# *Algorithms for Computing Intersection and Union of Toleranced Polygons with Applications*

F. Cazals[1], G.D. Ramkumar[2] *

[1] *i*MAGIS-IMAG
BP 53 - 38041 Grenoble cedex 09 - FRANCE
[2]Robotics Laboratory, Department of Computer Science, Stanford University,
Stanford, CA 94396 USA

*email:* Frederic.Cazals@imag.fr

**Abstract**

Since mechanical operations are performed only up to a certain precision, the geometry of parts involved in real life products is never known precisely. But if tolerance models for specifying acceptable variations have received a substantial attention, operations on toleranced objects have not been studied extensively. That is the reason why we address in this paper the computation of the union and the intersection of toleranced simple polygons, under a simple and already known tolerance model. Firstly, we provide a practical and efficient algorithm that stores in an implicit data structure the information necessary to answer a request for specific values of the tolerances without performing a computation from scratch. If the polygons are of sizes $m$ and $n$, and $s$ is the number of intersections between edges occurring for all the combinations of tolerance values, the pre-processed data structure takes $O(s)$ space and the algorithm that computes a union/intersection from it takes $O((n+m)\log s + k^{'} + k\log k)$ time where $k$ is the number of vertices of the union/intersection and $k \leq k^{'} \leq s$. Although the algorithm is not output sensitive, we show that the expectations of $k$ and $k^{'}$ remain within a constant factor $\tau$, a function of the input geometry. Secondly, we define and study the stability of union or intersection features. Thirdly, we list interesting applications of the algorithms related to feasibility of assembly and assembly sequencing of real assemblies.

# 1 Introduction

## 1.1 Geometric operations for toleranced objects

In the life cycle of a part in a manufactured product, say an engine, a piece of furniture, or a toy, etc., its exact geometry can never be described exactly. In fact, mechanical operations are performed only up to as much precision as to ensure that a feature of the part (for example a vertex) lies within a zone called its *tolerance* zone. Tolerance models and their role in product design have received substantial interest in the literature [21, 22, 18, 23, 14, 8]. Most of this work concentrates in defining models that enable a precise and mathematically accurate representation of manufacturing defects —imperfect edges and datums, existence of a maximum material part, etc. But geometric properties have not been looked at extensively, so that it is not known how easily these models can undergo the usual operations of solid modeling. Intersection volumes and surface areas for cylinders with tolerances are described in [7]. Planning the assembly of toleranced products is addressed in [12]. But basic operations such as

*Work on this paper was done was F. Cazals was visiting the Robotics Laboratory at Stanford University and was supported by Matra Datavision. G. D. Ramkumar is supported by NSF. Part of this work has also been supported by NSF/ARPA grant IRI-9306544.

intersection, union, convolution, and Minkowski sum of toleranced polygons have not yet been studied. The basic operations among these are the union and the intersection. We address their computation in this paper: given two simple polygons whose geometry (but not topology) can vary according to a simple tolerance model, pre-process the set of all possible intersections between edges of the two polygons so that for a particular set of tolerance values their intersection or union can be computed efficiently.
It should be observed that this is in essence different from D. Salesin's work on $\varepsilon$-geometries ([19]). Indeed, the focus of his work is to build robust geometric algorithms from imprecise geometric primitives, while we are aiming at performing operations on a continuum of exact geometries assuming precise geometric primitives. But of course, should we have addressed the robustness issue, the core problem would have been the same since the computations we end up with consist in evaluating the sign of algebraic expressions (sections 2.2.2 and 3.1). The reader is referred to [20, 1] for recent developments in this area.

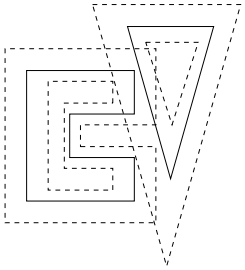
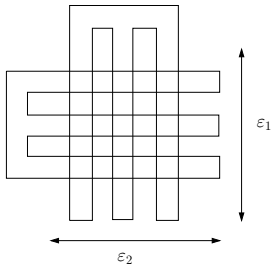


Figure 1: (a)Intersection of two toleranced polygons




(b)Worst-case example of intersection

Figure 1(a) shows an example of two toleranced polygons intersecting in a configuration that may produce zero, one, or two components in the intersection depending on the exact tolerance values. Figure 1(b) shows such an example of two intersecting comb-like polygons $R$, and $B$ — one vertical and one horizontal with tolerances $\varepsilon_1$ and $\varepsilon_2$ on their horizontal and vertical edges respectively. In this example, the number of components in the intersection can vary from 0 to $\Theta(m * n)$. In addition, the components are determined by different tolerance values, causing an exponential number of topologies. Hence it is infeasible to represent the topologies explicitly.

There is substantial literature related to computing union and intersection of polygons without tolerances. The intersection algorithm of [3] can be used to compute the trapezoidal decomposition of an arrangement of edges. This provides a union/intersection algorithm of simple polygons that runs in $O((m+n)\log(m+n)+k)$. Our method is simpler and thus easier to implement and also is more efficient if $|S| = o(n+m)$, that is the tolerances space does not admit a super-linear number of intersections; we believe this is the case in practical situations. Other previous work in the area of union/intersection computation is an integration of the linear time triangulation algorithm of [2] with the linear time map merging of [9], giving a space-time optimal solution. But this involved method does not give any feedback on the probability of occurrence of say an intersection feature —which we call stability in the sequel.

The specification of the problem we address if as follows. The input consists of two simple polygons, referred to as a red polygon of $n$ edges and a blue polygon of $m$ edges. Each edge has an associated tolerance parameter — denoted $r$ ($b$) for the red (blue) polygon, or $t$ if the polygon color is not specified — defined with respect to a reference frame attached to its polygon. The relative position of the two polygons is determined by those of their respective frames. The configuration space $\mathcal{T}$ has dimension $n+m$ and an instantiation of the two polygons is fully specified by a set of tolerance values $T = \{r_1, \ldots, r_n, b_1, \ldots, b_m\}$. Let $S$ refer to the set of intersections between pairs of edges occurring for all combinations of tolerance values and by $S_T$ those valid only for the input $T$.

Our results are three-fold. Firstly, we pre-process the toleranced polygons to enable efficient computation of the union or intersection for a query instance of tolerance values. The pre-processing results are stored in a interval-tree like data structure ([17, 5]) from which an instance of the intersection or union is computed in time $O((n+m)\log|S| + k' + k\log k)$ where $k$ is the number of vertices of the union or

intersection and $k^{'}$ is the cardinality of a super-set of the output vertices, with $k \leq k^{'} \leq |S|$. This term $k^{'}$ causes the complexity to be not output sensitive; however, this aspect seems necessary because the space of intersection of two toleranced edges only admits a complicated definition as a semi-algebraic set in dimension six defined by four inequalities involving polynomials of degree two. But we show that the expected running time of our algorithm is output-sensitive by proving that the expectations of $k$ and $k^{'}$ remain within a constant factor $\tau$, a function of the input geometry. Secondly, we use our data structure to report the set of all possible "local" topologies of the union and intersection of the polygons. Thirdly, we present several straightforward applications of our algorithms to feasibility of assembly and assembly sequencing.

### 1.2 Paper overview

In section 2 we introduce the tolerance model and study the intersection of the so-called toleranced edges, the basic primitive. In section 3 we present the algorithm that outputs $S_T$ from $S$. In section 4 we show how to perform the intersection/union operation from the set $S_T$. In section 5 we discuss the stability of intersection features. In section 6 we consider three applications of the toleranced intersection and in section 7 we list some outstanding problems and conclude.

## 2 Tolerance model

### 2.1 Toleranced polygon

By toleranced polygon, we mean a simple polygon with an attached coordinate system $(p, \vec{i}, \vec{j})$ and each edge of which is defined by a triple $(\theta, [a, b])$ as follows: the normal to the line supporting this edge makes an angle $\theta \in [0, 2\pi)$ with the frame attached to the reference point of the polygon; the distance between the edge and the polygon reference point is allowed to span the interval $[a, b]$ where $a > 0, b \geq a$. The model assumes that the parameter $\theta$ is constant; in more general models $\theta$ can have a tolerance range as well. An edge is therefore characterized in $(p, \vec{i}, \vec{j})$ by the equation $x \cos\theta + y \sin\theta - t = 0$ with $t \in [a, b]$.

For a given polygon, these tolerance zones should be small enough so that all instances of the polygon have the same topology. A sufficient condition is that no vertex falls into the intersection of more than two stripes spanned by the edges.

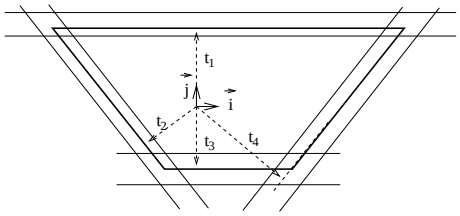
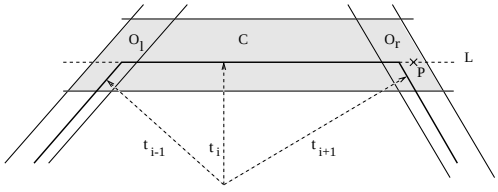



Figure 2: (a)Toleranced polygon




(b)Edge domain

In this model, the domain spanned by each edge is a trapezoid, as indicated in grey on figure 2(b). In fact, the lines supporting the two edges that have extremal values determine the parallel sides of the trapezoid, and the two other sides correspond to the maximal values of the two connected edges. More precisely, this trapezoid can be subdivided into three regions as indicated on figure 2(b):
-the $C - domain$ : whatever value $t_i \in [a_i, b_i]$ may have, the $i^{th}$ edge crosses this domain all the way ($C$ stands for compulsory),
-the $O - domains$ $O_l$ and $O_r$ : a point lying on the $i^{th}$ edge might not be attained because of the tolerance values of the adjacent edges ($O$ stands for optional, $l$ and $r$ for left and right). It is for example the case of the point $P$ in $O_r$ on figure 2(b).
This trapezoid is called the *toleranced edge* in the sequel of this paper, and is denoted with capital letters.

For example $R_i$ refers to the $i^{th}$ toleranced edge of the red polygon. It is important to observe that it depends on three variational parameters: the *main* parameter $t_i$, and the two *connected* parameters $t_{i-1}$ and $t_{i+1}$. The tolerance model we use was introduced in [12].

## 2.2 Intersection of toleranced edges

### 2.2.1 Problem statement

In dealing with geometric operations between toleranced polygons, a crucial step consists in intersecting toleranced edges. Given two toleranced edges, the status of their intersection is one of the following: $(i)$they never intersect $(ii)$they always intersect $(iii)$they intersect conditionally to the variational parameters involved. Intersections corresponding to the cases $(ii)$ and $(iii)$ are referred to as *compulsory* and *optional* respectively.

As already observed, each toleranced edge depends on three parameters. The intersection between edge $i$ of the red polygon and edge $j$ of the blue polygon therefore depends on the six parameters $\{r_{i-1}, r_i, r_{i+1}, b_{j-1}, b_j, b_{j+1}\}$. We note $\nu_{ij}$ the corresponding subspace of $\mathcal{T}$. If an intersection exists, it belongs to the polygon $I_{ij}$ intersection of the two toleranced edges. This polygon is shown in grey on figure 3(a) (see also fig. 5). Compulsory intersections require that $I_{ij}$ lies within the intersection of the $C - domain$ of the toleranced edges and that any of its section along the direction of a main parameter spans the whole domain of this parameter. This is the case on figure 3(a) for edges $R_i$ and $B_j$. On the other hand, finding a simple description of optional intersections is more laborious. Consider figure 3(b) where the three edges $R_i, R_{i+1}$ and $B_j$ have non null tolerances zones —all the other edges, tat is $R_{i-1}, B_{j-1}, B_{j+1}$ being reduced to line segments. The three dashed lines correspond to instances of these edges for the tolerance values $r_i = x, r_{i+1} = y$ and $b_j = z$. In this configuration, edges $R_i$ and $B_j$ intersects at point $I$. But in addition to $x$ and $z$, $I$ also depends on $y$. Indeed for any value of $y$ smaller than the one displayed, $B_j$ intersects $R_{i+1}$ but not $R_i$ anymore. It turns out that the condition under which the intersection occurs cannot be easily found by a geometric analysis of the edges interaction, which accounts for an algebraic computation of $\nu_{ij}$.

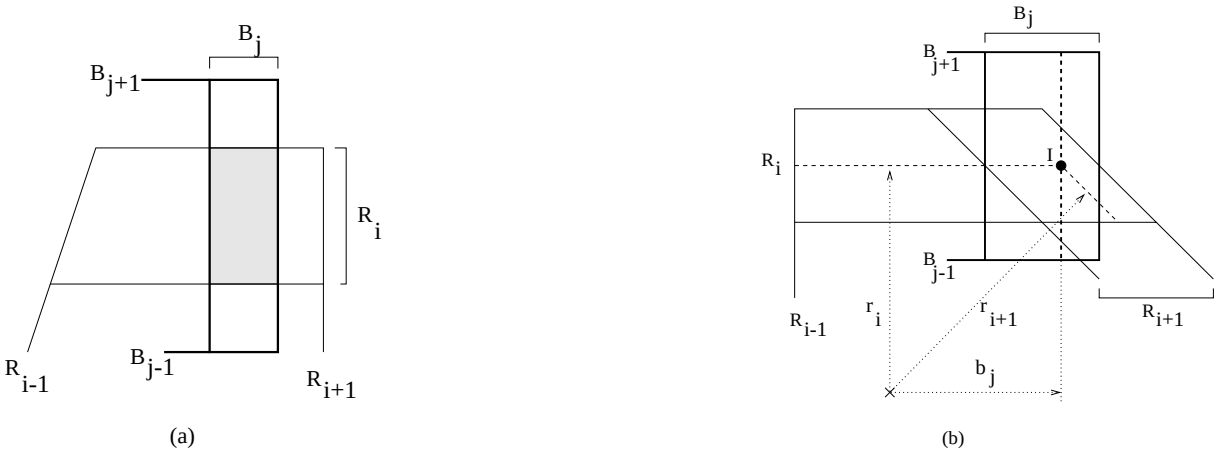



Figure 3: Edges intersection

### 2.2.2 Algebraic specification of toleranced intersection vertex

**Lemma 1** *Let $e_i$ and $e_j$ be two edges delimited by the vertices $(u_1, v_1)$ and $(u_2, v_2)$ respectively, and let $(a_1, a_2), (a_2, a_3), (b_1, b_2), (b_2, b_3)$ be the variational parameters these vertices depend on. Then, the subset of these parameters such that $e_i$ and $e_j$ intersect is defined by four quadratic inequalities in the space of 6 tolerance values*

Proof: Edges $e_i$ and $e_j$ intersect iff the line supporting $e_i$ divides the edge $e_j$ and vice-versa (see figure 4). Let $\mathrm{Det}(u, v, w)$ represent the signed area of the triangle $uvw$ defined by vertices $u = (u_x, u_y), v = (v_x, v_y)$, and $w = (w_x, w_y)$:

$$\begin{vmatrix} 1 & 1 & 1 \\ u_x & v_x & w_x \\ u_y & v_y & w_y \end{vmatrix}$$

The condition $\mathrm{Det}(u, v, w) > 0$ requires that $u, v, w$ are in counterclockwise order on the plane. The line supporting edge $e_i$ divides the edge $e_j$ iff $u_2$ and $v_2$ are on the opposite sides of the line supporting $e_i$, so that the conditions sought are

$$\mathrm{Det}(u_2, u_1, v_1)\,\mathrm{Det}(v_2, u_1, v_1) \;<\; 0 \text{ and } \mathrm{Det}(u_1, u_2, v_2)\,\mathrm{Det}(v_1, u_2, v_2) \;<\; 0 \quad (\mathcal{I})$$

Each of the vertices $u_1, v_1, u_2, v_2$ is a linear function of the tolerance parameters of the two edges intersecting to produce it. The set of constraints defines a volume bounded by quadratic surfaces in the space of 6 tolerance values. Since the edges preserve their orientation under variation of tolerance parameters, requiring the two edges to intersect is tantamount to fixing the sign of each determinant (a stronger condition than above). □

To illustrate the previous computation, we implemented condition $\mathcal{I}$ under Maple to run some experiments. For example, the configuration of figure 3(b), which is defined as follows

$$R_{i-1} := (Pi, a1 \in [5., 5.]);\; R_i := (Pi/2, a2 \in [5., 11.]);\; R_{i+1} := (Pi/4, a3 \in [4.5 * sqrt(2), 7.5 * sqrt(2)])$$

$$B_{j-1} := (Pi/2, b1 \in [3., 3.]);\; B_j := (0, b2 \in [1., 6.]);\; B_{j+1} := (Pi/2, b3 \in [14., 14.]);$$

leads to the two conditions

$$(b1 - a2)\left(-a2 + \sqrt{2}a3 + a1\right)^2 (b3 - a2) < 0$$

$$\left(a2 - \sqrt{2}a3 + b2\right)(-b3 + b1)^2\,(b2 + a1) < 0$$

As this example shows, this space is not linear and actually not even convex. Indeed, it is easily checked that the two following points $P_1 = [5.;\ 6.;\ 8, 48.;\ 3.;\ 6.;\ 14.]$ and $P_2 = [5.;\ 7.;\ 9.19;\ 3.;\ 6.;\ 14.]$ belong to $\nu_{ij}$ but not their middle. Computing the volume $p_{ij}$ of $\nu_{ij}$ cannot be performed using simple techniques as those of [13] for convex polytopes and this issue will be addressed in section 5.1 in dealing with the stability of the intersection components. Moreover, storing $\nu_{ij}$ in a data structure to efficiently select the relevant intersections with respect to a particular input of tolerances remains an open question.

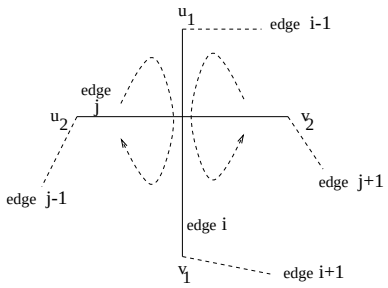



Figure 4: Intersection conditions

### 2.2.3 Approximate representation

For these reasons, we address the problem of finding an approximate representation of $\nu_{ij}$ as an hyper-rectangle $\nu'_{ij}$ of volume $p'_{ij}$ such that all the points located in this hyper-rectangle are candidates with a high probability to be a true intersection between the two edges.

As intersection of two four sided polygons each depending on at least three parameters, $I_{ij}$ is at most eight-sided and depends on at most six parameters. Moreover it is convex since the toleranced edges are convex. If $k$ is one of the six parameters involved in the intersection of edges $i$ and $j$, then let $stripe_k$ be the minimal closed stripe of the plane with sides parallel to the $k^{th}$ parameter direction and containing $I_{ij}$. The two lines defining a stripe correspond to two values in the variational space of the corresponding parameter, possibly out of the range $[a_k, b_k]$ so that it is possible to reduce any stripe to its intersection with the tolerance zone of its parameter. Let $rs_k$ be the bounds of the reduced stripe in the variational zone of parameter $k$. It is also possible that $rs_k = [a_k, b_k]$ in which case the corresponding parameter is called *useless* since an intersection can occur for any of its value. If the opposite holds, it is called $useful$.

An obvious necessary condition on the *main* parameters for an intersection to exist is that $r_i \in rs_i$ and $b_j \in rs_j$. See e.g. figures 5(a)(b). For the *optional* parameters, the situation is more involved. First observe that a useless parameter can be discarded since we are interested in the intersection of edges $i$ and $j$ and it does not carry any information. It is the case of $b_{j-1}$ on figure 5(a). Second, the notion of minimal stripe enclosing $I_{ij}$ is not sufficient any more: consider the parameter $r_{i+1}$ on figure 5(c) and the two vertical lines $V_1, V_2$ enclosing $I_{ij}$. If the $(i+1)^{th}$ edge lies to the left of $V_1$, no intersection is possible. But if it lies to the right of $V_2$ the intersection is possible. Therefore, for a *connected* parameter, the correct stripe is the stripe enclosing $I_{ij}$ whose upper bound is extended to this parameter maximal value.

In summary, each of the six parameters involved in the intersection of edges $i$ and $j$ produces a range in its tolerance zone. Such a range might be useless if it spans the whole tolerance zone, and in this case we discard it for a connected parameter. This leads to a representation of $I_{ij}$ as a hyper-rectangle $\nu'_{ij}$ of dimension at least two and at most six that partitions $\nu_{ij}$ in two: the set of points that fulfill a necessary condition for the edges to intersect, and its complement.

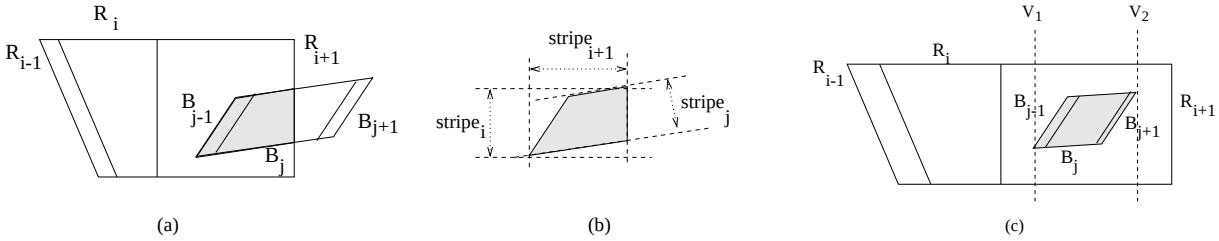



Figure 5: Intersection and stripes

### 2.3 Computing the intersection of toleranced edges

We use a traditional Bentley-Ottman style algorithm to compute the intersection of toleranced edges. This algorithm that runs in $O((m+n+s)\log(m+n))$ for line-segments is described in [17] and can be easily modified for trapezoids as follows. The idea consists in parsing the edges from left to right, the events to be handled being the insertion (deletion) of a new edge when its left-most (right-most) point is found. To get the ordering from left to right, we just have to sort the $n+m$ left-most and right-most points. Now, for a given abscissa $x$, we need to know which are the other edges in the $y$-range of the edge processed at this abscissa. This can be done using the interval tree data structure of [17, 5]. A precise analysis should however be performed to state a precise complexity for this Bentley-Ottman sweep applied to trapezoids. But this is not that important since this step is a pre-processing.

## 3 Selection of relevant intersections

### 3.1 Selecting a subset of events

Let $T = \{r_1, \ldots, r_n, b_1, \ldots, b_m\}$ be an instantiation of the two input polygons. Before we can get the topology of the intersection/union, we must select the subset $E_T$ of intersections that are valid for the particular input $T$. The problem we have to face here is that each $I_{ij}$ depends on at most six parameters

while the whole space has dimension $m+n$. Devising an efficient data structure to retrieve these small dimensional sub-spaces is a challenging problem. An easy way to get around it is therefore to allow the selection process to operate in two steps as follows: first select a subset $E_{AT} \subset E$ and then derive $E_T$ from $E_{AT}$.

This strategy allows us to represent $I_{ij}$ using the hyper-rectangle $\nu'_{ij}$ described above. For a given $t_i$, finding all the $\nu'_{ij}$ containing it reduces to a point-in-segment enclosure test. A suitable data structure to perform this is an interval tree ([17, 5]) for each parameter, which leads to a forest of interval trees for the $n+m$ parameters. But as noticed in section 2.2 some parameters appearing in $\nu'_{ij}$ do not express any constraint since their range is equal to their tolerance zone. We store them in linked lists instead of the interval tree. More precisely, each *main* parameter $k$ involved in the description of any $\nu'_{ij}$ is stored as follows:

- If it is useful, it is stored in its interval tree $IT_k$. In addition, each such range is given a pointer to the $\nu'_{ij}$ it comes from,
- If it is useless but is involved in a compulsory intersection, it is stored in the linked list $CL_k$ with a pointer to the $\nu'_{ij}$ it comes from,
- If it is useless and is involved in an optional intersection, it is stored in the linked list $OL_k$ with a pointer to the $\nu'_{ij}$ it comes from.

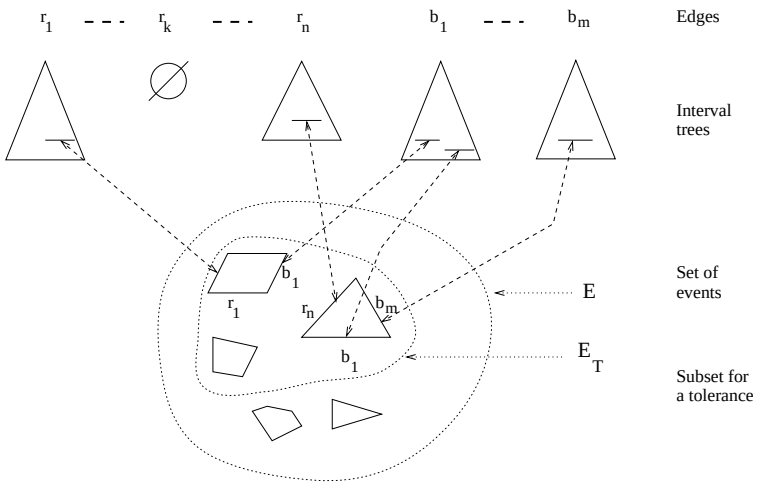



Figure 6: Selecting active edges

An example configuration of the forest of interval trees is shown in figure 6. Observe that we have not stored the connected parameters, but they are accessed through the pointer from the main parameters to the $\nu'_{ij}$ they belong to.

We now describe below the selection process that leads from $E$ to $E_T$.

Selection algorithm
**for** $i := 1$ **to** $n+m$ **do**
(1)Add to $E_T$ the intersections referenced by $CL_i$ still not accessed
(2)Add to $E_T$ the relevant intersections referenced by $OL_i$ still not checked
(3)Add to $E_T$ the intersections referenced by the segments of $IT_i$
containing $t_i$ , not accessed and relevant
**od**

First observe that any $\nu_{ij}^{'}$ involves two main ranges stored either in a list or an interval tree. Each intersection or hyper-rectangle can thus be accessed by either of the two pointers. We assume that the first access sets a flag. Now, there are two kinds of intersections: compulsory and optional, of which the latter have to be checked. If more than two parameters are involved in the description of its $\nu_{ij}^{'}$, the first step is to check that the connected parameters fulfill the required conditions. If so, the intersection has to be computed and a check has to be performed to see if it belongs to the two line segments supported by the main parameters. These tests should be applied in this order because of their respective computational costs.

### 3.2 Analysis

Let $s_i$ be the cumulated size of the $i^{th}$ interval tree and the associated lists. Selecting and reporting all the segments containing $t_i$ in these structures costs $O(\log s_i + k_i^{'})$ with $k_i^{'}$ the output size. The first term sums to $\sum_{i=1}^{n+m} O(\log s_i)$ which is easily seen to be smaller than $(n+m)\log s$ with $s = \sum_{i=1}^{n+m} s_i$. And the second one sums to $k^{'} + k$ with $k^{'} = \sum_{i=1}^{n+m} k_i^{'}$ and $k = \sum_{i=1}^{n+m} k_i$ since the relevance of each intersection has to be checked.

An important task is to compare the relative values of $k$ and $k^{'}$. Since our data structure is intended to process several queries, it would be nice to have an amortization phenomenon over these queries. Suppose we have to process $r$ queries, and for any any query $i$ in $1..r$ and any intersection $j$ in $1..s$, let $\varepsilon_{ij}^{'}$ be the random variable defined as 1 if the intersection $j$ is selected at stage $i$ through the $IT$ data structure and 0 otherwise. Also, let $\varepsilon_{ij}$ be a random variable defined as 1 if the previous intersection is not discarded and 0 otherwise. Of course, $\varepsilon_{ij}^{'} = 0$ implies $\varepsilon_{ij} = 0$, and a measure of the $IT$ data structure goodness to select the relevant intersections is the ratio of the number of intersections kept over the number of intersections initially selected. More precisely:

**Definition 1** *The acceptance rate $\tau$ of intersections selected with the interval-trees is defined as the ratio of the expectations of the number of events kept over the number of events initially selected, that is*

$$\tau = E(\sum_{i,j} \varepsilon_{ij}) / E(\sum_{i,j} \varepsilon_{ij}^{'})$$

And we have

**Lemma 2** *The value of $\tau$ is given by $\sum_j p_j / \sum_j p_j^{'}$ with $p_j$ the volume of the subset of parameters defining the $j^{th}$ intersection, and $p_j^{'}$ the volume of the approximation of this subset*

Proof: For any $j$, $\sum_i \varepsilon_{ij}^{'}$ and $\sum_i \varepsilon_{ij}$ are distributed as binomial random variables of parameters $B(r, p_j^{'})$ and $B(r, p_j)$ respectively, whence the result □

It is difficult to state precisely which value $\tau$ might have. But since the only case where $p_j / p_j^{'} = 0$ is when the edges involved in that intersection are parallel, while configurations such as the one figure 3(a)(b) respectively correspond to ratios of 1 and .549 (see section 5.1), it is reasonable to say that $k^{'}$ and $k$ are within a constant factor.

Another interesting problem is whether or not it is possible to apply the paradigm of divide-and-conquer to this selection process. The answer seems to be negative. Indeed, unlike for $d$-dimensional query using say a $d$-dimensional search tree where we start by looking at the first direction, then proceed along the second one for the subset selected so far, and so on, the edges independence make this process impossible here.

## 4 Intersection/Union

We now show how to reconstruct the intersection or the union from the intersection events. Decisions have to be made at the intersection points only since by following the boundary of a polygon we eventually

end up on an edge involved in an intersection ! But at any vertex of the intersection it is clear from the local structure which edge must be pursued next, as shown in figure 7(a). The only information we are missing when following an edge is therefore the label of the next intersection of that edge with the second polygon. For example on figure 7(b), after to have decided to follow edge 1 after the intersection $I_{1,f}$, we must know that the next intersection to be analyzed is $I_{1,b}$. This is not straightforward since what we get from the selection process of section 3.1, is for a given edge a sequence of edges of the second polygon. Ordering these intersections requires running a standard sorting algorithms which takes $O(k \log k)$ time for $k$ intersection points.

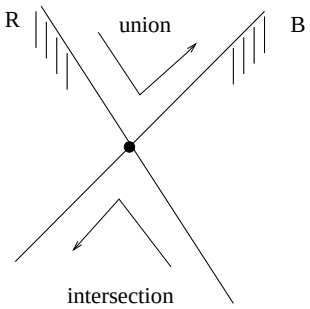
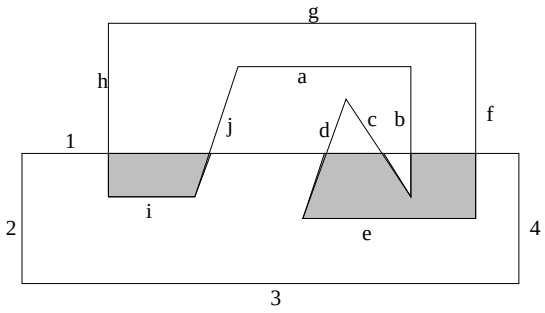



Figure 7: (a)Local analysis




(b)Computing the intersection

It should also be noticed that computing the sequence of intersections *along the edge* given the sequence *along the second polygon* is what does the so-called Jordan sorting algorithm described in [10]. Unfortunately, this algorithm does not apply here since the ordering of the segments imposed by the interval tree data structure ([5]) is incompatible with the lexicographic order along the polygon. But had this algorithm been appropriate, the complexity gain would have been minor —actually, the $O$ constant.

Once the previous ordering is known, getting a component of the union or the intersection consists in enumerating its vertices applying the rules explained above. And finding all the components consists in iterating the previous process until all the intersections have been used.

# 5 Stability of Intersection Features

In this section, we study the stability of features of the intersection. Features of the union can be handled similarly. There are obviously two kinds of features: vertices, and components that can involve several vertices. For each of these, *stability* refers the probability of occurrence, with the distinction of *stable* features that always exist, and *optional* features otherwise.

## 5.1 Vertices

Checking if a vertex is compulsory is easily done when computing the approximation $\nu'_{ij}$ of $\nu_{ij}$. If the vertex is optional, a measure of its stability is the volume $p_{ij}$ of $\nu_{ij}$. Computing the exact value of this volume seems difficult since we do not have a description of its boundary. Getting an $(\varepsilon, \delta)$ approximation $\tilde{p_{ij}}$ of $p_{ij}$ using a Monte Carlo method (see [15, 6]) is also an open problem since $\nu_{ij}$ might in general be non-convex. However, from a practical point of view, the following boot-strapping algorithm may give satisfactory results:

1. First, get a rough estimate $\tilde{p_{ij}}$ of $p_{ij}$ as the fraction of points satisfying condition $\mathcal{I}$ over a sample of 'reasonable' size uniformly drawn in $[0,1]^6$,

2. Second, plug $\tilde{p_{ij}}$ into the estimator theorem (theorem 11.1) of [15] which in turn gives the sample size such that $\tilde{p_{ij}}$ is accurate within a factor $\epsilon$ with a probability greater than $1 - \delta$.

For example, on the configuration of figure 3(b), an estimate on a sample of size 1000 gives $\tilde{p_{ij}} = .543$ while the corrected value according to a sample of size 11000 given by the estimator theorem for a $(.05, .05)$ approximation is $\tilde{p_{ij}} = .549$.

## 5.2 Components

Let a purple vertex be the intersection of a red edge with a blue edge, and let a component be defined as a sequence of of red, blue and purple vertices forming a simple polygon intersection of the original polygons. Extending this definition is tricky. Indeed, as depicted on figure 8(a), a component can be stable while none of its vertices are —when the triangle grows downward, edge $b$ is not intersected anymore by edges 1 and/or 2. In a similar way, a component can be stable while none of its edges are so: on figure 9(a)(b), by continuously transforming the leftmost configuration to the rightmost one, we go from a configuration where the intersection consists of the leftmost edge of the vertical rectangle together with the rightmost edges of the rotated square, to a configuration involving the rightmost edge of the rectangle and the leftmost edges of the square. For these reasons, we constrain the previous definition to a fixed sequence of edges from the two polygons. For example on figure 8, there are four different components, $\{2, b, 1\}, \{2, b, c, 1\}, \{2, a, b, 1\}$ and $\{2, a, b, c, 1\}$.

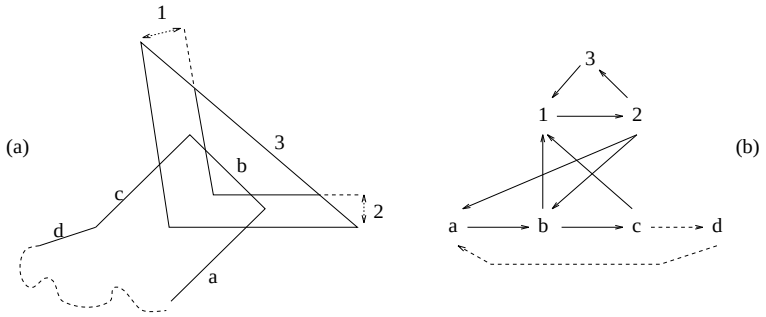



Figure 8: Components, stable vertices and $\mathcal{IVG}$ graph

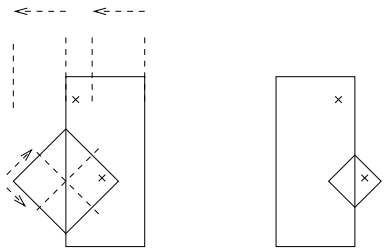


Figure 9: Components and stable edges

Enumerating these components is different from enumerating the set of all topologies of the intersection. Indeed, the later reflects the local interference between red and blue edges, while the former is the cartesian product of these. We therefore focus our attention on the computation of components involving a given intersecting edge. We do not claim any bound on the running time of the algorithm and just sketch it here. Let $\mathcal{IVG}$ be the intersection vertices graph that is the oriented graph defining the possible intersections between edges of the two polygons as well as the connectivity between the red and blue edges. Figure 8(b) depicts the $\mathcal{IVG}$ for figure 8(a). The convention used to assign directions to edges of $\mathcal{IVG}$ is the following: edge $u$ points to edge $v$ if a turn from $u$ into $v$ has interior to the left. The basic idea of the algorithm consists in considering each cycle of the $\mathcal{IVG}$ and checking if it corresponds to a valid component. But any cycle is not a good candidate, since the following constraints need to be satisfied:

- Each turn must have the interior to its left,
- If an edge $e$ of a polygon, say the blue edge, appears several times in the labeling, the red intersecting edges must respect the ordering along the blue edge,
- The labeling must respect as much as possible, the 'locality' of the intersection sought. For example on figure 8(b), the word $2abc$ can be expanded as $2abc1$ or $2abcd$. But $2abcd$ is obviously not valid. It is not clear however how to use this geometric information while generating good candidates cycles.

# 6 Applications

We briefly examine in this section two applications of the computations described so far and list another one that requires Minkowski operations to be defined for toleranced polygons. Since we do not make any contribution to these examples, the reader is referred to the original papers for the details.

## 6.1 Feasibility of assembly of two parts

A simple but instructive real life example on how useful operations between toleranced objects is the following one from [7]: a fly fishing reel consists of a spool mounted on high quality bearings spinning around a center pin assembly. The hole in the center of the reel has to be machined smaller than the bearing so that when the reel is pressed in place interference is observed. Performing the intersection operations between the different parts of the reel (on a cross-section so that cylinders are represented as rectangles) therefore turns out to be very useful: it gives the topology of the interference zones, which provides feedback to the designer on how precisely the parts have to be milled depending on the type of material used (aluminum in this case).

## 6.2 Assembly sequencing

As already mentioned in the introduction, results were obtained recently for assembly sequencing of toleranced assemblies ([12]). Without mentioning the details, the authors propose two algorithms:
-one that lists all the assembly sequences that are *always* feasible, whatever the tolerance values are
-another one that lists the sequences that may be feasible *only* for some combinations of the parameters.

A failure of the second algorithm means that no instance of the product is assemblable, which in turns implies that there is a collision between two parts, or that the product is 'intrinsically' infeasible for assembly. In the first case, applying our intersection algorithm to the toleranced polygons gives the parameters involved in the intersection. This is important because paying more attention to these parts in the manufacturing process –that is reducing the tolerance zones, might solve the problem. Handling the second case is much more difficult and goes beyond the scope of this paper.

It should be observed that using polygonal models as input for the union and intersection operations is not very restrictive since most of the contacts between real products parts are either cylindrical or between flat surfaces.

## 6.3 Collision detection

It is well known that one of the most elementary operations in robotics consists in computing the *Minkowski difference* between the robot and the obstacles, which gives the space of intersection free translations of the robot. In particular in configurations that contain degenerate inputs such as contacts, tangencies, etc, collision checking on the toleranced objects may give additional information.

# 7 Conclusion

In this paper we presented data structures and algorithms for computing the intersection and the union of simple polygons with tolerances on edges. Given two polygons of sizes $n$ and $m$ whose edges give rise to $s$ intersections for all the combinations of the tolerances values, our algorithm pre-computes in time $O((m+n+s)\log(m+n))$ a search structure that takes $O(s)$ space. Given specific values for the tolerances, we use the structure to output the specific intersection or union in time $O((m+n)\log s + k^{'} + k\log k)$ where $k$ is the output size and $k \leq k^{'} \leq s$. Although the algorithm is not output sensitive, the expected values of $k$ and $k^{'}$ remain within a constant factor $\tau$, a function of the input geometry. The algorithm is easy to implement, practical, and we believe it works well for realistic input instances. Also, several straightforward applications to feasibility of assembly and assembly sequencing are described.

Many difficult issues still remain. Firstly, directly storing the semi-algebraic set describing the intersection of toleranced edges may enable more efficient processing of union and intersection queries. Secondly, getting an $(\epsilon, \delta)$ approximation for the volume of the semi-algebraic set is an open question.

At last, the problems remain unaddressed for the case when the tolerance model is extended to include angles on the polygons. There is also scope for future work on computing convolutions and Minkowski sums of toleranced polygonal objects.

ACKNOWLEDGMENTS: F. Cazals is supported by Matra Datavision, G. D. Ramkumar is supported by NSF. Part of this work has also been supported by NSF/ARPA grant IRI-9306544.

The authors wish to thank Cyprien Godard and Danny Halperin for insightful discussions, as well as the reviewers and the editors for helpful comments.

# References

[1] F. Avnaim, J.-D. Boissonnat, O. Devillers, F. Preparata, and M. Yvinec. Evaluating signs of determinants using single-precision arithmetic. Research Report 2306, INRIA, BP93, 06902 Sophia-Antipolis, France, 1994. http://www.inria.fr/prisme/personnel/devillers/publis.html.

[2] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete and Computational Geometry*, 6:485–524, 1991.

[3] B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *J. of the ACM*, 39(1-54), 1992.

[4] P. Dagum, R. Karp, M. Luby, and S. Ross. An optimal algorithm for Monte Carlo estimation (extended abstract). In *36th Annual Symposium on Foundations of Computer Science*, pages 142–149, Milwaukee, Wisconsin, 1995. IEEE.

[5] M. de Berg et al. *Computational geometry by example.* Stanford University Press, draft version, 1996.

[6] M. Dyer, A. Frieze, and R. Kannan. A random polynomial time algorithm for approximating the volume of convex bodies. In *ACM STOC*, pages 375–381, 1989.

[7] W. H. ElMaraghy et al. Intersection volumes and surface areas of cylinders for geometrical modelling and tolerancing. *C.A.D*, 26(1):29–45, 1994.

[8] Shiaofen Fang and Beat Brüderlin. Robustness in geometric modeling — tolerance-based methods. In *Computational Geometry — Methods, Algorithms and Applications: Proc. Internat. Workshop Comput. Geom. CG '91*, volume 553 of *Lecture Notes in Computer Science*, pages 85–101. Springer-Verlag, 1991.

[9] U. Finke and K. Hinrichs. Overlaying simply connected planar subdivisions in linear time. In $11^{th}$ *ACM Symposium on Computational Geometry*, Vancouver, 1995.

[10] K.Y Fung et al. Simplified linear-time jordan sorting and polygon clipping. *IPL*, 35, 1990.

[11] L. J. Guibas, D. Salesin, and J. Stolfi. Epsilon geometry: building robust algorithms from imprecise computations. In *Proc. 5th Annu. ACM Sympos. Comput. Geom.*, pages 208–217, 1989.

[12] JC. Latombe, R.H. Wilson, and F. Cazals. Assembly sequencing with toleranced parts. *Computer Aided Design*, 29(2), 1997.

[13] J. Lawrence. Polytope volume computation. *Mathematics of Computation*, 57(195):259–71, 1991.

[14] B. Moller. Tolerances in product modelling. *Informatik, Informationen Reporte*, 1991(5):4–52, 1991.

[15] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[16] J. Nievergelt and F. Preparata. Plane-sweep algorithms for intersecting geometric figures. *Communications of the A.C.M*, 25(10), 1982.

[17] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, NY, 1985.

[18] U. Roy, C.R. Liu, and T.C. Woo. Review of dimensioning and tolerancing: representation and processing. *Computer Aided Design*, 23(7):466–83, 1991.

[19] D.H. Salesin. *Epsilon geometry: building robust algorithms from imprecise computations*. PhD thesis, Stanford University, 1991.

[20] J.R. Shewchuk. Robust adaptive floating-point geometric predicates. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, pages 141–150, 1996.

[21] V. Srinivasan. Recent efforts in mathematization of asme/ansi y14.5m standard. In $3^{rd}$ *CIRP seminars on computer aided tolerancing*, Ed. Eyrolles, Paris, 1993.

[22] R.K. Walker and V. Srinivasan. Creation and evolution of the asme y14.5.1 standard. *Manufacturing Review*, 7(1), 1994.

[23] N. Wang and T.M. Ozsoy. A scheme to represent features, dimensions, and tolerances in geometric modeling. *Journal of Manufacturing Systems*, 10(3):233–40, 1991.