle for the new part for every candidate orientation considered. In column 4, the table shows the number of interference pairs actually considered by the planner for c-space obstacle construction or extrapolation. As the part complexity increases, the percentage of convex components considered drops from 66% through 26% all the way to less than 0.1%. We can see that there are significant advantages to using partial computation of c-space obstacles for polyhedral sheet metal parts.

B. Interference Analysis by Stacking Planner

In this section we show the stacking plans generated by our planner for the three test parts from Fig. 14. The number of

[3]The total computation time includes computation time for procedures other than interference detection and c-space obstacle computation. Hence, the three computation times do not add up to give total computation time.

TABLE I
COMPUTATION TIMES FOR INTERFERENCE DETECTION AND C-SPACE OBSTACLE CONSTRUCTION AND EXTRAPOLATION FOR TWO-STAGE APPROACH IN SECTION V-C

Test Part	No. of Convex Components	Time for Interference Detection		Time for Obstacle Construction		Time for Obstacle Extrapolation		Total Computation Time
		(s)	(s/Call)	(s)	(s/Call)	(s)	(s/Call)	(s)
Channel Part	3	30.4	0.0014	5.1	0.0105	8.1	0.0013	188.4
Triangle Part	6	58.9	0.0015	22.6	0.0155	14.4	0.0016	298.2
Complex Part	79	151.2	6.3e-5	2.8	0.0111	8.7	0.0016	271.6

TABLE II
NUMBER OF INTERFERENCE PAIRS EVALUATED TO COMPUTE INTERFERENCE-FREE CONFIGURATION FOR TWO-STAGE APPROACH FROM SECTION V-C

Test Part	No. of Convex Components	Total No. of Interference Pairs	No. of Interference Pairs Evaluated
Channel Part	3	9720	6960
Triangle Part	6	38880	9951
Complex Part	79	226760	6751

parts, n from (1) is 15. For each test part, we have shown the stacking plan for three values of the parameter w from (1). When $w = 0.1$, part c.g. height is penalized much more than floor space utilization and the reverse is true for $w = 0.9$. For $w = 0.5$, the two cost components are comparable. The planner chooses candidate orientations that can be grasped using suction cups and are promising candidates from stability point of view. For each candidate orientation, the planner uses procedure from Section V-A to compute a list of interference-free positions sorted in increasing order of the cost computed using (1). These candidate positions are then evaluated for stability and stacking plan feasibility. The parts are separated in the plane by at least 5 mm to account for robot positioning error. The only difference in implementation between this section and Section VII-A is that the free space is decomposed into convex polyhedral components, each of which is represented by a set of linear inequalities. The c-space obstacles are also represented using linear inequalities. This eases the task of accounting for robot positioning errors. Further, this increases speed and robustness of subtraction of a c-space obstacle from the set of permissible positions.

The stacking plans for the three values of parameter w are shown in Fig. 18. For $w = 0.1$, the preferred orientation is the one with the largest face acting as the base. Nesting occurs once floor space used becomes large enough for space utilization cost to become comparable with the cost of increasing part c.g. Nesting is also preferred by the planner as it strives to keep the stack compact using the two stage approach described in Section V-B. As w increases, the preferred orientation changes to reduce space utilization. Most of the parts rest on the smaller flange. All the parts are in this sideward orientation for $w = 0.9$. Nesting does not occur in this orientation for $w = 0.9$ as nested parts cannot be added to the stack from the top using pure translation. All the part orientations chosen present a horizontal face for grasping using suction cups. While the interference analysis module supplies a list of promising configurations, stability, grasping, and plan feasibility concerns dominate the choice of the final part configurations. Therefore, it is useful to have algorithms that can generate multiple interference-free configurations.

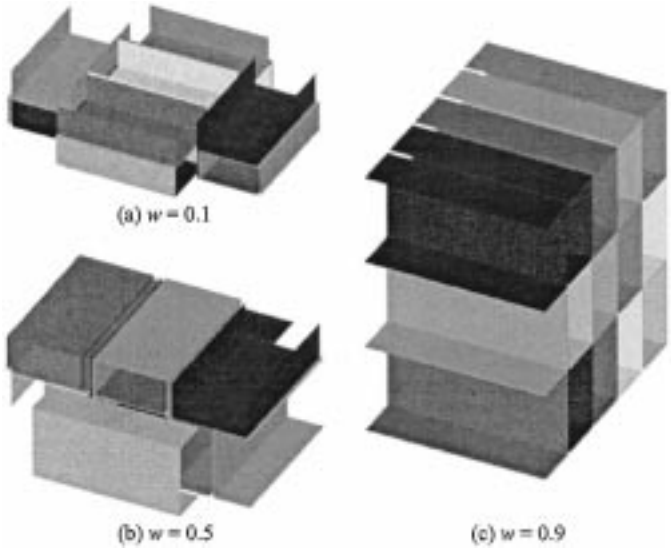
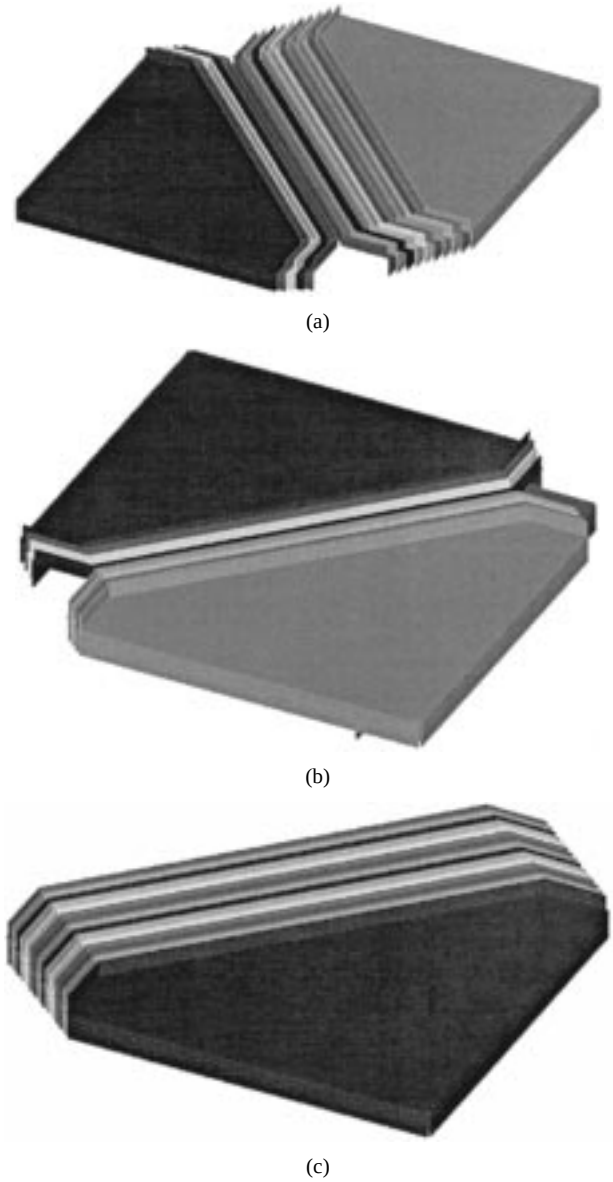


Fig. 18. Stacking plans for channel part.

The stacking plans for three values of w for Triangle Part are shown in Fig. 19. The planner shows a strong preference for nesting. For this part, nesting is preferred for low values of w because of the small increase in height of part c.g. when it is added to the stack. For large values of w, nesting is preferred because of the savings in floor space. The only effect of increasing w is the reduction from two nested stacks to one nested stack. This occurs because the planner is encouraged by the floor space utilization component of the cost function from (1) to move all parts as close to the floor space center (x_0, y_0, z_0) as possible. The orientation obtained by flipping the part upside down from the orientation shown in Fig. 19 results in a lower z-coordinate for the c.g. with the same floor space utilization. However, this upside down orientation is rejected because it is unstable when placed on the floor. Often for sheet metal parts, the orientation that minimizes the c.g. height may not be stable when placed on the floor or may not be supported contacts with other parts to render it stable. This justifies the presence of a stability check for all candidate configurations although the cost function has a component that is a measure of stability.

Fig. 20 shows the stacking plans generated for Complex Part. The tall flanges make this part a hard one to stack. For $w = 0.1$, most parts are placed on the floor with the largest face acting as a base face. Only a few parts are placed on top of the bottom layer of parts. This is due to the difficulty in accommodating the tall flanges while observing the separation distance required to account for robot positioning error. The planner is able to do this for only one part. For higher values of w, this orientation is not preferred as floor space is expensive and it is not possible



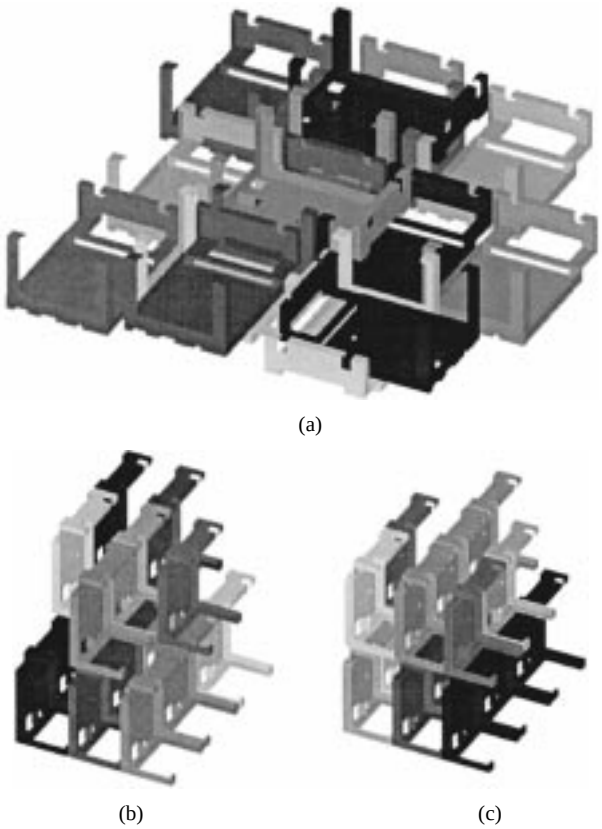
(a) (b) (c)

Fig. 19. Stacking plans for triangle part. (a) $w = 0.1$. (b) $w = 0.5$. (c) $w = 0.9$.

to stack parts in a column fashion because of the tall flanges. One drawback of the current cost function is reflected by the stacking plans for $w = 0.5$ and 0.9. The planner builds the stack as a set of columns. A stack with staggered parts is more stable than a set of columns.

C. Limitations

Our convex decomposition scheme triangulates concave faces and therefore produces an unnecessary large number of convex components. A more efficient convex decomposition scheme would help further speed up interference-free configuration computation. Currently we do not use part symmetry to prune out candidate orientations. This again would help us compute interference-free configurations faster without sacrificing solution quality. Our cost function computes floor space utilization for the parts individually and just sums them up. A cost function that looks at the properties of the stack as a whole might produce stacking plans that are closer to what a human would choose.



(a) (b) (c)

Fig. 20. Stacking plans for the complex part. (a) $w = 0.1$. (b) $w = 0.5$. (c) $w = 0.9$.

VIII. Previous Work

Configuration space (c-space) based approaches have been popular with researchers working in robot path planning, assembly planning, and mechanism design. A c-space is useful for abstracting away the geometry of the object of interest and focus on the property of interest, e.g., geometric interference and kinematic behavior, but it is expensive to compute. In this section, we discuss the application of c-space based techniques and what researchers have done to skirt around the problem of excessive computation time required for c-space obstacles.

The generation of a stacking plan involves determining interference-free configurations for all parts in a stack. We use an incremental approach to stacking. When a new part is added to an existing stack, interference-free configuration determination is performed in the c-space of the new part. Parts already in the stack are considered stationary obstacles. The desired configuration for the new part is one that minimizes a user specified cost function. This configuration may not be realizable because of interference with parts in the stack. Hence, it is required to compute an interference-free configuration that is as close as possible to the desired configuration. This is an instance of the *findspace* problem defined by [5].

Reference [5] addresses *findspace* and *findpath* problems. Findspace problem is the interference-free configuration determination for an object in a obstacle filled environment.

Findpath problem involves determining a interference-free path from a start to a goal configuration. Path planning in $\{p_x, p_y, \theta\}$-space is performed by computing slices that are unions of c-space obstacles in $\{p_x, p_y\}$ for small orientations. The slices are a conservative approximation of the real c-space obstacle and include all configurations leading to interference in a small orientation range.

A. Configuration Space and Motion Planning

C-space based planning has been used extensively for motion planning. Motion planning might involve computing an interference-free path for a robot or for a part to be added to an existing sub-assembly. If no such path exists, the planner should declare failure. The dimensionality of c-space is the number of degrees of freedom of the robot. For motion planning, the start and goal configurations are known beforehand. The challenge is computing a path that lies completely in free space for the robot or the part to be assembled. Complexity of computing a feasible path, irrespective of its optimality, has been shown to be exponential in the number of degrees of freedom by [8]. Hence, the focus in motion planning is on determination of the connectivity of the free space, not the more difficult problem of determination of topological and geometric details.

Computation of the stacking configuration involves determination of the goal configurations for all stacked parts. For stacking, we are interested in near-optimal configurations that minimize distance from the desired configuration. Hence, we need topological and geometric information along with connectivity information. Once the stacking plan is generated, assembly planning algorithms [9], [10] can be used to determine an interference-free path for the stack parts from a start configuration to their final configuration. Alternately, robot path planning algorithms can determine an interference-free path for a robot as it builds the part stack.

References [11]–[13] study path planning for robots in two and higher dimensions. The first paper performs path planning in $\{p_x, p_y, \theta\}$-space. The emphasis is on studying what happens to the connectivity of free space as the orientation changes. Free space connectivity information is essential to ensure that no collision occurs with an obstacle along the path.

Reference [14] presents a search algorithm for motion planning with six degrees of freedom. The planner is complete with a resolution. The emphasis is on finding a path in reasonable computational time. No attempt is made to optimize the path. Moves suggested by local experts are then implemented using a combination of pure translations and pure rotations. Metrics are provided to compute distance between two configurations in c-space. By using stability enhancing heuristics we have reduced the dimensionality of the search-space from six dimensional $\{p_x, p_y, p_z, \phi, \psi, \theta\}$-space to four dimensional $\{p_x, p_y, p_z, \theta\}$-space. The metric used by Donald might be useful if we want to include orientation also in the quadratic cost function used by the prototype planner.

Reference [15] performs path planning for an n degree-of-freedom manipulator by representing the c-space obstacles using $n-1$ dimensional slices. These slices are represented using slices in $n-2$ dimensions and so on. No attempt is made to characterize the c-space obstacle surfaces. The approximation is conservative and hence some feasible paths may not be detected. One advantage is that the free space is not divided into arbitrary subdivisions such as introduced by Donald. Instead the divisions represent the coherence of the free space. A feasible path is determined by looking at only a portion of the c-space. We are also able to compute good interference-free configurations by looking at a small portion of the c-space obstacle and are able to construct exact c-space obstacles in $\{p_x, p_y, p_z, \theta\}$-space.

References [16]–[18] use probabilistic methods to perform path planning. Kavraki *et al.* spend a lot of time building up information about the configuration space of a robot. Then, multiple queries for path from a given start to a given goal can be determined very fast. Hsu *et al.* research a assembly/disassembly problem where only a few queries may be made after building a portion of the c-space obstacle information. This approach is successful in finding paths through narrow channels for motion planning and assembly planning problems. However, no information is available about the geometric structure of the c-space obstacles. Hence, computing near-optimal interference-free configurations is not possible.

B. Configuration Space and Mechanism Design and Analysis

Configuration space is used in mechanism design for modeling, simulation, tolerance analysis and proposing alternative designs. References [19] and [20] survey some of the important work done in this area.

Mechanism modeling and classification require the complete characterization of one component of free space and its boundary. The component studied is the one containing the initial mechanism configuration. Mechanism simulation requires rapid incremental generation of c-space information. Joskowicz and Sacks present an algorithm for kinematic modeling of mechanisms [21]. Stacking requires the complete characterization of components of free space containing configurations positioning a part on the floor space or above it. We generate incremental c-space information to guide the search for an interference-free configuration, but often consider multiple free space components.

References [21] and [22] present a kinematic analysis algorithm for mechanisms containing higher pairs, such as door locks, gearboxes, and transmissions. The mechanism is decomposed into subassemblies. The behavior is described as a contact curve in 2D c-space of the subassembly. The curve partitions the c-space obstacle from the free region. The behavior of the mechanism is described as a composition of the 2D c-spaces of the constituent assemblies. The mechanism c-space is divided into regions that characterize its operating modes. An operating mode is defined by the contacts that exist between members of the assemblies. The dimensionality of the c-space of a mechanism with n assemblies, each with 2 degrees of freedom, is $2n$. However, the mechanism behavior has been captured by just studying n two-dimensional c-spaces. The dimensionality of c-space of a stack with n parts is $6n$. We determine near-optimal stacking configurations by analyzing n 3-dimensional c-spaces. Three of the parameters are orientations and are fixed using stability heuristics. This is achieved by incrementally adding parts to the stack.

References [23] and [24] use the c-space regions described above for kinematic simulation of mechanisms. The program takes a driving motion, internal forces, and time allotment as additional inputs and generates animation and an interpretation of the ensuing behavior. The focus is on rapid generation of partial c-spaces that are traversed by the mechanism in the course of the simulation. The user-provided driving motions determine which regions of the c-space are computed. For the stacking problem, the c-space regions to be computed are determined by checking for collision between convex components of the sheet metal parts using fast interference detection tools [4].

Mechanism design using c-space based techniques is more difficult than mechanism modeling and simulation. This is because transformation from a physical mechanism to its c-space is unique. However, the inverse transformation is not unique as many mechanisms with multiple parameter values can achieve the same kinematic behavior. Joskowicz and Sacks [19] have worked on interactive parametric design of mechanisms. They avoid the uniqueness problem by first modeling the c-space behavior of a given mechanism and modifying its parameters by small amounts to obtain a desired change in the c-space curve. Stahovich *et al.* [25] use qualitative c-space curves to propose alternate mechanisms to achieve the same kinematic behavior as a user-specified mechanism. The qualitative c-space curves approximate the real c-space curves and limit the number of designs to be considered while transforming back from c-space.

References [26] and [27] present an algorithm for worst-case and statistical kinematic tolerance analysis of mechanisms with parametric part tolerances. The kinematic variations are modeled as parametric surfaces in the mechanism c-space. C-space based tolerance analysis requires dividing the c-space into free zone, interference zone, and contact zone. In the free zone, there is no interference for all mechanism parameter values lying in the tolerance zone. Similarly, in the interference zone, there is interference for all mechanism parameter values. In the contact zone, interference exists only for certain parameter values. The inputs to the algorithm are the nominal motion path, the pair-wise contact zones of the interacting parts, and the parameter variations. The actual motion path is approximated by a sequence of c-space points. The output is the kinematic variation of each c-space coordinate at each path point. Currently, we account only for robot positioning errors while computing interference-free configurations. The stacking planner can be extended to use this representation to account for part tolerances also. Once the part configurations in a stack are determined, the effect of variation around the nominal configuration on interference and stability can be modeled.

C. Configuration Space Obstacle Computation

The success of configuration space based techniques for the applications discussed above hinges on computation of c-space obstacles in reasonable time. The particular application determines what is reasonable time. Some applications require rapid but approximate descriptions of the c-space. Others require an accurate description that can be computed off-line. Since our work deals with polyhedral parts, we will primarily discuss c-space obstacle computation for polyhedral objects.

For convex polyhedra, computation of a c-space obstacle in $\{p_x, p_y\}$-space and $\{p_x, p_y, p_z\}$-space can be performed in $O(n)$ and $O(n^2 \log n)$ time respectively [5], where n is the total number of vertices of the polyhedrons. One reason computation in these spaces is easy is that c-space obstacle corresponding to a polyhedron in the $\{p_x, p_y, p_z\}$-space of another polyhedron is polyhedral. This is not true once orientations are also considered. Hence obtaining a closed form description is far more difficult in c-space dealing with one or more of the parameters ϕ, ψ, and θ.

Reference [5] provides a conservative description of the c-space obstacle in $\{p_x, p_y, \theta\}$-space. The obstacle is represented as a set of slices. Each slice is a region in $\{p_x, p_y\}$-space representing union of positions resulting in interference for any orientation in the interval corresponding to that slice. We use stability enhancing heuristics to determine promising values of parameters ϕ and ψ. This reduces the c-space to be searched to four dimensional $\{p_x, p_y, p_z, \theta\}$-space. We use Lozano-Perez's algorithm to compute c-space obstacles in $\{p_x, p_y, p_z\}$-space. Further, we have an algorithm that characterizes the effect on the obstacle of rotating one of the objects about an axis. The input to the algorithm is the axis of rotation and the output is the orientation intervals within which the c-space obstacle topology stays the same. Within each interval the obstacle geometry for one orientation is computed. This information can be used to compute the obstacle for other orientations in the same interval. Therefore, by specifying the rotation axis as z-axis (see Fig. 4), we can obtain a closed form description of the obstacle in $\{p_x, p_y, p_z, \theta\}$-space.

References [11] and [13] study the effect of varying $\{\phi, \psi, \theta\}$ on the c-space obstacle in $\{p_x, p_y, p_z\}$-space and divide $\{\phi, \psi, \theta\}$ into noncritical and critical regions. In a noncritical region, the topology of the c-space obstacle stays the same. They also study the effect on a c-space obstacle in six-dimensional configuration space when a polyhedron is moving in a three-dimensional world with polyhedral obstacles. The significance of critical orientations is explained, but no algorithm is presented to compute the critical orientations.

Avnaim and Boissonnat [28] present a polynomial time algorithm for construction of configuration space obstacles for one set of planar polygons with respect to another set of planar polygons. In their work, a closed form description of the obstacle boundary is formulated as a transformation from a Φ-region. The intervals within which the region Φ is described by a constant analytic function is analogous to the orientation interval within which the topology of the obstacle in $\{p_x, p_y, p_z\}$-space stays the same. References [29] and [30] investigate the effect of change in θ on the c-space obstacle in $\{p_x, p_y\}$-space. The algorithm represents c-space obstacles in $\{p_x, p_y, \theta\}$-space as set of slices, each a region in $\{p_x, p_y\}$-space. It discretizes the c-space into intervals of equivalent slices separated by critical slices. In Chapter 4, we compute critical orientation ranges within which the c-space obstacle topology in $\{p_x, p_y, p_z\}$-space remains constant. The critical slices are analogous to these critical orientation ranges.

The algorithm developed by [6], [31] uses facet intersections to construct the obstacles in configuration space. Given two polygons as input, the algorithm computes the c-space obstacle

in $\{p_x, p_y, \theta\}$-space. The projection of the obstacle in a user-supplied direction is generated without constructing the actual obstacle. The representation includes a complete metric and topological description. This information is required to enumerate all the possible contact states between the polygons. This algorithm is polynomial in the number of vertices of the two polygons. There is however no straightforward way to extend this approach to higher dimensions. Information about contact states is useful for studying tangling of sheet metal parts. Parts are considered tangled when flanges of one part are jammed in holes or slots of another part and cannot be handled by a robot. A conservative method to identify tangle prone configurations is to check if small perturbations from a configuration lead to a change in the contact state.

D. Configuration Space Based Analysis for Sheet Metal Parts

Zussman and Horsch [32] present a method for extracting a bent part out of the press brake without colliding with the machine. The search for interference-free configurations is performed in three-dimensional discretized configuration space. The planning process is accelerated by identifying critical part profiles which are closest to the punch and die and hence, most likely to collide with the machine. The profiles are identified after decomposing the part into convex parts. They use a fast algorithm developed by [33] for computing distance between convex objects in three-dimensional space. An extension of this work could be used to perform path planning for the stacking robot once the stacking plan has been generated.

IX. Conclusion and Future Work

We have discussed two techniques to speed up the expensive step of c-space obstacle computation.

1) The first technique identifies orientation intervals within which topology of face-edge-vertex graph of a c-space obstacle topology for a pair of convex solids stays constant. Within this orientation interval, c-space obstacle geometry for one orientation can be extrapolated to obtain obstacle geometry for another orientation. In our experiments, extrapolation takes about 12% of the time it takes to compute an obstacle from scratch.
2) The second technique enables us to compute interference-free configurations for a pair of concave solids by partially constructing c-space obstacle geometry. This method works especially well for polyhedral sheet metal parts with a large base face and flanges. We have seen that for a sheet metal part with 79 convex components, less than 0.1% of the potential convex component pairs were evaluated to obtain an interference-free configuration.

We have used c-space based algorithms for interference analysis by a stacking planner. For every candidate orientation, the interference analysis module provides a list of interference-free positions sorted in increasing order of the cost. This list is searched for a position that satisfies stability, grasping and stacking plan feasibility concerns.

We need a method more efficient than triangulation for decomposing concave flanges of a sheet metal part. This would reduce the number of interference pairs we need to process to compute interference-free positions. One way to achieve fewer convex components is to partition the flange along lines that are parallel to edges of the flange and edges of the concavities and holes of the flange.

We also need a cost function that captures properties of the stack as a whole. At the same time, the cost function should be quadratic to enable use of closed-form quadratic optimization methods to compute interference-free positions for a candidate orientation. One way to achieve this objective is to have a stack cost function that uses the bounding box of the stack to measure floor space utilization and stack c.g. as a stability measure. The planner can be run for different values of w for the cost function currently used. The plan which gives the lowest cost as calculated by the stack cost function is then chosen as the final stacking plan.

Currently, the planner sometime builds stacks as a set of columns. The cost function should have a term that encourages the more stable alternative of staggering parts to increase the number of parts with which a part is in contact.

Appendix A
Configuration Space Terminology

The configuration parameters $\{p_x, p_y, p_z, \phi, \psi, \theta\}$ (see Fig. 4) of a rigid body define a six-dimensional *configuration space* (c-space). These six parameters describe the transformation from a global coordinate frame to a local coordinate frame attached to the rigid body. Every point in this space refers to a certain configuration of the rigid body in the real world.

The set of configurations of a rigid body A that result in interference with another rigid body B is called a *c-space obstacle* of B with respect to A denoted $C(A, B)$. Two bodies are considered to interfere if the volume of their intersection is nonzero. Hence bodies touching each other are not said to interfere. The set of configurations of A such that there is no interference between A and B is called the *free space* denoted $F(A, B)$. For any configuration of A lying in the interior of $C(A, B)$, there is penetration between the two bodies. For any configuration of A lying in the interior of $F(A, B)$, A and B are not in contact. For any configuration on the boundary separating $C(A, B)$ and $F(A, B)$, A and B are touching each other.

For example in Fig. 21, three positions of polygon A relative to polygon B are shown on the left hand side and the c-space obstacle $C(A, B)$ in $\{p_x, p_y\}$-space is shown on the right hand side. Position 1 lies on the boundary between the obstacle and free space, position 2 lies in the interior of the obstacle, and position 3 lies in the interior of the free space, i.e., outside the c-space obstacle.

Appendix B
Construction of C-Space Obstacle in $\{p_x, p_y, p_z\}$-Space

Consider the pair of convex polygons A and B in Fig. 22. B is stationary and A is allowed to translate, but not rotate. The polygons are shown on the left-hand side in world coordinates and the c-space obstacle corresponding to B in $\{p_x, p_y\}$-space is shown on the right hand side. The following discussion also applies to obstacles in $\{p_x, p_y, p_z\}$-space. The c-space obstacle $C(A, B)$ can be expressed as the Minkowski sum of B and A

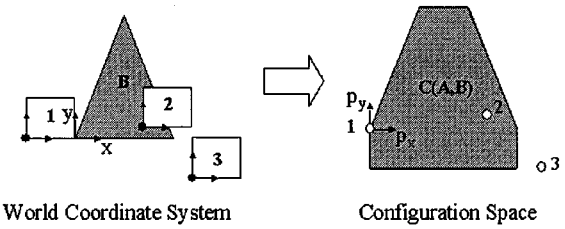


Fig. 21. Two polygons and their corresponding c-space obstacle.

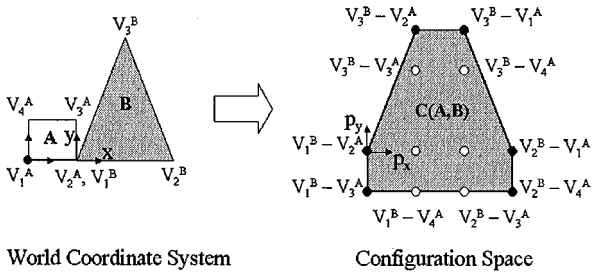


Fig. 22. Construction of c-space obstacle $C(A,\ B)$.

reflected about the origin of world coordinate frame in Fig. 22. Reference [5] shows that $C(A,\ B)$ is convex and can be constructed using vertices of A and B. The c-space obstacle is the convex hull of the points obtained by the pair-wise subtraction of vertices of A from vertices of B. Hence, we have the following equation:

$$C(A,\ B) = ConvexHull\left(\left\{P_{ij} \equiv V_j^B - V_i^A\right\}\right), \quad i = 1,\ 2,\ \ldots,\ n_A,\ j = 1,\ 2,\ \ldots,\ n_B \tag{12}$$

where V_i^A, $i = 1, 2, n_A$ are vertices of A and V_j^B, $j = 1, 2, n_B$ are vertices of B. This computation can be performed in $O(n_A \cdot n_B \cdot \log(n_A \cdot n_B))$ [3] time where n_A is the number of vertices of A, and n_B is the number of vertices of B.

Appendix C
Computation of Critical Orientation θ_C

Starting from (5), we show how to compute the critical orientation for a single point P_{ij} from Appendix B with respect to a face of the convex c-space obstacle. P_{ij} is defined in (12). Let F_0, F_1, and F_2 be three consecutive vertices of the convex face of interest. Since, these vertices are three of the extreme (black) points from Fig. 22, they are also formed by pair-wise subtraction of vertices F_0^A, F_1^A, and F_2^A of polygon A from vertices F_0^B, F_1^B, and F_2^B of polygon B respectively. Using (4), we can define the face normal $\mathbf{n}$ from (5) as

$$F_i^{BA}(\theta) = \left(F_i^B - R_{z,\theta} F_i^A\right), \quad i = 0,\ 1,\ 2$$
$$n = \frac{\left[F_1^{BA}(\theta) - F_0^{BA}(\theta)\right] \times \left[F_2^{BA}(\theta) - F_i^{BA}(\theta)\right]}{\left\|F_1^{BA}(\theta) - F_0^{BA}(\theta)\right\| \left\|F_2^{BA}(\theta) - F_0^{BA}(\theta)\right\|} \tag{13}$$

where $R_{z,\theta}$ is the rotation matrix from (4). We can then expand (5) as follows:

$$\begin{aligned}&\left[\left(V_j^B - F_0^B\right) - R_{z,\theta}\left(V_i^A - F_0^A\right)\right] \\ &\quad \bullet \left[\left(F_1^B - F_0^B\right) \times \left(F_2^B - F_0^B\right)\right. \\ &\qquad - \left(R_{z,\theta}\left(F_1^A - F_0^A\right)\right) \times \left(F_2^B - F_0^B\right) \\ &\qquad - \left(F_1^B - F_0^B\right) \times \left(R_{z,\theta}\left(F_2^A - F_0^A\right)\right) \\ &\qquad \left. + R_{z,\theta}\left(\left(F_1^A - F_0^A\right) \times \left(F_2^A - F_0^A\right)\right)\right] = 0.\end{aligned} \tag{14}$$

The rotation matrix is given by the following equation:

$$R_{z,\theta} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{15}$$

Using the above equation and the relationships

$$\sin(\theta) = \frac{2\tan(0.5\theta)}{1 + \tan^2(0.5\theta)} \quad \text{and} \quad \cos(\theta) = \frac{1 - \tan^2(\theta)}{1 + \tan^2(0.5\theta)}$$

we can simplify (14) to the following form:

$$At^2 + Bt + C = 0, \qquad t = \tan(0.5\theta) \tag{16}$$

where A, B, and C are constants. Solving this quadratic equation and taking the inverse tangent gives us the critical angle for one interior point for one face of the convex hull.

Acknowledgment

The authors are grateful to Mr. Ikeda for suggesting stacking of sheet metal parts as a research topic and K. Hazama for his valuable suggestions. Thanks are also due to Dr. C.-H. Wang for help with software development.

References

[1] Z. Li and V. Milenkovic, "Compaction and separation algorithms for nonconvex polygons and their applications," *Eur. J. Oper. Res.*, vol. 84, no. 3, pp. 539–561, 1995.

[2] V. R. Ayyadevara, "Automated planning for stacking polyhedral sheet metal parts," Ph.D. dissertation, Dep. Mech. Eng., Carnegie Mellon Univ., Pittsburgh, PA, 2000.

[3] J. O'Rourke, *Computational Geometry in C*, 2nd ed. New York: Cambridge Univ. Press, 1998.

[4] S. C. Gottschalk, M. C. Lin, and D. Manocha, "OBB-tree: A hierarchical structure for rapid interference detection," in *Proc. ACM SIGGRAPH Conf. Computer Graphics*, New Orleans, LA, 1996, pp. 171–180.

[5] T. Lozano-Perez, "Spatial planning: A configuration space approach," *IEEE Trans. Comput.*, vol. 32, pp. 108–120, 1983.

[6] R. C. Brost, "Analysis and planning of planar manipulation tasks," Ph.D. dissertation, Dep. Comp. Sci., Carnegie Mellon Univ., Pittsburgh, PA, 1991.

[7] C. B. Barber, D. P. Dobkin, and H. P. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Trans. Mathematical Software*, vol. 22, no. 4, pp. 469–483, 1996.

[8] J. F. Canny, *The Complexity of Robot Motion Planning*. Cambridge, MA: MIT Press, 1988.

[9] L. S. H. de Mello and A. C. Sanderson, "And/or graph representation of assembly plans," *IEEE Trans. Robot. Automat.*, vol. 6, pp. 188–199, Apr. 1990.

[10] R. H. Wilson, "On geometric assembly planning," Ph.D. dissertation, Department of Computer Science, Stanford University, Stanford, CA, USA, 1992.

[11] J. T. Schwartz and M. A. Sharir, "On the piano movers' problem I: A case of a two-dimensional rigid polygonal body moving amidst polygonal barriers," *Commun. Pure Appl. Math.*, vol. 36, pp. 345–398, 1983.

[12] ——, "On the piano movers' problem II: General techniques for computing topological properties of real algebraic manifolds," *Adv. Appl. Math.*, vol. 4, pp. 298–351, 1983.
[13] ——, "On the piano movers' problem V: The case of a rod moving in three-dimensional space amidst polyhedral obstacles," *Commun. Pure Appl. Math.*, vol. 37, pp. 815–848, 1984.
[14] B. R. Donald, "A search algorithm for motion planning with six degrees of freedom," *Artif. Intell.*, vol. 31, no. 3, pp. 295–353, 1987.
[15] T. Lozano-Perez, "A simple motion-planning algorithm for general robot manipulators," *IEEE J. Robot. Automat.*, vol. RA-3, pp. 224–238, June 1987.
[16] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, pp. 566–580, Aug. 1996.
[17] L. E. Kavraki and J-C. Latombe, "Probabilistic roadmaps for robot path planning," in *Practical Motion Planning in Robotics: Current Applications and Future Directions*, K. Gupta and A. de Pobil, Eds. New York: Wiley, 1998, pp. 33–53.
[18] D. Hsu, D. J-C., D. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," *Int. J. Comput. Geometry Applicat.*, vol. 9, no. 4–5, pp. 495–512, 1999.
[19] L. Joskowicz and E. Sacks, "Computer-aided mechanical assembly design using configuration spaces," Department of Computer Science, Purdue University, West Lafayette, IN, CS Tech. Rep. 97-001, 1997.
[20] ——, "Computer-aided mechanical design using configuration spaces," *IEEE Comput. Sci. Eng. Mag.*, vol. 1, pp. 14–21, 1999.
[21] ——, "Computational kinematics," *Artif. Intell.*, vol. 51, no. 1–3, pp. 381–416, 1991.
[22] E. Sacks and L. Joskowicz, "Computational kinematic analysis of higher pairs with multiple contacts," *ASME Trans. J. Mech. Design*, , vol. 117, no. 2A, pp. 269–277, 1995.
[23] D. A. Bourne, D. Navinchandra, and R. Ramaswamy, "Relating tolerances and kinematic behavior," Robotics Inst., Carnegie Mellon Univ., Pittsburgh, PA, CMU-RI-TR-89-102, Apr. 1989.
[24] E. Sacks and L. Joskowicz, "Automated modeling and kinematic simulation of mechanisms," *Computer-Aided Design*, vol. 25, no. 2, pp. 107–118, 1993.
[25] T. F. Stahovich, R. Davis, and H. Shrobe, "Generating multiple new designs from a sketch," *Artif. Intell.*, vol. 104, no. 1–2, pp. 211–264, 1998.
[26] E. Sacks and L. Joskowicz, "Parametric kinematic tolerance analysis of planar mechanisms," *Computer-Aided Design*, vol. 29, no. 5, pp. 333–342, 1997.
[27] ——, "Parametric kinematic tolerance analysis of general planar systems," *Computer-Aided Design*, vol. 30, no. 9, pp. 707–714, 1998.
[28] F. Avnaim and J. D. Boissonnat, "Polygon placement under translation and rotation," Institut National de Recherche en Informatique et en Automatique, Le Chesnay Cedex, France, INRIA Report #899, August 1988.
[29] E. Sacks and C. Bajaj, "Sliced configuration spaces for curved planar bodies," *Int. J. Robot. Res.*, vol. 17, no. 6, pp. 639–651, 1998.
[30] E. Sacks, "Practical sliced configuration spaces for curved planar pairs," *Int. J. Robot. Res.*, vol. 18, no. 1, pp. 59–63, 1999.
[31] R. C. Brost, "Computing metric and topological properties of configuration-space obstacles," in *Proc. IEEE Int. Conf. Robotics and Automation*, 1989, pp. 170–176.
[32] E. Zussman and T. Horsch, "A planning approach for robot-assisted multiple-bent profile handling," *Robot. Computer-Integrated Manufact.*, vol. 11, no. 1, pp. 35–40, 1994.
[33] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE J. Robot. Automat.*, vol. 4, pp. 193–203, Apr. 1988.

Venkateswara R. Ayyadevara received the Ph.D. degree in mechanical engineering from Carnegie-Mellon University, Pittsburgh, PA.

He is Director of Engineering at Evoxis Inc., Pittsburgh, and is heading software development for a system that delivers highly customized time-sensitive information for enterprises to a large customer base through audio. Prior to joining Evoxis, he served first as a Modeling Consultant and later as Senior Consultant at Trilogy, Inc., Austin, TX. He worked there in the area of Product Configuration systems for voice and data communication products. His research interests include CAD/CAM, process planning, geometric modeling, and robotics.

David A. Bourne received the B.S. degree in mathematics from the University of Vermont, Burlington, and the M.S. and Ph.D. degrees in computer and information science from the University of Pennsylvania, Philadelphia.

He is a Principal Scientist in the Robotics Institute at Carnegie Mellon University, Pittsburgh, PA, and has been researching artificial intelligence, robotics and manufacturing technologies for the last 25 years. He is the director of the Rapid Manufacturing Lab at Carnegie Mellon, which has built a large Flexible Manufacturing Cell for Westinghouse, the Intelligent Machining Workstation for the U.S. Air Force and the Intelligent Bending Workstation for Amada (the largest sheet metal machine tool company). He has written over 70 papers in areas applying artificial intelligence to manufacturing and holds 9 patents in related areas. In addition, he is the co-author of the book *Manufacturing Intelligence* (Boston, MA: Addison-Wesley) and he directed and produced a PBS short series on the topic of mass customization by applying high technology to manufacturing applications. He is also the current President and CEO of Design One Software, Inc.

Kenji Shimada received the B.S. and M.S. degrees from the University of Tokyo, Tokyo, Japan, and the Ph.D. degree from the Massachusetts Institute of Technology, Cambridge.

He is Associate Professor of the Department of Mechanical Engineering, the Robotics Institute, and Institute for Complex Engineered Systems at Carnegie Mellon University, Pittsburgh, PA. His research interests are in the areas of geometric modeling, computational geometry, computer graphics, CAD/CAM/CAE, and computer-assisted orthopedic surgery. Prior to joining Carnegie Mellon in 1996, he was Manager of Graphics Applications at IBM Research, Tokyo Research Laboratory. At Carnegie Mellon, he has explored a new physically-based approach to key geometric problems in engineering and medical applications, such as finite element mesh generation, interactive curve and surface design, and three-dimensional shape reconstruction. He holds 18 patents in the U.S., Japan, and Europe.

Dr. Shimada received the Yamashita Award from IPSJ and the Best Paper Award from Nicograph in 1994, the George Tallman Ladd Award for Excellence in Research from the Carnegie Institute of Technology in 1998, an NSF CAREER Award in 2000, and the IPSJ Best Paper Award in 2002. He is a member of ACM, ASME, and ASEE.

Robert H. Sturges, Jr. received the B.S. and M.S. degree from the Massachusetts Institute of Technology, Cambridge, and the Ph.D. degree from Carnegie-Mellon University, Pittsburgh, PA.

He has over 30 years of research experience in industry and academe including the development and teaching of integrated design/manufacturing methods, and the design and development of advanced automation equipment and robotics. His Wire Harness Flexible Manufacturing System is one of industry's first CAD driven systems producing finished wiring assemblies from raw materials. Early in his career, he pioneered the development master/slave remote robot systems for the nuclear service industry. He invented a design for assembly calculator and developed a quantification of dexterity. His current research interests include concurrent design-manufacturing systems for machining, assembly, and sheet metal processing. He holds 14 U.S. patents and has published over 90 papers in various international technical journals and conference proceedings.