1995

Sliced Configuration Spaces for Curved Planar Bodies

Elisha Sacks
Purdue University, eps@cs.purdue.edu

Chandrajit Bajaj

Report Number:
95-078

SLICED CONFIGURATION SPACES FOR CURVED PLANAR BODIES

Elisha Sacks
Chandrajit Bajaj

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

CSD-TR 95-078
December 1995
(Revised 12/96)

Sliced Configuration Spaces for Curved Planar Bodies

Elisha Sacks and Chandrajit Bajaj
1398 Computer Science Building
Purdue University
West Lafayette, IN 47907, USA

December 13, 1996

Abstract

We present the first practical, implemented configuration space computation algorithm for a curved planar object translating and rotating amidst stationary obstacles. The bodies are rigid, compact, regular, and bounded by a finite number of rational parametric curve segments. The algorithm represents the three-dimensional configuration space as two-dimensional slices in which the moving object has a fixed orientation. It discretizes the configuration space into intervals of equivalent slices separated by critical slices. The output is topologically correct and accurate to within a specified tolerance. We have implemented the algorithm for objects bounded by line segments and circular arcs, which is an important class for applications. The program is simple, fast, and robust. The slice representation is a natural and efficient abstract data type for geometric computations in robotics and engineering.

To appear in *International Journal of Robotics Research*

1 Introduction

We present the first practical, implemented configuration space computation algorithm for a curved planar object moving amidst stationary obstacles. Configuration space computation is important because it underlies algorithmic approaches to geometric reasoning. It supports the robotics tasks of path planning [20] and compliant assembly [9]. It supports mechanical engineering tasks that involve contact analysis, including mechanism design [26, 27], fastener design [12], part feeder design [9, 10], functional tolerancing [18], fixturing [8], and design for assembly [24, 29]. It supports the biomedical task of joint modeling [3, 19] (knees, hips, elbows) for diagnosis, therapy, and prosthesis design. It provides an alternative to collision detection for fast, robust dynamical simulation [28].

Configuration space is a concise, complete encoding of the motion constraints imposed on a rigid object by contacts with rigid obstacles. It is a manifold, the Cartesian product of the Cartesian plane and the unit circle, whose points represent the position and orientation of the moving object with respect to the obstacles. Configuration space partitions into free space where the object does not touch the obstacles and into blocked space where it overlaps an obstacle. The common boundary, called contact space, contains the configurations where the object touches an obstacle without overlap. Only free space and contact space are physically realizable. They represent the possible motions of the object and the couplings between its degrees of freedom induced by contacts with the obstacle. Configuration space computation is the task of constructing a representation of the configuration space partition.

Previous research provides configuration space computation algorithms for polygonal bodies. There are two main approaches. The first approach is to represent free space as a semi-algebraic set bounded by contact surface patches. Each patch represents the motion constraint when an object feature touches an obstacle feature. The patches intersect at multiple contact configurations. The second approach is to represent free space as a sequence of planar slices along the orientation axis where each slice represents the contact constraints when the object translates with a fixed orientation. The properties of the three-dimensional configuration space are derived from the slices. For example, intermediate configurations are interpolated from adjacent slices. The slices are chosen to ensure that the properties are computed correctly.

Previous research does not provide practical algorithms for curved bodies despite their importance in applications. Curved bodies are common in fine motion planning, feeder design, and fixture design. Polyhedral approximations are suboptimal because they introduce spurious discontinuities in the contact constraints that distort the system dynamics. Curved bodies are required to model coupled rotation, which occurs in pin joints, ball joints, gears, cams, and many other mechanical assemblies. Curved bodies are necessary for joint modeling because human joints exhibit compliant rotation.

We present a slice-based configuration space computation algorithm for bodies bounded by rational parametric curve segments. We partition the configuration space along the

orientation axis into intervals of equivalent slices separated by critical slices where the contact structure changes. We use the partition to compute the topology of the three-dimensional configuration space, to approximate the contact space geometry, and to derive specialized properties for robotics, mechanical design, and dynamical simulation. We have programmed a fast, robust implementation of our general algorithm for bodies bounded by line segments and circular arcs, which is an important class for applications.

Our research contributes to the theory and practice of configuration space computation. We extend the slice approach from point and line contacts to curve contacts. The slices generalize from polygons to semi-algebraic sets, slice computation generalizes from linear algebra to algebraic geometry, and the slice equivalence computations generalize from simple geometric tests, such as parallel object/obstacle edges, to transversality conditions on semi-algebraic sets. Our main theoretical result is an algebraic necessary condition for criticality that is completely general and efficiently solvable. We contribute the first practical algorithm and working program for curved bodies. It is also the first slice-based program that guarantees topological correctness for polygonal bodies. The program is simpler, more robust, and probably faster than boundary algorithms because it computes and intersects planar curve segments, whereas they compute and intersect surface patches.

The rest of the paper is organized as follows. The next section illustrates the role of configuration space in applications. Section 3 reviews previous work in configuration space computation. Sections 4–7 describe the components of our algorithm: the slice representation, criticality theorem, criticality computation, and slice construction. Sections 8 and 9 discuss the computational complexity of our algorithm and its robustness on non-generic input. The paper concludes with a discussion of our results and with a plan for future work on geometric computation using slices as abstract objects.

2 Applications

We illustrate the many applications of configuration spaces on examples from knee modeling, mechanism design, fixture design, and fastener design. The first two examples are open research problems, whereas the second two are simple instances of active research areas.

2.1 Knee modeling

Figure 1 shows a planar knee model obtained by slicing the femur and tibia on the sagittal plane. We hold the tibia fixed and allow the femur to move freely with position (x, y) and orientation θ. The figure shows the femur in configuration $(x = 5, y = -3, \theta = 0.5)$ and the corresponding (x, y) configuration space for this orientation. The shaded region is the blocked space where the femur and the tibia overlap. The white region is the free space. The dot in free space marks the displayed configuration of $(5, -3)$. The contact space consists

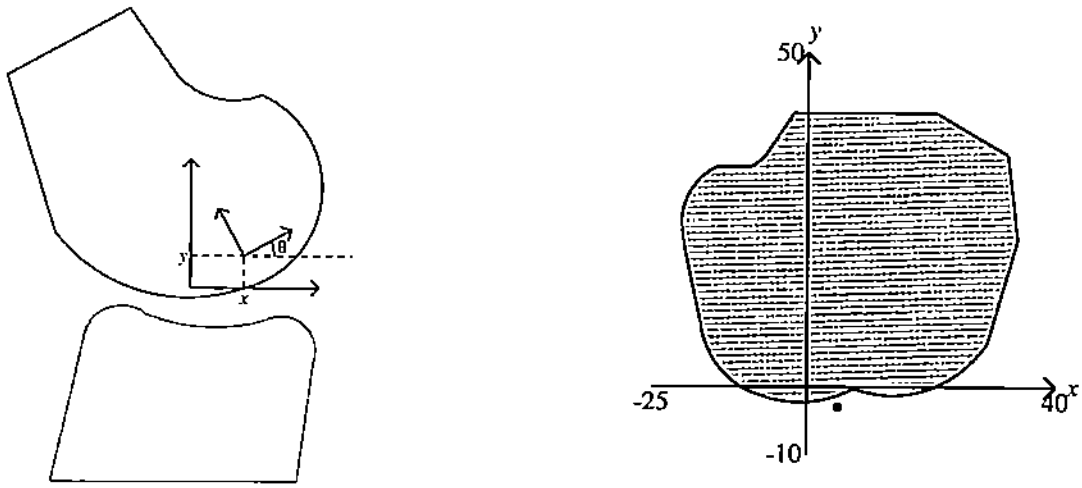


Figure 1: Planar knee model with $\theta = 0.5$ and its (x, y) configuration space.

of curve segments that represent contacts between the features of the tibia and the femur. Figure 2 shows the three-dimensional configuration space. The contact space is grey. The blocked space is the interior of the contact space and the free space is the exterior.

The configuration space supports joint modeling and analysis [3]. It describes the range of motion of the tibia with respect to the femur and the configurations where it hyperextends. Specific motions, such as flexion, extension, and rotation, correspond to paths in configuration space. The contact constraints allow us to formulate dynamical and stress equations directly from geometric models of the parts annotated with material properties and external forces.

2.2 Mechanism design

Figure 3 shows a cam mechanism from Artobolevsky's encyclopedia of mechanisms [1]. The function of the mechanism is to advance the film in a motion picture camera by repeatedly inserting the follower tip in a square hole on the side of the film, pushing the film down, and retracting. The cam has a constant-breadth profile consisting of four arc segments. The follower has a square profile whose sides are slightly longer than the cam breadth. The cam is mounted on a rotating shaft, indicated by a small circle. The follower is mounted on a frame that allows horizontal and vertical translation, but prevents rotation. Each rotation of the cam causes the follower tip to follow the square path marked by the dashed line. We compute the configuration space of the cam relative to the follower, which amounts to moving the cam while holding the follower fixed. The free space forms a narrow, spiral channel bounded by the contact space, which is shown in grey. The blocked space, everything outside the channel, is omitted for clarity.

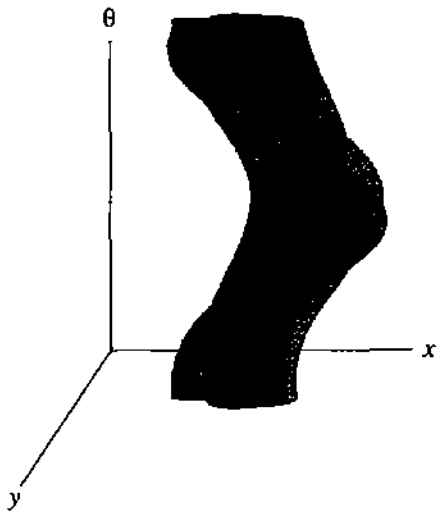


Figure 2: Configuration space of planar knee model.

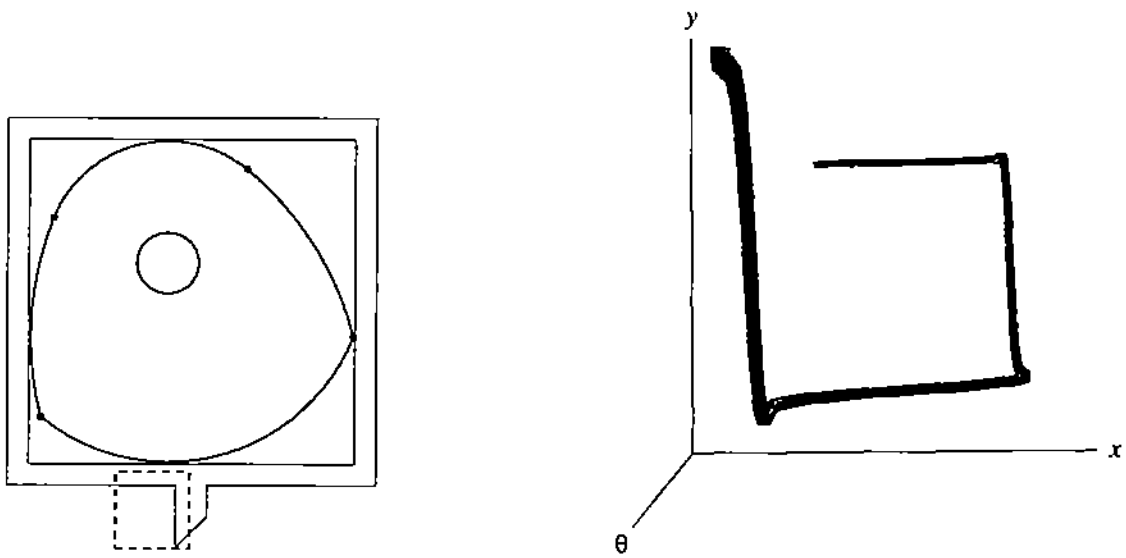


Figure 3: Cam mechanism and its configuration space. The dots on the cam profile delimit its four constituent arc segments.

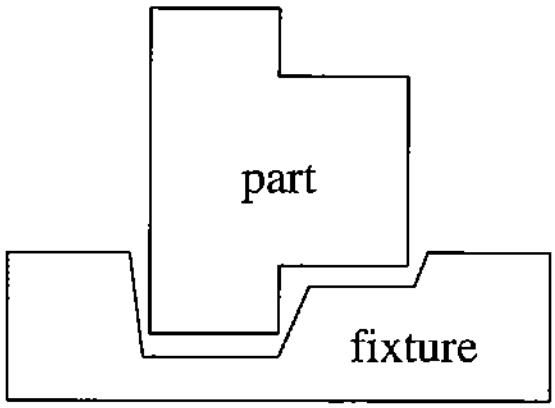


Figure 4: Part partially engaged in a fixture.

The configuration space reveals the qualitative and quantitative function of the cam mechanism to the designer, thus supporting functional design and tolerancing [17, 18]. The free space consists of four almost linear bands. Starting from $\theta = -\pi$, the first and third bands are parallel to the x axis, while the second and fourth are parallel to the y axis. The cam pushes the follower left in the first band, down in the second, right in the third, and up in the fourth. The fact that the channel spans the range of angles $(-\pi, \pi)$ indicates that the cam rotates without interference from the follower. The width of the channel bounds the relative motion of the follower with respect to the cam, called backlash, which can cause part wear, vibration, and failure. The channel width, hence the backlash, is zero in an ideal mechanism where the cam breadth equals the length of the inner sides of the follower. The ideal design fails if the follower is slightly too large for the cam due to manufacturing imprecision. The actual design accepts some backlash in order to guarantee correct function. The designer computes the function by dynamical simulation based on the contact relations encoded in the configuration space.

2.3 Fixture design

Figure 4 shows a fixture adapted from Caine [9]. The purpose of the fixture is to hold the part in a specific configuration for transport, machining, and assembly. It must hold the part stably, must provide access to the part by tools or other parts, and must easily engage and disengage the part. Brost [8] uses configuration spaces to support the design of polygonal fixture/part pairs that meet these goals. The goals translate into constraints on the contact space geometry that express part support, stability, frictional sticking, and accessibility. Our algorithm extends Brost's method to curved parts and fixtures.

Figure 5 shows the configuration space of the fixture pair and the slice $\theta = 0$ where the part engages in the fixture. The engaging configuration $(x = 2.1, y = -0.7, \theta = 0)$ is a local

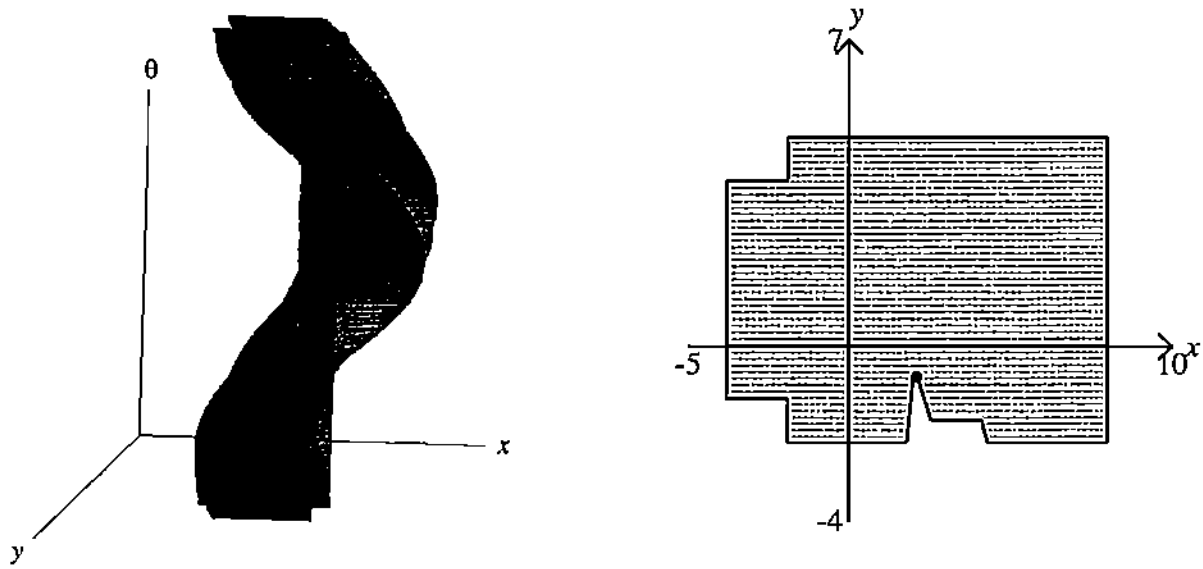


Figure 5: Fixture configuration space and $\theta = 0$ slice.

maximum of the slice contact space (marked by a dot) and of the three-dimensional contact space. The well around the maximum consists of configurations where the part partially engages. A similar well appears at $\theta = \pi$ where the part engages upside down. The rest of the contact space represents non-functional contacts between the part and the outer surfaces of the fixture.

2.4 Fastener design

Fastener design is another important application of configuration spaces [12]. Figure 6 shows a simple example of a moving pin and a fixed fastener. The user rotates the pin to the vertical position, translates it vertically into the fastener slot, and rotates it to the horizontal position. The free space consists of a single component in slices where the pin is near vertical and of two components otherwise. The components are tangent in the critical slices that separate the two cases. The critical slices are the key to a correct design because they model the transitions between the locked and open states.

Figure 7 shows the three-dimensional configuration space. The contact space consists of two roughly rectangular contours connected by narrow channels. The outer and inner contours represent contacts between the pin and the outside and the inside of the fastener. The channels, which occur near $\theta = 0$ and $\theta = \pi$, represent contacts between the pin and the slot. The channels begin and end at θ values where the pin simultaneously touches both sides of the slot.

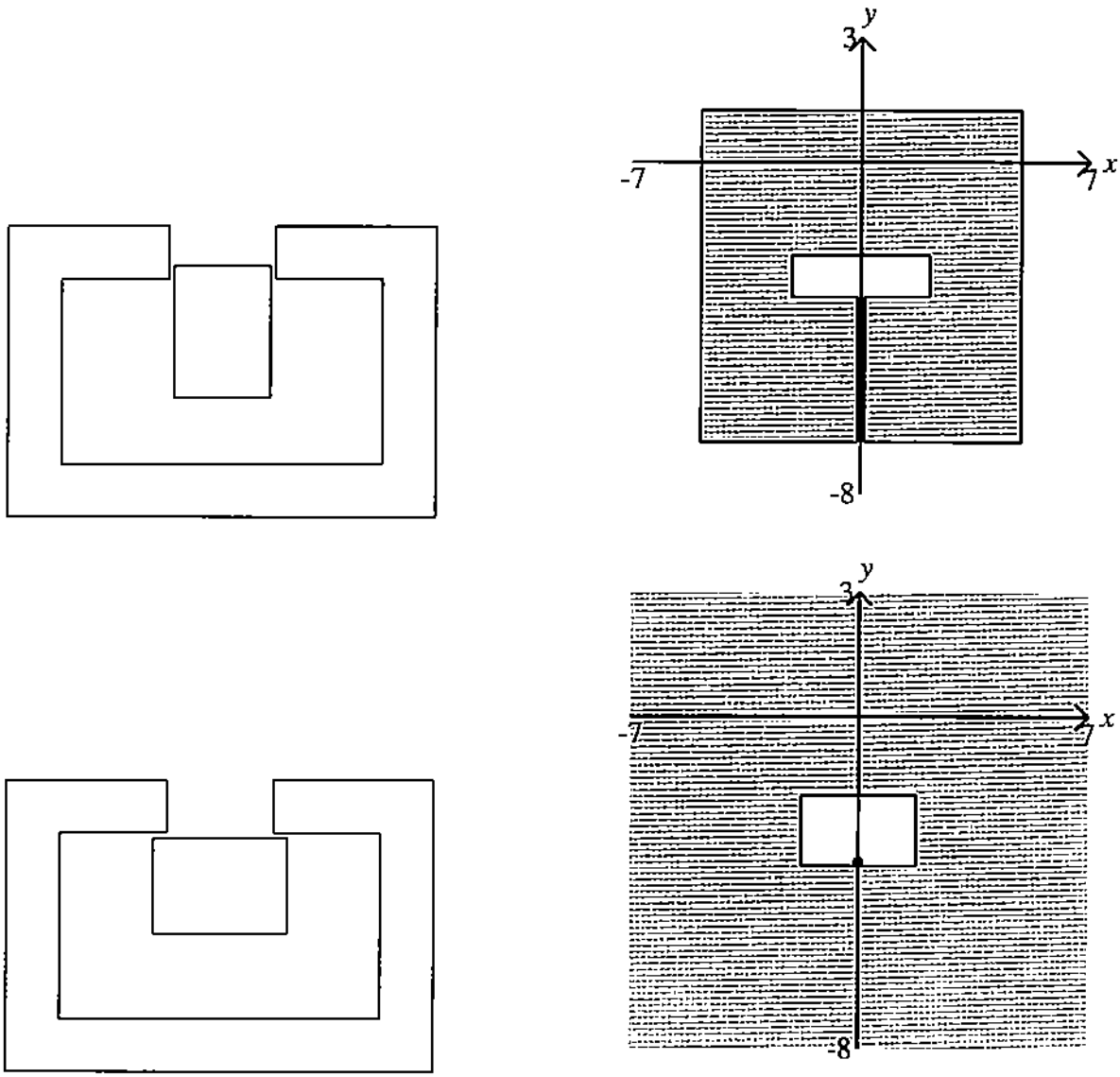


Figure 6: Fastener pair vertical and horizontal configuration space slices.

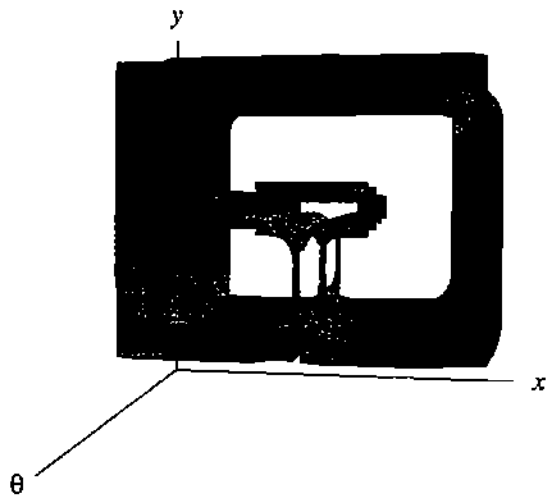


Figure 7: Fastener configuration space.

2.5 Reconstruction

We construct boundary representations of sliced configuration spaces with an algorithm that we developed in previous work [5]. We render the boundary representation to obtain the images shown in Figures 2, 3, 5, and 7. We can selectively construct the portion of the boundary within a bounding box or between two slices. Partial reconstruction is the key to efficient implementations of the manipulation tasks described in the conclusion. The inputs are the slices and the correspondence between the segments in adjacent slices. The output is a triangulated contact patch built by fitting triangles between adjacent slices and sometimes within slices. We can fit smooth patches to the triangles in a variety of ways [4].

The algorithm provides general solutions to the inherent reconstruction problems of contour tiling, correspondence, and contour branching given only the slices [25]. Tiling means triangulating the strip between two adjacent slices. The results must be topologically correct and can also optimize metric properties such as surface area and enclosed volume. Correspondence means matching contours in adjacent slices. Contour branching means matching several contours in one slice to a single contour in an adjacent slice. We specialize the algorithm to exploit the fact that the segment correspondence is known, in contrast to image reconstruction where missing critical slices create ambiguous topologies. The correspondence and contour branching problem vanish, tiling is simpler, and the algorithm is much faster.

	knee	mechanism	fixture	fastener
features	22	16	36	32
criticalities	125	64	155	118
time	4	1	1	0
slices	165	133	156	133
time	27	6	7	4
total time	31	7	8	4

Figure 8: Runtime statistics for the applications (times in seconds).

2.6 Run times

Figure 8 summarizes the runtime statistics on the four examples. The program is written in Allegro Common Lisp. The running times are on an SGI Indigo 2 workstation with 64MB of main memory and a 250 Mhz processor. The *features* entry lists the total number of line segments, circular arcs, and endpoints on the object and obstacle boundaries. The endpoints are counted because they generate their own contacts and criticalities. The times are in seconds averaged over three runs.

3 Previous work

The configuration space approach to robot motion planning originates in the work of Lozano-Pérez [22]. It has spawned a large literature that is surveyed by Latombe [20]. We discuss only the portions relevant to our work.

Several boundary-based configuration space computation methods have been developed for bodies bounded by algebraic curve segments [20]. The task is formulated algebraically. The condition that the moving object touch an obstacle without overlap yields multivariate polynomial inequalities in the configuration space coordinates. The equations are solved by algebraic methods, such as cylindrical decomposition or Gröbner basis calculation. The solution set is the contact space. None of the algorithms has been implemented, perhaps because of their intricacy or because of the slowness of the underlying algebraic solvers.

Several researchers have implemented boundary-based algorithms for polygonal bodies. The contact space consists of ruled surface patches generated by contacts between a moving vertex and an obstacle edge or between a moving edge and an obstacle vertex. Avnaim et al. [2] compute the contact patches by tracing their generating line segments through the range of orientations over which the patches are defined. They compute the singular orientations where changes occur in the number of segments or in the expressions that define their endpoints. They link adjacent patches to obtain the boundary topology. Brost [8] presents a similar algorithm that produces correct configuration spaces on 1599 out of 1600 challenging test cases. He computes contact patches by tracing the boundary curve segments

where vertex/vertex contacts occur. He intersects the patches and analyzes the arrangement of intersection edges to compute the contact space. Caine [9] computes contact patches at interactive speeds as part of an interactive design algorithm. He does not compute the patch intersections, so he cannot tell which contacts are adjacent or subsumed. Donald and Pai [12] compute paths through configuration space by quasi-static simulation.

The algorithms do not readily extend to curved bodies because they rely on the special structure of polygonal contact patches, which are simple, ruled surface patches. The boundary computations are complicated and subject to topological errors due to small numerical errors. The algorithms cannot detect or correct the errors. One error corrupts the entire boundary representation, often causing the algorithm to fail dramatically. In a personal communication, Brost reports that these problems are common in his applications.

Lozano-Pérez [23] suggests configuration space slicing for robot motion planning. Erdmann [13] develops a slicing algorithm for configuration spaces and non-directional backprojections, which have the same geometry. The algorithm handles polygonal bodies. It computes slices at fixed intervals and interpolates linearly between them. It does not compute the configuration space topology or bound the interpolation error. The algorithm is unreliable when it interpolates through critical slices because the interpolated curves can vanish or become singular.

Donald [11] and Briggs [7] compute the critical slices of non-directional backprojections and use them to derive global properties relevant to path planning under motion uncertainty. The criticality conditions and the slice construction algorithms rely on the special structure of polygonal contact. We extend the approach to curved bodies by deriving general criticality conditions and general algorithms for computing criticalities and slices.

4 Slice representation

We model configuration space as a three-dimensional Euclidean space whose coordinates are the x position, y position, and orientation θ of the moving object reference frame with respect to a global frame. The orientation is modeled as an interval $[-\pi, \pi]$, rather than as a circle. In manifold terminology, we work in a fixed chart. We can recover the correct topology by identifying the $-\pi$ and π slices. We must parameterize the orientations by $\tan\frac{\theta}{2}$ to obtain the semi-algebraic representations that we use below, but the actual computations are more convenient with θ. We suppress this distinction from here on.

Free space is a three-dimensional semi-algebraic set in configuration space. Contact space is a two-dimensional semi-algebraic set comprised of realizable contact patches. Each realizable patch is a subset of a larger patch, called a contact patch, that consists of the contact configurations for a single feature pair, regardless of whether other features overlap. The realizable patches of a feature pair contain the subset of its contact patches that lies outside blocked space.

The algebraic equations that define a contact patch are

$$R_\theta[m(u)] + O = f(v) \quad (1)$$
$$R_\theta[m'(u)] \times f'(v) = 0 \quad (2)$$

with $O = (x, y)$ the reference point of the moving object reference frame, θ the orientation of the moving object, R_θ the rotation operator, $m(u)$ the moving feature in body coordinates, $f(v)$ the fixed feature in world coordinates, and the primed quantities derivatives with respect to u and v. The first equation (two scalar equations) states that the features intersect and the second equation states that their tangents are parallel at the intersection point. The solution set is a two-dimensional algebraic set that we intersect with the parameter limits $0 \leq u, v \leq 1$ and project into configuration space to obtain the contact patch. We define the patch boundary as the set of configurations in which either parameter equals one of its limits. It is a one-dimensional semi-algebraic set comprised of curve segments along which an endpoint of one feature touches another feature.

We represent configuration space as a sequence of slices along the orientation axis. Each slice consists of the free, blocked, and contact configurations at a single orientation. The free configurations form open regions bounded by realizable contact segments along which the slice intersects the contact space. Each realizable segment is a subset a larger segment, called a contact segment, formed by the intersection between the slice and a contact patch. The realizable segments of a feature pair contain the subset of its contact segments that lies outside blocked space.

We represent the contact spaces as graphs whose nodes represent realizable contact segments and whose edges represent segment intersection points. We call two nodes equivalent when their segments arise from the same pair of object/obstacle features. We call two slices equivalent when their graphs are isomorphic and the isomorphism preserves node equivalence. We define the distance between two slices as the distance between their orientations. We call a slice regular if it has a neighborhood of equivalent slices, and critical otherwise.

5 Criticality theorem

Our main theoretical result is a necessary condition for slice criticality in the form of polynomial equations whose roots include all critical orientations. We first develop sufficient conditions for a slice to be regular. The converse of these conditions are the necessary conditions for criticality.

Sufficient conditions for a slice to be regular are that (1) it intersects every contact patch transversely, (2) it intersects every contact patch boundary transversely, and (3) its contact segments intersect transversely (Figure 9). Transversal intersection means that the tangent spaces of the intersecting sets span the ambient space at every intersection point, which implies that the intersections persist under small perturbations [15]. The theorem applies

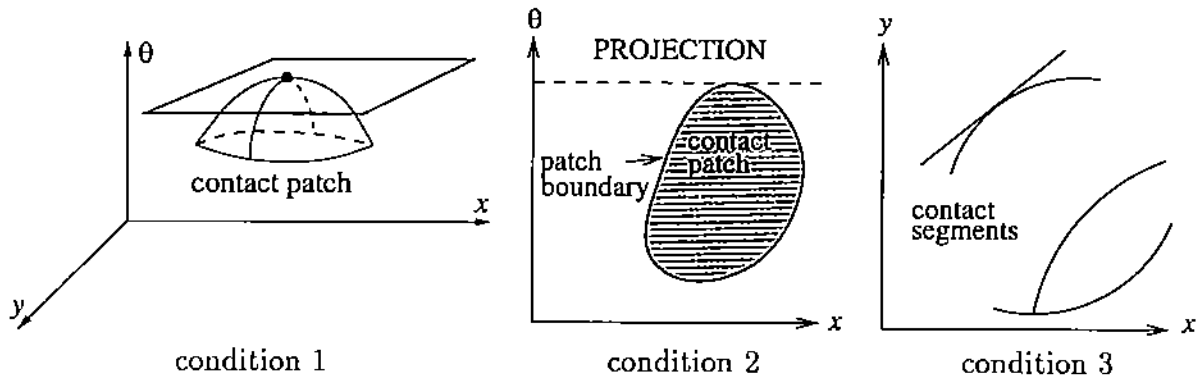


Figure 9: Critical slices for the three regularity conditions.

to differentiable manifolds, which have a tangent space everywhere. We extend it to semi-algebraic sets by requiring that both sets be regular at each intersection point, which implies that they are manifolds in a neighborhood of the intersection. Condition 1 means that the patch normal never vanishes (manifold) and is nowhere orthogonal to the slice (transversal). Condition 2 means that the slice contains no endpoints of contact patch boundary segments (manifold) and no segments are tangent to the slice (transversal). Condition 3 means that no contact segments meet at an endpoint (manifold), that three segments never meet at a point (manifold), and that no segments intersect tangentially (transversal). A triple intersection means that the endpoint of a realizable contact segment lies on another segment.

The sufficiency proof is a direct application of the transversality theorem. Condition 1 implies that the contact segments in the slice belong to smooth families of segments in nearby slices. Each realizable segment in the slice contact space is a portion of one of these segments bounded by contact segment endpoints or intersection points. Conditions 2 and 3 imply that these boundary points belong to smooth, disjoint families of curves in nearby slices. Hence, the segments belong to smooth families of segments, which means that all nearby contact graphs have equivalent vertex sets. Condition 3 also implies that the intersecting segments intersect in nearby slices, whereas the non-intersecting segments (a form of transversal intersection) do not intersect in nearby slices. This means that nearby contact graphs have isomorphic edges, hence are equivalent.

We formulate the criticality equations by expressing algebraically the converses of the regularity conditions. Figure 10 lists the criticality equations. We develop each in turn.

Condition 1 holds when the contact patch is singular or is tangent to the xy plane. We derive the criticality equations from contact equation (2), which we abbreviate as $g(\theta, u, v) = 0$. The equation defines an algebraic surface in the (θ, u, v) space. The surface is singular at points where all the partials of g equal zero. These are also the singular points of the contact patch because Equation (1) defines a regular function from (θ, u, v) to (x, y). The patch is tangent to the xy plane at regular points where $g_u = 0$ and $g_v = 0$. We can express θ in

condition	variables	equations
1	θ, u, v	$g(\theta,u,v)$; $g_u(\theta,u,v)=0$; $g_v(\theta,u,v)=0$
2 endpoint	θ, u, v	$g(\theta,u,v)=0$; $b(u)$; $b(v)$
2 tangent	θ, u, v	$g(\theta,u,v)=0$; $b(u)$ and $g_v=0$ or $b(v)$ and $g_u=0$
3 endpoint	θ, u, v, s, t	$g(\theta,u,v)=0$; $h(\theta,s,t)=0$; $p(\theta,u,v)=q(\theta,s,t)$; $b(u)$, $b(v)$, $b(s)$, or $b(t)$
3 triple	$x, y, \theta, u, v, s, t, q, r$	three sets of contact equations (1–2)
3 tangent	θ, u, v, s, t	$g(\theta,u,v)=0$; $h(\theta,s,t)=0$; $p(\theta,u,v)=q(\theta,s,t)$; Equation (4)

Figure 10: Criticality equations.

terms of u and v by the implicit function theorem and substitute into Equation (1) to obtain a parametric representation of the contact patch in terms of u and v. We derive $\frac{\partial}{\partial u}\theta = 0$ at this point from the equation

$$\frac{d}{du}g(\theta,u,v) = g_\theta \frac{\partial}{\partial u}\theta + g_u = 0 \tag{3}$$

using $g_u = 0$, and compute $\frac{\partial}{\partial v}\theta = 0$ analogously. We conclude that the θ component of the tangent vector is zero in both parameter directions, hence the patch is tangent to the xy plane. Conversely if g_u or g_v is nonzero, we eliminate u or v and obtain a parametric representation of the contact patch in which θ is one of the two parameters. The contact patch is not tangent to the xy plane because $\frac{\partial}{\partial\theta}\theta = 1$. The tangency equations subsume the singularity equations, so we need not solve them separately.

Condition 2 holds when a contact patch boundary segment is singular, when a segment is tangent to the xy plane, or when two segments intersect. The segment equations are the algebraic patch equations along with one of the boundary conditions $b(u)$, meaning that $u = 0$ or $u = 1$, or $b(v)$, meaning that $v = 0$ or $v = 1$. The two disjunctions yield four sets of equations. The equations for singularity and tangency are analogous to the patch ones, but simpler because one of the parameters is replaced by 0 or 1. Tangency again subsumes singularity. Two segments intersect when the contact equation holds along with two boundary conditions.

Condition 3 holds when two contact segments meet at an endpoint, when three segments meet, or when two segments intersect tangentially. We write Equation (1) as $O = p(\theta,u,v)$ and Equation (2) as $g(\theta,u,v) = 0$ for the first contact. We write $O = q(\theta,s,t)$ and $h(\theta,s,t) = 0$ for the second contact. The equations $g = 0$ and $h = 0$ define contact segments at each nonsingular θ value. The singular case is covered by condition 1. The endpoint equations state that both contacts occur in the same configuration and that one contact is at an endpoint. The triple intersection equations state that three contacts occur. The tangency equations are the same as the endpoint equations, except that the last equation is replaced

by

$$\begin{aligned}(p_u\dot{u} + p_v\dot{v}) \times (q_s\dot{s} + q_t\dot{t}) &= 0 \\ g_u\dot{u} + g_v\dot{v} &= 0 \\ \dot{u}^2 + \dot{v}^2 &= 1 \\ h_s\dot{s} + h_t\dot{t} &= 0 \\ \dot{s}^2 + \dot{t}^2 &= 1.\end{aligned} \tag{4}$$

The first equation states that the contact segments are tangent. The tangents are computed by the chain rule from the unit tangent $(\dot{u}, \dot{v})$ to the curve $g = 0$ and from the unit tangent $(\dot{s}, \dot{t})$ to $h = 0$. The last four equations define these tangents.

6 Computing critical orientations

We compute the critical orientations by solving the criticality equations symbolically or numerically. We can obtain extraneous roots because the regularity conditions are sufficient, but not necessary. The extraneous roots cause no problems; they just add a few slices to the output. Before turning to the general case, we develop specialized solutions to the criticality equations for important classes of problems.

6.1 Polygons

We first consider polygonal bodies. Polygons are a good model for applications where smoothness is unimportant because they yield closed-form solutions for the critical orientations and the contact segments. We need only consider contacts between vertices and line segments because they subsume contacts between pairs of line segments. Figure 11 shows a contact between a vertex r and a line segment lm. Contact equation (1) is

$$R_\theta r + O = um + (1 - u)l \tag{5}$$

for a moving vertex and a fixed segment, and is

$$R_\theta[um + (1 - u)l] + O = r \tag{6}$$

for a moving segment and a fixed vertex. Equation (2) holds identically because the vertex is independent of its parameter v. The solution sets for both contact equations are ruled surface patches whose boundary segments represent vertex/vertex contacts.

Direct calculation shows that criticality conditions 1 and 2 never hold. Endpoint condition 3 subsumes the tangent condition because of the ruled form of the contact patches. The contact segments are line segments, so segments that intersect tangentially must intersect at

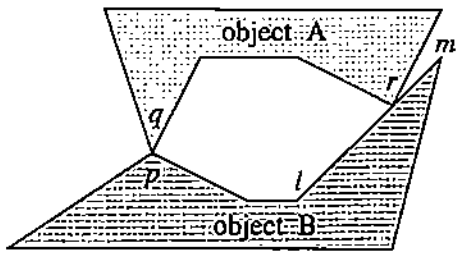


Figure 11: Critical contact configuration.

an endpoint. This occurs when a vertex touches a line segment and a second vertex touches a third vertex (Figure 11). We formulate four equations in x, y, θ, and u for each choice of these features. The first two are the contact Equations (5) or (6) and the last two are

$$R_\theta p + O = q \tag{7}$$

with p the part coordinates of the moving vertex and q the global coordinates of the fixed vertex. We subtract the last two from the first two to obtain

$$R_\theta(p - r) = q - um - (1 - u)l \text{ or} \tag{8}$$

$$R_\theta(p - um - (1 - u)l) = q - r, \tag{9}$$

which is free of x and y. Equating the norm of the argument to R_θ and the norm of the right-hand side yields quadratic equations in u,

$$||l - m||^2u^2 + 2(l - m) \cdot (q - l)u + ||q - l||^2 - ||p - r||^2 = 0 \text{ or} \tag{10}$$

$$||l - m||^2u^2 + 2(l - m) \cdot (p - l)u + ||p - l||^2 - ||q - r||^2 = 0. \tag{11}$$

We solve for u, substitute the roots into the previous equation, solve for θ, substitute into equations (5) or (6) and solve for x and y.

6.2 Circular arcs

We generalize the treatment of polygons to objects bounded by line segments and circular arcs. We obtain the contact segments and the critical values in closed form, although the computations are more complicated than for polygons. The closed-form solutions are much faster and more robust than the general numerical solution methods below. These advantages are important in applications with many curved feature contacts, which are the norm in mechanical design. We found that many mechanisms can be modeled with lines and arcs based on an extensive survey [16].

We need to analyze three types of contacts: moving arc/fixed line, moving line/fixed arc, and moving arc/fixed arc (Figure 12). We can treat vertices as circles of radius zero and

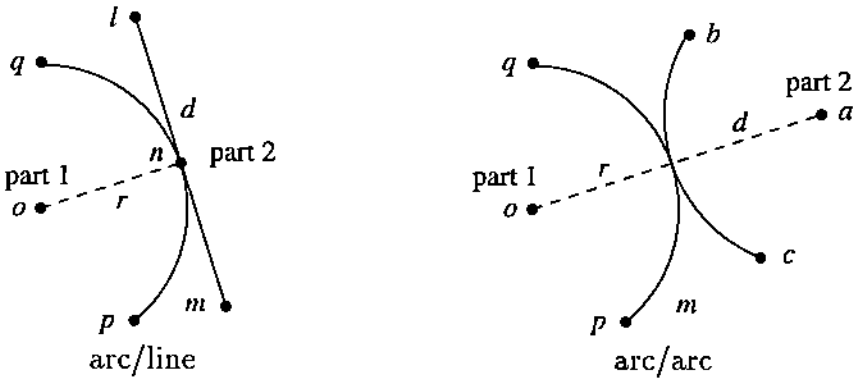


Figure 12: Contacts between line segments and circular arcs.

type	moving feature	fixed feature	contact equation	segment type
line/arc	$(1-u)l + um$	$o + r(\cos v, \sin v)$	$v - \theta - \alpha = n\pi$	line segment
arc/line	$o + r(\cos u, \sin u)$	$(1-v)l + vm$	$u + \theta - \alpha = n\pi$	line segment
arc/arc	$o + r(\cos u, \sin u)$	$a + d(\cos v, \sin v)$	$u - v + \theta = n\pi$	circular arc

Figure 13: Contact equations for line segments and circular arcs.

can continue to ignore contacts between line segments. Figure 13 lists the contact equations, which we derive from Equation (2) by trigonometric manipulation, with α the angle between the line and the local y axis and with n an integer. The simple form of the equations allows us to solve the criticality equations in closed form. Condition 1 never holds. Condition 2 holds when an arc endpoint touches a line or another arc endpoint. Condition 3 holds when two line/arc segments meet at an endpoint, a line/arc and an arc/arc meet at an endpoint, or two arc/arcs are tangent. We omit the equations, which would fill several pages.

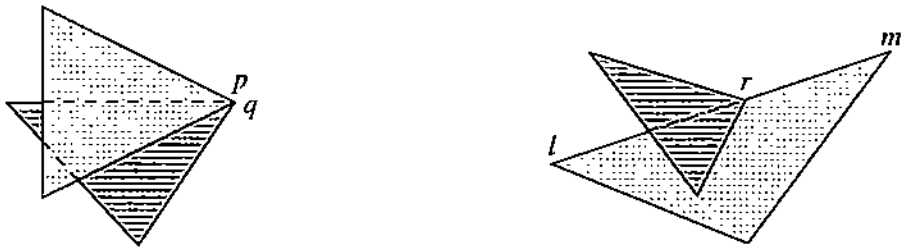
6.3 Conics

We develop a specialized method for computing the critical slices of objects with conic boundaries. A conic is a parametric curve whose coordinates are given by ratios of quadratic functions in the parameter. Conics are flexible enough to model C^1 smooth objects with an extra degree of freedom for local shape control. We reformulate contact equation (2) as

$$\frac{x'_f}{y'_f} = \frac{x'_m \cos\theta - y'_m \sin\theta}{x'_m \sin\theta + y'_m \cos\theta} \tag{12}$$

and substitute the conics $x_m = a/c$, $y_m = b/c$, $x_f = d/f$, and $y_f = e/f$ with a, b, c quadratic in u and d, e, f quadratic in v. We obtain

$$\frac{p_1}{p_2} = \frac{q_1 \cos\theta - q_2 \sin\theta}{q_1 \sin\theta + q_2 \cos\theta} \tag{13}$$



(a) (b)

Figure 14: Criticality filtering: (a) vertex/vertex interference; (b) vertex/segment interference.

with $p_1 = d'f - df'$ and $p_2 = e'f - ef'$ quadratic in u and $q_1 = a'c - ac'$ and $q_2 = b'c - bc'$ quadratic in v. This yields a quadratic equation in u whose coefficients are quadratic in v. We solve the equation symbolically to obtain u as a function of v and substitute into equation (1) to obtain the contact segments. We solve the first two criticality equations in closed form and the third numerically.

6.4 Rational parametric curves

We can solve the criticality equations for general rational parametric curves by resultants or by homotopy. Most of the computation will probably be wasted because we only need real critical points that are not in blocked space. Alternately, we can compute the critical slices by binary search along the θ axis. We generate two slices and recurse if they are not equivalent. We can replace the equivalence test with Sturm sequence tests for the criticality equations, which are slower but require fewer bisections. We have not implemented these algorithms.

6.5 Local overlap filtering

We significantly reduce the number of criticalities by deleting ones that cannot occur because they imply overlap between the immediate neighbors of the critical features, which is physically impossible (Figure 14). Contact between vertices p and q causes overlap if the interior angles between the adjacent segment tangents overlap. Contact between vertex r and segment lm causes overlap if the outward segment normal lies outside the sector bounded by the normals to the features adjacent to r. These overlap criteria are special cases of the general rule that criticalities cannot lie in blocked space. We could implement the general rule by constructing the configuration space of every critical slice and computing blocked space membership, but the overhead appears excessive.

7 Computing slice configuration spaces

After computing the critical θ values, we construct the (x, y) configuration spaces of the critical slices and of enough intermediate slices to guarantee accuracy to within the input tolerance. We use the HIPAIR program of Sacks and Joskowicz [26]. HIPAIR handles objects whose boundaries consist of line segments and circular arcs. It computes contact segments for every pair of interacting features by instantiating precomputed solutions to the contact equations. The totality of contact segments partitions the configuration space into the connected components of the free and blocked spaces. HIPAIR computes the partition by a line sweep and retrieves the free components by depth first search. It has been tested on 1,000 parametric variations of 50 higher pairs with up to 10,000 contacts.

We can easily extend HIPAIR to conics because we have solved their contact equations in closed form. We can extend it to rational parametric curves by solving Equations (1–2) by resultants or by tracing. Resultants are robust, but slow. Tracing is robust and fast given a point on each curve component [6]. These points can be computed with resultants, by random sampling, or by homotopy. The homotopy method finds points on compact components by solving the equations along with their partials with respect to one parameter. It finds points on unbounded components by intersecting with the plane at infinity or with the bounding box of parameter values. We can represent the traced segment by line segments, by conic splines, by other splines, or by recovering its implicit equations from the sampled points.

We compute each configuration space slice independently of the other slices. We can reduce the running time with an incremental approach suggested by Donald [11]. We compute an initial contact graph at $\theta = -\pi$, update the vertex and edge data (curve segments and intersection points) at each regular slice, and update the graph structure and data at each critical slice. We eliminate the cost of analyzing all the feature contacts at each slice. We estimate a 25% savings based on the HIPAIR data. The incremental approach is less robust than the independent approach because an error in one slice can corrupt the entire configuration space and because of the non-generic criticalities discussed below.

8 Computational complexity

The computational complexity of criticality computation depends on the geometric and algebraic complexity of the moving object and the obstacles, that is on the number and maximal degree of the boundary segments. The worst case complexity is the product of the number of criticality conditions and the time to solve one set of criticality equations. In the worst case, there are $O(n^2)$ object/obstacle feature pairs for n boundary segments, which yields $O(n^2)$ criticality conditions 1 and 2 and $O(n^6)$ criticality conditions 3. The time to solve one set of equations in closed form is exponential in their degree. It is a small constant for line segments and circular arcs, as seen in the applications run times, and is a moderate

constant for conics.

The computational complexity of computing one slice configuration space is $O(n^2)$ when the features have bounded geometric complexity, which is the case we study. Several simple algorithms achieve this bound, but with unacceptable constant factors. Our HIPAIR program reduces the constant factors enough to analyze pairs with 10,000 features in one second [26]. The complexity of computing all the slices is $O(n^8)$ because there are $O(n^6)$ slices due to the bounded feature complexity. General features are best analyzed by numerical methods for which discrete complexity analysis is inappropriate.

We have found that criticality filtering reduces the number of contacts to $O(n)$, hence the number of criticality conditions to $O(n^3)$, the slice computation time to $O(n \log n)$, and the overall complexity to $O(n^4 \log n)$. These findings are based on extensive testing on hundreds of pairs [26] and are consistent with the run times in the applications section.

9 Robustness

Our sliced configuration space computation algorithm handles most non-generic situations correctly and is insensitive to numerical errors in its input or in its intermediate computations. The criticality equations are completely general. We assume that the equations have isolated solutions. This situation is generic, meaning that it is true of almost all equations and is achieved by almost all small perturbations in the constant terms of the remaining equations. We can compute the criticalities of line segments and circular arcs without additional assumptions because we use symbolic methods. The numerical methods that we propose for other rational parametric features work best when the solutions are nonsingular, which is also generic.

Designed artifacts may embody constraints that invalidate genericity by restricting the space of equations and perturbations. We see no such problem with isolated, nonsingular solutions, but problems may arise when several criticalities occur in a slice. Although non-generic, multiple criticalities are common in practice due to symmetries, parallelism, and the like. They appear as distinct, closely spaced singularities due to input imprecision compounded by numerical errors. The criticality and slice algorithms make no assumptions about multiple criticalities. The incremental slice computation algorithm and the reconstruction algorithm need the correct criticality ordering. A simple interval grouping works on the examples we have studied, but we may need to devise better heuristics for more complex shapes.

Designed artifacts may also invalidate the generic assumption that the contact space consists solely of surface patches, without isolated points or dangling curves. These artifacts occur when the object mates perfectly with an obstacle, for example a round pin in a round hole of the same radius. Criticality computation does not depend on this assumption, but HIPAIR does. Extending HIPAIR to isolated points and dangling curves amounts to adding

special cases to the line sweep that computes the connected components of planar partitions. We would treat them specially in reconstruction and in applications. We expect these cases to prove irrelevant in applications because the only real parts that fit perfectly are permanently connected, hence can be modeled as standard joint constraints.

10 Conclusion

We have presented a general algorithm for computing the configuration space of a planar object moving amidst planar obstacles. We have implemented the algorithm for objects bounded by line segments and circular arcs, which cover many applications, and have described extensions to conics and to rational parametric curves. The novelty of the algorithm lies in the general algebraic method of computing critical slices for curved bodies. We expect that the slice representation will prove to be a useful abstract data type for computational geometry, since many problems are far simpler in the plane than in three dimensions.

We conclude with a brief discussion of how to automate configuration space manipulation in support of robotics, mechanical design, and joint modeling. The examples in the applications section involve several operations on configuration space, all of which we performed manually. We plan to implement these operations via abstract operators on sliced configuration spaces. The critical slices are crucial for every operation.

Each of the examples requires dynamical simulation to compute the motion of parts due to external forces and contact constraints. The hardest computational problem, called contact analysis, is to determine the touching parts and the ensuing contact forces at each time step. Current simulators perform contact analysis at each time step. They try to avoid the quadratic worst-case performance with collision detection heuristics, such as spatial partitioning, which avoids comparisons between distant parts, and coherent computation, which predicts current contacts based on past [21]. It is unclear how well the heuristics work in the mechanical domain where most parts interact, contact changes are common, clearances are small, and parts are driven fast. The algorithms approximate curved boundary parts with polyhedra, which creates spurious discontinuities in the contact functions that distort the dynamics of high-speed systems and increase the simulation time.

We are developing a simulator that replaces collision detection with configuration space computation. The program computes the configuration spaces of all pairs of parts before the simulation. It handles contacts between parts with curved boundaries without approximation. The worst-case running time of the computation is quadratic in the geometric complexity of the parts, as is a single collision detection. At each simulation step, the algorithm obtains the necessary contact data by querying the pairwise configuration spaces. Each query takes linear time in the number of parts and is independent of their geometric complexity. We have implemented this algorithm for two-dimensional configuration spaces [28] and achieved twice real-time performance. Matching this performance in three dimen-

sions requires novel geometric algorithms for locating a point in the configuration space partition and for intersecting a parametric curve with the free space boundary.

Fixture, fastener, prosthesis, and mechanism design involve parametric design. The input is a parametric model in which the shapes and the positions of the parts are specified in terms of symbolic parameters. The designer searches the space of allowable parameter values for points that realize or optimize behaviors. If the designer can specify the objective as a smooth function of the parameters, he may be able to achieve it by nonlinear optimization. This is impossible if the design involves contact changes, if the objective is to achieve a behavior, or if the objective is qualitative. The traditional approach to these problems is direct search of the parameter space. This is often impractical, especially when there are more than three parameters. The designer must examine many points to assure that he has not overlooked a good design. Each point requires a time consuming analysis. The search is even harder when the behavior is sensitive to small perturbations in the parameter values.

We aim to reduce search by interactively inverting the mapping from parameter values to configuration spaces. The goal is to allow the designer to modify an assembly configuration space interactively while the design program updates the assembly to realize the changing kinematics. We have implemented a naive linearization algorithm for inverting the nonlinear, many to one mapping from assembly to configuration space and have tested it on a few planar, two degree of freedom pairs [17]. The program uses simple heuristics and mouse input to traverse the space of solution parameter values. Caine [9, 10] takes a similar approach to polygonal feeder design. We need an efficient, robust algorithm that handles curved parts with three degrees of freedom and larger assemblies.

The knee example requires configuration space composition to model the three way interactions among the femur, tibia, and patella (knee cap). We can embed the tibia/femur, tibia/patella, and femur/patella configuration spaces in the nine-dimensional joint configuration space then intersect the pairwise contact spaces to obtain the joint contact space. We [16] found this approach highly effective for pairs with two degrees of freedom. It may prove impractical for pairs with three degrees of freedom because of the large number of regions in the pairwise spaces, which is worst-case exponential in the number of parts with large constant factors [20]. We can fall back on dynamical simulation, for which pairwise configuration spaces suffice, coupled with local computation of the assembly configuration space in key regions.

Acknowledgments

Leo Joskowicz provided excellent comments that helped us significantly improve this paper. Kwun-Nan Lin developed the reconstruction program and Dan Schikore helped produce the pictures of the three-dimensional configuration spaces. Supported in part by NSF grant CCR 92-22467, NSF grant CCR-9505745, AFOSR grant F49620-94-1-0080, and ONR grant N00014-94-1-0370.

References

[1] Artobolevsky, I. *Mechanisms in Modern Engineering Design*, volume 1–4. (MIR Publishers, Moscow, 1979). English translation.

[2] Avnaim, F., Boissonnat, J. D., and Faverjon, B. A practical exact motion planning algorithm for polygonal objects amidst polygonal obstacles. in: *IEEE International Conference on Robotics and Automation*, 1988.

[3] Bajaj, C., Bernardini, F., Lin, K.-N., et al. Physical simulation of the visible human joints. in: *National Library of Medicine Visible Human Project Conference*, 1996.

[4] Bajaj, C. L. Electronic models of the human anatomy. in: *Curves and Surfaces in Computer Vision and Graphics 2: Proceedings of the Symposium on Electronic Imaging Science and Technology*, volume 1610, pages 230–237, Boston, MA, 1991.

[5] Bajaj, C. L., Coyle, E., and Lin, K.-N. Arbitrary topology shape reconstruction from planar cross sections. *Graphical Models and Image Processing* **58** (1996). to appear.

[6] Bajaj, C. L., Hoffmann, C., Hopcroft, J., et al. Tracing surface intersections. *Computer Aided Geometric Design* **5** (1988) 285–307.

[7] Briggs, A. An efficient algorithm for one-step compliant motion planning with uncertainty. *Algorithmica* **8** (1992) 195–208.

[8] Brost, R. C. *Analysis and planning of planar manipulation tasks.* PhD thesis, Carnegie-Mellon University, 1991. Available as Technical Report CMU-CS-91-149.

[9] Caine, M. E. The design of shape from motion contraints. AI-TR 1425, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 545 Technology Square, Cambridge, MA, 02139, 1993.

[10] Caine, M. E. The design of shape interactions using motion constraints. in: *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 366–371, 1994.

[11] Donald, B. R. The complexity of planar compliant motion planning with uncertainty. *Algorithmica* **5** (1990) 353–382.

[12] Donald, B. R. and Pai, D. The motion of planar, compliantly connected rigid bodies in contact, with applications to automatic fastening. *International Journal of Robotics Research* **12** (1993) 307–338.

[13] Erdmann, M. A. On motion planning with uncertainty. AI-TR 810, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1984.

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

[15] Guillemin, V. and Pollack, A. *Differential Topology.* (Prentice-Hall, Englewood Cliffs, NJ, 1974).

[16] Joskowicz, L. and Sacks, E. Computational kinematics. *Artificial Intelligence* **51** (1991) 381–416. reprinted in [14].

[17] Joskowicz, L. and Sacks, E. Configuration space computation for mechanism design. in: *Proceedings of the 1994 IEEE International Conference on Robotics and Automation.* IEEE Computer Society Press, 1994.

[18] Joskowicz, L., Sacks, E., and Srinivasan, V. Kinematic tolerance analysis. *Computer-Aided Design* (1996). to appear.

[19] Joskowicz, L. and Taylor, R. H. Interference-free insertion of a solid body into a cavity: An algorithm and a medical application. *International Journal of Robotics Research* **15** (1996) 211–229.

[20] Latombe, J.-C. *Robot Motion Planning.* (Kluwer Academic Publishers, 1991).

[21] Lin, M. C., Manocha, D., Cohen, J., et al. Collision detection: Algorithms and applications. in: *Proceedings of the 2nd Workshop on Algorithmic Foundations of Robotics*, 1996.

[22] Lozano-Pérez, T. Spatial planning: A configuration space approach. in: *IEEE Transactions on Computers*, volume C-32. IEEE Press, 1983.

[23] Lozano-Pérez, T. A simple motion-planning algorithm for general robot manipulators. *IEEE Journal of Robotics and Automation* **RA-3** (1987).

[24] Lozano-Pérez, T. and Wilson, R. H. Assembly sequencing for arbitrary motions. in: *IEEE Conference on Robotics and Automation*, 1993.

[25] Meyers, D., Skinner, S., and Sloan, K. Surfaces from contours. *ACM Transactions on Graphics* **11** (1992) 228–258.

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

[27] Sacks, E. and Joskowicz, L. Mechanism design and analysis using configuration spaces. in: *Proceedings of the Ninth World Congress on the Theory of Machines and Mechanisms*, 1995.

[28] Sacks, E. and Joskowicz, L. Mechanical systems simulation using configuration spaces. Technical Report 96-046, Purdue University, 1996.

[29] Wilson, R., Kavraki, Latombe, J.-C., et al. Two handed assembly sequencing. *International Journal of Robotics Research* **14** (1995) 335–350.