Relative position computation for assembly planning with planar tolerated parts

Yaron Ostrovsky-Berman and Leo Joskowicz
School of Engineering and Computer Science
The Hebrew University of Jerusalem, Israel
Email: yaronber@cs.huji.ac.il

Abstract

This paper presents a new framework for worst-case tolerated assembly planning of planar mechanical systems. Unlike most assembly planners, which produce plans for nominal parts, our framework incorporates the inherent imprecision of the manufacturing process, which introduces uncertainties in the shape and size of the assembly parts. It accounts for the uncertainty of the relative part placements in the assembled state and allows planning for all assembly instances. Our framework is more general than existing approaches in terms of the part model, the tolerance specification model, and the type of motions used in the assembly sequences. We describe efficient algorithms for computing the sensitivity of part positioning to variations in part geometries, and for incorporating these computations into the geometric core of existing assembly planners for nominal parts. We show that the cost of accounting for tolerated parts in planning is a multiplicative factor which is a polynomial of low degree in the number of tolerance parameters. Our implementation and experiments on five assembly models show that tolerancing significantly reduces the volume of the space of valid motions for assembly sequence planning, and that for translational motions this reduction depends linearly on the size of the tolerance intervals. We conclude that geometric computation for assembly planning with tolerance parts is time efficient and practical.

Keywords: motion planning, assembly planning, geometric constraint solving, tolerance analysis, tolerance envelopes.

1 Introduction

Assembly sequence planning of mechanical systems has received significant attention due to its practical importance in manufacturing. Nearly all assembly planners produce plans for nominal parts, assuming that the part geometry, position, and orientation in the assembled state are fixed and known. However, the inherent imprecision of the manufacturing process introduces uncertainties in the shape and size of the assembly parts, which results in uncertainty in their relative placement in the assembled state. Thus, the nominal assembly plan may not be feasible for certain instances of the parts, and a valid plan for one instance may not be suitable for another.

Manufacturing variability is controlled through tolerances, which allow engineers to specify limits on part and assembly variability. A manufactured part instance passes inspection if its...
dimensions (lengths, radii, and angles) and geometric characteristics (form, profile, orientation, location, and runout) meet the design specifications. Since it is impossible to know in advance which instances of the parts will have to be assembled and to plan an appropriate assembly sequence, an assembly sequence for tolerated parts must be feasible for the parts that meet the design specifications.

To achieve this goal, the assembly planner must perform worst case tolerance analysis based on the geometry of the parts and their relative positioning relations. The geometry of the parts is affected by tolerancing on feature dimensions. A chain of related features accumulates tolerances such that the last feature in the chain can tolerate larger variations. This is called a stack up analysis of tolerances in parts. Similarly, the relative positioning of parts is affected by tolerances on the precision of the assembly operations. The last part in a chain of related parts accumulates tolerances due to imperfect placement of parts relative to each other and imperfect geometry of parts in the chain. The tolerance stack up in this case is on the translations and rotations that the parts have to undergo to satisfy the relative positioning relations.

From the perspective of mechanical engineering, the goal is to perform geometric worst case analysis of tolerance chains of two-dimensional elements (parts) with changing contacts and closed loops, for functionality analysis and assembly sequence planning. Chains of one-dimensional elements are well understood, and assemblies with open and closed loops of fixed contacts (joints) have been studied extensively, but assembly planning requires a two-dimensional tolerance stack up analysis of parts with varying contacts.

In this paper we present a framework for geometric worst case analysis of assemblies with tolerated parts. We introduce a constraint-based assembly specification which determines the relative positioning of parts with imperfect geometries. We analyze the sensitivity of part positions to variation in part dimensions, and perform geometric computations for assembly planning of tolerated parts that replace their counterparts for nominal geometries in many existing planners.

The framework is based on our previously developed parametric tolerancing model for planar parts whose boundaries consist of line and circular arc segments, and whose vertices are described by standard elementary functions of part dimensions, which vary within tolerance intervals. This model is general in its semantics and complies with a large subset of the international tolerancing standard [3]. It allows dependencies between neighboring vertices, and variations in lengths, radii, and angles, and therefore subsumes simpler tolerancing models used for assembly planning [8, 21].

We model the relation between parts in the assembly with geometric constraints based on distances between points, lines, and arcs. These are sufficient to model all types of clearance, contact, and angularity relations between part features. The relations constrain the degrees of freedom of parts such that for every instance of part geometry, there exists a unique rigid transformation for relative positioning. To model the relative position constraints in the entire assembly, we introduce the assembly graph, a generalization of Latombe’s relation graph [21] that includes closed chains, parts with three degrees of freedom, and conditional constraints, which are deliberate over-constrained specifications.

From the tolerance specification of the individual parts, and from the relative positioning constraints, we obtain the sensitivity of individual part positioning to the variation in assembly parameters (part dimensions and distance relations) in the form of sensitivity matrices. For each assembly part vertex, we compute the sensitivity matrix, which accounts for all possible variations of the vertex coordinates. Using the sensitivity matrices, we obtain the relative position envelopes and the configuration space envelopes of parts, which are the main geometric
tools for functionality analysis, and assembly planning algorithms, respectively.

We demonstrate our framework on algorithms based on the motion space approach for assembly planning [15]. We show how to replace the nominal motion space objects with their toleranced counterparts in probabilistic roadmap methods and in three useful assembly sequence types: single step infinite translations, infinitesimal rigid motions, and multiple step translations. Our analysis shows that the added complexity of accounting for tolerancing in assembly planning is a multiplicative factor, which is a polynomial of low degree in the number of tolerance parameters. In a series of experiments on five assembly models, we quantify the effect of tolerancing on the assembly by measuring the increase in blocked translational motions as a function of the tolerance interval size, and determine that the reduction in the volume of valid motions is linear in the size of the tolerance intervals. We conclude that geometric computation for assembly planning with tolerance parts is time efficient and practical.

This paper is organized as follows. In Section 2 we review work related to tolerancing, relative position computation, and assembly planning. In Section 3 we describe the relative positioning constraints and the assembly graph. In Section 4 we demonstrate the framework on the motion space algorithms for assembly planning. In Section 5 we discuss the implementation and experiments. We conclude in Section 6 with extensions and future work.

2 Related work

The current dimensioning and tolerancing standard [3] defines geometric tolerance zones for tolerated part features. The features must be contained in this zone, which is typically two parallel planes or two concentric cylinders. Since the feature can vary in an arbitrary manner within the tolerance zone, there are many ways to model the geometry of a valid part instance. The standard does not specify how the part should be represented for assembly tolerance analysis.

There are numerous models and representations of tolerated parts and features. Juster [18] surveys models developed prior to 1992. Models based on simple bounding volumes for vertex variation [1, 7, 9, 17, 21] have the advantage of allowing exact geometrical analysis of tolerance accumulation but cannot model parameter dependency induced by actual tolerance specifications and assembly relations. Requicha [26] and Rossignac [27] present a tolerancing model based on offsetting operations on solid model boundaries. Their models have a strong mathematical basis but depart from practices defined by current standards. Models of tolerated features based on rigid transformations of small displacements and rotations were recently developed to model variation within the geometric tolerance zone defined by the standard [6, 10, 28, 39]. These models were used to analyze chains of parts in the assembly. However, the acceptable domain for the transformation variables (called deviation space or tolerance map) is a high dimensional convex polytope, which necessitates exponential assembly instance samples for worst case analysis. The tolerancing model we use in this paper [24] is equivalent to the parametric tolerance specifications [2, 35] and complies with a large subset of geometric specifications [3] while allowing polynomial time worst case geometric analysis.

The relative position of imperfect planar parts was studied by Turner [38], who reduces the problem to solving a non-linear system of constraints for a given cost function. Sodhi and Turner [33] later extended this work for 3D parts. Li and Roy [22] show how to find the relative position of polyhedral parts with mating planes constraints. These methods compute the placement of a single instance of the assembly, and cannot be extended to analyze the entire variational class of the assembly. Inui et al. [17] propose a method for bounding the volume of the configuration space representing position uncertainties between two parts. However, their
Figure 1: Example of a simplified eight part planar mechanism with all types of contacts between parts (model 1).

method is only applicable to polygonal parts and is computationally prohibitive. Whitney [39] models relative positions and part variations using matrix transforms, and analyzes open chains of parts that deliver the key characteristics of the assembly. His approach is based on variation parameter space sampling. It requires a number of samples which is exponential in the number of parameters to perform worst case analysis. The relative position specification we introduce in this paper supports closed chains of parts consisting of line segments and circular arcs, and allows fast geometric analysis of the entire variational class of the assembly.

The general assembly planning problem is known to be PSPACE-hard, even for nominal parts [42]. With additional assumptions about the parts, motion types, or sequence types, efficient and practical algorithms have been developed (see [15, 30, 40] for surveys).

Very few works deal with assembly planning for toleranced parts. Thomas et al. [37] compute translational assembly sequences for polyhedral parts with uniform tolerances. They model geometric variation by scaling the parts by a constant factor and ignore the effect of tolerancing on the relative positioning of parts. Latombe et al. [21] present a simple tolerancing model in which polygonal parts vary in the distance of their edges from the part origin, but not in their orientation. The relative positioning of parts in this model serves as a basic step in computing the assembly plan [8]. The assembly planner is a special case of the motion space approach for two-handed single translations to infinity [15]. Latombe et al. acknowledge the limitations of their model and point to the need for developing a more general tolerancing model and for supporting other motion types.

3 Toleranced assembly specification

Assemblies of toleranced parts require a representation that accounts for part variations. The goal is to develop a framework within which part variations can be represented and efficiently computed. Our starting point is the general model of planar toleranced parts that we developed in previous research, which we briefly describe in Section 3.1. In Section 3.2, we describe the seven types of relative position constraints between two parts based on point, line, and arc distances. In Section 3.3, we introduce the assembly graph and describe an algorithm for computing the variability of part positions in the assembly. In Section 3.4 we describe the datum reference feature which serves as a global frame defining translation directions and a
3.1 Toleranced parts

We model part variation with the parametric tolerancing model described in [24]. This model is general, reflects current tolerancing practice, incorporates common tolerancing assumptions, and has good computational properties. In this model, part variation is determined by \( m \) tolerance parameter values \( p = (p_1, \ldots, p_m) \), specifying lengths, angles, and radii of part features. The parameters have nominal values and can vary along small tolerance intervals. The coordinates of the part vertices are standard elementary functions of a subset of the \( m \) parameters. An instance of the parameter values determines the geometry of the part. Figure 2 shows the tolerance specification of part \( P_6 \).

In [24], we describe algorithms for computing the outer and inner tolerance envelopes of individual parts, which are the boundaries of the union and the intersection of all possible part instances, respectively. The algorithms input the partial derivatives of the vertices according to the \( m \) tolerance parameters, and compute the envelopes under the linear approximation of the model. For a part with \( n \) vertices, the Best Segment Approximation (BSA) algorithm computes the most accurate tolerance envelope in \( O(nm^2) \) space and \( O(nm^2 \log m) \) time. The Independent Parameters Approximation (IPA) algorithm computes a conservative approximation of the envelope, based on the assumption that the endpoints depend on different parameters, in \( O(nm) \) space and \( O(nm \log m) \) time. Figure 2 shows the tolerance envelopes of part \( P_6 \) produced by both algorithms.

3.2 Relative position of two parts

The relative position of one part with respect to another is modeled with contact, clearance, and angle constraints. These constraints specify the location of the part boundaries with respect to each other. For planar parts consisting of line and arc segments, the constraints
describe how to position a part feature (vertex, edge, or arc) with respect to another feature or with respect to a datum (reference frame). Distance relations between these features are sufficient to model all types of contact, clearance, and angle constraints.

The constraints yield the possible variation in the position of part $B$ (the free part) relative to part $A$ (the fixed part) when $B$ is positioned according to the specification and the tolerance parameters of both parts span their allowed values. We say that we know the position of part $B$ relative to $A$ if we can determine, for any given instance of the parameters, the shape and orientation of $B$ with respect to a reference frame on $A$. Thus, the goal is to compute for each vertex $u$ of $B$ a transformation matrix that describes its sensitivity to variations in the parameters of parts $A$ and $B$. The $2 \times m$ sensitivity matrix $S_u$ has one column for each of the tolerance parameters. By applying the transformation $u + S_u p$, we obtain the correct position of $u$ for the instance of $p$, and from the positions of all the vertices, we obtain the shape and orientation of part $B$ in a global coordinate frame.

In Section 3.2.1 we describe the relative position constraints and their associated equations. In Section 3.2.2 we show how to solve the resulting system of equations and compute the sensitivity matrices of $B$. We give a simple example for sensitivity matrix computation in Section 3.2.3.

### 3.2.1 Relative position constraints

Planar part $B$ has three degrees of freedom, two for translation and one for rotation. Thus, to uniquely determine its position relative to $A$, three independent constraints are required. For each instance of the parts, there is a rigid transformation $T = (t_x, t_y, \theta)$ that positions $B$ relative to $A$ and satisfies the constraints. The precision required from tolerated parts is usually measured in terms of the ratio between the largest dimension and the smallest tolerance interval. This typically varies from $10^2$ (normal machining) through $10^4$ (precision machining) to $10^6$ (high-precision processes). Since the part variations are typically small, the translation offsets and rotation angle are also small. We can thus approximate the transformation with $\cos(\theta) \approx 1$ and $\sin(\theta) \approx \theta$.

The geometric entities participating in the constraints between parts are points, lines, and circular arcs. Distance constraints between these entities are sufficient to model all types of clearance, contact, and angularity relations. Following, we describe five common types of distance constraints, as well as two common types of deliberate over-constrained specifications: 1. vertex-line; 2. edge-line; 3. vertex-point; 4. arc-line; 4. arc-arc; 6. conditional edge-edge, and; 7. conditional arc-arc. For each constraint type we write in parentheses the number of degrees of freedom it constrains. The assembly in Figure 1 has all the seven types of constraints.

1. vertex-line constraint (1): this constraint is used to describe distance and angle relationships between two linear features. For example, in Figure 1, the flush relationship between the top of parts $P_3$ and $P_4$ is described with a zero-distance vertex-line constraint between the right vertex of part $P_3$ and the line supporting the vertical right edge of $P_3$. The vertex-line constraint equation is derived as follows. Let $v_i = (v_{ix}, v_{iy})$ be a vertex of $B$ and let $e_i$ be an edge of $A$ supported by the line $n_{ix} x + n_{iy} y + c_i = 0$, where $n_{ix}^2 + n_{iy}^2 = 1$. Since $B$ undergoes a rigid transformation, the coordinates of $v_i$ are functions of the original coordinates and the transformation variables: $w_{ix} = v_{ix} - v_{iy} \theta + t_x$ and $w_{iy} = v_{iy} + v_{ix} \theta + t_y$. To constrain the distance between $v_i$ and $e_i$ to be $d_i$, we write the equation $n_{ix} w_{ix} + n_{iy} w_{iy} + c_i = d_i$, or:

$$n_{ix} t_x + n_{iy} t_y + (v_{ix} n_{iy} - v_{iy} n_{ix}) \theta + v_{ix} n_{ix} + v_{iy} n_{iy} + c_i - d_i = 0 \quad (1)$$

6
which is linear in \((t_x, t_y, \theta)\). When \(e_i \in B\) and \(v_i \in A\), the vertices of \(e_i\), denoted \(e_{i1}\) and \(e_{i2}\), are transformed by \(T\), and the resulting distance constraint equation is:

\[
(c_{i2y} - e_{i1y})t_x + (c_{i1x} - e_{i2x})t_y + \left( (c_{i1x} - e_{i2x})v_{ix} + (e_{i1y} - c_{i2y})v_{iy} \right) \theta + \left( e_{i1y} - c_{i2y} \right) v_{ix} + (e_{i2x} - c_{i1x})v_{iy} + c_{i1z}c_{i2y} - e_{i1y}c_{i2x} - d_i = 0
\]  

(2)

2. **edge-line constraint** (2): this constraint is used to describe parallelism or angularity relationship between two edges. For example, in Figure 1, the contact relationship between the bottom edge of \(P_2\) and the upper horizontal edge of \(P_1\) is described with an edge-line constraint. The edge-line constraints are expressed as two vertex-line constrains, one for each vertex of the edge, with two Equations (1) or (2), depending on whether the edge belongs to \(A\) or \(B\).

3. **vertex-point constraint** (2): this constraint is used to model a two point contact which is equivalent to a pin joint with a single rotational degree of freedom. For example, part \(P_3\) connects to part \(P_1\) at the middle of its right vertical segment. After making this contact, \(P_3\) can only rotate about this point to satisfy the third constraint. A vertex-point contact constraint is expressed as two vertex line constraints. The two lines participating in the constraints are the axis-parallel lines intersecting at the desired point.

4. **arc-line constraint** (1): this constraint is used to describe clearance or contact relationships between an arc and an edge, such as the contact between parts \(P_3\) and \(P_2\) in Figure 1. An arc-line constraint entails a linear equation defining the distance of the arc center to the line supporting the edge as the required distance plus the arc radius. In our tolerancing model [24], circular arc segments are specified by the two endpoint vertices \(v_1, v_2\), and either the radius \(r\) or the arc angle \(\alpha \leq \pi\), all of which are functions of the tolerance parameters. The center and radius of the circle supporting the arc is derived from the relations: \(c = 0.5(v_1 + v_2) + v_{12}^{-1} \tan(0.5(\pi - \alpha))\) and \(r = |v_1 - v_2|/(2 \cos(0.5(\pi - \alpha)))\), where \(v_{12}^{-1}\) is counterclockwise perpendicular to \(v_2 - v_1\). To derive the constraint equation, we substitute vertex \(v_i\) in Equations (1) and (2) with \(c_i\), the arc center defined by the relation above, and set \(d_i\) to be \(d_i + r\). For example, Equation (1) becomes:

\[
n_{ix}t_x + n_{iy}t_y + (c_{ix}n_{iy} - c_{iy}n_{ix})\theta + c_{ix}n_{ix} + c_{iy}n_{iy} + c_i - (d_i + r) = 0
\]  

(3)

5. **arc-arc constraint** (1): this constraint is used to describe clearances or contacts between two arcs, which are common in mechanisms with rotating parts, such as the contact between parts \(P_3\) and \(P_5\) in Figure 1. The arc-arc constraint entails a quadratic equation setting the distance \(d\) between the two arc centers to the required distance plus either \(r_1 + r_2\), when the curvature vectors of the arcs face toward each other (case a), or \(|r_1 - r_2|\), when the curvature vectors of the arcs are in the same direction (case b), as in parts \(P_5\) and \(P_6\). Let \(c_1\) be the center of the fixed part arc and \(c_2\) be the center of the free arc, then the constraint is \((c_{2x} - \theta c_{2y} + t_x - c_{1x})^2 + (c_{2y} + \theta c_{2x} + t_y - c_{1y})^2 = d^2\). We collect the terms according to the transformation variables to get a quadratic equation in the transformation variables:

\[
t_x^2 + t_y^2 + (c_{2y} + c_{2x})\theta^2 - 2c_{2y}t_x \theta + 2c_{2x}t_y \theta + 2(c_{2x} - c_{1x})t_x + 2(c_{2y} - c_{1y})t_y + (-2(c_{2x} - c_{1x})c_{2y} + 2(c_{2y} - c_{1y})c_{2x})\theta + (c_{2x} - c_{1x})^2 + (c_{2y} - c_{1y})^2 - d^2 = 0
\]  

(4)
Figure 3: Conditional constraint cases. (a) contact between the secondary mating edge of A and the conditional edge of B. (b) contact between the interiors of the conditional arc of A and the arc of B. (c) contact between the interior of part B’s arc and conditional arc endpoint.

When the parts have the nominal geometry, more than three constraints can be satisfied simultaneously due to symmetry. When their geometry varies from the nominal, this symmetry breaks down and only three constraints may be satisfied simultaneously (assuming the parts are rigid, non-compliant). In practice, engineers often deliberately over-constrain the assemblies to specify desired relations when the symmetry breaks down.

The following two constraints model desired over-constraint in the assembly in the form of conditional constraints. The specification of conditional constraints allows additional constraints by defining a condition for determining which three of the constraints apply for a specific instance of the parts. In the nominal (symmetrical) case all constraints are satisfied simultaneously, and any variation in the part shapes either forces the assembly into a state where only three constraints can be satisfied or leaves the assembly in the same state. We distinguish between these cases based on the geometric condition.

6. **conditional edge-edge constraint** (1): this constraint is used to specify contacts between nominally parallel edges. In the nominal case, two parallel edges make contact in a line segment; in toleranced assemblies, the contact is usually a point. For example, consider parts $P_1$ and $P_2$ in Figure 1. The design intent is to make contact between both pairs of edges: first with the horizontal edges (which are wider and therefore provide more stable contacts), then with the vertical edges. Thus, the rotational degree of freedom of $P_2$ is over-constrained. For example, if topmost horizontal edge of $P_2$ is longer than nominal, then the top left vertex of $P_2$ will be in contact; if it is shorter, then the lower left vertex will be in contact. Both cases yield linear equations between the secondary mating edge’s supporting line and one of the two endpoint vertices of the conditional edge as in Equations (1),(2). The condition that distinguishes between the cases for a specific instance of the parts is based on the distance of the conditional vertices from the line. Figure 3(a) shows one of the two conditional cases.

7. **conditional arc-arc constraint** (1): this constraint is used to specify contact between arcs of equal radii. This nominal contact between arcs, as in parts $P_6$ and $P_8$ in Figure 1, is in a circular arc, but when the geometries vary, there are three possible solutions: contact between the interiors of the circular arcs, and contact between an arc and either the first or second endpoint of the other arc, termed the conditional arc. Figure 3(b,c) shows two of the conditional cases. The first case yields a constraint in the form of Equation (4), and the other cases entail a similar quadratic equation with the conditional arc vertices replacing the arc center. The condition that distinguishes between the cases is based on distances from the center of the first arc to the vertices of the conditional arc.
Part variations can break the symmetry in complex ways that are hard to analyze. For example, several dimensions may vary simultaneously with adverse effects on the geometric condition. Another complexity occurs when the state shown in Figure 3(a) is the nominal case: the upper horizontal edge of B has to shorten by at least some \( \zeta > 0 \) before the bottom left vertex makes contact instead of the top left vertex. In our model we assume that the nominal state is symmetric, or that \( \zeta \) is larger than all cumulative variations affecting the dimension (i.e. there are no ‘almost symmetrical’ cases). This assumption holds in most practical cases.

### 3.2.2 Computation of the sensitivity matrices

We now present a four-step algorithm to compute the vertex sensitivity matrices. The steps are: 1. model the relations between the two parts; 2. construct the corresponding system of equations; 3. compute the transformation relating the parts and its partial derivatives according to the tolerance parameters, and; 4. apply the transformation on the vertices of the free part to obtain the sensitivity matrix of each vertex.

In step 1, the relations are determined by three constraints of the seven types described in Section 3.2.1. In step 2, the equations are constructed according to Section 3.2.1, and stored in the following generic form: for linear equations (Eqs (1,2,3)), the form is: \( A_i t_x + B_i t_y + C_i = 0 \); for quadratic equations such as Equation (4), the form is: \( t_x^2 + t_y^2 + E_i t_x t_y + H_i t_x + I_i t_y + J_i \).

In step 3, we solve the system of generic equations constructed in step 2 by substituting the coefficients into general solution templates that we derived for the following three types of systems: 1. three linear equations; 2. two linear equations plus one quadratic equation; 3. two quadratic equations plus one linear equation. Note that three quadratic constraints (equivalent to the case where three circles have non-empty intersections), which do not occur when the geometries vary. The general solution to the system of three linear equations is:

\[
\begin{align*}
\text{denom} &= (A_2 C_1 B_3 - A_1 C_2 B_3 - C_3 A_2 B_1 - A_3 C_1 B_2 + C_2 A_3 B_1 + A_1 C_3 B_2) \\
t_x &= (B_1 C_3 D_2 - B_1 C_2 D_3 + C_1 B_2 D_3 - C_1 D_2 B_3 + D_1 C_2 B_3 - D_1 C_3 B_2) / \text{denom} \\
t_y &= (C_3 A_2 D_1 - A_2 C_1 D_3 - C_3 A_1 D_2 + A_1 C_2 D_3 + C_1 A_3 D_2 - C_2 A_3 D_1) / \text{denom} \\
\theta &= (A_3 B_1 D_2 - A_3 D_1 B_2 + B_3 A_2 D_1 - B_3 A_1 D_2 + D_3 A_1 B_2 - D_3 A_2 B_1) / \text{denom}
\end{align*}
\]

The system of equations resulting from \( 0 \leq z \leq 2 \) arc-arc (quadratic) constraints and \( 3-z \) linear constraints has \( 2^z \) solutions. However, only one of them corresponds to the nominal positions of the parts. The correct solution is identified by comparing the transformed vertices with the nominal vertex positions. The general solutions for systems with quadratic equations were derived using the symbolic computation software MAPLE, and are too lengthy to reproduce here.

For the partial derivatives of each of the template solutions, we derived corresponding templates, consisting of the coefficients and their partial derivatives. Since these were computed in...
step 2, the nominal solution $T = (t_x, t_y, \theta)$ and its derivatives $\partial T/\partial p_j = (\partial t_x/\partial p_j, \partial t_y/\partial p_j, \partial \theta/\partial p_j)$ are computed with a constant number of elementary arithmetic operations.

In step 4, we use the transformation derivatives to compute the sensitivity matrices of the vertices of $B$. Each vertex $u \in B$ undergoes the transformation $T$ in order to satisfy the relations. Thus, its nominal position is: $w_x = u_x - u_y \theta + t_x$ and $w_y = u_y + u_x \theta + t_y$. Its partial derivatives according to the tolerance parameters of both $A$ and $B$ define the sensitivity matrix $S_u$, whose $j^{th}$ column is:

$$\frac{\partial w}{\partial p_j} = \begin{pmatrix} \frac{\partial u_x}{\partial p_j} - \frac{\partial u_y}{\partial p_j} \theta - \frac{\partial \theta}{\partial p_j} u_y + \frac{\partial t_x}{\partial p_j} & \frac{\partial u_y}{\partial p_j} + \frac{\partial u_x}{\partial p_j} \theta - \frac{\partial \theta}{\partial p_j} u_x + \frac{\partial t_y}{\partial p_j} \end{pmatrix}^T$$

Note that the transformation of the vertices is sufficient to describe the segments of the transformed part, as line segments are fully defined by their endpoints, and the radii and angles of circular arcs remain constant under rigid body transformations.

When one of the constraints is conditional, the transformation $T$, which accounts for all possible conditional cases, is computed as follows. First, we solve the system of equations once for each of the cases (two or three solutions), and denote the resulting transformations $T_i = (t_{ix}, t_{iy}, \theta_i), 1 \leq i \leq 3$. An infinitesimal change in a single parameter $p_j$ either results in one of the solutions correct, or leaves all solutions correct. In the latter case, $\partial T/\partial p_j = \partial T_i/\partial p_j = \partial T_{i+1}/\partial p_j$. In the former case, we first determine which of the solutions is correct (by checking distance relations) for an infinitesimal increase (decrease) of $p_j$, and denote it by $T_j^+$ ($T_j^-$), respectively. We then compute the left-hand and right-hand derivatives as follows: $\partial T^+/\partial p_j = \partial T_i^+/\partial p_j$, and $\partial T^-/\partial p_j = \partial T_{i+1}^-/\partial p_j$, where $\partial T_i^+/\partial p_j$ ($\partial T_i^-/\partial p_j$) is the right-hand (left-hand) derivative of $T_j$. The algorithm for computing tolerance envelopes in [24] can be used with a straightforward modification to handle input divided into left and right-hand derivatives, at no asymptotic extra cost.

The tolerance envelope computed with the sensitivity matrices obtained from conditional constraints is conservative: it never underestimates the worst case variation. When the conditional constraint feature (edge or arc) depends on a single parameter, the envelope is an accurate bound on the variation. When the feature depends on two or more parameters, the envelope is overly conservative. This is a limitation of the linear approximation, which sums the effects of all the parameters and ignores parameter coupling. Accounting for parameter coupling results in exponential complexity in both the number of parameters $m$ and the length of the chain of related parts.

3.2.3 Example

We illustrate the relative position computation on parts $P_3$ and $P_4$ of Figure 1 where the tolerancing and relative position constraints are as shown in Figure 4.

In step 1, we specify an edge-line constraint and a vertex-line constraint designed to get a flush relationship between the parts (distances are zero). In step 2, we construct the equations based on the lines supporting the edges. The equations of lines $L_1$ and $L_2$ are, respectively: $(p_1 - p_2)x + 30y + 30p_2 - 60p_1 - 3000 = 0$ and $(p_2 + 65)x - 60p_2 - 3900 = 0$ (the line equations are not normalized, but since the constraint distances are zero this does not matter). Substituting the coefficients into Equation (1) we get the following system of linear equations:

$$\begin{align*}
(p_1 - p_2)t_x + 30t_y + (100p_2 - 100p_1 + 900)\theta - 30p_1 &= 0 \\
(p_1 - p_2)t_x + 30t_y + (100p_2 - 100p_1 + 1800)\theta - 30p_2 &= 0 \\
(p_2 + 65)t_x - (100p_2 + 6500)\theta &= 0
\end{align*}$$
Figure 4: Relative position constraints between parts $P_3$ and $P_4$. Distance constraints are denoted by two headed arrows (distances are zero). The horizontal edges are forced to overlap by an edge-line contact constraint, and their right endpoints are forced to coincide by a vertex-line contact constraint.

In step 3 we substitute the coefficients into Equation (5) to get the nominal solution \( \{t_x = 10/3(p_2 - p_1), t_y = 2p_1 - p_2, \theta = 1/30(p_2 - p_1)\} \). Notice that when \( p_1 = p_2 \), no rotation or horizontal translation is required. We substitute the coefficients and their derivatives into the solution derivative template and get: \( \frac{dT}{dp_1} = \left( \frac{10}{3}, 2, \frac{1}{30} \right) \) and \( \frac{dT}{dp_2} = \left( \frac{10}{3}, -1, \frac{1}{30} \right) \). Parameter \( p_3 \) has no effect on the transformation as it does not affect any of the constrained features. We illustrate step 4 on the vertex \( u = (50 + p_3, 110) \). Substituting the computed transformation derivatives and the vertex nominal coordinates and derivatives into Equation (6), we get the sensitivity matrix \( S_u = \begin{bmatrix} 1/3 & -1/3 & 1 \\ 1/3 & 2/3 & 0 \end{bmatrix} \).

3.3 Relative positions of parts in an assembly

We now describe how to model the relative position of parts in the entire assembly. Previous work by Latombe et al. [21] introduces the relation graph to describe the relative position constraints between nominal parts with two degrees of freedom each. We extend this graph to include cycles and support parts with general tolerances and three degrees of freedom, and call it the extended relation graph, or simply the assembly graph.

Graph nodes correspond to parts and undirected edges correspond to constraints between parts. Edge weights are 1, 2, or 3, indicating the number of degrees of freedom reduced by the constraints between the two parts. The edge data structure holds additional information about each constraint, such as the feature names of parts \( A \) and \( B \), the type of constraint, and the value or parametric expression of the distance between these features. Figure 5 shows the assembly graph of the assembly in Figure 1.

The assembly specification is properly constrained if it is both complete and non-redundant. The specification is complete if the relative position of all pairs can be determined from the constraints. It is non-redundant if the removal of any constraint results in incompleteness. A necessary and sufficient condition for a properly constrained assembly of \( N \) parts is that the sum of edge weights is \( 3(N - 1) \), and that for each cycle in the graph with \( N_c \) nodes, the sum of weights is \( 3(N_c - 1) \) and there is exactly one edge of weight 2 and one edge of weight 1 (a cycle with three edges of weight 2 results in a non-linear system of six equations with no general solution). The above conditions are a special case of the Gr"ubler equation for planar mechanisms [39]. Properly constrained assembly graphs have two important properties:
1. When two parts are connected by a chain of edges of weight 3, their relative position is determined link by link, where each link is solved as in Section 3.2. Such a chain of parts can be regarded as a single rigid part, because any rigid transformation on the parts as a group preserves the relation constraints.

2. A cycle of $N_c$ parts has exactly $N_c - 2$ edges of weight 3, one edge of weight 2, and one edge of weight 1. The last two edges divide the cycle into two disjoint sets of parts $X$ and $Y$ connected by the edges. The relative position between parts connected by edges of weight 1 or 2 cannot be determined because it is under-constrained, but the relative positions between the rigid bodies corresponding to $X$ and $Y$ is properly constrained. Figure 5 shows the sets $X,Y$ for the assembly in Figure 1.

Table 1 shows the algorithm to compute sensitivity matrices of part $P_j$ relative to $P_i$. Its inputs are the assembly graph and the tolerated part models, including initial vertex partial derivatives. The sensitivity matrices of the vertices of $P_j$ depend on the tolerance parameters of all the parts in the path from $P_i$ to $P_j$. The algorithm computes the relative position transformations between pairs of parts on the path from $P_i$ to $P_j$. The transformation $T_{XY}$ positioning the two sets of cycle parts is applicable for each part in $Y$. Since the parameters of a part participate in two transformations at most, there are at most two transformations which have non-zero $k^{th}$ columns in the sensitivity matrices of the vertices of $P_j$. The algorithm applies only the appropriate transformations when computing the matrix columns. The outputs of the algorithm are the sensitivity matrices $S_u$ for each $u \in P_j$. The position of $P_j$ relative to $P_i$ is determined by applying the appropriate $S_u$ on each vertex, and computing the coordinates of vertices in the frame attached to part $P_i$ (the position and orientation of the reference frame may change due to shape variations in $P_i$, and is determined from the sensitivity matrices of $P_i$, which belong to the original part input).

The complexity of the algorithm is as follows. Let $r_{ij}$ denote the number of parts in the path from $P_i$ to $P_j$ (including cycle parts). Let $q_k$ be the maximal number of tolerance parameters affecting a vertex of the $k^{th}$ part, and let $q = \max_k \{q_k\}$. At each iteration of step 2, we compute the transformation $T_{kl} = (t_x, t_y, \theta)$ between two parts and its partial derivatives according to the local tolerance parameters. There are up to 12 vertices participating in each set of constraints (when there are three arc-edge or arc-arc constraints), so the number of local parameters is at most $12q$. From Section 3.2, the solution for $T_{kl}$ has constant size, and so do each of its $O(q)$ derivatives. The $n_j$ vertices of $P_j$ depend on at most $12q r_{ij}$ tolerance
Input: Assembly graph, tolerated parts models, $P_i$, $P_j$.
Output: Sensitivity matrices $S_u$ of $P_j$’s vertices.

1. Find a path in the assembly graph from $P_i$ to $P_j$.

2. Iterate on the path edges $e = (P_k, P_l)$ in order:
   - If $weight(e) = 3$ then compute transformation $T_{kl}$ positioning $P_l$ relative to $P_k$.
   - Else if $weight(e) < 3$ (cycle edge) then
     a. Identify cycle edges $e_1, e_2$ of weight 1 and 2, respectively (Figure 5).
     b. Identify rigid bodies $X, Y$ bridged by edges $e_1, e_2$ ($X$ containing $P_k$, $Y$ containing $P_l$).
     c. Compute $T_{xi}$ positioning parts $P_x \in X$ relative to $P_i$.
     d. Compute $T_{yl}$ positioning parts $P_y \in Y$ relative to $P_l$.
     e. Compute the transformation $T_{XY}$ positioning $Y$ relative to $X$ according to constraints in $e_1, e_2$.
     f. Continue path from the exit edge (if it exists).

3. For each $u \in P_j$:
   For each tolerance parameter $p_k$ in parts from $P_i$ to $P_j$:
     a. Find the two transformations that depend on $p_k$.
     b. Apply each transformation on $u$.
     c. Compute $k^{th}$ column of $S_u$ using derivatives of transformations and shape sensitivity matrices.

4. Return $S_u$ for all $u \in P_j$

Table 1: Algorithm for computing sensitivity matrices of $P_j$ relative to $P_i$

The complexity is linear in the size of the input.

Figure 6: Tolerance envelope and relative position envelope of part $P_6$.

parameters, so the computation of their sensitivity matrices in step 3 takes $O(n_j q_{rij})$ time. The complexity is linear in the size of the input.

Note that this result is a generalization of the result of Cazals and Latombe [8] for parts with two degrees of freedom. In their tolerancing model, only translations are used to satisfy the constraints. This means that all the vertices of $P_j$ are translated uniformly, and the translation depends on $O(q_{rij})$ parameters. The tolerance zone of the translation is a convex polygon inside which the origin of part $P_j$ varies. It can be computed in $O(q_{ij} \log(q_{rij}))$ time (see [24]), and since $q$ is constant (at most six), this compares with their result. Note also that in their model, the vertices are linear functions of the tolerance parameters, so the linear approximation is in fact exact.
Figure 6 shows the nominal shape of part $P_6$ (innermost curve), its tolerance envelope as a result of geometric variations within the tolerance specification (middle curve), and its relative position tolerance envelope (outermost curve) which accounts for geometric variations of all the assembly parts affecting the position of $P_6$ relative to the global reference frame. The relative position envelope is computed with the sensitivity matrices of the part. It is useful in tolerance analysis of key characteristics of the assembly [39], which determine its functionality.

3.4 The datum frame

Datums are features, such as points, lines, and planes, from which the location, or geometric relationship of other part features may be established. Datum specification is an integral part of the tolerancing standard [3]. In our framework, the datum may be an edge of a part with a fixed origin and orientation, or it may be a reference frame which does not belong to the assembly. In either case, the datum is represented as a node in the assembly graph. The datum serves as a global coordinate frame which defines translation directions and a rotation axis for assembly sequence motions. It is therefore important to compute the position of all the parts relative to the datum as a function of the tolerance parameters. In the rest of the paper we denote the datum by $P_0$.

Observe that in order to know the position of all the parts relative to each other, it is sufficient to compute the sensitivity matrices of all the parts relative to the datum, because this computation requires finding the transformations linking all the edges of the assembly graph. Since the transformation between two neighboring parts $P_i$ and $P_j$ (or rigid bodies $X$, $Y$, see Section 3.3) is the inverse of the transformation from $P_j$ to $P_i$, the sensitivity of any part relative to another can be determined by following the path between them and using the precomputed transformations (in fact, the relative positioning can be determined directly from the sensitivity matrices, but this requires more care in the implementation).

Let $I = \{p_1, p_2, \ldots, p_m\}$ be the set of all the tolerance parameters, and consider the path from the datum $P_0$ through $P_i$ to $P_j$. We separate the tolerance parameters into two sets: $I_{ij} \subseteq I$ contains the parameters affecting the shapes of the parts from $P_i$ up to and including $P_j$, and $I_{0i} \subseteq I$ contains parameters affecting the shapes of the rest of the parts in the path. As noted in Section 3.3, variations in the parameters of $I_{0i}$ cause the parts from $P_i$ to $P_j$ to undergo an identical rigid transformation. Observe that when two parts $A$ and $B$ undergo an identical rigid transformation $(t_x, t_y, \theta)$, their configuration space (C-space) obstacle, $B \cap A = \{b-a|a \in A, b \in B\}$, rotates in angle $\theta$ with respect to the origin, but does not translate.

From this observation we obtain an efficient method for computing the effect of parameters in $I_{0i}$ on the relative position of two parts. Let $\theta_{k1}$ and $\theta_{k2}$ be the rotations in the two rigid transformations relating variations in $p_k \in I_{0i}$ to the positions of $P_i$ and $P_j$, and let $\delta_k$ be the maximal variation of the parameter $p_k$, then the maximal rotation angle as a result of combined variations of parameters in $I_{0i}$ is $\theta_{max} = \sum_{p_k \in I_{0i}}(|\frac{\partial \theta_{k1}}{\partial p_k} + \frac{\partial \theta_{k2}}{\partial p_k}|)\delta_k$. We term it the cumulative rotation angle.

4 Geometric computation for assembly planning

We now show how to incorporate our framework into existing assembly planners to support tolerated parts. Assembly planning algorithms are typically categorized in terms of the number of robotic hands (two-handed or $m$-handed), the number of allowed steps in removal of a sub-assembly (single or multiple steps), the motion type (translational, rotational, infinitesimal),
and the sequence type (linear and/or monotone).

In his PhD thesis, Wilson [40] describes a general approach for assembly planning. To efficiently search the space of disassembly sequences, his method searches in restricted spaces of increasing complexity, and, if all else fails, applies the general motion planning algorithm, whose complexity is exponential. His method was later generalized into the motion space approach [15].

In the motion space approach, assembly motions are parameterized such that each point in motion space represents a mating motion that is independent of the moving part set. The geometry of the assembly partitions the motion space into an arrangement of cells such that within each cell, the blocking relations between parts remains fixed. The arrangement is called the Non Directional Blocking Graph (NDBG), and the blocking relations of each cell is called the Directional Blocking Graph (DBG). The algorithm searches the NDBG for a DBG with a strongly connected component, which corresponds to a subassembly that can be separated from the other parts. The motion space approach is applied to one-step translations, infinitesimal rigid motions, and multi-step translations.

In the toleranced case, we replace the nominal motion space obstacles with their toleranced counterparts, which are the tolerance envelopes of the obstacles computed with the sensitivity matrices of their vertices. We obtain the strong NDBG, which accounts for all geometric and relative positioning variations, and enables planning for all feasible instances.

In Section 4.1, we show how to use the relative position envelopes of parts in workspace to analyze the assembly. In Sections 4.2, 4.3, 4.4 we demonstrate our method on the three motion space algorithms. In Section 4.5 we show how to adapt the probabilistic roadmap methods to toleranced parts. In Section 4.6 we compare the complexities of the geometric computations in the nominal and toleranced cases.

4.1 Relative position envelopes in workspace

Since relative position envelopes account for all shape and position variations of parts, it seems natural to replace the parts in workspace with their relative position envelopes for worst case tolerance analysis of assembly sequencing. This approach is overly conservative because the envelopes of two parts may account twice for shared parameters that possibly cancel each other partially or completely. Figure 7 demonstrates how these conservative envelopes lead to incorrect analysis.

To correctly account for variations due to parameters shared by a pair of parts $P_i$ and $P_j$, their effect on the sensitivity matrices of the vertices must be combined. This is done in the C-space of the parts as follows. For each pair of segments $s_k \in P_i$ and $s_l \in P_j$, compute the C-space obstacle. Vertices of the C-space obstacle are of the form $v_l(p) - v_k(p)$, where $v_l$ is an endpoint of $s_l$, $v_k$ is an endpoint of $s_k$, and $p$ is the tolerance parameter vector. Thus the sensitivity matrices of the obstacle can be computed from the sensitivity matrices of the endpoints of $s_k$ and $s_l$, and they combine the effect of parameters shared by both segments. In the example of Figure 7, the sensitivity matrices of the C-space obstacle defined by the contacting vertical edges have zero columns for the parameter $p_1$ because the effects cancel each other out (the matrices of other pairs of segments are not entirely zero).

The tolerance envelopes of the C-space obstacles correctly bound the space occupied by obstacles defined by all assembly instances, and we use them in the assembly planning algorithms of Sections 4.2, 4.4, and 4.5.

Relative position envelopes of parts in workspace are nevertheless very useful, since many geometric calculations, such as finding separating directions or intersection of envelopes, are
Figure 7: Example for incorrect analysis using relative position envelopes. (a) Tolerance and relative position specification of two parts. Part $P_1$ is toleranced such that its right vertical edge can translate horizontally. Part $P_2$ is not toleranced, but is constrained to be in contact with part $P_1$. (b) The relative position envelope of part $P_1$ (dotted) and of part $P_2$ (dashed) overlap, because the dependency in parameter $p_1$ was considered twice. This implies that there is an instance for which these parts overlap, which is incorrect.

much more efficient when computed in workspace (Sections 4.2, 4.5). To make the analysis correct, we compute the tolerance envelopes of the workspace parts using only parameters that are independent, and then augment these envelopes based on the shared parameters.

Let $P_i$ and $P_j$ be two parts connected by a path in the assembly graph. Then according to Section 3.4, parameters in $I_{ij}$ affect the shape of parts in the path from $P_i$ to $P_j$. We further divide this set as follows: parameters in $I_{ij}^{shr} \subseteq I_{ij}$ have a non-zero column in at the sensitivity matrix of at least one vertex of $P_i$ and at least one vertex of $P_j$ (shared parameters), and the rest of the parameters, $I_{ij}^{ind} = I_{ij} \setminus I_{ij}^{shr}$ (independent parameters). This classification of parameters is similar to (but slightly different from) the classification of Latombe et al. [21].

Note the following observations:

- Parameters in $I_{ij}^{shr}$ affect both the shape of $P_i$ and the position of $P_j$, while the parameters in $I_{ij}^{ind}$ affect the shapes of either $P_i$ or $P_j$, or affect the position of $P_j$, but not both.

- Parameters in $I_{ij}^{shr}$ appear only in the tolerance specification of $P_i$.

- Parameters in $I_{ij}^{shr}$ affect the coordinates of the vertices participating in the relative position constraints from $P_i$ to the next part on the path to $P_j$.

- The vertices of $P_i$ that depend on parameters in $I_{ij}^{shr}$ are the (at most) six vertices participating in the constraints, plus vertices related to them in the tolerance specification by the same parameters. We call segments of $P_i$ that contain at least one such vertex shape-position segments and denote them by $S_{ij}^{sp}$ and denote by $g_{ij}$.

In general, all the vertices of $P_i$ can depend on the parameters in $I_{ij}^{shr}$. But in practice $g_{ij}$ is a small constant since long chains of toleranced dimensions are discouraged. To sum up: for each pair of parts $P_i$ and $P_j$, the tolerance parameters’ index set $I = \{1, 2, \ldots, m\}$ is divided into subsets as follows: $I = I_{0i} \cup I_{ij}$, where $I_{ij} = I_{ij}^{ind} \cup I_{ij}^{shr}$.

From these observations we obtain a method for computing the tolerance envelopes of parts $P_i$ and $P_j$ that account for all the independent parameters: for both parts, compute the envelope using only the columns of the sensitivity matrices which correspond to parameters in
**Input:** Assembly graph, source and target parts $P_i, P_j$.

**Output:** $C_{ij}$ - cone of translation directions of $P_i$ blocked by $P_j$.

1. a. Find the path from the datum $P_0$ that goes through $P_i$ and $P_j$.
   b. Identify the parameter sets $I_{0i}$, $I_{ij}$, $I_{ij}^{ind}$, and $I_{ij}^{shr}$ (Sections 3.4, 4.1).
   c. Identify the shape-position segment set $S_{ij}^{sp}$ (Section 4.1).
2. Compute the relative position transformations from $P_0$ to $P_i$.
3. Compute the cumulative rotation angle $\theta_{max}$ for parameters in $I_{0i}$.
4. Compute $E_i, E_j$ - relative position envelopes of $P_i$ and $P_j$ from parameters in $I_{ij}^{ind}$.
5. Check if $E_i$ and $E_j$ intersect, and if so, return the entire unit circle.
6. Compute $C_{ij}$ - cone of translation directions of $E_i$ blocked by $E_j$.
7. For each segment $s_k \in S_{ij}^{sp}$ and for each segment $s_l \in P_j$ do:
   a. Compute the parametric C-space obstacle $s_l \ominus s_k$.
   b. Compute the outer tolerance envelope of $s_l \ominus s_k$ from parameters in $I_{ij}$.
   c. Compute $C_{kl}$ - blocking directions of the obstacle envelope.
   d. Add $C_{kl}$ to the cone $C_{ij}$.
8. Augment $C_{ij}$ by $\theta_{max}$ clockwise and counterclockwise, return $C_{ij}$.

Table 2: Algorithm for computing the blocked translation directions

Figure 8: Illustration of the blocking directions of two line segments $s_l$ and $s_k$ (step 8 in Table 2). For clarity, the tolerance envelope of the C-space obstacle is exaggerated.

$I_{ij}^{ind}$. In Sections 4.2, 4.5 we show how to augment these envelopes to correctly account for all the tolerance parameters.

**4.2 One step infinite translations**

The assembly planning problem for two-handed monotone sequences of one step translations to infinity was solved by Latombe et al. for toleranced polygons with fixed edge orientations [21]. The planner computes the strong NDBG to find separating translation directions that are feasible for all part instances. The geometric primitive in this method is the computation of the cone of translation directions for which part $P_i$ is blocked by part $P_j$.

Table 2 presents the algorithm for computing the cone of translation directions of part
P_i blocked by part P_j when the assembly parts are toleranced with our tolerancing model and all the assembly parameters span their tolerance intervals. The algorithm first finds a path from the datum to the two parts, and classifies the parameters and segments as follows: \( I_{0i} \) - parameters which cause both parts to undergo an identical rigid transformation; \( I_{ij} \) - parameters that affect each part differently; \( I_{ij}^{shr} \) - parameters which affect both the shape of \( P_i \) and the position of \( P_j \); \( I_{ij}^{ind} \) - parameters which cause independent variations in the shape and position of \( P_i \) and \( P_j \); \( S_{ij}^{sp} \) - segments of \( P_i \) depending on parameters in \( I_{ij}^{shr} \). In step 2, the algorithm computes the transformations from the datum to \( P_i \) in order to find the cumulative rotation angle of the C-space obstacle \( P_j \oplus P_i \) effected by parameters in \( I_{0i} \). Next it computes the sensitivity matrices of part \( P_j \) relative to \( P_i \) (Section 3.3).

From these matrices, the algorithm computes the relative position envelopes of \( P_i \) and \( P_j \), which account for variations in all independent parameters. If these envelopes intersect, then there does not exist a direction which separates these parts under all possible instances of the assembly, which means there is no feasible assembly plan. The algorithm returns the entire unit circle, which signifies this problem. If the envelopes do not intersect, then the algorithm computes the cone of blocked directions between the two envelopes.

To do this, the algorithm uses the method of Sack and Toussaint [29], modified to handle the conic sections that appear as segments in the tolerance envelopes. The algorithm for computing separating translations for polygonal parts requires three major geometric computations: finding the convex hull of a simple polygon, finding the visibility region of a polygon edge, and processing for efficient point location. The extension to curved parts utilizes the algorithms developed by Dobkin and Souvaine [11] and by Bajaj and Kim [4]. The tolerance envelope is a simple special case of the splinagon introduced in [11]. The convex hull of the envelope is computed with the efficient algorithm of [4]. The visibility of the hull edges are computed with the bounding polygon approach of [11], and the point location query structure combines the hierarchy approach of Kirkpatrick [20] with the carrier polygon method for splinagons [11]. The computed cone of blocked translations accounts only for independent parameters, and is expanded in the iterations of step 8.

The algorithm iterates on the shape-position segments \( s_k \) of part \( P_i \), and for each of them considers all the segments \( s_l \) of \( P_j \). For each pair of these segments the algorithm computes the configuration space obstacle, as explained in Section 4.1 and illustrated in Figure 8. It then scans the envelope from the origin of the configuration space to obtain the cone \( C_{kl} \) of translation directions of \( s_k \) blocked by \( s_l \) (step 8c). This cone accounts for all the parameters affecting the two segment lengths and orientations, including parameters in \( I_{ij}^{shr} \). It is either contained in \( C_{ij} \), in which case \( C_{ij} \) remains unchanged, or it only partially overlaps with \( C_{ij} \), in which case it expands \( C_{ij} \) clockwise and/or counterclockwise. Finally, the algorithm accounts for parameters in \( I_{0i} \) by augmenting \( C_{ij} \) with the cumulative rotation angle both clockwise and counterclockwise. The output is a cone \( C_{ij} \) representing directions \( d \) for which there exist instances of \( P_i \) and \( P_j \) such that translation of \( P_i \) in \( d \) is blocked by \( P_j \). If \( C_{ij} \) is the entire unit circle, then there is no feasible assembly sequence.

We now analyze the complexity of the algorithm. Let \( n_i \) and \( n_j \) be the number of vertices in parts \( P_i \) and \( P_j \), respectively, let \( r_{ij} \) be the length of the path from \( P_i \) to \( P_j \), and let \( q \) be the maximal number of tolerance parameters affecting a single vertex of a part in the assembly. Step 1 takes \( O(r_{0i} + r_{ij}) \) time. Computing the transformations in step 2 and the cumulative rotation angle in step 3 takes \( O(qr_{0i}) \) time. Step 4 requires \( O(n_j q r_{ij}) \) time. Using the BSA algorithm [24], step 5 takes \( O(n_j q^2 r_{ij}^2 \log n + n_j q^2 r_{ij}^2 \log(qr_{ij})) \), and produces envelopes \( E_i \) and \( E_j \) of size \( O(n_i q^2) \) and \( O(n_j q^2 r_{ij}^2) \), respectively. Checking whether two planar objects intersect takes \( O(\tilde{n} \log \tilde{n}) \), where \( \tilde{n} \) is the size of the objects. The modified algorithm of Sack
and Toussaint [29] runs in \( O(\bar{n} \log \bar{n}) \) time. Let \( n_{ij} = \max\{n_i, n_j\} \), then steps 6 and 7 run in \( O(n_{ij}q^2r_{ij}\log(n_{ij}rq_{ij})) \) time.

Step 8 is repeated \( g_{ij}n_{ij} \) times, where \( g_{ij} \) is the number of shape-position segments of \( P_i \).

Step 8a takes constant time while step 8b depends on the type of segments in the current iteration. When both segments \( s_k \) and \( s_l \) are line segments, it suffices to compute the IPA envelope approximation of the obstacle (Section 3.1) because the difference between the IPA and the BSA envelope is a pocket, and the ray emanating from the origin cannot enter and exit this pocket without colliding with the obstacle (the cones corresponding to edges for which configuration space origin lies in the pocket are subsumed by the cones of the rest of the edges, see Figure 8). Otherwise, the BSA algorithm must be used to compute the envelope. For simplicity we make a conservative analysis treating all segments as circular arcs. Thus, step 8c takes \( O(q^2r_{ij}^2 \log(qr_{ij})) \) time, and steps 8d and 9 take constant time. The overall complexity of the algorithm is \( O(qr_{ij} + n_{ij}(1 + g_{ij})q^2r_{ij}^2 \log(n_{ij}qr_{ij})) \).

To construct the NDBG, we compute the cones of all pairs of \( N \) parts and then sort them on the unit circle. The combined complexity of the relative position and NDBG computation is \( O(N^2\log N + N^2n(1+g)q^2r^2\log(nrq)) \), where \( n = \max_{i\in\text{parts}}\{n_i\} \), \( r = \max_{i,j\in\text{parts}}\{r_{ij}\} \), and \( g = \max_{i,j\in\text{parts}}\{g_{ij}\} \).

For polygonal parts with two degrees of freedom, as in [21], all the segments in step 8 are line segments so the more efficient IPA algorithm is used. In addition, since all the vertices undergo the same positioning translation, it suffices to compute the tolerance envelope of the translation once for every pair of parts, and the running time reduces to \( O(N^2\log N + N^2r(n(1+g) + q\log(nrq))) \). In the restricted model of [21], both \( q \) and \( g \) are considered constants, so the running time becomes \( O(N^2\log N + N^2r(n + \log(nrq))) \), which improves on the quadratic complexity of \( n \) in [21]. The improvement is due to the use of the algorithm of Sack and Toussaint [29].

4.3 Infinitesimal rigid motions

As observed in [14, 40], when all the parts of the assembly are in contact, an effective way of pruning the space of disassembly motions is to consider infinitesimal rigid motions first. If the infinitesimal motion of a set of parts is unblocked, then the extended motion of the same set is checked for validity. The assembly planning algorithms in [14, 40] consider only contacts between linear features (points, lines, and planes), and do not readily extend to curved parts, since the resulting inequalities are non-linear. We also restrict the current description to polygonal parts.

Let \( A \) and \( B \) be two parts in contact defined by a zero-distance constraint as in Section 3.2. The infinitesimal motion of \( B \) is represented by a unit vector \( dX = (dx, dy, d\theta)^T \), where rotations are with respect to the datum’s global frame. This motion causes a vertex \( v = (v_x, v_y) \in B \) to undergo a translation \( t_v = J_v dX \), where \( J_v = \begin{pmatrix} 1 & 0 & -v_y \\ 0 & 1 & v_x \end{pmatrix} \) is the Jacobian relating the differential motion of part \( B \) to the motion of \( v \). Figure 9 shows the contact between parts \( P_1 \) and \( P_3 \) nominally and after two possible variations.

Let \( e \in A \) be an edge with normal \( n_e \). Then vertex \( v \) undergoing infinitesimal motion \( dX \) will not collide with \( e \) if the following condition holds: \( n_eJ_v dX \geq 0 \). The condition partitions the space of infinitesimal motions into an open halfspace of blocked motions, an open halfspace of non-blocked motions, and a plane of motions for which the contact is preserved. The plane intersects the sphere of unit motions in a great circle, and its normal is \( n = n_e J_v \). Since both \( J_v \) and \( n_e \) are functions of the tolerance parameter vector \( p \), so is the normal: \( n = n(p) \). Let
Figure 9: Nominal contact and variations. (a) the nominal contact between parts \( P_1 \) and \( P_3 \). Part \( P_3 \) can translate vertically and to the right. It can rotate only clockwise with respect to the datum’s origin, which is located at the bottom left corner of \( P_1 \); (b) the contact varies because edge \( e \in P_1 \) changes orientation, ruling out downward translation of part \( P_3 \); (c) the contact varies because vertex \( v \in P_3 \) moves horizontally, changing the Jacobian of \( v \).

Figure 10: The contact zonotope and its contribution of arcs to the strong NDBG. The zonotope is centrally projected from the motion space origin to find the vertices which contribute to the strong NDBG. The figure shows the great circles \( c_1, c_2, c_3, c_4 \) corresponding to the projected zonotope vertices \( v_1, v_2, v_3, v_4 \), respectively (circles \( c_5, c_6 \) are not shown to keep the figure clean).

\( \bar{p} \) denote the nominal values of the tolerance parameters and \( p_i \) denote the \( i^{th} \) parameter out of \( k \), then the linear approximation of the normal is: \( n(p) \approx n(\bar{p}) + \sum_{i=1}^{k} \left( \frac{\partial n(\bar{p})}{\partial p_i} \delta_i^+ + \frac{\partial n(\bar{p})}{\partial p_i} \delta_i^- \right) \).

The volume swept by the approximation of \( n(p) \) when \( p \) spans the allowed tolerance hyper-box is the Minkowski sum of the nominal point \( n(\bar{p}) \) and the \( k \) line segments \( \text{conv}\{ \frac{\partial n(\bar{p})}{\partial p_i} \delta_i^-, \frac{\partial n(\bar{p})}{\partial p_i} \delta_i^+ \} \), where \( \delta_i^- \) and \( \delta_i^+ \) are the minimal and maximal variations of \( p_i \), respectively. The segments are called generators and the volume is a zonotope [12, 43], a convex centrally symmetric polytope whose center is the nominal normal \( n(\bar{p}) \). The complexity of the zonotope is \( O(k^2) \) and it can be computed in \( O(k^2) \) time using the correspondence between the topology of the zonotope and an arrangement of lines in the plane [12]. Figure 10 shows an example of the contact zonotope.

In the nominal case, the normal corresponding to a contact defines one great circle on the
Input: Assembly graph.

Output: Strong NDBG - an arrangement $\mathcal{A}$ of great circles.

1. Compute the sensitivity matrices of all the parts relative to datum $P_0$.

2. For each contact between parts $A$ and $B$:
   a. Substitute coefficients in template for partials of contact normal: $\frac{\partial n(\vec{p})}{\partial \vec{p}}$.
   b. Compute generators of contact zonotope: $\text{conv}\{\frac{\partial n(\vec{p})}{\partial \vec{p}} \delta_1^-, \frac{\partial n(\vec{p})}{\partial \vec{p}} \delta_1^+\}$
   c. Compute central projection of zonotope onto a plane.
   d. For each vertex $v$ of projection:
      - Compute plane $\Pi$ incident on origin and normal to vector $v$.
      - Compute intersection $c$ of unit sphere $S^2$ and plane $\Pi$.
      - Add great circle $c$ to arrangement $\mathcal{A}$.

3. Compute the structure of arrangement $\mathcal{A}$ of great circles, return $\mathcal{A}$.

Table 3: Algorithm for computing the strong NDBG of infinitesimal motions

The arrangement of great circles defined by all the contacts is the NDBG of the assembly. Every point in a two-dimensional cell of the NDBG defines an infinitesimal motion for which the same subset of the contacts interfere and the rest do not.

In the toleranced case, the normal zonotope determines the great circles of the strong NDBG as follows. Each point in the zonotope represents an instance of the contacting vertex and edge. The vector from the origin of the motion space to the zonotope point is normal to a plane that divides the unit sphere into two half-spheres. For the instance considered, one half-sphere represents motions that cause collision between the parts (the negative half-sphere), and the other side represents motions that are collision free (the positive half-sphere). Any other point in the zonotope corresponds to two different half-spheres. Thus, if a point of the unit sphere lies on the positive half-spheres defined by all zonotope points, then it represents a motion for which the current contact is non-interfering.

Since all the points of the zonotope are convex combinations of its vertices, it suffices to consider the zonotope vertices. In fact, since only the direction of the zonotope vertices matters, it suffices to consider the central projection of the zonotope (from the motion space origin) onto a plane. Bern et al. [5] show that the complexity of the central projection of a zonotope with $k$ generators in $\mathbb{R}^d$ is $\Theta(k^{d-2})$. They also present an $O(k \log k)$ time algorithm for computing the projection of a 3D zonotope defined by $k$ generators without explicitly computing the zonotope.

The great circles corresponding to the normals defined by the projection vertices of a single contact make up its contribution to the strong NDBG of the toleranced assembly. Each two-dimensional cell of the strong NDBG corresponds to motions for which either all, some, or none of the instances of the parts are blocked by this contact. Figure 10 shows the central projection of the contact zonotope and part of its contribution to the strong NDBG.

Table 3 presents the algorithm for computing the strong NDBG. It begins by computing the sensitivity matrices of all the parts relative to the datum. This is necessary because the axis of rotation and the infinitesimal translations are with respect to the datum’s frame. The coefficients of the edge normals are obtained from the sensitivity matrices and the derivatives of the Jacobian entries. These are used in step 2 to obtain the zonotope generators. The vertices of the projected zonotope define the great circles which form the strong NDBG arrangement.

The complexity of the algorithm depends on the number of contacts in the assembly, $c$,
Input: Assembly graph.

Output: Strong interference diagram \( S\text{ID} \).

1. Compute sensitivity matrices of all the parts relative to each other.
2. Store rotation angle and partial derivatives of transformations positioning all parts relative to \( P_0 \).
3. **For each** pair of parts \( P_i \) and \( P_j \):
   - If \( i < j \):
     a. Identify parameter indices \( I_{ij} \) and \( I_{0i} \).
     b. **For each** pair of segments \( s_k \in P_i, s_l \in P_j \):
        - Compute the parametric C-space obstacle \( s_l \ominus s_k \).
        - Compute the outer tolerance envelope of \( s_l \ominus s_k \): \( E_{kl} \).
        - Insert \( E_{kl} \) into arrangement \( A \).
     c. Compute outer cell of \( A \): \( E_{ij} \).
     d. Compute the cumulative rotation angle \( \theta_{\text{max}} \).
     e. Compute outer boundary of \( E_{ij} \) swept CW/CCW by \( \theta_{\text{max}} \) around origin: \( SE_{ij} \)
     f. Insert \( SE_{ij} \) into strong interference diagram \( S\text{ID} \).
   - Else: insert \( SE_{ji} \) into strong interference diagram \( S\text{ID} \).
4. Compute the overlay of pairwise obstacle envelopes in \( S\text{ID} \), return \( S\text{ID} \).

<table>
<thead>
<tr>
<th>Table 4: Algorithm for computing the strong interference diagram</th>
</tr>
</thead>
</table>
and the number of parameters affecting a single contact, \( k \leq \min\{m, 12qr\} \). According to Section 3.3, step 1 takes \( O(Nnk) \) space and time. Each iteration of step 2 takes \( O(k \log k) \) time and produces \( O(k) \) great circles. Step 3 dominates the complexity of the algorithm, requiring \( O(c^2k^2) \) space and time. Thus, the number of cells in the strong NDBG is \( O(c^2k^2) \) and the overall space and time complexity of the algorithm is \( O(Nnk + c^2k^2) \).

4.4 Multi-step translational motions

A single translational motion is insufficient for assemblies, in which two or more consecutive motions are necessary to remove or assemble a set of parts. Wilson et al. [41] introduce the interference diagram for planning assembly sequences with multiple translations per step. Halperin and Wilson [16] present a specialized algorithm for the construction of the interference diagram when only two translations are allowed: one finite translation followed by a translation to infinity.

The interference diagram is the overlay of the outer boundaries of the configuration space obstacles of all the pairs of parts, and its origin represents the assembled state. Edges of the diagram represent transitions between motions that cause some two parts to overlap and motions for which they do not. A path from the origin to the external diagram cell defines a multi-step motion which possibly separates the assembly into two sets. The DBG of the motion is constructed by tracing the edges crossed by the path. To account for toleranced parts, we compute the **strong interference diagram** (SID), which replaces the C-space obstacles with their outer tolerance envelopes.

Table 4 presents the algorithm. First, it computes the relative position sensitivity matrices of all the pairs of parts. To compute the cumulative rotation angles of parts positioned relative to the datum \( P_0 \), the algorithm stores the relevant transformation angles and their partial derivatives. Step 3 iterates on all pairs of parts, computing the outer tolerance envelope of the pairwise C-space obstacle. Since the envelope of \( P_j \ominus P_i \) is symmetric around the origin to
Figure 11: Strong interference diagram of parts $P_2$, $P_4$, and $P_7$ from Figure 1. For each pair of these parts, the figure shows the nominal C-space obstacle (inner closed curve) and its tolerance envelope (outer closed curve). The path marked by arrow 1 is a single translation. Nominally, any of the three parts can translate in this direction without colliding with each other, as the path does not cross any of the obstacles. Specifically, part $P_2$ can move without colliding with the other parts. The strong interference diagram shows that with tolerated parts, this path will cause $P_2$ to collide with $P_4$ and $P_7$. The path marked by arrow 2 is a two-step translation for which $P_2$ does not collide with either part under any instance of the part geometry and relative positioning. The diagram also shows that parts $P_2$ and $P_4$ can move together without colliding with $P_7$ as long as the path does not cross the solid or dashed obstacles.

the envelope of $P_i \ominus P_j$, the algorithm only computes the envelopes of $i < j$. The envelope is computed similar to algorithm 2, where each pair of segments contributes its obstacle envelope to the pairwise envelope. The arrangement $\mathcal{A}$ is composed of the tolerance envelopes of the pairwise segment obstacles. Its outer cell is the tolerance envelope of the C-space obstacle of $P_j \ominus P_i$ when only parameters in $I_{ij}$ are considered. To account for parameters in $I_{0i}$, the algorithm computes the cumulative rotation angle $\theta_{\text{max}}$ in step 3d, and then computes the swept area of the envelope when rotated by $\theta_{\text{max}}$ clockwise and counter-clockwise around the datum’s origin. This is accomplished using a rotational sweepline centered at the origin. The swept envelope boundary $SE_{ij}$ is the C-space obstacle of $P_j \ominus P_i$ when all assembly parameters are accounted for. The SID is the overlay of the envelopes computed in the iterations of step 3. Figure 11 shows the strong interference diagram of three parts from the example in Figure 1.

We now analyze the complexity of the algorithm. For simplicity, we make a conservative analysis which assumes that all the parts depend on $k \leq \min\{m, 12qr\}$ assembly parameters. Steps 1 and 2 require $O(N^2 nk)$ space and time. Step 3 iterates $N(N - 1)$ times: step 3a takes $O(k)$ time; step 3b is repeated $O(n^2)$ times, and each iteration takes $O(k^2)$ space and $O(k^2 \log k)$ time.
The complexity of the outer cell of a planar arrangement of $n$ curves, where each curve can intersect with another curve at most $s$ times, is $O(\lambda_s(n))$, where $\lambda_s(n)$ is the maximal complexity of a Davenport-Schinzel sequence of order $s$ on $n$ symbols [32]. The curves of the arrangement $A$ are line segments, circular arcs, and conic sections, so $s = 4$. The current best bound for $\lambda_6(n)$ is $O(n2^{\alpha(n)})$, where $\alpha(n)$ is the extremely slowly growing inverse of Ackerman’s function (for $n$ equal to the number of atoms in the universe, $\alpha(n) = 4$). Thus, the space complexity of $A$ is $O(\lambda_6(n^2k^2))$, and it can be computed in $O(\lambda_6(n^2k^2) \log nk)$ time. Step 3e has the same space and time complexity as 3c. Step 4 computes the overlay of the envelopes computed by step 3, which consist of a total of $S = O((N^2\lambda_6(n^2k^2)))$ segments. This is done in $O(S\lambda_6(S))$ space and time, which dominates the complexity of the algorithm.

The specialized algorithm of Halperin and Wilson [16] can be similarly extended to compute the strong interference diagram of two-step translations.

4.5 Probabilistic roadmap methods

The general assembly sequence planning problem, in which all the parts have three degrees of freedom and any collision free motion is valid, is a special case of the general motion planning problem, which is NP-hard. The current best complete solution has exponential complexity in the total number of degrees of freedom. Since this is not practical, the attention has recently shifted to randomized methods, which sacrifice completeness in favor of efficiency. The most commonly used approach is the Probabilistic Roadmap (PRM) method [19, 36, 13], which has proved to be practical and efficient in high-dimensional problems.

The idea of PRMs is to sample the composite C-space of the parts, which is $3(N - 1)$-dimensional when there are $N$ parts with three degrees of freedom each, until sufficient samples are determined to be collision-free. Then, pairs of points are tested for possible collision-free connecting paths, until the graph, whose edges correspond to collision free paths between configurations, covers the connectedness of the C-space. The different algorithms vary in the way they choose the sample points, in the way they pick pairs for connection, and in the way the connecting path is chosen. Sundaram et al. [36] show how to construct PRMs with heuristics specifically suited for assembly planning. Geraertz and Overmars [13] compare the performance of different PRM methods.

The geometric core of the PRM method is collision detection: testing for collision free configurations, and determining if a path connecting good points is collision free (local planning). Since most methods of local planning are based on sampling of the C-space curve connecting the points, either incrementally or by binary search, until the sample points are sufficiently close to one another, we only need to show how to test a specific (static) configuration for collision with tolerated parts.

In the nominal case, a static collision testing query asks whether any of the $N$ parts intersect each other. The lower bound on the complexity of this query is $\Omega(\tilde{n} \log \tilde{n})$, where $\tilde{n}$ is the total number of segments in all the parts, that is $\tilde{n} \leq Nn$. The algorithm transforms the parts to the desired configuration and checks in optimal time whether any of the segments strictly intersect. PRMs for nominal geometries utilize the spatial coherence between consecutive static checks to improve the cost of each check to amortized $O(1)$ time, provided there are sufficient sample points.

Table 5 presents the algorithm for collision detection in the tolerated case. The input is the assembly graph and the C-space point $c \in \mathbb{R}^{3N-3}$ that describes the configuration of all the parts relative to the assembled state. The assembled state of an assembly instance is determined according to the relative position constraints. Each of the parts has three
Input: Assembly graph, C-space point $c$.

Output: Whether configuration $c$ is valid for all assembly instances.

1. Compute sensitivity matrices of all the parts relative to the datum.
2. For each part $P_i$, apply transformation $T_c$ to get $P'_i$.
3. For each pair of parts $P_i$ and $P_j$ with $i < j$:
   a. Classify parameters into $I_{0i}$, $I_{ij}^{ind}$, and $I_{ij}^{shr}$.
   b. Identify part $P_i$’s shape-parameter segments $S_{ij}^{sp}$.
   c. Compute $E_{0i}$, $E_{0j}$ - relative position envelopes of $P'_i$, $P'_j$ from parameters in $I_{ij}^{ind}$.
   d. Check if $E'_{ij}$ and $E'_{ij}$ intersect, and if so return false.
4. Return true.

Table 5: Algorithm for static collision detection of toleranced parts.

The complexity analysis of the algorithm is similar to that of Section 4.2. The overall complexity is $O(N^2n(1 + g)q^2r^2 \log(nqr))$.

### 4.6 Comparison with nominal planning

We now compare the complexity of assembly sequence planning with nominal parts and with tolerated parts. In the nominal case, the input to the planner is the geometry of the assembly parts and possibly contact relations. Thus, the complexity is measured in terms of the number of assembly parts, $N$, the maximal complexity of an individual part, $n$, and the number of part contacts, $c$.

In the tolerated case, the description of the parts includes tolerancing information about individual parts in the form of initial sensitivity to local parameter variations (parameters that affect the part shape), as well as relative positioning constraints which define the assembled
Table 6: Comparison of the complexities of the geometric computations for assembly planning in the nominal and toleranced cases. The first to fourth rows correspond to the algorithms in Sections 4.2, 4.3, 4.4, and 4.5, respectively. For rows one to three, the space complexity is the upper bound on the number of vertices in the NDBG, and the time complexity is the time spent computing the NDBG elements and their arrangement. For the PRM methods, the space and time complexities are the amount of storage and the time required to perform a single static collision detection. The time complexity of the algorithm for infinitesimal motions of toleranced parts includes a dependency in \( N \) and \( n \) which does not exist in the nominal case. This is due to the datum frame, whose position relative to all parts must be determined in order to plan infinitesimal motions.

### 5 Implementation and experiments

We now describe the implementation of our algorithms and experiments that quantify the effect of tolerancing on the valid assembly sequence motions.

In our previous paper [24], we described the computation of tolerance envelopes of individual parts, and compared their quality and efficiency with common sampling methods. The same algorithms and results apply for the computation of the relative position envelope de-
scribed in Section 3.3, and for computing the tolerance envelopes of C-obstacles (Sections 4.2 and 4.4). In addition, we implemented the algorithm for computing the sensitivity matrices between two parts in an assembly (Table 1) and the algorithm for computing the directions of blocked translational motions (Table 2). The computation of the blocking cones uses the less efficient algorithm presented in an earlier paper [23], and has quadratic complexity in the number of vertices of each part. The implementation is in MATLAB (un-optimized code), and all the following experiments were run on a computer with Pentium IV 2.4 GHz CPU, 1GB of RAM, running under Windows XP. The experiments were designed to achieve the following goals:

1. Measure the time complexity of relative position and motion space obstacle computation.
2. Quantify the effect of dimensional tolerancing on the space of allowed assembly motions.
3. Quantify the error of the linear approximations.
4. Measure the effect of the size of the tolerance intervals on the space of allowed motions.
### 5.1 Running times

In addition to assembly model 1 in Figure 1, we experimented with the four assembly models shown in Figure 12. Table 7 shows the model characteristics and the measurements of time to compute the sensitivity matrices of all the part vertices relative to the datum part (Table 1). For models of up to nine parts and up to 58 tolerance parameters, the running time is less than a second, which is very practical. The table also shows the time spent computing the cones of directions blocked by all the parts relative to the datum. This time can be significantly improved by optimizing the MATLAB code, but even so all the running times are below 180 seconds, which is practical for assembly planning tasks.

### 5.2 Expansion of blocked motions

In Section 4.2 we described how to compute the cone of blocked translational directions between two parts, which is the key geometric computation for assembly planning with translations to infinity. Figure 13 demonstrates how the nominal cone between two parts from model 1 expands when the parts are toleranced. In the nominal assembly, parts $P_2, P_3, P_4$ of Figure 1 can translate upwards as a group and be separated from the rest of the assembly. But when the parts are toleranced, this motion may be blocked by part $P_6$. Thus, tolerancing reduces the number of assembly sequences that are valid for all part instances.

Counting the number of valid assembly sequences is a discrete measure of the effect of tolerancing on the relative position of parts in the assembly. In the example given above,
Figure 13: Nominal positions of parts $P_2$ and $P_6$ from Figure 1 and the directions for which $P_2$ is blocked by $P_6$. The shaded cone represents directions for which the nominal parts always collide. The expanded cone represents directions for which there is at least one instance of the parts which causes collision. This expansion of $22.8^\circ$ was achieved with a dimensional working window of roughly 3, which is typical of normal precision processes.

even the smallest part tolerances would cause assembly sequences that remove $P_2$ before $P_6$ to be invalid. To quantify the effect of tolerancing in a continuous manner, we measured the expansion (in degrees) of the cone of blocked translational motions for each pair of parts. Table 7 shows the maximal and average expansion of the cones over all the pairs of parts in each model.

The robotic arm assembly (model 4) was designed to demonstrate how a long chain of connected parts causes large sensitivity in the last part of the chain. The tolerance intervals are small compared to the other assemblies, yet the maximal increase in the size of the blocking cone (due to parts $P_1$ and $P_8$) is relatively large. The effect would have been even more dramatic had the parts been closer to each other.

5.3 Linearization error and the cumulative rotation angle

The rigid body transformations satisfying the relative position constraints determine the small adjustments needed for correct positioning with small variations in the part shapes. When two parts are related through a long chain of constraints, these adjustments accumulate. Of the cumulative transformations, the rotation angle is especially important since its effect grows linearly with the distance from the rotation center.

The cumulative rotation angle (Section 3.4) measures the largest possible rotational adjustment of a part due to simultaneous variations of all the tolerance parameters. The column marked $\theta_{\text{Max}}$ in Table 7 shows the cumulative rotation angle of the part that maximizes this quantity in the assembly. The largest cumulative rotation angle was found in model 3 between parts $P_1$ (the datum) and $P_2$, due to their proximity and cyclic relations.

Throughout this paper we apply the linear approximation in computations of shape and position variation. The approximation error of individual shape variation was studied in our previous paper [24]. The error in relative positioning stems from the linear approximation of the rotations in the rigid body transformations, which sums the contribution to the rotation of each tolerance parameter. Thus, the worst case error occurs with the largest cumulative rotation angle.

Let $\theta$ be the rotation angle and let $L$ be the length of the feature undergoing the rotation. Since the rotation error is invariant to translations and rotations, we may assume that the
feature coincides with the datum’s frame. Thus, the error is $|\sin(\theta) - \theta| \cdot L$. With the wide tolerances used for model 3, the largest error is approximately four times the average tolerance interval. But when the tolerances are tightened by a factor of 10 (normal precision tolerance values), the error is only $0.0414$ of the average tolerance interval. When the tolerances are tightened by an additional factor of 10, the error becomes $4.11 \cdot 10^{-4}$ of the average tolerance interval, which is insignificant.

We conclude that the error is less significant when the tolerances are tighter. Since manufacturing processes become increasingly more precise, the quality of the linear approximation will increase in time. The following section describes the effect of the tolerance interval size on the space of assembly sequence motions.

5.4 Effect of tolerance interval size

To measure the effect of the size of the tolerance intervals on the expansion of the cones, we repeated the runs on model 5, each time scaling the tolerance intervals by a factor of two (rows 5a to 5f in Table 7). The resulting average increase in the size (expanded minus nominal) of the cones is approximately a similar factor of two. Similar experiments can be performed to quantify the effects of tolerancing for other motion types. For example, the area of the projected contact zonotope (Section 4.3) corresponds to the reduction in the area of allowed motions in the motion space sphere, and the area bounded by the C-space envelope (Section 4.4) corresponds to the reduction of the free space of multiple step motions. Our experiments on tolerance envelopes [24] show that the area they bound increases linearly with the size of the tolerance intervals. We conjecture that the increase in the volume (expanded minus nominal) of motion space occupied by obstacles as a function of the tolerance intervals is roughly linear.

We conclude that tolerancing significantly reduces volume of the space of valid motions for assembly sequence planning. For translational motions, the reduction depends roughly linearly in the size of the tolerance intervals, and can be efficiently computed in practical time.

6 Conclusion and extensions

In this paper, we have presented a framework for toleranced planar assembly planning which is more general than existing approaches in terms of the part model, the tolerance specification model, and the type of motions used in the assembly sequences. We have developed efficient algorithms for computing the sensitivity of part positioning to variations in part geometries, and for incorporating these computations into the geometric core of existing assembly planners for nominal parts. We have shown that the cost of accounting for toleranced parts in planning is a multiplicative factor which is polynomial of low degree in the number of tolerance parameters. We have demonstrated the feasibility and efficiency of these algorithms by implementation and experiments on five assembly models. Our experiments show that tolerancing significantly reduces the space of valid motions for assembly sequence planning, and that this reduction depends linearly on the size of the tolerance intervals. We have demonstrated that geometric computation for assembly planning with tolerance parts is efficient and practical.

We now briefly discuss the extension of our framework to other assembly sequence types and for polyhedral parts.

Schwarzer et al. [31] study the problem of $m$-handed simultaneous translations, in which all the assembly parts are translated simultaneously such that each part has its own translation direction and velocity. The algorithm reduces the problem to finding an unbounded direction in the composite C-space of the parts. The C-space is defined by the pairwise cones of blocked
translations. Our framework can be used to account for tolerancing by modeling the assembly using relative position constraints, and then computing the tolerated cones as in Section 4.2.

Srinivasan and Gadh [34] propose an efficient method for disassembly aimed at removing a selected polyhedral component from the assembly for maintenance purposes. The geometric part of their algorithm computes blocked translation directions based on contacts between parts. In the nominal case, the blocked directions are half spaces. In the tolerated case, the blocked directions are defined by the union of half spaces derived from the contacts of all valid instances of the parts. Our framework can be used to find the sensitivity of the contacts to parameter variations, and to compute the blocked directions from the projected vertices of the contact zonotope, as in Section 4.3.

Thomas et al. [37] compute translational assembly sequences for polyhedral parts by using the stereographic projection of the C-space obstacles to find translations that separate parts. They propose scaling the parts by a small factor to simulate variability, but this approach does not account for actual tolerance specifications, parameter dependencies in the shape variation, and relative positioning constraints. Our framework can be used to compute the tolerance envelopes of the projected C-space obstacles based on the tolerance specification and relative position constraints, as described in Section 4.4.

The framework can be extended to support polyhedral parts as follows. Tolerancing of individual parts is based on the parametrization of the spatial coordinates of the vertices, which follows directly from parametric tolerance specifications, or from the translation of geometric tolerance zones. The tolerance envelopes of polyhedral parts consist of envelopes of tolerated triangles in 3D, which in turn consist of linear and ruled surfaces [25].

The relative positioning of polyhedral parts is determined by six distance constraints between points, lines, and planes. The assembly graph edges have weights ranging from one to six, and the assembly is properly constrained if a special case of the Kutsbach criterion is satisfied [39].

For assembly planning with tolerated polyhedral parts, computation of tolerance envelopes of motion space obstacles is extended as follows. Directions of blocked translations (Section 4.2) are computed from the pairwise C-space obstacle envelopes of polyhedral facets, which consist of tolerated triangles. The strong interference diagram (Section 4.4) is the overlay of the boundaries of these facet obstacle envelopes. Infinitesimal rigid body motion (Section 4.3) is encoded on the sphere $\mathbb{S}^5$. Each contact between parts defines a six-dimensional zonotope whose central projection has $O(k^4)$ vertices, each vertex contributing one great circle to the strong NDBG.

Future work will focus on improving the PRM method running times (utilizing the spatial coherence between consecutive samples), devising efficient algorithms for polyhedral assemblies, and extending the set of supported features to cylinders and other non-linear features.

Acknowledgement: This work was partially supported by a grant from the Israeli Ministry of Science and Technology, Grant Number 01-01509.

References


