Polyhedral Tracings and their Convolution[*]

Julien Basch, *Stanford University, Stanford, CA, USA*
Leonidas J. Guibas, *Stanford University, Stanford, CA, USA*
G. D. Ramkumar, *Stanford University, Stanford, CA, USA*
Lyle Ramshaw, *Digital Equipment Corporation, Palo Alto, CA, USA*

1 Introduction

Over ten years ago, Guibas, Ramshaw, and Stolfi [8] introduced the *kinetic framework* for Computational Geometry in two dimensions. They augmented the standard boundary representation of planar shapes via polygons or closed curves with certain additional information, to make them into closed *tracings*. The advantage of tracings is that they admit of a multilinear calculus of operations that generalizes standard boolean operations on shapes such as union, intersection, set difference, etc. A particular accomplishment of the kinetic framework was to define the operation of *convolution* on planar tracings and to illustrate its use in a variety of algorithmic problems. Loosely speaking, the convolution $B * R$ of two tracings B (the barrier) and R (the robot) is another tracing that encodes the interaction of B with all possible translations of a copy of R mirrored through the origin. The convolution allows us to transform problems about two tracings B and R into problems about a point and a composite tracing, the convolution $B * R$. A related composition concept is familiar in robotics, under the names Minkowski sum or configuration space obstacle [15, 14]; but that concept lacks the advantageous multilinear structure of the convolution.

In two dimensions, a closed curve is normally defined via a continuous map from the unit circle to the plane. To augment the curve into a tracing, we make a value of this map be not only a point in the plane, but an associated direction as well, with the two related by a 'no-slipping' condition. Intuitively, we can think of a tracing as being drawn by a tiny car; the car can move forward, move backward, or turn while standing still, but may not skid sideways. A standard curve can be given an orientation at each point, corresponding to the sequence in which the points of the curve are traversed by the car. A tracing in addition specifies the direction the car is facing at each point, and that direction must be along the tangent to the curve whenever the latter is defined. The combination of {location, direction} is called a *state* in [8]. The direction of the car is the crucial new component that enables the definition of the convolution operation and gives it its desirable properties. The convolution is defined as a fiber product in [8], by matching states from the two factor tracings with the same direction, and then assembling those into a tracing.

Almost nothing is known about the convolution operation in dimensions higher than two, except in the special case of convex bodies, where it is equivalent to the Minkowski sum; this case for 3-D polyhedral objects has been treated in [9, 2, 12, 17]. Unfortunately the convex case is too special to suggest the proper extension of the concept of a tracing to three and higher dimensions. If, in two dimensions, one thinks of a tracing as a rule for assigning winding numbers to the elements (faces, but also edges and vertices) of the planar subdivision defined by the tracing, then we get a structure we call a *painting*. Paintings of arbitrary dimension were studied by Schapira [18] under the name of *constructible functions*. Using results from sheaf theory, Schapira was able to define a convolution operation on constructible functions and to show how it corresponds to the convolution of tracings discussed above. Nevertheless, Schapira's method does not give

[*]Published in *Algorithms for Robotic Motion and Manipulation*, Jean-Paul Laumond and Mark Overmars, eds., A. K. Peters, Wellesley, Massachusetts, 1996. All rights reserved.

a definition of a tracing in higher dimensions, and as such it is not directly useful in algorithmic problems. A tracing can be thought of as a boundary representation of a painting; because of this, it can be cheaper to represent, since it need not encode features, such as self-intersections, that are artifacts of the embedding of the domain manifold into the range space, and not features of the mapping itself.

The main contribution of this paper is to define the notion of *polyhedral tracings*, which extends the classical notion of a polyhedron in exactly the same way a polygonal tracing extends the notion of a polygon. We also define the convolution $B * R$ of two polyhedral tracings B and R and show that it has the same desirable properties as in two dimensions. The technical challenge in getting this theory to work lies in how to extend normal polyhedra into tracings and, less obviously but equally importantly, how to orient the elements of the fiber product defining the convolution so that together they form another tracing manifold. We describe a data structure for representing polyhedral tracings based on the *quad-edge* data structure of Guibas and Stolfi [10] and give an efficient algorithm for computing the convolution of such tracings. Given tracings B and R of size m and n respectively, their convolution $Q = B * R$ can have size $\Theta(mn)$ in the worst case. If the actual size of Q is k, then we can compute Q in output-sensitive time $O(k\alpha(k)\log^3 k)$. This algorithm, which requires novel data structures, is based on a new method we have for detecting red-blue intersections between two families of segments, the red and the blue; within each family the segments need not be disjoint — in fact we require that they be connected. To keep the length of this paper within reason, the details of these data structures and the associated analysis will be reported in a separate paper [3]. We are currently working on extending to arbitrary dimensions and to more general types of tracings our current results on orienting fiber products, defining tracings and convolutions, and the relationship of tracings with paintings.

We view the importance of the work presented here to robotics as follows. For a polyhedral robot translating amidst polyhedral barriers in the Euclidean three dimensional space E^3 (or for polyhedral approximations of more general objects), the standard motion-planning solution involves the calculation of the free space, i.e. the locus of all placements of the robot where it does not collide with barriers. It is well-known that the free space is also a polyhedral domain and can be computed as the complement of the Minkowski sums of the barriers with the robot reflected through the origin. Once a description of the free space is available, one can develop algorithms for checking possible collisions during proposed robot motions, answering connectivity queries (the classical motion planning problem), rendering portions of the free space for visualization purposes, etc. In this paper we propose to accomplish the same goals, but using convolutions instead of the classical Minkowski sums.

To focus the issues, let us assume there is only one barrier B that the robot R must avoid. Why should we prefer the convolution $B * R$ over the Minkowski sum $B \oplus R$ (actually we should be using $-R$ in both cases)? There are two answers. First, the convolution gives us more information. Given a proposed placement of the robot, the convolution gives us information about the topology of the collision region between the barrier and the robot (its Euler-Poincaré characteristic) — not simply a bit on whether they collide or not; this is an advantage of the multilinear over the boolean formulation of the problem. More importantly, the convolution can be combinatorially much simpler than the Minkowski sum, while still allowing us to develop efficient algorithms for the kinds of motion queries described above. If, as above, B and R have sizes m and n, then the Minkowski sum of B and R can have size $\Theta(m^3n^3)$, while the convolution is always of size at most $O(mn)$. Again, this is because the Minkowski sum has to explicitly represent features that correspond to self-intersections of the embedding of the convolution in $\mathcal{R}^3$. Thus, even when the Minkowski sum is what we really want, it is useful to think of the convolution as an *implicit representation* of it. The convolution is a structure that can be used to answer efficiently queries about the Minkowski sum, while being both smaller in size and easier to compute.

This paper is organized as follows. Section 2 introduces the general framework of tracings and paintings. Section 3 defines polyhedral tracings and introduces a data structure for representing them. Section 4 describes the convolution of two polyhedral tracings in terms of these data structures, while Section 5 sketches an algorithm to compute this convolution that runs in time nearly linear in input plus output size. Section 6 investigates in more detail the relationship between the Minkowski sum and the convolution, and Section 7 concludes.

2 Tracings and paintings

In this section, we generalize to higher dimensions the notion of *tracings* introduced in the kinetic framework, and relate that notion to Schapira's constructible functions, or *paintings*. We briefly describe his convolution operation on paintings and its relation to the Minkowski sum, and proceed to define an equivalent convolution operation on tracings. In this paper we restrict the discussion to the 3-D case: all manifolds below are 2-manifolds; S^2 denotes the 2-dimensional sphere, and E^3 denotes the oriented 3-dimensional Euclidean space. A basis of E^3 is called *direct* (resp. *indirect*) if it has positive (resp. negative) orientation. The *indicator* of a set $X \in E^3$ is the function whose value is 1 on X and 0 everywhere else.

For the definitions below to make sense, we need to place ourselves in a consistent framework. We may for instance follow Schapira, and assume that our manifolds are real analytic, that the sets are subanalytic, and that our functions are morphisms of real analytic manifolds (the focus of this paper, however, is on piecewise linear tracings).

2.1 Tracings

In the kinetic framework, a point on a curve is enriched into a state by the addition of a direction vector which can point either forward or backward along the curve. In three dimensions, we enrich a point on a surface into a state by adding a *whisker* vector at that point, which can point along either of the two directions normal to the surface.

More formally, a tracing R is given by an *oriented compact manifold without boundary* M_R, together with a pair of smooth functions: the location map $r : M_R \to E^3$, and the whisker map $\dot{r} : M_R \to S^2$. The two are related by a *no-slipping* condition:

$$\forall x \in M_R, \operatorname{Im}(dr)_x \perp \dot{r}(x)$$

where $\operatorname{Im}(dr)_x$ is the image of the differential of r at x. That is, the location map r is restricted locally to move in a direction orthogonal to $\dot{r}$.

The orientation of the manifold M_R can be thought of as a tiny rotating circle around each of its points. If the image under the location map of this circle around x is replaced by a rotating corkscrew, the direction in which this familiar object starts moving will be referred to as *locally outward* for M_R at x, while the opposite direction will be called *locally inward*.

There are two senses in which the whisker map $\dot{r}$ gives more information than the differential of the location map:

- When the tangent space $\operatorname{Im}(dr)_x$ is a plane, the whisker provides exactly one bit of information, by selecting the normal to this plane which points locally inward or outward.
- The whisker map smoothly extends the notion of tangent plane to ridges and corners (where $\operatorname{Im}(dr)_x$ is only a line or a point). Crossing a ridge causes the whisker vector to swing through an arc in S^2, while the whiskers associated with a corner fill some region of S^2.

2.2 Paintings

A painting ϕ on E^3 is a function that associates an integer to each point of E^3, in a "not too wild" fashion. The paintings we will deal with correspond to polyhedral subdivisions of E^3 with an associated constant integer value for each feature (vertex, edge, face, chamber) of the subdivision. For instance, there are two natural paintings that can be associated with a polyhedron P. The *closed painting* of P has value 1 everywhere inside the polyhedron and on its boundary, and zero elsewhere. The *open painting* of P is the same except for a value of 0 on the boundary. A painting ϕ is said to be *locally closed* at $p \in E^3$ if there exists an open neighborhood A of p on which $\max_{q\in A} \phi(q) = \phi(p)$ Likewise, we define *locally open* by replacing max with min. These definitions generalize the notions of open and closed sets to paintings. Since the overlay of spatial subdivisions is another spatial subdivision, algebraic operations (sum, product) on paintings can be defined in an obvious point-wise fashion.

Any tracing R has an associated painting ϕ, defined via the *winding number* function. We can compute the winding number at a point p of E^3 that does not lie on the range of the location map $r(M_R)$ of R as follows. Take a smooth path π from p to infinity, and count $+1$ or -1 each time the path crosses $r(M_R)$. We count the crossing as $+1$ if the 'velocity vector' of π at the crossing point is directed locally outwards (as defined by the local orientation), and -1 otherwise. The total count (the winding number) is independent of the path chosen, and thus defines a unique value except for those points that lie on $r(M_R)$. If p is such a point, one would like to perturb the location map by a tiny amount so as to move its image away from p, and proceed as before. One role of the whisker map $\dot{r}$ is to indicate the direction in which this perturbation should happen: the winding number is computed not with respect to r, but with respect to $r + \epsilon\dot{r}$, for some small enough $\epsilon > 0$. This perturbation transforms a ridge into a sector of a cylinder and a corner into a region of a small sphere centered at that corner. The winding number is thus established on every point of E^3 and defines a painting ϕ associated with R.

The distinction between open and closed sets is crucial in Schapira's theory of constructible functions. Tracings also make this distinction: it is one of their novel features as a boundary representation for solids. Consider a point $p \in E^3$ and assume, to simplify matters, that it has only one pre-image under r, say $x \in M_R$. If the whisker at p (i.e. $\dot{r}(x)$) points locally outward on the surface of r around p, the associated painting is locally closed at p, otherwise it is locally open.

2.3 Convolution

Any polyhedral painting ϕ can be triangulated, and thus it is not hard to see that it can be expressed as a finite weighted sum of indicators of closed, simply connected sets — such a decomposition, however, is not unique. Schapira defines the *integral* of ϕ, $\int \phi$, as the algebraic sum of these weights and shows that it is independent of the choice of decomposition. This integral has a familiar interpretation in certain cases. For example, if ϕ is the indicator of a closed set S, this integral is simply the Euler-Poincaré characteristic of the set, i.e., in three dimensions, the number of components, minus the number of tunnels, plus the number of internal holes of S.

The convolution of two paintings ϕ, ψ is then defined with respect to this integral $\int$, as

$$(\phi * \psi)(p) = \int_{q \in E^3} \phi(q)\psi(p-q)$$

and it relates to intersection problems in the following way [18]:

Theorem 2.1. *If ϕ, ψ are indicators of closed sets B, R, then the value of $\phi * \psi$ at a point $p \in E^3$ is the Euler-Poincaré characteristic of the intersection of B with R reflected through the origin and translated by p.*

We now wish to describe a convolution operation on tracings such that, given tracings B and R (blue and red) that define paintings ϕ and ψ, the convolution $Q = B * R$ defines the painting $\phi * \psi$. Disregarding the topology of tracings, and viewing them as bags of states as in [8], the convolution operation remains exactly the same as in two dimensions: points $x \in M_B$ and $y \in M_R$ define a state (x, y) in the convolution just when their whiskers match. In that case, the output feature keeps the same whisker and its location is $r(x) + b(y)$.

In our framework for tracings, the convolution domain is defined to be the following subset of $M_B \times M_R$:

$$M_Q = \{(x, y) \in M_B \times M_R \mid \dot{r}(x) = \dot{b}(y)\}$$

This is an instance of a general categorical construction known as a *fiber product* [13] applied to the two whisker maps $\dot{r}$ and $\dot{b}$. Under mild assumptions (*transversality*) on the pair $(\dot{r}, \dot{b})$, the fiber product M_Q is an oriented manifold of the same dimension as M_B and M_R, which makes the above define a valid tracing. The topology of the fiber product M_Q is not simply related to that of M_B and M_R; for instance, M_Q can have any number of components, even when both M_B and M_R are connected. Deriving a consistent local orientation rule for M_Q is a problem which is fully addressed in [4].

Note that when defining the winding number at a point in the image of the location map, the perturbed tracing can be viewed as the convolution of the original tracing with a ball of radius ϵ.

3 Polyhedral tracings

In this section, we define a polyhedral tracing as a specialized tracing defined in the framework of piecewise-linear geometry. We describe an efficient way to represent a polyhedral tracing using a *signed oriented quad-edge* structure, and show how an ordinary polyhedron can be made into a tracing using a sound set of conventions.

A polyhedral tracing is a tracing such that its domain manifold can be decomposed into (interior-disjoint) simply connected subsets of three types: vertex domains, edge domains, and face domains, with connectivity properties akin to a subdivision as defined in [10], and such that (Figure 1):

1. On a face domain, the whisker map is constant and the location map is a bijection whose range is a simple planar polygon,
2. On an edge domain, the location map range is a line segment (corresponding to a shared edge between the polygonal ranges of the two adjacent face domains), and the whisker map range is a great circle arc of length less than π on S^2, and
3. On a vertex domain, the location map is constant, while the whisker map is a bijection, and its range is a simple spherical polygon on S^2 entirely contained within a hemisphere [*the hemisphere assumption*].

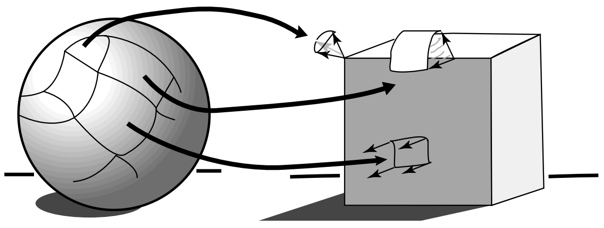

Figure 1: *A cube tracing indexed by a sphere. Note that the inverse image under the location map of points on faces, edges, and vertices of the cube are 0-dimensional, 1-dimensional, and 2-dimensional respectively.*

Using a tracing to describe a polyhedron boundary has several advantages. First, a tracing can encode both open, closed, and mixed subsets of E^3, whereas no such distinction exists for a polyhedron. Second, a tracing can represent a larger variety of paintings with piecewise planar boundaries, and not just indicators of polyhedra. Third, the convolution of two (closed) polyhedra is in general not a polyhedron. Polyhedral tracings have in fact just the right expressiveness to allow the representation of convolutions of general non-convex polyhedra.

3.1 Enriched quad-edge structure

A quad-edge structure [10] is an algebra $(V, E, F, \mathrm{Onext}, \mathrm{Org}, \mathrm{Lface}, \mathrm{Sym})$, where V (resp. E, F) is the set of vertices (resp. edges, faces), with a number of relations between the operations which we do not review here. In such a structure, an edge has a *direction*, from its *origin* (Org) to its *destination* (Dest), and an *orientation*, from its *left face* (Lface) to its *right face* (Rface). The *symmetric* (Sym) of an edge is the same edge with both its orientation and direction reversed. The *next edge* (Onext) to an edge e whose origin is a vertex v is the edge with origin v whose Rface is the Lface of e. In the case of a connected orientable quad-edge, the quad-edge algebra is the union of two disjoint subalgebras: an edge with a given direction is present in both subalgebras, but with a distinct orientation. An oriented quad-edge structure is simply the choice of one of these subalgebras for each connected component. In this case, the direction of an edge determines its orientation and vice-versa, so that a face f has a ring of *directed* edges associated with it (those whose Lface is f).

We represent the domain manifold of a polyhedral tracing by an oriented quad-edge structure (V, E, F). In this setting, a vertex and an edge in the structure also represent two-dimensional domains (vertex domains and edge domains) in the underlying manifold. Next, we encode the location and whisker maps by keeping only their values at vertices and faces respectively, via the two functions:

$$\begin{array}{rccc} \lambda: & V & \mapsto & E^3 \\ \mu: & F & \mapsto & S^2 \end{array}$$

where λ gives the location of each vertex, and μ gives the whisker at each face.

Obviously, this representation loses some information: the original domain manifold of the polyhedral tracing is known only up to a homeomorphism, but this level of precision is enough to define a painting. The location map can be smoothly reconstructed by linear interpolation from the data for vertices, to edges and faces, due to the assumption that the location map is a bijection on a face domain.

The reconstruction of the whisker map for edges and vertices from the whiskers of the faces is more delicate. Consider first the case of an edge. Its whiskers describe an arc joining the whiskers of its two adjacent faces. There are two such arcs, but only the smallest of the two satisfies the requirement that the whisker set of an edge be an arc of length less than π on S^2. For a vertex domain, the assumption that the whisker map is a bijection allows its specification using the values on its boundary, but, again, this boundary defines two polygons on the sphere. It is the hemisphere assumption that allows us to choose the smaller one without ambiguity. From this choice, the whisker map on a vertex domain is reconstructed up to an isomorphism.

3.2 Whisker and outward-pointing normal

Traditionally, when a quad-edge structure is used to represent a polyhedron boundary, a record for a vertex v stores its location $\lambda(v)$, and a record for a face f stores its outward-pointing normal $\nu(f)$. The outward-pointing normal is such that, when an observer is looking at a face f with $\nu(f)$ pointing towards her, the ring of oriented edges around f describes a counter-clockwise cycle.

Due to the no-slipping condition, the outward-pointing normal to a face is either the same as, or opposite to, the whisker at that face. Thus, it is not necessary to store it in our structure, as it is possible to reconstruct it at the additional cost of only one extra bit per face. We define

$$\sigma : F \mapsto \{-1, +1\}$$

such that for each face f, $\nu(f) = \sigma(f)\mu(f)$, and store this sign bit with each face (hence the name “signed quad-edge”).

This sign bit has an important meaning for the corresponding painting. If the sign at a face is $+1$, the whisker at that face points locally outwards, which defines a locally closed painting. If the sign is -1, the tracing defines a locally open painting. Storing this sign is redundant, as it can be reconstructed from the whisker and the orientation of the ring of edges around one face. However, this reconstruction is costly, as it requires tracing the entire ring of edges for each face.

3.3 From a polyhedron to a polyhedral tracing

In most applications, a polyhedron is defined by its boundary, represented by an oriented quad-edge structure, the location of each vertex, and the outward-pointing normal to each face. This structure can be easily converted into a polyhedral tracing, but there are two issues that require some discussion: what signs to choose for the faces, and how to enforce the hemisphere requirements for the vertex whisker maps.

The choice of signs for faces depends on whether the polyhedron is intended to represent a topologically closed or open set. While this distinction is rarely if ever considered in computational geometry, it may have some relevance in solid modeling. As we saw before, having the whisker map point locally outwards defines a locally closed painting. Thus, in order to represent a topologically closed polyhedron, one would set the sign bit of all faces to +1. To represent an open polyhedron, one would set the sign bit of all faces to −1. Once the signs are given, the 'tracification' boils down to replacing the outward-pointing normal data with the whisker data for each face (in effect multiplying the face normal by the sign). This clearly sets the values of the associated painting correctly on all faces, but it remains to check whether the edges and vertices have the correct values as well. We address this question while checking for the other requirements of a polyhedral tracing.

The choice of signs defines the whisker on each face. We take the convention that the whisker set of an edge is the shortest arc that joins the whiskers of its adjacent faces. Therefore, there is no special conversion necessary to represent an edge in the tracing. This convention mirrors a similar one used for turns in two dimensions [8]. It gets its full relevance from the open/closed interpretation of tracings. Indeed, with this choice, an edge adjacent to two closed faces is closed (i.e. the painting has the same value on the edge as in the polyhedron interior). An edge adjacent to two open faces is open.

The vertex case is more delicate. The path stroked by the whiskers of the edges whose origin is a vertex v might not fit in a hemisphere (Figure 2), or might even self-intersect (Figure 3), hence requiring a normalization process. There is no canonical way to normalize, but the simplest is probably to "triangulate" the vertex as follows: single out a face f adjacent to v, and introduce zero-length edges between f and each of the other faces adjacent to v. This cuts v into many vertices, each of which has degree three and defines a triangle on the sphere of directions. A triangle is not self-intersecting and is contained within a hemisphere, so this defines a valid tracing. Note that the edges introduced in this process are zero-length, but their whiskers span an arc on the sphere of directions (Figure 3).

It remains to see whether vertices have the desired value in the associated painting, i.e. whether they have value 1 if one settled for the closed polyhedron, and value 0 for the open polyhedron. This is indeed the case, provided that, for each vertex, there exists a hemisphere that contains all the whiskers to the faces

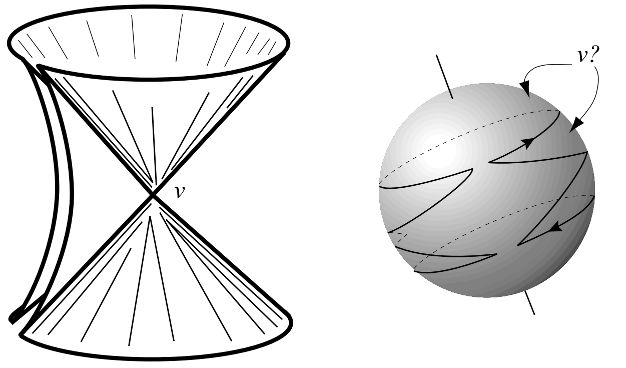

Figure 2: *A vertex v whose whisker map does not fit in any hemisphere. We can choose either of the two regions bounded by the spherical polygon on the right to be the range of the whisker map on the domain of v. Reversing this choice changes, by 1, the value of the associated painting at v.*

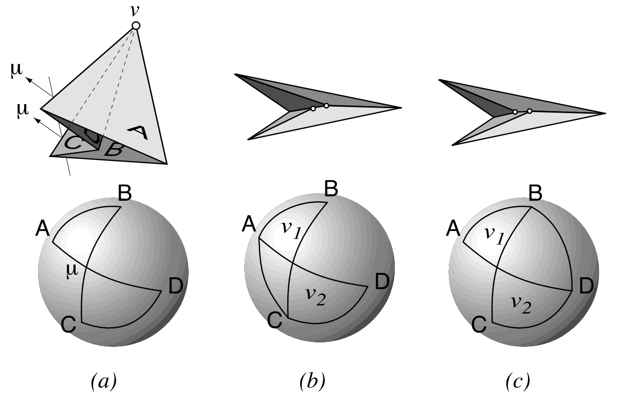

Figure 3: *(a) The simplest vertex that has a self intersecting path on the sphere of directions; (b,c) two ways to normalize it, cutting it into two vertices v_1, v_2 at the same location, by an adjunction of a zero-length edge. The top view shows the two vertices slightly displaced to emphasize the connectivity.*

adjacent to this vertex. Indeed, consider the closed case: for each vertex v the corresponding hemisphere gives us a direction (a whisker) with which the whiskers of all adjacent faces form an angle of less than $\pi/2$. Say that this direction is "up". Whichever normalization is used (if any is needed) at v, a convolution with a ball of radius ϵ will locally pull up all the adjacent faces.

Hence the winding number at v is the same as that of a point slightly inside the polyhedron, i.e. $+1$. If the hemisphere requirement is not satisfied at a vertex (Figure 2), it is unclear how to choose the whisker map so that the associated painting has value 1 at that vertex.

4 Convolution of polyhedral tracings

Since a polyhedral tracing is a special case of a tracing as defined in Section 2, the convolution operation is well defined. Moreover, the result is also a polyhedral tracing. Given the signed quad-edge representations of two input polyhedral tracings B and R, we show in this section how to obtain the convolution Q in the same representation. In the rest of the paper, we refer to features of B, R, and Q as *blue, red,* and *purple* respectively. The underlying manifold of the convolution is the fiber product of the whisker maps of B and R. It is described implicitly by the signed quad-edge structure of Q.

The features of the convolution are obtained by convolving pairs of matching features of B and R. A pair of features is said to *match* if the whisker sets of the two features have a non-empty intersection. In order to keep the notation simple, we assume that the intersection of the whisker sets of any two features is simply connected (and thus defines only one feature, whose whisker set is the intersection of the two original sets). In two dimensions, a forward move combined with a right turn becomes a backward move in the convolution. Likewise, in three dimensions, we expect that some faces will change sign in the convolution. A face f' in the convolution can arise when a face f of one tracing is convolved with a matching vertex v in the other. As f and f' have the same whisker, a change of sign requires a reversal of the edge ring (Figure 4). A face in the convolution is also obtained when a blue and a red whisker arc intersect on the sphere; their corresponding edges generate a parallelogram (the Minkowski sum of the two edges). The whisker of this new face is well defined, but its sign has to be determined.

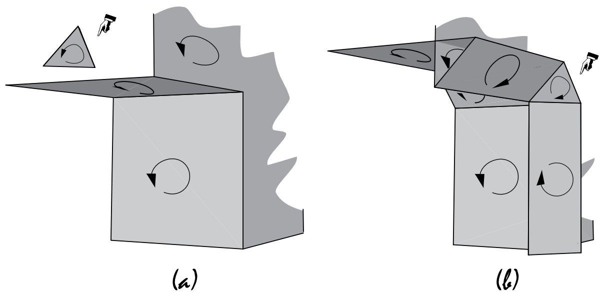

Figure 4: *In this convolution, the orientation of the little triangle is switched to create a consistent tracing. This reversal is captured by a negative sign assigned to the saddle vertex. Some pairs of edges whose whisker sets intersect create parallelogram faces in the convolution.*

4.1 Orienting the convolution

In the two-dimensional kinetic framework, the authors captured the reversals of sign in a very simple rule: forward moves and left turns have sign $+1$, backward moves and right turns have sign -1; furthermore, the sign of an output feature is the product of the signs of the corresponding input features. Guessing a similar sign rule in three dimensions, if it exists, is more difficult. We can deduce this sign rule from the general convolution definitions by calculating the orientation of the fiber product manifold on the domains corresponding to each feature of the convolution. The topology of this manifold also gives the connectivity of its quad-edge representation. We now proceed to define signs for edges and vertices for a tracing; we assume that these signs are also stored in the signed quad-edge structure.

1. The sign of an edge e, denoted $\sigma(e)$, is positive (resp. negative) if the triplet of vectors $(e\,\mathrm{Dest} - e\,\mathrm{Org}, e\,\mathrm{Lface}, e\mathrm{Rface})$ defines a direct (resp. indirect) basis.[1]

2. The sign of a vertex v, denoted $\sigma(v)$, is positive (resp. negative) if a traversal of the ring of edges around v via the Onext operator corresponds to a counterclockwise (resp. clockwise) traversal of the boundary of the whisker of v on the sphere of directions (from the point of view of an observer,

[1] A zero-length edge can be given an arbitrary sign

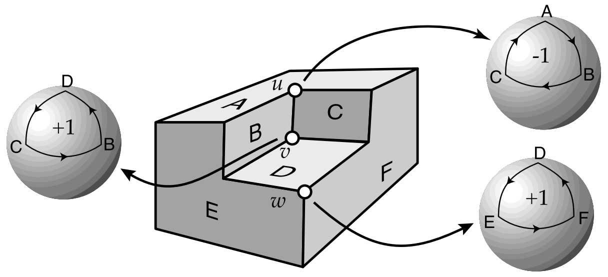

Figure 5: *The three simplest types of vertices with their ring of edges on the sphere of directions and their signs.*

facing from the outside, the hemisphere containing the whisker).

From the above definition, it follows that the sign of an edge e is the same as that of $e\,\mathrm{Sym}$. Also, when a tracing represents a closed polyhedron the above sign rule leads to giving convex edges a positive sign and concave edges a negative sign (much like left and right turns in polygonal tracings). Both convex and concave vertices have a positive sign, whereas a vertex like the one in Figure 4 has negative sign (Figure 5). If we view the boundary of the whisker set of a vertex as a 2-D tracing on the sphere of directions, then the sign of the vertex is exactly the winding number this tracing assigns to the region defining the whisker set. This is not a coincidence, and further work is needed to clarify and take advantage of this relationship.

4.2 Connectivity of the convolution

Let the vertex, edge, and face sets of B be V_b, E_b, and F_b, and those of R be V_r, E_r, and F_r. If features h_b and h_r of B and R match, the resulting convolution feature is denoted by the unordered pair (h_b, h_r). The assumption that every such pair produces at most one feature in the convolution makes the notation unambiguous. In an implementation, an output feature would be identified by a pointer, so that this assumption would be unnecessary. We denote $-e$ for $e\,\mathrm{Sym}$, allowing expressions like $\sigma(v)e$, to refer to e or $e\,\mathrm{Sym}$ depending on the sign of v.

We now define the signed quad-edge structure of P by enumerating its features and their connectivity.

$$\begin{array}{rcl}
V(P) & = & \{(v_b, v_r) \in V_b \times V_r \mid \mu(v_b) \cap \mu(v_r) \neq \emptyset\} \\
E(P) & = & \{(e, v) \in E_b \times V_r \ \cup\ E_r \times V_b \mid \\
 & & \qquad\qquad \mu(e) \ \cap\ \mu(v) \neq \emptyset\} \\
F(P) & = & \{(v, f) \in V_b \times F_r \ \cup\ V_r \times F_b \mid \mu(f) \in \mu(v)\} \\
 & \cup & \{(e_b, e_r) \in E_b \times E_r \mid \mu(e_b) \cap \mu(e_r) \neq \emptyset\}
\end{array}$$

The pairs (e, e'), $(e\,\mathrm{Sym}, e')$, $(e\,\mathrm{Sym}, e')$, and $(e\,\mathrm{Sym}, e'\,\mathrm{Sym})$ refer to the same parallelogram face. Before we define the topological connectivity of the convolution and the geometry and signs of its features, we need to define a primitive $\mathrm{Fedge}(v, e)$. This primitive is defined only when $\mu(v)$ and $\mu(e)$ overlap.

Definition 4.1. If e is an edge whose whisker intersects the boundary of the whisker set of a vertex v, we denote by $\mathrm{Fedge}(v, e)$ the edge e' with $e'Org = v$, and such that e' is the first such edge crossed when going from $(\sigma(v)\,\sigma(e)\,e)\,\mathrm{Lface}$ to $(\sigma(v)\,\sigma(e)\,e)\mathrm{Rface}$ on S^2.

The way to compute $\mathrm{Fedge}(v, e)$ is not addressed here. It will be a direct bi-product of the sweep-line algorithm described in the next section. The rules for constructing the polyhedral tracing representing the convolution are summarized in table 4.2.

Note that, just like in the two dimensional case, the topological adjacencies between features in the convolution are induced by those of the factor tracings and that the signs of its features are the products of the corresponding signs of the matching factor features. General results on tracings imply that the quad-edge structure defined above is consistent, and represents the tracing of the convolution. Except for the case of closed convex polyhedra, where it is the well-known Minkowski sum, the resulting convolution will in general self-intersect and have locally open faces (Figure 4 again).

4.3 Normalization and convolution size

When discussing output sensitive algorithms, there is an issue of what should be considered as the size of the output. This issue was discussed at length by Guibas and Seidel [9] for two-dimensional convolutions, and it

Table 1: *Connectivity and geometry of the convolution of two polyhedral tracings.*

$$
\begin{aligned}
(v,e)\,\mathrm{Sym} &= (v, e\,\mathrm{Sym})\\
(v,e)\,\mathrm{Org} &= (v, \sigma(v)\,e\,\mathrm{Org})\\
(v,e)\,\mathrm{Onext} &= \begin{cases} (v, (\sigma(v)\,e)\,\mathrm{Onext}^{\sigma(e)}) & \text{if } \mu(e\,\mathrm{Lface}) \in \mu(v)\\ ((\sigma(v)\,e)\,\mathrm{Org}, \mathrm{Fedge}(v,e)) & \text{otherwise}\end{cases}\\
(v,e)\,\mathrm{Lface} &= \begin{cases} (v, (\sigma(e)\,e)\,\mathrm{Lface}) & \text{if } \mu(e\,\mathrm{Lface}) \in \mu(v)\\ (e, \mathrm{Fedge}(v,e)) & \text{otherwise}\end{cases}\\
\lambda(v_b, v_r) &= \lambda(v_b) + \lambda(v_r)\\
\mu(e_b, e_r) &= \mu(e_b) \cap \mu(e_r)\\
\mu(v, f) &= \mu(f)\\
\sigma(v, f) &= \sigma(v)\,\sigma(f)\\
\sigma(e_b, e_r) &= \sigma(e_b)\,\sigma(e_r)
\end{aligned}
$$

is not surprising that it pops up again in three dimensions. In the present setting, two tracings that have different quad-edge representations are always considered distinct, although they might define the same painting (and the same "bag of states" in the vocabulary of the original kinetic framework). In this sense, the output described above is the only one that properly describes the exact topology of the fiber product manifold of the convolution.

This view is however slightly hypocritical, as the objects we are ultimately interested in are paintings, and not one specific representation. As in the two dimensional case, it is possible to construct two tracings of sizes m, n, such that their convolution is of size $\Theta(mn)$, while $\Theta(m+n)$ features are sufficient to represent the same painting. Even worse, the normalization process used to obtain a polyhedral tracing from a polyhedron may also lead to variations in the output size: if one takes a polyhedron with n vertices like the one of Figure 3, and another polyhedron with m features crossing one of the normalization lines, the convolution may have a size that varies from linear to quadratic depending on which normalization line is used. It would be desirable to review the normalization process to avoid this inconsistent behavior.

5 An efficient algorithm for the convolution

In this section, we briefly describe an algorithm to compute the signed quad-edge structure of a convolution Q of two tracings B and R with a total of n vertices in time $O(k\alpha(n)\log^3 n)$, where k is the number of vertices of Q. We will assume that B and R all have faces of sign $+1$, from which it follows that their whisker maps cover the sphere of directions S^2, and therefore $k \geq n$. The key algorithmic problem we must solve is to find the matching features of B and R. Geometrically, this reduces to finding the overlay, on the sphere of directions, of the whisker sets of the two tracings. This overlay will give us in fact the whisker sets of the convolution — its features will directly correspond to the elements of Q. The subdivision overlay problem requires us to locate all vertices of one subdivision into regions of the other, and to discover all pairs of intersecting red-blue edges. From the complexity point of view, the dominant cost of such an algorithm is the latter computation of all the bichromatic edge intersections. In the rest of the paper, we will assume that k actually represents the number of these bichromatic pairs. In the case where B and R are convex tracings (that is, when each whisker value appears only once per tracing) this problem has already been optimally solved [9] in $O(n+k)$ time. The difficulty with general tracings is that their whisker maps can multiply cover S^2 — they are effectively Riemann surfaces over S^2. During the overlay operation we want to discover all bichromatic edge intersections, but we cannot afford to pay for looking at crossings of edges of the same color (which would arise if we were to 'flatten out' each Riemann surface onto S^2).

Through an appropriate polar map, our problem is the same as the well-known problem of detecting red-blue intersections among a set of red and blue segments in the plane. This problem traditionally appears in two versions: general and disjoint (more precisely, the latter means that segments in each collection are required to have disjoint interiors). Neither is appropriate for our purposes: our collections need not be disjoint, as we pointed out above (monochromatic pseudo-intersections can arise from the overlay of different Riemann sheets), while at the same time a nearly linear red-blue merge algorithm seems highly unlikely for the general case. We show below how to exploit the fact that our segment collections are connected — that is, that the union of all points in the red segment collection forms a connected subset of the plane (and similarly for the blue). Agarwal and Sharir [1] investigated this version, and gave an $O(n\alpha(n)\log^2 n)$ algorithm to find *one* purple intersection when one exists, but their method could not be extended to find all purple intersections.

A classical Bentley-Ottmann sweep technique [5] can be applied for this problem, but it does not avoid the possibly quadratic cost of processing all blue-blue and red-red intersections (Figure 6). Our new algorithm, called HeapSweep, revisits the sweep paradigm by relaxing the constraint that the segments be completely ordered along the sweep line. It uses a novel data structure that can be used to efficiently sweep the upper envelope of a family of segments in the plane by maintaining only a partial vertical ordering among the segments. We will report this algorithm and its detailed analysis elsewhere [3] — below we just indicate some of the ideas of the method.

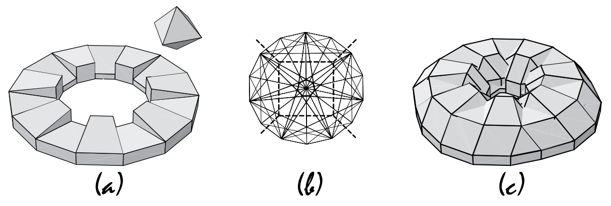

Figure 6: *(a) A diamond and a synchronizing gear with $n = 7$ teeth; (b) the arcs defined by the edges of the gear (in thin stroke) have $\Theta(n^2)$ intersections on the sphere, but only $\Theta(n)$ intersections with the features of the diamond (dashed stroke); (c) the convolution has only $\Theta(n)$ features.*

The upper envelope of an arrangement of n line segments is a well-known concept in computational geometry [7] and it has size $O(n\alpha(n))$ (where $\alpha(n)$ is the familiar inverse Ackermann function). We propose a new (non-optimal) algorithm to compute this structure that can be used as a building block for a sweep-line procedure in the red-blue intersection problem. The idea is to perform a line sweep on the segments, keeping track of a partial (instead of exact) order of the segment intersections on the sweep line in a heap-like data structure. Indeed, for the sweep line structure to be maintained in the totally ordered case, all segment intersections are events that have to be processed in the priority queue, leading to a possibly bad running time. In contrast, to maintain the heap-like structure, only intersections that involve a parent and child in the heap need to be scheduled and examined — we call these *tournament intersections*. No result is known about the maximum number of tournament intersections for standard heap implementations. In [3] we introduce several heap-like structures for which we can prove that the number of tournament intersections is roughly linear.

We now make use of the above as a basic component for an algorithm that computes the red-blue intersections in time $O(k\alpha(n)\log^3 n)$ in the connected version of the problem. At a given position, the sweep line is divided into *blue blocks* and *red blocks*. A block is a maximal interval on the line that intersect only one color of edges. All edges in a block are stored in a heap-like structure, so that the two extreme edges of the block are known at all times. Purple intersections

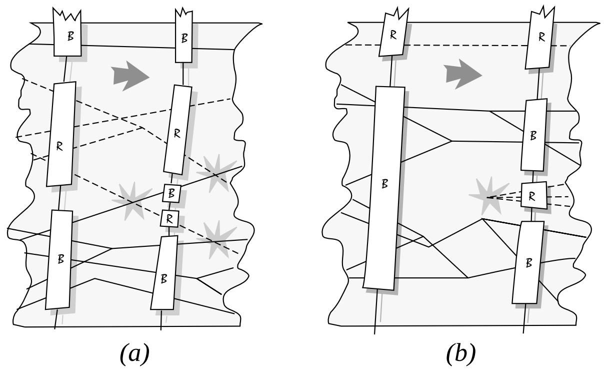

Figure 7: *(a) A purple intersection: two new blocks are created, and new purple intersection events are scheduled. (b) A SPLIT of a blue block in two, triggered by an unexpected red node*

can be discovered and scheduled by comparing the top and bottom segments of every pair of adjacent blocks, and can be processed in polylogarithmic time (see the schematic representation in figure 7-a).

Sometimes along the sweep, a blue block may encounter a red node that forces a split of the block into two blocks (Figure 7-b), a possibly expensive operation. Similarly, all arcs of a blue block may end at the same node, without a new arc to replace them, requiring a costly merge of the two adjacent red blocks in order to maintain the prescribed sweep line structure (these two possibilities arise in the computation of the convolution as soon as the input polyhedra are allowed to be non-convex). The connectedness of our families is the crucial hypothesis that makes the amortized cost of these operations polylogarithmic.

Theorem 5.1. *Let B, R be two polyhedral tracings with n vertices in total, whose faces all have sign $+1$, and whose convolution Q has k features. The algorithm* HeapSweep *computes the signed quad-edge structure of Q in time $O(k\alpha(n)\log^3 n)$*

6 Minkowski sum using convolution

Minkowski sums of polyhedra arise frequently in motion planning algorithms based on the configuration space approach [15, 14]. Though the case of computing the Minkowski sum has been well studied when both the polyhedra are convex [9, 6], non-trivial bounds are not known for the computation of Minkowski sum for general non-convex polyhedra [11]. We argue that the convolution of polyhedra provides a low-storage alternate representation which encodes much of the same (and often more) information as the Minkowski sum.

First let us address the size issue. If the two polyhedra are of sizes m and n their convolution is always of size $O(mn)$, whereas the Minkowski sum can be $\Theta(m^3n^3)$ in the worst case (an example where this happens can be found in [11]). The key savings in space and computation cost result from not computing or storing self-intersections of the convolution; these still give rise to features on the boundary of the Minkowski sum.

By Theorem 2.1, the winding number at x with respect to the convolution of closed tracings B and R is the Euler-Poincaré characteristic of the intersection of B with R reflected and translated by x. Thus the winding number captures information about how the robot and the barrier intersect. The intersection of two tracings representing simple closed polygons in the plane is always a set of simply connected components with Euler-Poincaré characteristic equal to the number of components. This means that the winding number with respect to the convolution is equal to the number of components in the corresponding intersection, and thus we can interpret this winding number as ‘depth of collision.’

However, the correspondence between convolutions and Minkowski sums of polyhedral tracings is not as simple as in the case of polygonal tracings because the intersection of closed polyhedral tracings may not be simply connected. For example, imagine pushing two glasses close to each other until their mouths interpenetrate. The intersection is a torus of zero Euler-Poincaré characteristic (one component – minus one tunnel) even though it is a non-empty solid. Thus for polyhedral tracings, the region of the convolution of zero winding number is not the same as the complement of the Minkowski sum. However, the *outer cell* of the Minkowski sum is in fact the outer cell of the convolution in E^3.

6.1 Algorithmic issues

We expect that many algorithms using the Minkowski sum can be speeded up by using the combinatorially smaller convolution instead. For example, in order to render the outer face of the Minkowski sum from an external viewpoint, it is sufficient to render the faces of the convolution in any sequence. The depth buffer of current graphics hardware will automatically ensure that only the visible sections of the Minkowski sum are produced. Similarly, algorithms that interrogate the Minkowski sum in various ways, such as ray-shooting, winding number and distance computations, planar sectioning, extrema and bounding volume computations, etc., can all benefit from using the smaller convolution in lieu of the Minkowski sum. Note that the solution to several of these problems makes use of the sign (orientation) information stored with the convolution. For example, a winding number computation at a point p, based on connecting p to another point of known winding number and then algebraically summing the crossings with the convolution along the connecting path, makes use of the signs. We also anticipate that the convolution can be used in novel ways along the lines of [16]. We intend to develop some of these applications in greater detail in a future publication.

7 Conclusion and further work

In this paper we have presented the theory of polyhedral tracings and their convolutions. Polyhedral tracings generalize ordinary polyhedra by providing an explicit set of normals at all features of the polyhedron: at vertices and edges, as well as at faces. This additional structure is not costly; it can be adequately represented by keeping just one additional bit (a "sign") with each face of the polyhedron. It allows us to properly state which pieces of the boundary are part of the polyhedron, and which are not. Furthermore, it permits the definition of the convolution of two polyhedral tracings, which compactly encodes information about the topology of the intersection of the two polyhedra under all possible translations. The convolution can serve as an alternate representation of the Minkowski sum of the polyhedra, which has found numerous applications in motion planning problems. When we compare the two, we see that the convolution gives us more information about the relative placement of the two polyhedra, while at the same time being much smaller combinatorially in the worst case. And while computing the Minkowski sum can be a challenging problem in three-dimensional computational geometry, our essentially optimal and output-sensitive convolution algorithm uses efficient planar data structures for its operation.

Clearly much remains to be done in further developing the theory of convolutions for polyhedral tracings and for more general instances. Other possible applications need to be investigated: object placement, distance computations, and more complex motion planning problems.

Acknowledgments

We wish to thank Rajeev Motwani and Dan Halperin for helpful discussions and Harish Devarajan, Scott Cohen, and Héloise Mallet for valuable comments on the manuscript. This work was supported by National Science Fundation grant NSF CCR-9215219, NSF/ARPA grant IRI-9306544, and US Army Grant 5-23542A.

References

[1] P. K. Agarwal and M. Sharir. Red-blue intersection detection algorithms, with applications to motion planning and collision detection. *SIAM J. Comput.*, 19(2):297–321, 1990.

[2] Chanderjit L. Bajaj and Myung-Soo Kim. Generation of configuration space obstacles: The case of moving algebraic curves. *Algorithmica*, 4:157–172, 1989.

[3] J. Basch, L.J. Guibas, and G.D. Ramkumar. Reporting red-blue intersections between connected sets of lines segments. *To appear in ESA '96.*

[4] J. Basch and L. Ramshaw. Orienting the fiber product of smooth manifolds. *In preparation.*

[5] J. L. Bentley and T. A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Comput.*, C-28:643–647, 1979.

[6] B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. Algorithms for bichromatic line segment problems and polyhedral terrains. *Algorithmica*, 11:116–132, 1994.

[7] H. Edelsbrunner, L. J. Guibas, and M. Sharir. The complexity and construction of many faces in arrangements of lines and of segments. *Discrete Comput. Geom.*, 5:161–196, 1990.

[8] L. J. Guibas, L. Ramshaw, and J. Stolfi. A kinetic framework for computational geometry. In *Proc. 24th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 100–111, 1983.

[9] L. J. Guibas and R. Seidel. Computing convolutions by reciprocal search. *Discrete Comput. Geom.*, 2:175–193, 1987.

[10] L. J. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Trans. Graph.*, 4:74–123, 1985.

[11] A. Kaul and M. A. O'Connor. Computing minkowski sums of regular polyhedra. Report RC 18891 (82557) 5/12/93, IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, 1993.

[12] A. Kaul and J. Rossignac. Solid-interpolating deformations: construction and animation of PIPs. *Comput. & Graphics*, 16:107–116, 1992.

[13] Serge Lang. *Differential Manifolds.* Addison-Wesley, 1972.

[14] J.-C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, 1991.

[15] T. Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Trans. Comput.*, C-32:108–120, 1983.

[16] G. D. Ramkumar. Algorithms to compute the convolution and minkowski sum outer-face of two simple polygons. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, 1996.

[17] J. Rossignac and A. Kaul. Agrels and bips: Metamorphosis as a bézier curve in the space of polyhedra. In *Eurographics '94 Proceedings*, volume 13, pages 179–184, 1994.

[18] P. Schapira. Operations on constructible functions. In *Journal of pure and applied algebra* [8], pages 83–93.