# Robotics Research

The Sixth International Symposium

edited by
Takeo Kanade
and
Richard Paul

The International Foundation for Robotics Research
Cambridge, USA.

# Motion Planning

Jean-Claude Latombe
Robotics Laboratory, Department of Computer Science
Stanford University, Stanford, CA 94305, USA

One important goal in robotics is to make it possible for robots to perform tasks whose goals are expressed in high-level declarative terms. In this context, researchers in motion planning develop algorithms to automatically generate motions to achieve goals formulated as geometric arrangements of the robot and its workspace. Motions must avoid collisions with obstacles. They must deal adequately with the laws of nature (e.g., friction, gravity, inertia). They must also make enough information available to the robot controller, either by sensing the environment or by reasoning about motion mechanics, or both, so that the successive states of the robot relative to its environment are reliably recognized.

The first paper, *Robot Algorithms*, by Jean-Claude Latombe, takes the stand that, like computer science, robotics is fundamentally about algorithms. Robot algorithms, however, differ in significant ways from computer algorithms. Latombe's paper suggests that a unique characteristic of robot algorithms is how they combine a control component involving very basic control issues, such as controllability and observability, and a planning component raising fundamental computational issues, such as calculability and complexity.

The second paper, *Motion Planning for Mobile Robots: From Academic to Practical Issues*, by Jean-Paul Laumond, investigates in more detail the relationship between controllability and complexity for one particular class of robots: the mobile robots. Laumond stresses the difference between holonomic motion planning, which essentially lies in the realm of computational geometry and can now be solved efficiently, and nonholonomic motion planning, which also raises control theory issue. This comparison illuminates the relation between controllability and motion planning complexity.

The third paper, *Towards a Theory of Information Invariants for Cooperating Autonomous Mobile Robots*, by Bruce Donald, James Jennings, and Daniela Rus relates to the observability issue in robot algorithms. More specifically, it considers cooperating robots pushing large, heavy objects. Donald, Jennings, and Rus develop "information invariants" that make it possible to establish some explicit trade-of equivalence between the internal state retained by each robot, communication among robots, sensory information, and information derived from task mechanics.

The fourth paper, *Feeding and Sorting Algorithms for the Parallel-Jaw Gripper*, by Ken Goldberg, explores algorithms to orient and recognize parts with a parallel-jaw gripper equipped with a passive translating bearing. The algorithm orienting parts uses information derived from task mechanics to iteratively reduce the set of possible orientations of a part. Part recognition uses a low-cost linear sensor. Goldberg's work is a good demonstration that neatly designed robot algorithms may yield efficient and cost-effective hardware implementation.

The fifth paper, *A Safe Swept Volume Method for Collision Detection*, by André Foisy and Vincent Hayward, addresses a basic problem in most motion planners, namely collision detection. Foisy and Hayward propose an elegant algorithm that detects collision along a given robot path in a given environment. The algorithm is both safe – it never classifies a subset of the path as collision-free if it isn't – and reliable – if enough time is available, it computes the colliding subsets of a path with arbitrary precision.

These papers illustrate some of the major trends in motion planning research over the past few years. In particular, motion planning has evolved from the pure geometric problem of finding collision-free paths in known environments, to the more complicated problem of generating motion strategies involving sensory interaction and physical reasoning. New basic issues have been identified (e.g., the interaction between controllability and complexity) and new algorithms have been proposed (e.g., criticality-based discretization of continuous spaces). Research now also attacks a wider variety of challenging tasks, for example complex robot coordination. Concurrently, motion planning algorithms become more efficient and researchers consider tasks of greater practical significance.

# A Safe Swept Volume Method for Collision Detection

André Foisy Vincent Hayward
McGill Research Center for Intelligent Machines,
3480 University Street,
Montreal, Quebec, Canada, H3A 2A7.

**Abstract**

*This paper presents a collision detection method based on a swept volume approach. The proposed method computes a convex approximation (CSV) guaranteed to encompass the real swept volume (RSV). It is shown to be robust, which means that small errors in the model result in small errors in the result. It is first shown to be safe, which means that detections can ever be missed. It is then shown to be reliable, meaning that the exact result can be approached as closely as desired for a known cost.*

## 1 Introduction

The configuration space is called $CS$ and the free space $FS$. The collision detection problem is defined as follows:

**Def.: 1** *Given a path* $\Psi(t) : [0,1] \rightarrow CS$ *for a robotic system, a* collision detection *algorithm partitions the interval* $[0,1]$ *into* $I = I_1, I_2, \cdots, I_n$ *such that* $\Psi(I_i) \subset FS$, non-overlap, *or* $\Psi(I_i) \subset CS \backslash FS$, overlap. *A* solution *to an instance of the collision detection problem is the set of overlap intervals.*

Robotic systems and their surroundings are often modeled using polyhedra, convex or concave bounded flat faced geometrical objects. In this paper, only convex polyhedra, known as polytopes, are considered. Polytopes are defined as bounded sets created by the intersection of some finite number of half spaces in $\mathcal{R}^3$. Any object which is not convex is decomposed into convex shapes, each one being modeled by a polytope.

The collision detection technique presented here is an "off-line" method based on a swept volume approach. In contrast with other swept volume approaches, the result of the calculations are convex approximations (CSV), hence polytopes, guaranteed to encompass the real swept volume (RSV).

The calculation of the convex approximation involves finding the bounding volume of a moving point. To ensure that the algorithm is safe and reliable, bounding values are computed using the theory of interval analysis [Moore 66, Moore 79].

The main design criterion is stated in section 2. Section 3 is a short review of collision detection methods akin to ours. Section 4 introduces the definitions of the CSV and RSV operations. Section 5 presents the basic relations between the CSV and RSV. Section 6 presents a form of decomposition which trades off computations of the CSV against goodness of fit. Section 7 shows how the basic computations of the CSV operations can be carried out by a digital computer to ensure safety and reliability. Section 8 describes the collision detection algorithm. Section 9 illustrates the application of the algorithm to an open kinematic chain. Finally, section 10 offers some concluding remarks.

## 2 Design Criterion

It is well known that any model of a physical object is approximate. For example, the position of a vertex may be known only within a tolerance. Moreover, the coordinates of a vertex are generally represented in a computer as an approximation. This suggests the definition of a design criterion for a collision detection algorithm called *robustness*.

**Def.: 2** *A collision detection algorithm is said to be* robust *if it does not fail to detect any true overlaps and degrades gracefully as errors accumulate.*

Robustness conveys an idea of resistance to perturbations. The sources of these perturbations, or more generally errors, are multiple:

- Collision detection algorithms are developed to solve real physical problems. Hence, the first source of errors is the modeling of the moving parts and of the surroundings. Of course, a bad model cannot be improved by a computer, it can only make it worst.
- The model is the input data to a series of computational problems that require numerical solutions. Because these problems are solved with digital computers, small perturbations in the coefficients may appear. These perturbations are a source of errors which can degenerate into unbounded discrepancies in the solution if the problem itself is ill-defined [Young *et al.* 72].
- An important source of errors comes from the algorithm itself. Certain algorithms or steps of an algorithm may propagate errors very rapidly.

In the light of the previous enumeration, we propose that robustness, a desirable property of a collision detector, conveys the ideas of *safety* (no misses) and of *reliability* (convergence to the exact solution).

With respect to an input model, a collision detection problem always has an exact solution, one for which each bound is known with infinite precision. Let this set of intervals be noted $C_0$, then:[1]

**Def.: 3** *A collision detection algorithm is* partially safe *if its solution to an instance of the problem,* $C_1$, *includes the exact solution,* $C_0 \subseteq C_1$.

**Def.: 4** *A collision detection algorithm is* safe *if it is partially safe for all instances of the collision detection problem.*

In other words, a collision detection algorithm is safe if it never classifies an overlap interval as a non-overlap interval.

Whether there are errors or not, safety is a required property, but safety alone is not enough. Indeed, a collision detection algorithm that always classifies any complete trajectory as an overlap is surely safe but useless. Thus, it is desirable that the outcome of a algorithm be reliable.

**Def.: 5** *Given an instance of the collision detection problem and given* $C_1$ *and* $C_2$, *two solutions to the same problem from two different algorithms. Then, the first algorithm is* partially more reliable *than the second algorithm if* $d(C_0, C_1) < d(C_0, C_2)$.

**Def.: 6** *A collision detection algorithm is* reliable, *compared to another algorithm, if it is partially reliable for all instances of the collision detection problem.*

It is important to notice that a robust collision detection algorithm is not guaranteed to refrain from producing spurious overlap intervals, termed *glitches*. The most common sources of numerical glitches are calculations on inaccurate representations. For a collision detection algorithm, a glitch should always be a safe decision. When in doubt it is better to produce a spurious overlap interval than to cause a collision inadvertently.

## 3 Review

The simplest collision detection method consists of sampling the trajectory and to check for static interferences [Boyse 79, Hurteau *et al.* 83, Dobkin *et al.* 83, Dobkin *et al.* 85, Gilbert *et al.* 85, Gilbert *et al.* 88, Rimon *et al.* 92]. However, this approach cannot be safe because the motion between samples is not checked for potential collisions.

[1] The notion of ordering and of a metric for intervals are easily derived and well described in [Moore 66, Moore 79]

The swept volume approach is an improvement over the sampling approach. A swept volume algorithm takes the description of a moving object in two or three dimensions as input, and produces another object in two or three dimensions that contains the volume swept by the moving object. The resulting swept volume is checked for static interference with other swept volumes.

The intersection of two swept volumes is not a necessary and sufficient condition to infer a collision. If the moving objects collide, their swept volumes obviously intersect, but if they do not collide, their swept volume may still intersect. The use of relative motion between objects transforms the swept volume intersection test into a necessary and sufficient condition. However, if the computed swept volume is an approximation of the real swept volume, the intersection test is at best a sufficient condition.

A swept volume approach to collision detection differs from an extrusion technique [Cameron 85, Cameron 90, Canny 86, Foisy *et al.* 92]. A collision detector build on extrusion works in space-time, whereas an algorithm built on the swept volume approach only works in the space in which the problem is given.

Of interest to the method developed in this paper are the methods of Cameron [Cameron 85], Ganter [Ganter 85] and Von Herzen [Von Herzen *et al.* 90].

Cameron and Ganter compute approximations that are not sure to encompass the real swept volume of the moving object. For example, Ganter approximates the real swept volume by taking the convex hull of successive samples.

In contrast, Von Herzen develops a approach applicable to time dependent parametric surfaces. He presents two techniques, both of which depend on Lipschitz values (the constant in a Lipschitz condition), themselves depending on bounds on the velocity. The first approximation is a sphere called a "Lipschitz bounding sphere" and the second approximation, which is a tighter bound than the sphere, is a bounding box called a "Jacobian bounding box".

The algorithm developed by Von Herzen is not directly applicable to polyhedra, however, the previous bounding techniques are applicable in the framework of the algorithm developed in this paper. Indeed, the basic computational step is to bound the motion of a point. Thus, the methods of Von Herzen are directly applicable and so is the method of the reachable set constrained by bounds on the acceleration presented in [Foisy *et al.* 90].

## 4 Definition of the RSV and CSV Operations

The swept volume of a moving polyhedron is defined as the trace of that moving polyhedron in Euclidean space. It is described by the following set of points:

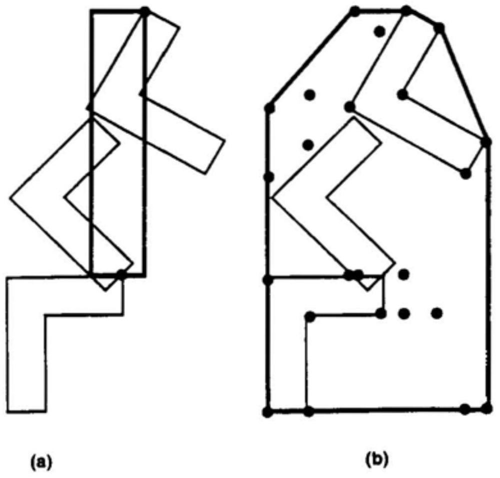

Figure 1: Construction of the CSV; (a) the bounding box of a point, (b) the convex hull of all the vertices of the bounding boxes.

**Def.: 7** *Given a displacement* $\Gamma(t)$, $t \in I \subseteq [0,1]$, *and* $P$ *a polyhedron, the* $\mathrm{RSV}(\Gamma(I),\ P)$ *is*

$$\mathrm{RSV}(\Gamma(I),\ P) \ = \ \{p \mid \exists (q \in P, t \in I)\ (p \ = \ \Gamma(t)q)\}.$$

In practice, displacements or rigid body motions are expressed in projective space and represented as 4x4 matrices in homogeneous coordinates.

Clearly, for a general displacement, the RSV of a moving polyhedron is difficult to compute. In practice, a convex approximation of the RSV will be computed, namely the CSV.

Before defining the CSV, consider the construction of a bounding volume encompassing the trajectory of a point. As we have seen in the review section, there are many methods for bounding the trajectory of a point. To accommodate the development of the forthcoming sections, a bounding volume is required to converge to the point itself as the width of the interval on which it is computed goes to zero. In this paper, we propose a simple method which does not depend on kinematic quantities. It will be shown (section 7) that it is readily computable using interval analysis methods.

Since a displacement cannot send a Euclidean point to infinity, it is possible to find a box aligned on the axis of the coordinate system frame that includes the trajectory of a point. The set of points defined by this bounding box is noted $\mathrm{BB}(\boldsymbol{\Gamma}(I)\boldsymbol{p})$, where $\boldsymbol{\Gamma}(I)\boldsymbol{p}$ is the expression of the moving point in some coordinate system.

Building on the bounding box, the CSV is defined to be (see figure 1)

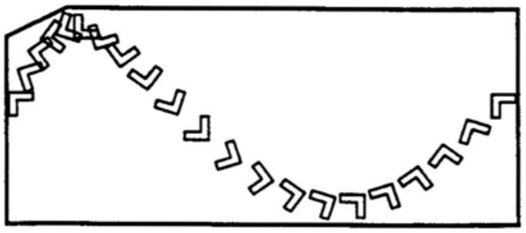

Figure 2: Inclusion of the RSV in its CSV.

**Def.: 8** *Given a displacement* $\Gamma(t)$, $t \in I \subseteq [0,1]$, *and a polyhedron* $P$, *the* $\mathrm{CSV}(\Gamma(I),\ P)$ *is the convex hull of the bounding boxes of the vertices of* $P$. [2]

The set of points defined by the CSV operation is:

**Def.: 9** *Given a displacement* $\Gamma(t)$, $t \in I \subset [0,1]$, *and a polyhedron* $P$, *then the CSV of definition 8 is*

$$\begin{aligned}\mathrm{CSV}(\Gamma(I),\ P) = \{p \ \mid \ & p = \overline{p_1 p_2}, \\ & \exists q_1 \in P\ (p_1 \in \mathrm{BB}(\boldsymbol{\Gamma}(I)\boldsymbol{q}_1), \\ & \exists q_2 \in P\ (p_2 \in \mathrm{BB}(\boldsymbol{\Gamma}(I)\boldsymbol{q}_2))\}.\end{aligned}$$

## 5 Basic Relations Between RSV and CSV

The first basic relation between the RSV and CSV is the cornerstone of a safe and reliable implementation (see figure 2).

**Theorem 1** *Given a displacement* $\Gamma(t)$, $t \in I \subset [0,1]$, *and a polyhedron* $P$, *then* $\mathrm{RSV}(\Gamma(I),\ P) \subseteq \mathrm{CSV}(\Gamma(I),\ P)$

**Proof:** By definition, the segment between any two points of a convex set is comprised in that convex set. In particular, this is true for any moving segment because its vertices are comprised in bounding boxes. Finally, the CSV operation takes the convex hull of the bounding boxes that contain the moving vertices of $P$. It follows that the motion of $P$ must be included in its CSV. **qed.**

The second basic relation between the RSV and the CSV is one that enables us to compute a better approximation of the RSV by taking the union of a sequence of CSVs. Each CSV of the sequence is computed over as subtrajectory. The union of the subtrajectories is such that it covers the entire trajectory (see figure 3).

**Theorem 2**

$$\mathrm{RSV}(\Gamma([0,1]),\ P) \subseteq \bigcup_{i=1..n} \mathrm{CSV}(\Gamma(I_i),\ P),$$

*where* $I_i$ *is* $[a_{i-1}..a_i]$ *with* $0 = a_0 < a_i < a_n = 1$.

[2]Notice that the convex hull operation is assumed to return a set of points encompassing its interior and its boundary.

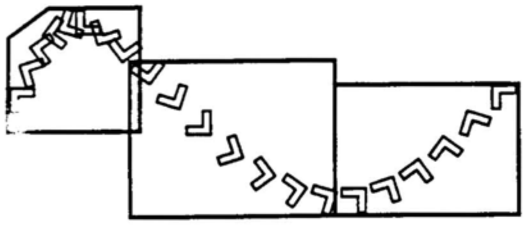
Figure 3: Using many CSVs to describe a RSV.

**Proof:** The RSV can be written as

$$\mathrm{RSV}(\Gamma([0,1]),\ P) \ = \ \bigcup_{i=1..n} \mathrm{RSV}(\Gamma(I_i),\ P),$$

and by theorem 1

$$\mathrm{RSV}(\Gamma(I_i),\ P) \subseteq \mathrm{CSV}(\Gamma(I_i),\ P) \ \textbf{qed}.$$

To see that the union of CSV becomes a better approximation as $n$ grows, consider the case $n = \infty$:

$$\bigcup_{i=1..\infty} \mathrm{CSV}(\Gamma(I_i),\ P) \ = \ \bigcup_{\forall t\in[0,1]} \mathrm{CSV}(\Gamma(t),\ P) \ = \ \bigcup_{\forall t\in[0,1]} \mathrm{CH}(\Gamma(t), P) \ = \ \mathrm{RSV}(\Gamma([0,1]),\ \mathrm{CH}(P)).$$

Notice that equality is only possible with polytopes, however, it is enough to motivate the decomposition of non convex polyhedra into polytopes (see figure 4).

## 6 Basic Property

A transform $\Gamma$ can be quite complex. The following paragraphs outline how the decomposition of $\Gamma$ into simpler transformations leads to simpler computations of the CSV over $P$. The price to pay is that the successive CSVs are not as good a fit as CSV($\Gamma$, $P$).

**Theorem 3** *Given a polyhedron $P$ and a displacement $\Gamma(t) = \Gamma_1(t), \cdots, \Gamma_n(t)$, $t \in I \subseteq [0,1]$, then*

$$\mathrm{RSV}((\Gamma_1\cdots\Gamma_n)(I),\ P) \subseteq \mathrm{RSV}((\Gamma_1\cdots\Gamma_{n-1})(I),\ \mathrm{RSV}(\Gamma_n(I),\ P)),$$

$$\mathrm{CSV}((\Gamma_1\cdots\Gamma_n)(I),\ P) \subseteq \mathrm{CSV}((\Gamma_1\cdots\Gamma_{n-1})(I),\ \mathrm{CSV}(\Gamma_n(I),\ P)).$$

**Proof:** By definition,

$$RSV((\Gamma^1\cdots\Gamma^n)(I), P) = \{p | (\exists q \in P) \wedge (\exists t \in I \subseteq [0,1]), p = \Gamma^1(t)\cdots\Gamma^n(t)q\}.$$

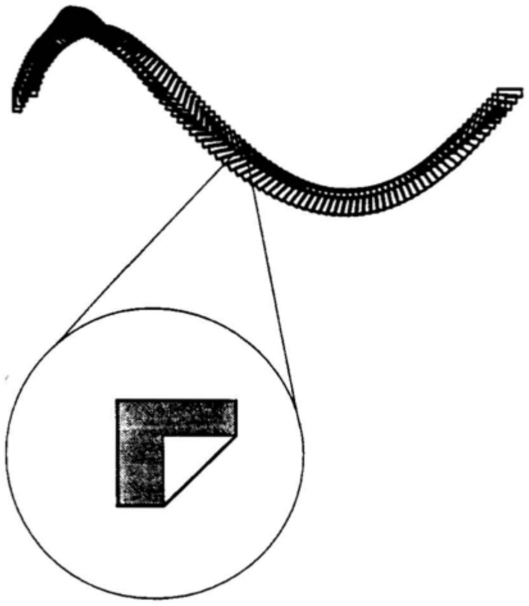
Figure 4: A RSV and its relation to an infinity of CSVs.

But for $t \in I$,

$$\Gamma^n(t)q \subseteq RSV(\Gamma^n(I), P).$$

Hence,

$$\Gamma^1(t)\cdots\Gamma^n(t)q \subseteq \Gamma^1(t)\cdots\Gamma^{n-1}(t)RSV(\Gamma^n(I), P).$$

Since this is true for all $q \in P$, the desired result follows.

The proof for the CSV operation follows a similar pattern. **qed.**

When applied to translations, the CSV definition is simplified and a direct link between the RSV and the CSV can be established.

There is a direct link between the RSV of the convex hull of a polyhedron and the CSV of that polyhedron when the motion is a translation parallel to an axis of the coordinate system. Indeed, the trace and the interval bounded by the minimum and maximum coordinates along an axis of the coordinate system are the same.

**Lemma 1** *Given $T(t)$, $t \in I \subseteq [0,1]$, a translation parallel to an axis of the coordinate system, and $P$ a polyhedron, then*

$$\mathrm{RSV}(T(I),\ \mathrm{CH}(P)) \ = \ \mathrm{CSV}(T(I),\ P)$$

**Proof:** Without loss of generality, consider the motion of a point of $CH(P)$. The trace of the moving point along an axis of the coordinate system has a width equal to the difference of the extremal values taken on that axis. Since this is true for all points of $CH(P)$, the desired result follows. **qed.**

Building on the previous results it is also possible to show that

**Corollary 1** *Given a polyhedron $P$ and a translation $T(t) = T_x(t)T_y(t)T_z(t)$, $t \in I \subseteq [0,1]$, then*

$$\begin{aligned}&\mathrm{CSV}(T(I),\ P) = \\ &\quad \mathrm{RSV}(T_x(I),\ \mathrm{RSV}(T_y(I),\ \mathrm{RSV}(T_z(I),\ \mathrm{CH}(P)))).\end{aligned}$$

## 7 Computing the CSV of a Polytope

Following definition 8, the computation of the $CSV(\Gamma(I), P)$ entails the computation of the bounding boxes around the vertices of $P$, and then, the computation of the convex hull of these bounding boxes.

To enforce our design criterion, we resort to interval analysis.

### 7.1 Interval Analysis Methods

To compute $f(I)$ over interval $I$, it is necessary to have interval expressions for the functions composing $f(I)$. An interval expression of a function is called an *inclusion function* and the simplest ones are those for the arithmetic operators.

**Def.: 10** *Let the intervals $[a,b]$ and $[c,d]$ be subsets of $\mathcal{R}$, with $a \leq b$ and $c \leq d$. Then, the inclusion functions of the arithmetic operators are*

$$\begin{array}{lcll}
[a,b] & + & [c,d] & \equiv [a+c, b+d], \\
[a,b] & - & [c,d] & \equiv [a-d, b-c], \\
[a,b] & * & [c,d] & \equiv [\min(ac,ad,bc,bd), \\
 & & & \quad \max(ac,ad,bc,bd)], \\
[a,b] & / & [c,d] & \equiv [\min(a/c,a/d,b/c,b/d), \\
 & & & \quad \max(a/c,a/d,b/c,b/d)], \\
 & & & \quad 0 \notin [c,d].
\end{array}$$

It is important to mention that the bounds of an interval may be real numbers not representable on a digital computer. Since a bound cannot be an interval, it must be a correctly rounded approximation either toward $+\infty$, for the upper bound, or toward $-\infty$, for the lower bound. In practice, this problem is solved using the round$-\infty$ and round$+\infty$ modes of the IEEE floating point standard.

Also, for other operators, namely the trigonometric operators sine and cosine, it is simple to construct their interval functions once their monotonicity intervals are known [Snyder 92]:

**Def.: 11**

$$\cos([a,b]) = \begin{cases}
[-1,1], & \text{if} 1 + \lceil \frac{a}{\pi} \rceil \leq \frac{a}{\pi} \\
[-1, \max(\cos(a),\cos(b))], & \text{if} \lceil \frac{a}{\pi} \rceil \leq \frac{b}{\pi} \wedge \\
 & \quad \lceil \frac{a}{\pi} \rceil \bmod 2 = 1 \\
[\min(\cos(a),\cos(b)), 1], & \text{if} \lceil \frac{a}{\pi} \rceil \leq \frac{b}{\pi} \wedge \\
 & \quad \lceil \frac{a}{\pi} \rceil \bmod 2 = 0 \\
[\min(\cos(a),\cos(b)), & \\
\ \max(\cos(a),\cos(b))], & \text{otherwise.}
\end{cases}$$

Thus, it is now possible to evaluate functions composed of simple arithmetic and trigonometric operators. For example, the evaluation of a polynomial $P(I)$ yields an interval that bounds all the possible values of $P(I)$.

### 7.2 Computing the Bounding Boxes

To compute the bounding box of a moving vertex it is necessary to compute the extrema of each parameter dependent coordinate of the Cartesian trajectory of the vertex. Without loss of generality, let $\xi(x, I)$ be the set of relative extrema of $x(t)$, $t \in I$. Then, the extremal values of $x(t)$ on $I = [b_1, b_2] \subseteq [0,1]$ are given by:

$$\begin{aligned}&\min\{x(b_1),\ x(b_2),\ \xi(x,I)\} \quad \text{and} \\ &\max\{x(b_1),\ x(b_2),\ \xi(x,I)\}.\end{aligned}$$

An extremum is found in $I$ if its bracketing interval intersects $I$. This behavior ensures that the robustness criterion is enforced.

### 7.3 Evaluation of the Extrema

To compute the extrema, any numerical or algebraic method is valid, but even with interval methods the computational cost is high. A simpler method would be to approximate the extremal values of $x(t)$ on $I$ by simply evaluating $x(I)$. If this is carried out with interval analysis methods, the result is an interval sure to contain the maximum and minimum values of $x(t)$ on $I$. However, it is not clear which of the two methods of evaluating extrema is more efficient.

Interval analysis methods suffer from *numerical swell.* Interval bounds tend to become very large as the number of evaluated inclusion functions grows. This translates into intervals that still include the theoretical result but for which the interval bounds are very distant. In our algorithm this translates into more and more evaluated subintervals to get the desired precision.

The implementation being underway, it is difficult to judge which method of evaluating extrema is best.

## 8 Algorithm

The results of the previous sections lead to a simple algorithm for collision detection. First, the complete trajectory is tested for collisions. If a collision is found, then the trajectory is split and the algorithm is recursively called for each subinterval. This process continues until no collision occurs or the tested subinterval is too small [Hayward 86].

```
ColDet( $O_1$, $\Gamma_1$, $O_2$, $\Gamma_2$ ): RETURN
 A list of collision subintervals.
 — $O_i$: set of polytopes modeling an object.
 — $\Gamma_i$: trajectory of an object.

   RETURN
        $\bigcup_{P_1 \in O_1, P_2 \in O_2}$ PolytopeColDet( $P_1$, $\Gamma_1$, $P_2$, $\Gamma_2$ )
END

PolytopeColDet( $P_1$, $\Gamma_1$, $P_2$, $\Gamma_2$ ): RETURN
 A list of collision subintervals.
 — $P_i$ is a polytope.

   RETURN ColDetInt( $P_1$, $\Gamma_1$, $P_2$, $\Gamma_2$, $[0,1]$ )
END

ColDetInt( $P_1$, $\Gamma_1$, $P_2$, $\Gamma_2$, $I$ ): RETURN
 A list of collision subintervals.
 — This is the function that really does the work.
 — $P_i$ is a polytope and $I$ is the subinterval.

   IF Interference( CSV($P_1$, $\Gamma_1$, $I$), CSV($P_2$, $\Gamma_2$, $I$ ) )
      AND NotTooSmall( $I$ ) THEN
      Split $I$ in $I_1$ and $I_2$
      RETURN
        ColDetInt($P_1$, $\Gamma_1$, $P_2$, $\Gamma_2$, $I_1$) $\bigcup$
          ColDetInt($P_1$, $\Gamma_1$, $P_2$, $\Gamma_2$, $I_2$)
   ELSE IF Interference( CSV($P_1$, $I$), CSV($P_2$, $I$ ) )
           AND TooSmall( $I$ ) THEN
           RETURN $I$
   ELSE
        RETURN $\emptyset$
END

Interference( $C_1$, $C_2$ ): RETURN Boolean.
 — Returns true if there is an interference.
 — $C_i$: a polytope.

Interference is detected using one of the mentioned
methods in the review:
   linear programming,
   computational geometry methods,
   distance computation (quadratic programming)...
END

CSV( $\Gamma(I)$, $P$ ): RETURN
 A polytope encompassing the RSV.
 — $P$ is the input, a moving polytope.
 — $\Gamma(I)$, $I \subset [0,1]$, is the trajectory.

   BB = $\emptyset$
   for each vertex $p$ of $P$
      BB = BB $\cup$ ComputeBoundingBox( $\Gamma(I)$, $p$ )

   RETURN ComputeConvexHull( BB )
   /* For the convex hull, use a known algorithm
      as long as interval analysis methods
      are applied. */
END
```

## 9 Application to an Open Kinematic Chain

To put into perspective the material of the previous sections, consider its application to an open kinematic chain with $n$ degrees of freedom. The transforms of the kinematic chains are $\boldsymbol{\Gamma}_1(t) \cdots \boldsymbol{\Gamma}_n(t)$, where $\boldsymbol{\Gamma}_i(t)$ is the local transform from frame $\{i\}$ attached to link $i$ to frame $\{i-1\}$ attached to link $\{i-1\}$.

The following computations are necessary to determine if a collision occurs on the interval $I$. Apply the collision detection algorithm of section 8 to each link. More precisely, let $\mathcal{O}$ be the set of polytopes modeling the manipulator and the obstacles, and let $\mathcal{L}_i$ be the set of polytopes modeling link $i$, then:

```
For $i = n..1$ do
   For each $P \in \mathcal{L}_i$ do
      For each $O \in \mathcal{O} \backslash \mathcal{L}_i$ do
         ColDet($P$, $\Gamma_1 \cdots \Gamma_i$, $O$, $\Gamma$)
      end
   end
end
```

Notice that computations for link $i$ only necessitate the knowledge of the transforms $\Gamma_1(t) \cdots \Gamma_i(t)$ and that the following $(n-i)$ links are not involved.

Since computing the CSV is generally a complex task, the application of theorem 3 will ease the computational burden at the expense of the goodness of fit of the output of the CSV. Indeed, successive CSVs generate larger and larger swept volumes which generate more and more collisions. Thus, to counteract the volume growth and to reach the desired precision, the interval $I$ will be split into finer and finer subintervals.

If the open kinematic chain is composed of revolute and prismatic joints, then the simplicity of the transforms greatly facilitates the computation of the CSV (see next subsection). Then, the partioning of $I$ into finer and finer subintervals is not a very high price to pay. Surely, an optimum tradeoff should be achieved between the repeated use of theorem 3 and the complexity of computing a CSV for more complicated transforms.

Concretely, one must be guided by the architecture of the manipulator. For example consider a wrist partitioned manipulator, like the PUMA-560. Let the transforms be $\boldsymbol{\Gamma}_1(t) \cdots \boldsymbol{\Gamma}_6(t)$. The computation could be,

**Wrist :**

- Big payload - Compute the CSV of the polytopes rigidly attached to the end-effector. Apply theorem 3 to the wrist,

$$\mathrm{CSV}(\Gamma_1(I) \cdots \Gamma_3(I),\ \mathrm{CSV}(\Gamma_4(I) \cdots \Gamma_6(I),\ P)),$$

or to each part of the wrist,

$$\mathrm{CSV}(\Gamma_1(I)\cdots\Gamma_3(I),\\ \mathrm{CSV}(\Gamma_4(I),\mathrm{CSV}(\Gamma_5(I),\mathrm{CSV}(\Gamma_6(I),\ P)))).$$

The first of these decomposition computes the CSV as seen from frame 3 and then takes that CSV as a starting point to compute the final CSV seen from the base. The second decomposition successively computes the CSVs from frame 6 to frame 3.

• Small payload - The links of the wrist are small and may be modeled by a single polytope. Thus no CSVs are computed for the links in the wrist and the modeling polytope is taken to be attached rigidly to the third link.

**Body :** There are not many variations possible for the first three links, because they generally induce large displacements compared to the links of the wrist. Hence, compute the CSVs of the first three links and take as input the CSV output from the wrist.

## 9.1 Bounding Box Computations for Prismatic and Revolute Joints

Most manipulators are constructed with prismatic and revolute joints. As a consequence of using the Denavit-Hartenberg notational scheme, it is rather easy to compute the bounding boxes and hence the CSV of a prismatic or a revolute joint.

**Prismatic Joint :** To define the bounding box of a vertex translating along the $z$ axis, it is necessary to find the minimum and maximum values on that axis. This is quite simple if the variation of $z$ is monotonic, otherwise the extrema must be estimated.

**Revolute Joint :** A rotating vertex traces an arc in the $x-y$ plane of its local coordinate system. To compute the bounding box of an arc, the minimum and maximum values of its angle is computed. If the angle variation is monotonic, this is a simple task, otherwise the extrema must be calculated.

### 9.1.1 The Bounding Box of an Arc

Since revolute joints are so common, we propose a method for the derivation of the optimal bounding box of an arc in local coordinates. Four cases occur: no angular displacement, an angular displacement smaller than or equal to $\pi$, strictly bounded by $\pi$ and $2\pi$, and greater then or equal to $2\pi$ (see figure 5).

Without loss of generality, let $\boldsymbol{a}$ and $\boldsymbol{b}$ be the end-points of an arc expressed in two dimensional Cartesian coordinates. Also, let $\theta$ be the angular displacement between $\boldsymbol{a}$ and $\boldsymbol{b}$ and let $\theta_0$ be the angular displacement of $\boldsymbol{a}$ with respect to the $x$ axis.

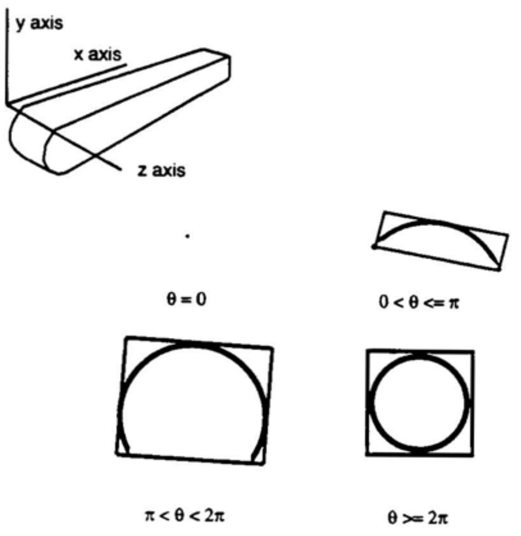

Figure 5: The bounding boxes of an arc.

1. No displacement - When there is no displacement, the bounding box is the point itself.

2. $0<\theta\leq\pi$ - The point on the arc furthest away from the segment joining $\boldsymbol{a}$ and $\boldsymbol{b}$ is attained at $\theta/2$. This corresponds to the middle vector

   $$\boldsymbol{c}\ =\ (\ \|\ \boldsymbol{a}\ \|\cos(\theta/2+\theta_0),\ \|\ \boldsymbol{a}\ \|\sin(\theta/2+\theta_0)\ ).$$

   One side of the optimal bounding box passes by $\boldsymbol{a}$ and $\boldsymbol{b}$, the other passes by $\boldsymbol{c}$, parallel to $\boldsymbol{b}-\boldsymbol{a}$. Hence the four following points

   $$\begin{array}{cc}\boldsymbol{a}, & \boldsymbol{b}\\ \boldsymbol{c}+\boldsymbol{b}/2-\boldsymbol{a}/2, & \boldsymbol{c}+\boldsymbol{a}/2-\boldsymbol{b}/2.\end{array}$$

3. $\pi<\theta<2\pi$ - In this case, the middle vector $\boldsymbol{c}$ is still usable but the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ are not corners of the bounding box. The vector $\boldsymbol{d}=(\ \|\ \boldsymbol{a}\ \|\sin(\theta/2+\theta_0),\ -\ \|\ \boldsymbol{a}\ \|\cos(\theta/2+\theta_0)\ )$, which is perpendicular to $\boldsymbol{c}$, is used to define the bounding box:

   $$\begin{array}{cc}\boldsymbol{c}+\boldsymbol{d}, & \boldsymbol{c}+-\boldsymbol{d}\\ \boldsymbol{a}/2+\boldsymbol{b}/2+\boldsymbol{d}, & \boldsymbol{a}/2+\boldsymbol{b}/2+-\boldsymbol{d}.\end{array}$$

4. $\theta\geq 2\pi$ - When the angular displacement is greater than $2\pi$, then any square of side $2\ \|\ \boldsymbol{a}\ \|$ is a good bounding box. For example, picking the square aligned on the axes of the local coordinate system gives the bounding box,

   $$\begin{array}{cc}(\|\ \boldsymbol{a}\ \|,\ \|\ \boldsymbol{a}\ \|), & (-\ \|\ \boldsymbol{a}\ \|,\ \|\ \boldsymbol{a}\ \|)\\ (\|\ \boldsymbol{a}\ \|,\ -\ \|\ \boldsymbol{a}\ \|), & (-\ \|\ \boldsymbol{a}\ \|,\ -\ \|\ \boldsymbol{a}\ \|),\end{array}$$

   or letting $\boldsymbol{a}^{\perp}$ a vector perpendicular to $\boldsymbol{a}$,

   $$\begin{array}{cc}\boldsymbol{a}^{\perp}+\boldsymbol{a}, & \boldsymbol{a}^{\perp}+-\boldsymbol{a}\\ -\boldsymbol{a}^{\perp}+\boldsymbol{a} & -\boldsymbol{a}^{\perp}+-\boldsymbol{a}.\end{array}$$

## 10 Conclusion

This paper presented a safe and reliable algorithm to implement a collision detection method based on the swept volume approach.

The algorithm is safe in the sense that the convex approximation (CSV) of the real swept volume (RSV) is sure to encompass the RSV. Moreover, the algorithm is safe because it relies on interval analysis methods.

Further work needs to be done on the bounding box computations, a better bounding box implies a better CSV.

Finally, the practicality of the algorithm will be established when its implementation is finished and tested.

## 11 Acknowledgments

This work has been supported in part by a research contract with SPAR Aerospace Ltd of Toronto, and by grants from FCAR (Fonds pour les Chercheurs et Aide à la Recherche) and NSERC (National Sciences and Engineering Research Council). Additional support was provided by the Department of Mathematics and Computer Science of the Université du Québec à Trois-Rivières.

## References

[Boyse 79] Boyse, J. W. 1979. Interference detection among solids and surfaces, *Communications of the ACM*, Vol. 22, pp. 3–9.

[Cameron 85] Cameron, S., 1985. A study of the clash detection problem in robotics, *IEEE Int. Conf. on Robotics and Automation*, St-Louis, MO, pp. 488–493.

[Cameron 90] Cameron, S., 1990. Collision Detection by Four-Dimensional Intersection Testing, *IEEE Transactions on Robotics and Automation*, Vol. RA-6, No. 3, pp. 291–302.

[Canny 86] Canny, J. 1986. Collision detection for moving polyhedra, *IEEE PAMI*, Vol. 8, No. 2, pp. 200–209.

[Dobkin *et al.* 83] Dobkin, D.P, and Kirkpatrick, D.G. 1983. Fast detection of polyhedral intersection, *Theoretical Computer Science*, Vol. 27, pp. 241–253.

[Dobkin *et al.* 85] Dobkin, D.P. and Kirkpatrick, D.G., 1985. A linear algorithm for determining the separation of convex polyhedra, *Journal of Algorithms*, Vol. 6, pp. 381–392.

[Foisy *et al.* 90] Foisy, A., Hayward, V., and Aubry, S. 1990. The use of "awareness" in collision prediction. *IEEE Int. Conf. on Robotics and Automation*, Cincinnati, OH, pp. 338–343.

[Foisy *et al.* 92] Foisy, A., Hayward, V. 1992. *Final Report for Development of Collision-Free Motion Planning Technique for the MCPL, the MSS Command and Programming Language*, Contract #09021 TF with SPAR Aerospace Ltd, Toronto.

[Ganter 85] Gartner, M.A. 1985. *Dynamic collision Detection using Kinematics and Solid Modeling Techniques*, Ph.D. Thesis (Mechanical Engineering), University of Wisconsin.

[Gilbert *et al.* 85] Gilbert, E.G. and Johnson, D.W., 1985. Distance functions and their application to robot path planning in the presence of obstacles, *IEEE J. of Robotics and Automation*, Vol. 1, No. 1.

[Gilbert *et al.* 88] Gilbert, E.G, Johnson, D.W., and Keerthi, S.S., 1988. A fast procedure for computing the distance between complex objects in three-dimensional space", *IEEE Journal of Robotics and Automation*, Vol. 4, No. 2, pp. 193–203.

[Hayward 86] Hayward, V., 1986. Fast collision detection scheme by recursive decomposition of a manipulator workspace, *IEEE Int. Conf. on Robotics and Automation*, San-Francisco, CA, pp. 1044–1049.

[Hurteau *et al.* 83] Hurteau, G., and Stewart, N.F., 1983. Distance Calculation for Imminent Collision Indication in a Robot System Simulation, *Robotica*, Vol. 6, pp. 47–51.

[Moore 66] Moore, R.E. 1966. *Interval Analysis*, Prentice Hall, Englewood Cliffs, New Jersey.

[Moore 79] Moore, R.E., 1979. *Methods and Applications of Interval Analysis*, SIAM, Philadelphia.

[Rimon *et al.* 92] Rimon, E. and Boyd, S. P., 1992. (February) *Efficient Distance Computation Using Best Ellipsoid Fit*, Tech. Rep., Stanford University, Dept of Electrical Engineering.

[Snyder 92] Snyder, J. M. 1992. *Generative Modeling for Computer Graphics and CAD*, Academic Press.

[Von Herzen *et al.* 90] Von Herzen, B., Barr, A., and Zatz, H. R. 1990. Geometric collisions for time-dependent parametric surfaces, *Computer Graphics*, vol. 24, no 4, pp. 39-48.

[Young *et al.* 72] Young, D. and Gregory, R. 1972. *A Survey of Numerical Mathematics*, Addison Wesley.