Practical Sliced Configuration Spaces for Curved Planar Pairs

Elisha Sacks

Purdue University, eps@cs.purdue.edu

PRACTICAL SLICED CONFIGURATION
SPACES FOR CURVED PLANAR PAIRS

Elisha Sacks
Department of Computer Sciences
Purdue University
West Lafayette, IN   47907

Practical Sliced Configuration Spaces for Curved Planar Pairs

Elisha Sacks

1398 Computer Science Building

Purdue University

West Lafayette, IN 47907, USA

Email: eps@cs.purdue.edu

August 5, 1997

Abstract

I present a practical configuration space computation algorithm for pairs of curved planar parts based on the general algorithm developed by Bajaj and me. The general algorithm advances the theoretical understanding of configuration space computation, but is too slow and fragile for some applications. The new algorithm solves these problems by restricting the analysis to parts bounded by line segments and circular arcs, whereas the general algorithm handles rational parametric curves. The trade-off is worthwhile because the restricted class handles most robotics and mechanical engineering applications. The algorithm reduces runtime by a factor of 60 on nine representative engineering pairs and by a factor of 9 on two human knee pairs. It also handles common special pairs by specialized methods. A survey of 2500 mechanisms shows that these methods cover 90% of pairs and yield an additional factor of 10 reduction in average runtime. The theme of this article is that applications requirements, as well as intrinsic theoretical interest, should drive configuration space research.

Submitted to *International Journal of Robotics Research.*

1 Introduction

I present a practical configuration space computation algorithm for pairs of curved planar parts based on the general algorithm developed by Bajaj and me [5]. The purpose of the algorithm is to support robotics and mechanical engineering tasks that involve contact analysis, such as dynamical simulation, functional tolerance analysis, and assembly planning. It has to be general enough to cover the intended applications, robust enough to be worth using, and ideally fast enough to be used interactively. The general algorithm advances the theoretical understanding of configuration space computation, but initial experience shows that it is fragile and slow on some applications.

The unnecessary generality of the algorithm limits its robustness and speed. The problems arise from the assumptions that parts are bounded by rational parametric curves and have three degrees of freedom. Empirical evidence shows that few applications require this generality. We can make do with line segments and circular arcs in all robotics applications and in most mechanical engineering applications [4]. We need the circular arcs because they model the fundamental motion of compliant rotation, but we can adequately approximate everything else. Few useful pairs consist of two independent parts each with three degrees of freedom. Instead, the parts are connected by simple joints or have one degree of freedom. Many complex parts consist of symmetric patterns of simple features.

Based on these observations, I have developed a new algorithm that trades generality for robust speed. It analyzes complex pairs in a few minutes on a workstation, versus hours for the general algorithm. The rest of the paper is organized as follows. Section 2 reviews the general algorithm. Section 3 describes the new algorithm. In Section 4, I evaluate the new algorithm on 11 representative pairs and estimate its expected performance based on a survey of 2500 mechanisms. I conclude with a discussion whose theme is that applications requirements, as well as intrinsic theoretical interest, should drive configuration space research.

2 General algorithm

The general algorithm computes the configuration space of a moving part with respect to a stationary obstacle. (We reduce a pair of moving parts to this case by attaching the reference coordinate frame to one part.) It partitions the configuration space along the orientation axis into intervals of equivalent slices separated by critical slices where the contact structure changes (Figure 1). The slices are planar configuration spaces in which the moving part translates at fixed orientations. Slice equivalence implies that the portion of the configuration space between adjacent critical slices, called a band, is expressible as closed loops of contact patches that enclose free regions. Each patch has left and right neighbors within its band, upper neighbors in the band above, and lower neighbors in the band below.
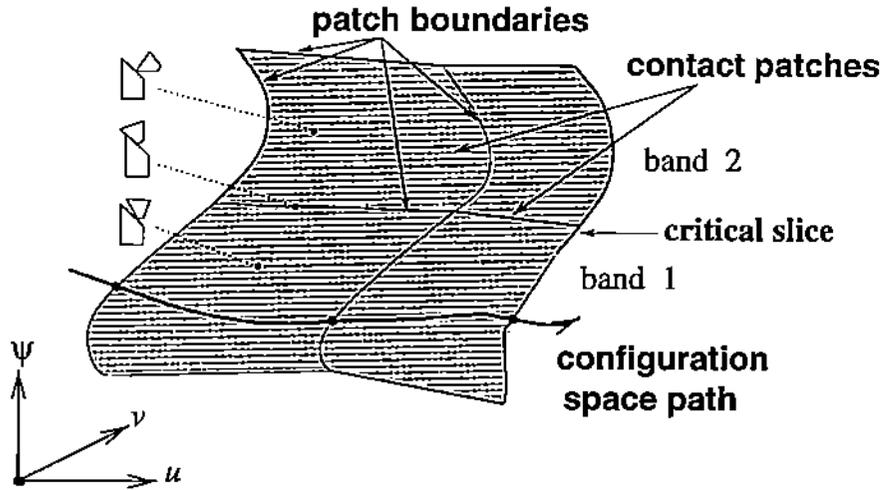
Figure 1: Illustration of contact space. Part contacts on the left show typical configurations.

The algorithm consists of three steps. The first step finds the critical orientations by formulating and solving algebraic equations for all combinations of feature contacts. We assume that the criticalities are finite in number, which is generically true. The second step generates configuration space slices at closely spaced orientations. Each slice consists of closed contours of contact curve segments that bound open free space regions. The third step constructs a polyhedral approximation of the contact space from the contours. We did not attempt to construct the exact contact space, which consists of algebraic patches that intersect along multiple-contact curves.

3   Specialized algorithm

The new algorithm handles parts bounded by line segments and circular arcs. It allows three degrees of freedom per part, but analyzes common special cases with specialized methods. It reduces redundant criticality computation, reduces the number of slices, and replaces polyhedral approximation of contact space with exact patch computation.

3.1   Contact space computation

The algorithm constructs an exact representation of the contact space: a graph whose nodes represent contact patches and whose links represent patch adjacency. Each node contains a subroutine that computes an implicit contact function that is zero on the patch, positive in nearby free configurations, and negative in nearby blocked configurations. Each link contains pointers to the adjacent patches along with subroutines that compute the boundary curves.

The contact graph is not a full boundary representation because it does not encode the edge ordering at vertices, but it suffices for contact analysis.

The algorithm consists of two phases. The first phase independently constructs the contact patches in each band. It constructs the slice midway between the upper and lower critical orientations that bound the band. Slice equivalence guarantees that the segments are in direct correspondence with the patches in the band and that the left/right patch adjacencies are the same as the corresponding curve adjacencies [5]. The algorithm construct a node (patch) for each segment and a link (patch adjacency) for each segment adjacency. The second phase links upper/lower neighbors whose top/bottom curves overlap. These curves are lines or circles, as explained below.

The algorithm derives the contact functions from the geometry of the touching features. There is one type of contact patch for each combination of moving and fixed features: moving arc/fixed line, moving line/fixed arc, and moving arc/fixed arc. Contacts involving points are identical to those for arcs of radius zero. Line/line contacts are subsumed by line/point contacts. The arc/line contact condition (Figure 2a) is that the distance d between the center o_A of the arc and the line lm equals the arc radius r_A, which is expressible as

$$(\mathbf{o}_A - \mathbf{l}) \times (\mathbf{m} - \mathbf{l}) - dr_A = 0 \tag{1}$$

where $\times$ denotes the vector cross-product. The arc/arc contact condition (Figure 2b) is that the distance between the centers equals the sum of the radii, which is expressible as

$$(\mathbf{o}_A - \mathbf{o}_B) \cdot (\mathbf{o}_A - \mathbf{o}_B) = (r_A + r_B)^2 \tag{2}$$

where r_A and r_B are positive for convex arcs and negative for concave arcs. We obtain the contact functions by expressing the points in coordinate form: a fixed point p becomes (p_x, p_y) and a moving point q becomes $(x, y) + R_\theta(q_x, q_y)$ with (x, y) the part position, θ its orientation, and R_θ the rotation operator.

The algorithm derives the boundary curves from the geometry of the two pairs of touching features in the adjacent patches. There is one type of patch boundary curve for each of the six pairs of patch types, for example moving arc/fixed arc patch intersects moving arc/fixed line patch. The algorithm derives parametric expressions $(x(\theta), y(\theta))$ in the orientation θ for each type of curve by extending Donald's [2] method from polygons to circular arcs. Donald observes that the contact patches of polygons are expressible as lines parameterized by θ. He obtains a parametric expression for the intersection curve of two patches by intersecting their parametric lines. We observe from Equations 1 and 2 that the contact patches of arc/line contacts are parametric lines, while the contact curves of arc/arc contacts are parametric circles. For example, the arc/arc equation is

$$x^2 + y^2 + 2(o_x \cos\theta - o_y \sin\theta)x + 2(o_x \sin\theta + o_y \cos\theta)y + k = 0 \tag{3}$$

with o the center of the moving arc and k a constant. The algorithm intersects two lines, a line and an arc, or two arcs to obtain the parametric boundary curve. The computation
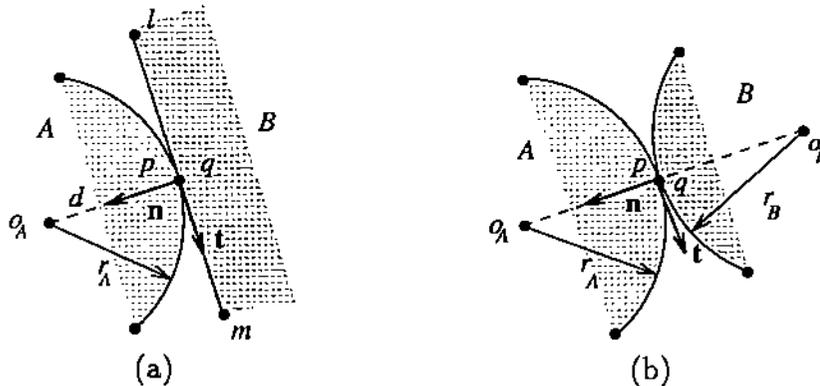
4

Figure 2: Planar feature contacts: (a) contact between a circular arc and a line segment, and (b) contact between two circular arcs. Shading indicates part interior.

is well-conditioned unless the contact curves are almost tangent or almost parallel, in which case the algorithm interpolates from nearby slices.

The need to compute exact patch boundary curves follows from the empirical failings of the simpler approach of approximating boundary curves by line segments through their upper and lower boundary points. That algorithm has to generate enough slices to approximate the true boundary curve to a specified tolerance. The extra slices, beyond one per critical θ to compute patch adjacency, slow down the computation. The gaps between the linear boundary curves and the adjacent patches violate the configuration space topology by connecting the free and blocked spaces. This violation sometimes causes a dynamical simulator that tracks part contacts in configuration space to fail at contact transitions. It will probably cause problems for other algorithms, such as path planning and assembly planning, that use the configuration space topology.

3.2   Special pairs

Many interesting pairs are too complex for the general algorithm, but are readily analyzable by specialized algorithms. The most important classes are lower pairs, higher pairs with two degrees of freedom, and symmetric pairs. (A lower pair consists of two parts that interact via a permanent surface contact. Everything else is a higher pair.) The new algorithm analyzes each class with its own method, but packages the results in the general format, so that applications programs can interact uniformly with all types of configuration spaces. The specialized algorithms are much faster than the general algorithm. Lower pairs take essentially no time, two degree of freedom pairs take under a tenth of a second, and k-symmetric pairs run almost k times faster than before.

Figure 3 shows a mechanism that illustrates the three classes. The mechanism consists of a driver, a link, a pawl, a ratchet, and a frame. None of the six interacting pairs requires a
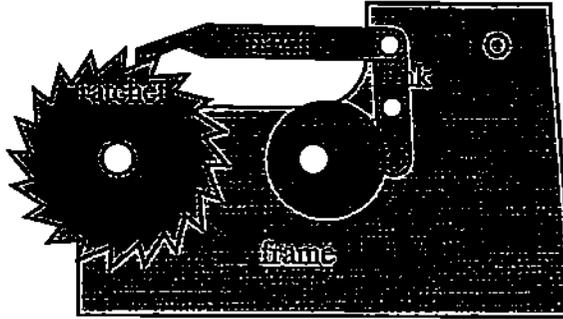
Figure 3: Ratchet mechanism with specialized pairs.

general analysis. The driver/frame, link/frame, ratchet/frame, and link/pawl form revolute joints, the most common lower pair. The driver/link pair has two degrees of freedom because each part has one (rotational) degree of freedom. The ratchet has 20 identical teeth, so the ratchet/pawl pair is symmetric under an 18 degree rotation.

The algorithm uses table lookup to compute lower pair configuration spaces. A general analysis is unnecessary because each pair imposes a single, permanent contact with a known contact equation. The equations represent point/line and point/arc contacts, so the contact spaces have the same structure as in the general case. The only difference is a flag that indicates that the contact cannot break, which saves the dynamical simulator from needless contact change tests.

The specialized algorithm for higher pairs with two degrees of freedom is described elsewhere [6]. The output is in a two-dimensional coordinate system that is incompatible with the patch representation. For example, the configuration space of a rotation/rotation pair is a torus whose coordinates are the part orientations relative to a fixed frame. The conversion to the patched format is straightforward and fast. The contact graph consists of one slice. The converter computes a patch for each planar contact curve. It derives the contact function by the standard method. The left and right neighbors of a patch are the patches for the corresponding contact curves. There are no top or bottom neighbors. Patch transitions occur in configurations where the motion coordinate of the first part equals its left or right limiting value, which differs from the standard semantics.

Symmetric pairs are analyzed by a simple extension of the general algorithm. The algorithm computes the configuration space of the symmetric part with respect to the other part. If the part is symmetric under rotation by α degrees, the configuration space consists of $k = 360/\alpha$ identical slabs along the orientation axis. The algorithm computes the first slab with the general algorithm then make $k - 1$ copies shifted by $\alpha, 2\alpha, \ldots, (k-1)\alpha$ degrees. Figure 4 shows the shift operation.

6

field	original	shifted by β
lower bound	l	$l + \beta$
upper bound	u	$u + \beta$
contact function	$f(x, y, \theta)$	$f(x, y, \theta - \beta)$
boundary function	$(u(\theta), v(\theta))$	$(u(\theta - \beta), v(\theta - \beta))$
neighbors	p	shifted p

Figure 4: Shifting a patch along the θ axis.

3.3 Redundant criticalities

The general algorithm computes the critical configurations of the pair, prunes unrealizable criticalities, and groups the rest by orientation. Experience shows that the pruning step takes 80% of the running time. The new algorithm reduces this time by testing if each criticality has the same orientation (to a tolerance) as some previously computed criticality. If so, it does not prune, since the orientation is already known and the exact critical configuration is not needed in the final output. The new algorithm also ignores criticalities that fall outside an input interval of orientations. It uses this feature to compute only the relevant criticalities of symmetric pairs. For example, the new algorithm computes the ratchet/pawl criticalities in 30 seconds, versus 100 seconds for the general algorithm.

4 Evaluation

I first estimated the speedup on general pairs (pairs with three degrees of freedom) due to exact patch computation and symmetry reduction. Figure 5 shows the results for 11 pairs. The number of old slices is for a separation of 0.005 radians, which is the minimum needed for topological correctness and for 1% accuracy. The numbers of new slice in parentheses are without symmetry reduction. On average, the new algorithm reduces the number of slices by a factor of 62 on the engineering pairs (the first nine examples). I expect similar results in general because the examples span a broad range of applications. The program reduces the slice count only by a factor of 9 on the human joints because they have many features without any symmetry.

I then estimated the prevalence of special case pairs (lower pairs and higher pairs with two degrees of freedom) in mechanical assemblies based on a survey of 2500 mechanisms in an engineering encyclopedia [4]. The survey consists of 559 pairs of which 72% are planar and of 1912 multi-pair mechanisms of which 85% are planar. The special cases account for 92% of the planar pairs. Of the planar mechanisms, 71% consist solely of special-case pairs and the rest contain on average one general pair and six special-case pairs. All told, the general algorithm is needed 10% of the time. We obtain a factor of 10 speedup on average because the running time of the special-case algorithms is negligible. The special-case algorithms

pair	features	old slices	new slices	reference
cam	16	1265	12	[5]
fixture	36	1293	72	[5]
fastener	32	1265	16	[5]
ratchet/pawl	96	1434	17 (325)	Section 3.2
geneva	61	1394	64 (256)	[8]
film advance	53	1261	4 (13)	[7]
crank/slider	24	1261	5	none
blocks	20	1261	5	none
chain gear	166	1262	3 (42)	[7]
tibia/femur	52	1366	196	[1]
patella/femur	24	1321	107	[1]

Figure 5: Slice reduction in new algorithm.

also cover 93% of the *spatial* pairs and 70% of the spatial mechanisms, although these cases are not implemented.

The survey covers traditional mechanisms, such as transmissions, ratchets, cams, and gears. An informal survey of modern mechanisms, such as VCR's and photocopiers, yields similar results. The mix of pairs is different in robots, part feeder, and other applications. Robots contain primarily lower pairs with the exception of contacts between anthropomorphic grippers and the environment. Most part feeders have a fixed frame that interacts with moving parts. There are few special pairs. Symmetry appears common (nuts, washers, gears, faucets), but I have not quantified this.

5 Conclusions

The message of this paper is that practical experience can drive configuration space research. One cannot anticipate the limitations of the general algorithm until one studies complex assemblies. Some problems have simple software engineering solutions. One example, redundant criticalities, is described, but there are many more. Other problems require special treatment of important special cases. Still others require algorithmic improvements. Conversely, the primary theoretical limitation of the algorithm, the restricted feature set of line segments and circular arcs, has proven irrelevant in applications so far. The next task is to compute configuration spaces for spatial pairs. General solutions appear infeasible, so empirical guidance is essential. The special cases in this paper cover many pairs, but the hardest case, higher pairs with three or more degrees of freedom, remains open.

Acknowledgments

References

[1] Bajaj, C., Bernardini, F., Cutchin, S., et al. Comprehensive analysis of joints from patient clinical data. Technical Report 97-019, Purdue University, 1997. submitted for publication.

[2] Donald, B. R. A search algorithm for motion planning with six degrees of freedom. *Artificial Intelligence* **31** (1987) 295–353.

[3] Goldberg, K., Halperin, D., Latombe, J., et al. (Eds.). *The Algorithmic Foundations of Robotics*. (A. K. Peters, Boston, MA, 1995).

[4] Joskowicz, L. and Sacks, E. Computational kinematics. *Artificial Intelligence* **51** (1991) 381–416. reprinted in [3].

[5] Sacks, E. and Bajaj, C. Sliced configuration spaces for curved planar bodies. *International Journal of Robotics Research* (1997). to appear.

[6] Sacks, E. and Joskowicz, L. Computational kinematic analysis of higher pairs with multiple contacts. *Journal of Mechanical Design* **117** (1995) 269–277.

[7] Sacks, E. and Joskowicz, L. Dynamical simulation of planar assemblies with changing contacts using configuration spaces. Technical Report 97-008, Purdue University, 1997. submitted for publication.

[8] Sacks, E. and Joskowicz, L. Parametric tolerance analysis of part contacts in general planar assemblies. Technical Report 97-027, Purdue University, 1997. submitted for publication.