Exact Convex Hull Computation for Plane and Space Parametric Curves

Christina Katsamaki, Fabrice Rouillier, Elias Tsigaridas

Exact Convex Hull Computation for Plane and Space Parametric Curves

Christina Katsamaki, Fabrice Rouillier, Elias Tsigaridas

INRIA Paris, IMJ-PRG, Sorbonne Université, Paris Université, Paris , France

Abstract

We consider the problem of the computing the boundary of the convex hull of rational parametric curves in $\mathbb{R}^2$ and in $\mathbb{R}^3$. The boundary of the convex hull is a semi-algebraic set and an exact representation of it breaks it down to a combination of line segments and arcs of the curve for the 2D case, and of triangles and surface patches for the 3D case. For both plane and space curves, we provide an algorithmic solution together with bit-complexity estimates. We show that the computation of the convex hull's boundary reduces in both cases to univariate and bivariate solving and to isolating roots of a univariate polynomial with coefficients in a multiple field extension. We express the bit-complexity with respect to the bit-complexity of the latter operation. Leveraging the results of Katsamaki and Rouillier [ISSAC '23], offers asymptotic upper bounds on the total bit-complexity.

Keywords: Convex hull, parametric curve, bit-complexity, polynomial system

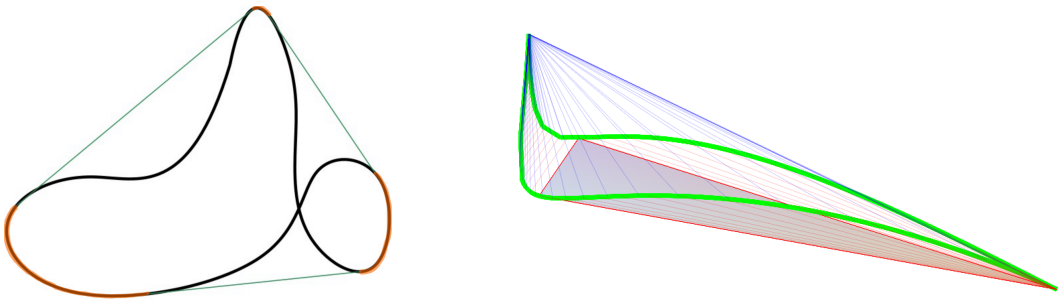
1. Introduction

For any subset of $\mathbb{R}^n$, where $n \geq 1$, its convex hull is defined as the smallest convex set that contains it, or in more technical terms, as the intersection of its supporting halfspaces [8, Thm. 4.5]. Convex hull computations are fundamental in computational geometry with direct applications in motion planning [46, 45], computer vision [13] and geometric modeling systems [17, 18]. Notably, convex hulls of non-linear objects arise naturally in optimization [28, 4] and learning theory [11, 41]. We focus on non-linear convex hulls, and in particular of parametric curves in $\mathbb{R}^2$ and in $\mathbb{R}^3$. Let C be an algebraic curve over $\mathbb{R}^n$, parametrized by

$$\begin{aligned} \phi: \quad & \mathbb{R} \dashrightarrow \mathbb{R}^n \\ & t \mapsto (\phi_1(t), \ldots, \phi_n(t)) = \Big(\frac{p_1(t)}{q_1(t)}, \ldots, \frac{p_n(t)}{q_n(t)}\Big), \end{aligned} \tag{1}$$

where $p_i, q_i \in \mathbb{Z}[t]$ have degree at most d and bitsize τ and $\gcd(p_i(t), q_i(t)) = 1$, $i \in [n]$. For $I \subseteq \mathbb{R}$, we denote by $\mathrm{conv}(\phi(I))$ the convex hull of $\phi(I)$. For the rest of this paper, $\phi(I)$ is a compact subset of $\mathbb{R}^n$. When $n = 2$, the boundary of $\mathrm{conv}(\phi(I))$ consists of a combination of

Email addresses: christina.katsamaki@inria.fr (Christina Katsamaki), Fabrice.Rouillier@inria.fr (Fabrice Rouillier), elias.tsigaridas@inria.fr (Elias Tsigaridas)



(a) (b)

Figure 1: (a) A plane curve and its convex hull; the boundary consists of the green line segments and the orange arcs. (b) A curve in $\mathbb{R}^3$ and its convex hull; the boundary consists of one triangle and four ruled (developable) surface patches (in red and in blue).

smooth curved arcs and segments joining two points on the curve (Fig. 1(a)). When $n = 3$, it is a combination of triangles and ruled (developable) surface patches (Fig. 1(b)). We design algorithms for the boundary description in these cases and we study their bit-complexity.

Parametric curves stimulated our interest since the parametric representation has already been proven advantageous in the topology computation; for parametric curves in $\mathbb{R}^n$ the bit-complexity is linear in n, whereas for the implicit case the dependence in n seems to be exponential. In particular, for plane curves the bit-complexity of the implicit and parametric case coincides [24] but for space curves the bit-complexity gap is already of order $O(d^2)$, where d is the degree of the polynomials involved [9]. Parametric curves come up in various applications in control theory [27], in machine learning [42] or in chemical engineering [10].

Previous work. There is a series of combinatorial algorithms for computing the convex hull of plane curved arcs in parametric form [37, 15, 1, 20]. However, in all these approaches, there are assumptions on the monotonicity and/or the total curvature of the arcs and they rely on several oracles whose running time can be expensive, e.g., oracles for the computation of bitangents, that are lines tangent to the curve at two points. Bitangents' computation is a problem of independent interest; among other works (e.g. [30, 33]), of particular interest is the one of Johnstone [21], that reduces the problem of finding common tangents between a pair of parametric curves to the intersection of parametric curves in a dual space. Then, he applies his method on the convex hull computation of a smooth paramatric plane curve [22]; the bitangents now correspond to multiple points of the dual curve. After computing the bitangents, he constructs the convex hull by walking along the curve, starting from a point on the convex hull and leaping, every time a bitangent is found, to the other end of the segment. Essentially, with this method we walk along the curve by walking along the parameter interval. However, it applies only to smooth curves. A more general approach for plane parametric curves with possible self-intersections is proposed by Elber et al. [16]. They find the parameter intervals that correspond to the arcs of the curve on the boundary, by means of a projection of a plane curve. Then, they sort the arcs with respect to their normal angles and connect them accordingly with segments whenever there is a gap. So, here, the sorting compensates for the fact that now the curve has self-intersections and the connections of the arcs of the curve are not trivial. In practice, computing the arcs of the curve that are on the boundary of the convex hull and sorting them is not straightforward. For an exact

algorithm and an analysis of its bit-complexity, a precise algebraic formulation of the problem is needed. Moreover, the presence of cusps is not considered.

For space curves, Sedykh provides a detailed study of the singularities of the convex hull's boundary [38]. Nevertheless, in our case, a complete classification is not necessary. Seong et al. [40] present an algorithm for the computation of the convex hull, that is extension of the work of [16] for plane curves. They describe the boundary of the convex hull as a combination of developable surface patches and plane patches (triangles). The plane patches are found by solving systems of polynomial equations and the developable surface patches by tracing an implicit plane curve that expresses a bitangent condition. As with the algorithm for plane curves, cusps are not treated and there are no bit-complexity estimates. There is a series of important works that nevertheless do not give any information on the facial structure of the boundary; this includes the computation of the algebraic boundary [35, 36] or the representation as the projection of a spectrahedron [19, 43].

Contribution. We propose an algorithm for the convex hull computation of plane parametric curves, as well as one for parametric curves in $\mathbb{R}^3$. The algorithms apply to any curve, as long as it is properly reparametrized (see [34] for an algorithm and [24] for an analysis of its complexity) and require in the input a parameter interval $I \subset \mathbb{R}$, such that $\phi(I)$ is compact (Rem. 1). The algorithms share the basic idea and they both rely on our Support predicate (Sec. 3) that applies to any dimension and checks if a given hyperplane is a supporting hyperplane for the curve (see Sec. 2 for a definition). The latter problem is reduced to a condition of sign-invariance of a univariate polynomial. We remark that for implicit curves designing such a predicate is not straightforward.

The boundary of the convex hull of plane parametric curves comprises of arcs of the curve and line segments, whose types we classify in Def. 2. The first step of the algorithm is to compute all these line segments, by solving some polynomial systems. However, among the solutions we obtain segments that do not lie on a supporting line to the curve and thus, they are not on the boundary of $\mathrm{conv}(\phi(I))$ (see Fig. 2 for an example). The second step consists of decomposing the curve at the endpoints of these segments; in this way, every arc of the decomposition is either entirely on the boundary of the convex hull, or it is contained in its interior. This is equivalent to decomposing the parameter interval I at the parameters that correspond to the segments' endpoints. At the third step, we start from a point on $\phi(I)$, we follow the branches of the curve one by one and we check for every branch if it is on the boundary of the convex hull by calling the Support predicate for an arbitrary tangent line at the interior of the branch. The last step, amounts to connecting the branches that are on the boundary accordingly with line segments. If the endpoint of a branch is smooth, then determining the segment is trivial, since it lies on the tangent line (and the other endpoint has been found at the first step). When the endpoint is not smooth (i.e., cusp or endpoint of $\phi(I)$), we check all the possible segments by employing the Support predicate of Sec. 3.

The boundary of the convex hull of curves in $\mathbb{R}^3$ consists of a combination of triangle facets, whose types we classify in Def. 8, and of some surface patches (Def. 9). The surface patches are ruled (they are generated by line segments connecting two points on the curve) and also developable. They are part of the bisecant surface (Eq. (8)), which is traced through the bisecant curve (Eq. (7)), an implicit plane curve in the space of parameters. Finding the surface patches on the boundary is equivalent to determining certain branches of the bisecant curve. The algorithm follows the same line as in the 2D case. First, we compute the triples of parameters that

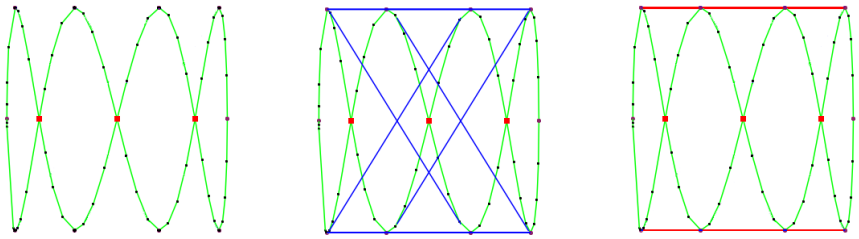
correspond to triangle facets, through solving some polynomial systems. At the second step, we split the graph of the bisecant curve into branches by computing a special purpose Cylindrical Algebraic Decomposition (CAD) of I. The third step is to identify the surface patches that are on the boundary of the convex hull. They correspond to branches of the bisecant curve. The SUPPORT predicate is used again to find the branches of the bisecant curve that correspond to surface patches. At the last step, we connect the surface patches with triangle facets. Triangles with a smooth vertex that are on the boundary are revealed by the endpoints of the branches of the bisecant curve. We call the predicate SUPPORT for the planes on which there are triangles that do not have any smooth vertex. Then, the triangles that do not belong on a supporting plane are discarded.

Our algorithms are built upon the algorithms of [16] for plane curves and of [40] for 3D curves. These approaches present challenges when it comes to implementation and lack analysis of their computational complexity. So, our goal is to bridge the gap between a theoretical approach and one that can be practically implemented in a working system. Moreover, linear facets (segments or triangles) with non-smooth vertices (cusps or endpoints of $\phi(I)$) require special treatment. This is done at the last step of our algorithm, where we check if the corresponding lines or planes are supporting for $\phi(I)$. We also prove that this operation dominates the bit-complexity. When the facets have a smooth vertex, it is not necessary, since the tangency condition at the smooth point defines the facet uniquely.

A bit-complexity analysis, up to our knowledge, has been missing from literature. Our main challenge was to minimize the bit-complexity by identifying the appropriate algebraic formulation that could eliminate expensive operations. By exploiting the problem's geometry, we show that treating the linear facets with non-smooth vertices at the last step amounts to isolating the roots of a zero-dimensional system of the form $\{F_1(X_1) = \cdots = F_N(X_N) = F(\mathbf{X}, Y)\}$. We denote the bit-complexity of this operation by $C(M, \Lambda, N)$ (Def. 3). We emphasize that the last polynomial might not be square-free, if we consider it as univariate polynomial in Y. Since there has not been extensive work on the complexity of isolating roots of systems of this form, we express the bit-complexity with respect to d, τ and $C(M, \Lambda, N)$. We employ algorithms for univariate and bivariate solving, and the resulting bit-complexities are in $\widetilde{O}_B\left(d^7 + d^6\tau\right) + C\left(d, \tau, 2\right)$ for 2D curves (Thm. 4), and in $\widetilde{O}_B(d^{10} + d^9\tau) + C(d, d^6\tau, 1) + C(d, d + \tau, 3)$ for 3D curves (Thm. 11). Using [14, Prop.19] for the case where $N = 1$ and the results of [23], the previous bit-complexities become $\widetilde{O}_B\left(d^{10} + d^9\tau\right)$ for plane curves and $\widetilde{O}_B\left(d^{13} + d^{12}\tau\right)$ for space curves. Alternatively, one can employ the Las-Vegas algorithm of [7] for $N \times N$ zero-dimensional polynomial systems. Then, by denoting by $\omega \approx 2.372873$ the exponent in the complexity of matrix multiplication, the bit-complexities become $\widetilde{O}_B(d^{2\omega+5}(d + \tau)) \approx \widetilde{O}_B(d^{9.75}(d + \tau))$ for plane curves and $\widetilde{O}_B(d^{3\omega+7}(d + \tau)) \approx \widetilde{O}_B(d^{14.12}\ (d + \tau))$ for 3D curves. So, our results lead to lower bit-complexity estimates. A preliminary version of our algorithms is implemented in MAPLE (examples in Sec. 6).

2. Background on rational curves

For the sake of generality, we introduce some basic notions for rational curves in $\mathbb{R}^n$ and their convex hull. Let C be an algebraic curve over $\mathbb{R}^n$, parametrized by ϕ in Eq. (1). Then, C is the Zariski closure of $\phi(\mathbb{R})$. We call $\phi(t)$ a *parametrization* of C. We say that the parametrization is of size (d, τ) when the polynomials p_i, q_i have degree at most d and bitsize τ, $i \in [n]$. The parametrization is *proper*, if it is injective for almost all points on C [39, Ch. 4]. If it is not



(a) (b) (c)

Figure 2: *(a) The graph of the curve parametrized by* $\left(\frac{-t^2+1}{t^2+1}, \frac{-8t(t^2-1)(t^4-6t^2+1)}{t^8+4t^6+6t^4+4t^2+1}\right)$*, (b) segments tangent to the curve at two points or more (in blue), (c) the ones that are on the convex hull (in red).*

proper, reparametrization algorithms exist, e.g., [34] (see [24] for an analysis of its complexity). Without loss of generality, we assume that no component of the parametrization ϕ is constant. The *point at infinity*, $\mathbf{p}_\infty$, is the point on C we obtain for $t \to \pm\infty$ (if it exists). The *tangent vector* of the curve at a point $\mathbf{p} = \phi(t)$ is $\phi'(t)$.

Singular points on the curve correspond to shape features that are known as cusps and self-intersections of smooth branches. Self-intersections are *multiple* points, i.e., points on C with more than one preimages. The parameters that correspond to pairs of multiple points are obtained by solving the square polynomial system [24, Lem. 11]:

$$h_i(s,t) := \frac{p_i(s)q_i(t) - q_i(s)p_i(t)}{s-t}, \quad \text{for } i \in [n]. \tag{2}$$

Cusps are points on the curve where the tangent vector is the zero vector. This is a necessary and sufficient condition when the parametrization is proper [31]. It holds that $h_i(t,t) = \phi_i'(t)q_i^2(t)$, for $i \in [n]$. Therefore, a parameter that gives a cusp of C is a root of

$$H(t) := \sum_{i=1}^{n} h_i^2(t,t) \tag{3}$$

(see [24, Lem. 4.1]). *Poles* are parameters for whom ϕ is not well-defined, i.e., $t \in \mathbb{R}$ such that $\prod_{i\in[n]} q_i(t) = 0$. We adopt the definitions of [8] of a supporting halfspace and supporting hyperplane for sets in $\mathbb{R}^n$. Let C be a non-empty set in $\mathbb{R}^n$. A *supporting halfspace* of C is a closed halfspace K in $\mathbb{R}^n$ such that $C \subset K$ and $H \cap C \neq \emptyset$, where H denotes the bounding hyperplane of K. A *supporting hyperplane* of C is a hyperplane H in $\mathbb{R}^n$ which bounds a supporting halfspace. If C is not contained in H, then H is a *proper supporting hyperplane*. Then, the convex hull of a set in $\mathbb{R}^n$, is the intersection of its supporting halfspaces.

Remark 1. *Let $I \subset \mathbb{R}$. For the convex hull conv$(\phi(I))$ to be a compact subset of $\mathbb{R}^n$, I has to be either a union of closed intervals not containing any poles of ϕ or a union of such closed intervals and intervals of the form $(-\infty, a], [a', +\infty)$, for $a \leq a'$, $a, a' \in \mathbb{R}$, if $\mathbf{p}_\infty$ exists.*

3. Support: a predicate for parametric curves in $\mathbb{R}^n$

Given the implicit equation of a hyperplane $\langle \mathbf{a}, \mathbf{x} \rangle + c = 0$, where $\mathbf{a}, \mathbf{x} \in \mathbb{R}^n$ and $c \in \mathbb{R}$, the Support predicate decides if it is a supporting hyperplane of $\phi(I)$. By definition, this holds if

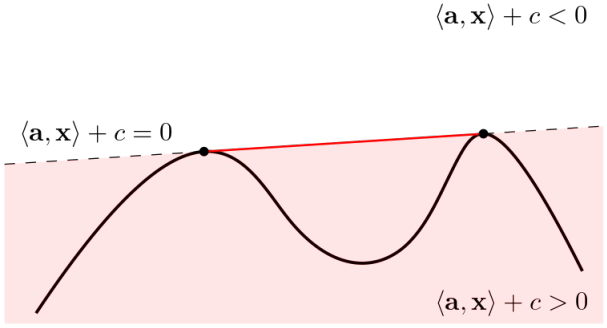


Figure 3: The line $\langle \mathbf{a}, \mathbf{x} \rangle + c = 0$ is supporting for $\phi(I)$; $\phi(I)$ lies entirely in the halfplane where $\langle \mathbf{a}, \mathbf{x} \rangle + c$ is positive. The line segment (in red) is on the boundary of the convex hull.

the hyperplane bounds a supporting halfspace of $\phi(I)$. Then, the convex hull of the intersection points of $\phi(I)$ and the hyperplane is a face on the boundary of the convex hull of $\phi(I)$ (see Fig. 3 for an example in two dimensions). Given the parametric representation of the curve, we can reduce the support test for the hyperplane in question to a condition of sign-invariance; for the hyperplane defined by $\langle \mathbf{a}, \mathbf{x} \rangle + c = 0$ is supporting for $\phi(I)$ if and only if $\langle \mathbf{a}, \phi(t) \rangle + c \in \mathbb{R}(t)$ has at least one root in I and is sign-invariant in I. So, we just have to isolate the real roots of $\langle \mathbf{a}, \phi(t) \rangle + c$, since this allows to determine its sign over I.

4. Convex hull of parametric curves in $\mathbb{R}^2$

Let $\phi(t) = (\phi_1(t), \phi_2(t)) = \left(\frac{p_1(t)}{q_1(t)}, \frac{p_2(t)}{q_2(t)}\right)$ (defined as in Eq. (1)) be a proper parametrization of a plane curve C and $I \subseteq \mathbb{R}$, s.t. $\phi(I)$ is compact (see Rem. 1).

Assumptions. *(i) The point at infinity, if it exists and it is in $\phi(I)$, is not endpoint of any segment. Otherwise, we will not be able to obtain these segments as solutions of a polynomial system. We can reparametrize the curve to ensure that the point at infinity is not a segment's endpoint (Las Vegas algorithm in expected time $\widetilde{O}_B(d^2 + d\tau)$ [24, Lem.7]).*

(ii) The real subset I has $k \in O(1)$ boundary points of constant bitsize.

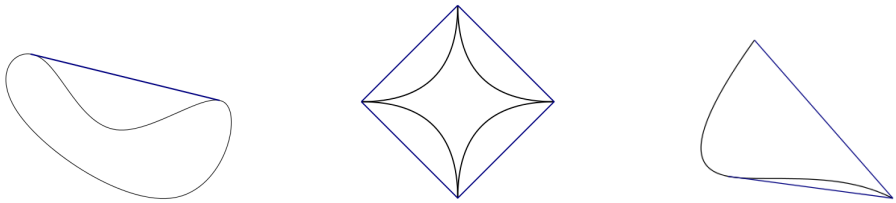
The facets of $\mathrm{conv}(\phi(I))$ are line segments joining two (or more) points on the curve. The boundary also consists of one-dimensional families of points of C, which are zero-dimensional faces; they assemble to smooth arcs of the curve (Fig. 1(b)). We will consider these arcs also as facets, by abuse of terminology. In the next definition, we classify the line segments on the boundary in several types (see also Fig. 4). For the set $I \subset \mathbb{R}$, we denote by $\mathrm{bd}(I)$ its boundary.

Definition 2 (Types of line segments)**.** *We distinguish the following types of line segments (I-VI) on the boundary of $\mathrm{conv}(\phi(I))$ according to their endpoints:*

I. bitangent segments: on the common tangent line of two or more smooth points of $\phi(I)$,

II. cusp-curve segments: connecting a cusp and a smooth point on the curve, lying on the latter point's tangent line,

III. cusp-cusp segments: whose endpoints are both cusps,



(a) (b) (c)

Figure 4: *Segments of type (a) I, (b) III and (c) IV and VI (in blue).*

IV. endpoint-curve segments: connecting a point given by a parameter in bd(I) and a smooth point on the curve, lying on the latter point's tangent line,

V. endpoint-cusp segments: connecting a point given by a parameter in bd(I) and a cusp,

VI. endpoint-endpoint segments: segments connecting two points given by parameters in bd(I).

4.1. Algorithm

Step 1: Computing the segments. We compute all segments of types I to VI for $\phi(I)$ (Def. 2). They form a superset of the set of segments that are on the boundary of conv($\phi(I)$) (Fig. 2). Each segment has a type (I to VI) and it is determined by two parameter values, say a and b, such that $\phi(a)$ and $\phi(b)$ are the segment's endpoints respectively. We find the parameters that correspond to each type of segments by solving a polynomial system and we denote by $\mathcal{S}$ the set of all these pairs of parameters. We assume that for a given pair of parameters in $\mathcal{S}$ we can determine the type of the corresponding segment in constant time.

We consider the equations in $\mathbb{Z}(s,t)$

$$\begin{aligned} \langle \phi(s) - \phi(t), (-\phi_2'(s), \phi_1'(s)) \rangle &= 0\,, \\ \langle \phi(s) - \phi(t), (-\phi_2'(t), \phi_1'(t)) \rangle &= 0\,. \end{aligned} \tag{4}$$

We can easily verify that the pairs of parameters corresponding to segments of Types I to III satisfy these equations. We divide them by $(s-t)^2$, since it is always a factor and we let $H_1(s,t), H_2(s,t) \in \mathbb{Z}[s,t]$ be their numerators respectively. We now have to compute the isolated roots of the system $\{H_1(s,t) = H_2(s,t) = 0\}$ over $\mathbb{R}^2$. This system has as roots also the tuple of parameters that correspond to multiple points of the curve and the tuples of poles. The pairs that correspond to multiple points can be viewed as bitangents of length zero, and thus do not harm the correctness of the algorithm. As for the pairs of poles, they can easily be excluded, by checking if any parameter is also a root of $q_1(s) \cdot q_2(s)$. For the segments of types IV and V, that have an endpoint of the form $\phi(a)$, with $a \in \mathrm{bd}(I)$, we isolate the roots of $H_1(s,a) \in \mathbb{Q}[s]$ and we consider all the pairs (s,a) such that s is a root of $H_1(s,a)$. For segments of type VI we just have to consider all pairs of parameters in bd(I). In that way, we can construct the set $\mathcal{S}$, containing all the pairs of parameters.

Step 2: Decomposition of the curve. We subdivide I into a finite number of open intervals and points, where the points of the decomposition are the parameters of all pairs in $\mathcal{S}$. Let

$R_t(s) = \mathtt{res}_t(H_1, H_2)$. The points of the decomposition are roots of the polynomial

$$P(s) := R_t(s) \prod_{a \in \mathrm{bd}(I)} (s - a) \cdot H_1(s, a)\,. \tag{5}$$

Now, every sub-interval corresponds to a parametric arc that does not contain endpoints of any segment and thus, it is either contained in the boundary of $\mathrm{conv}(\phi(I))$ or in its interior.

Step 3: Computing the arcs. For every interval I_j of the decomposition of I computed in Step 2, we work as follows: We pick any rational $u \in I_j$; $\phi(u)$ is always a smooth point on C since the singular points are given by parameters that are among the points of decomposition of I. We check if $\phi(u)$ is on the boundary of $\mathrm{conv}(\phi(I))$, which is the case if and only if there is a supporting line of $\phi(I)$ passing from $\phi(u)$ [8, Thm.4.3]. But since $\phi(u)$ is smooth, the only possible supporting line is the tangent line to C at $\phi(u)$. So we call the predicate Support for the tangent line at $\phi(u)$. Its equation is $\langle (x, y) - \phi(u), (-\phi_2'(u), \phi_1'(u)\rangle = 0$. Let $\ell_u(\lambda) \in \mathbb{Q}[\lambda]$ be the numerator of the tangent line equation after substituting (x, y) with the parametrization $\phi(\lambda)$. We isolate the real roots of $\ell_u(\lambda)$ in order to determine its sign in I. If it is supporting to $\phi(I)$, then the parametric arc $\phi(I_j)$ is on the boundary of $\mathrm{conv}(\phi(I))$.

Step 4: Connecting the facets. Having determined the curved facets on the boundary at the previous step, we now describe how they connect with each other. In other words, we compute the segments, among the ones found in the first step, that are on the boundary of the convex hull. Two neighboring facets, intersect at a point of the curve. Since this point is on the boundary, there has to be a supporting line to the curve passing from it [8, Thm.4.3]. If it is a smooth point, then the supporting line can only be the tangent line. When the point is not smooth, there is not a unique possible choice for the supporting line. So, for every arc of the curve that is on the boundary, we check each of its endpoints:

- If the endpoint is smooth, then the parametric arc is adjacent to the bitangent segment passing from this point.
- If the endpoint is not smooth but it belongs to a segment that has only one non-smooth point, then, the segment will be found with this procedure from the other one of its two adjacent curved facets.
- If the endpoint is not smooth and it belongs to a segment with two non-smooth endpoints, that is cusp-cusp (type III), endpoint-cusp (type IV) and endpoint-endpoint (type V) segment, we need to check every possible combination to find the ones that contribute to the boundary. Thus, for every such segment we check if it lies on a supporting line to the curve by calling the predicate Support. The parameters corresponding to cusp-cusp segments are also among the solutions of the system $\{H(s) = H(t) = 0\}$. The equation of the line passing through $\phi(s)$ and $\phi(t)$ is $\langle \phi(s) - (x, y), (-(\phi_2(s) - \phi_2(t)), (\phi_1(s) - \phi_1(t))\rangle = 0$. We substitute (x, y) with $\phi(\lambda)$ in the previous equation. Let $L(s, t, \lambda) \in \mathbb{Z}[s, t, \lambda]$ be the polynomial obtained after clearing the denominators . Calling the predicate Support for all the lines connecting cusps, amounts to computing the isolated roots of

$$\begin{aligned} H(s) &= 0\,, \\ H(t) &= 0\,, \\ L(s, t, \lambda) &= 0\,. \end{aligned} \tag{6}$$

For the segments of type V, for all $a \in \mathrm{bd}(I)$, we compute the polynomial $L(a, t, \lambda) \in \mathbb{Q}[t, \lambda]$ and then solve the system $\{H(t) = L(a, t, \lambda) = 0\}$. For the endpoint-endpoint segments, we solve for λ the polynomials $L(a, b, \lambda) \in \mathbb{Q}[\lambda]$ for every $a, b \in \mathrm{bd}(I)$.

Complexity analysis. The following definition serves in expressing the bit-complexity.

Definition 3. *We consider the zero-dimensional polynomial system*

$$\{F_1(X_1) = \cdots = F_N(X_N) = F(X_1, \ldots, X_N, Y) = 0\}$$

in $\mathbb{Z}[X_1, \ldots, X_N, Y]$. *If all the polynomials have degree* $O(M)$ *in each variable and bitsize in* $\widetilde{O}(\Lambda)$, *then* $C(M, \Lambda, N)$ *is the bit-complexity of computing isolating intervals for the roots.* $C(M, \Lambda, N)$ *is linear in* Λ *and is increasing with* N.

The next theorem summarizes our result. Its proof follows in the next subsection.

Theorem 4. *Let* C *be a curve with a proper parametrization* $\phi(t)$ *as in Eq. (1), of size* (d, τ). *Let* $I \subset \mathbb{R}$ *such that* $\phi(I)$ *is compact in* $\mathbb{R}^2$. *There is an algorithm that computes the convex hull of* $\phi(I)$ *in* $\widetilde{O}_B(d^7 + d^6\tau) + C(d, d + \tau, 2)$. *The output of the algorithm is a circular doubly linked list (with* $N \in O(d^2)$ *elements), containing the parameter intervals and the corresponding parametrizations for each facet; the image of the interval over the parametrization is a parametric arc or a segment on the boundary of* $conv(\phi(I))$.

Corollary 5. *Using the bit-complexity results of [23] for* $C(d, d + \tau, 2)$, *the bit-complexity of the previous theorem becomes* $\widetilde{O}_B(d^{10} + d^9\tau)$.

4.2. Proof of Theorem 4

4.2.1. Correctness

The correctness of the algorithm is based on the fact that the Support predicate, correctly detects the curve branches that are on the boundary of the convex hull (Step 3). On connecting the curve branches with segments, we refer to the selection criterion of the gift-wrapping algorithm for the set of points $\phi(I)$; given a point p on the boundary of the convex hull, the next point with whom it connects, is a point $q \in \phi(I)$ such that the line that passes from p and q is strictly supporting for the convex hull. The connection phase in Step 4 respects this criterion.

Remark 6 (Degeneracies)**.** *In a degenerate situation, several line segments on the boundary of the convex hull can lie on the same supporting line (e.g., Fig. 2). Consequently, each segment will be accounted for multiple times with different pairs of endpoints. However, this problem can be resolved during a post-processing stage, where the systems' solutions from the first step are analyzed to identify such situations.*

4.2.2. Bit-complexity

Step 1. We compute the polynomials H_1, H_2 in $\widetilde{O}_B(d^3 + d^2\tau)$: To construct the numerator of each H_i we perform multiplications of univariate polynomials in the variable s and in the variable t; this costs $\widetilde{O}_B(d^2\tau)$ [44, Ch.8]. The bi-degree of the resulting expression is in $(O(d), O(d))$ and its bitsize in $\widetilde{O}(\tau)$. Then, we divide by $(s - t)^2$ and this costs $\widetilde{O}_B(d^3 + d^2\tau)$ by adapting [44, Ex.10.21] to the bivariate case. The polynomials H_1 and H_2 are then of size $(O(d), \widetilde{O}(d + \tau))$.

Isolating the isolated roots of the system $\{H_1(s,t) = H_2(s,t) = 0\}$ costs $\widetilde{O}_B(d^6 + d^5\tau)$ [26]. If it is not zero-dimensional, then we have first to compute the gcd of H_1 and H_2 and their gcd-free parts. This is done in $\widetilde{O}_B(d^6 + d^5\tau)$. We can also compute the resultant of H_1 and H_2 with respect to s or t at no extra cost. Notice that both resultants are the same polynomial, since the system is symmetric. Let $R_t(s) = \mathtt{res}_t(H_1, H_2)$. It is of size $(O(d^2), O(d^2 + d\tau))$ [2, Prop. 8.46]. For the segments of types IV-V, for every $a \in \mathrm{bd}(I)$, we construct the polynomial $H_a(t)$ by performing $O(d)$ evaluations of polynomials of size $(O(d), \widetilde{O}(d + \tau))$ at a in $\widetilde{O}_B(d(d + \tau))$ [6, Lem. 6]. Then we isolate its roots in $\widetilde{O}_B(d^3 + d^2\tau)$ [32, Thm. 5].

Step 2. We decompose I at the endpoints of all the segments, which are roots of the polynomial P (Eq. (5)). Every endpoint is approximated by an isolating interval, so what it suffices, is that for two consecutive parameters in the sorted list, the respective isolating intervals do not overlap. The polynomial P is of size $(O(d^2), \widetilde{O}(d\tau))$ and we find isolating intervals of its roots in $\widetilde{O}_B(d^6 + d^5\tau)$ [32, Thm. 5]. Every endpoint of an isolating interval is a rational of size $\widetilde{O}(\sigma_i)$, and for all of them the bitsizes sum to $\widetilde{O}(d^3\tau)$.

Step 3. We take a rational inside every interval of the decomposition, say u_i with $i = 1, \ldots, O(d^2)$. The sum of their bitsizes is again in $\widetilde{O}(d^3\tau)$. For every u_i, $\ell_{u_i}(\lambda) \in \mathbb{Q}[\lambda]$ is a polynomial of size $(d, \widetilde{O}(d\sigma_i + \tau))$, so we isolate its roots in $\widetilde{O}_B(d^3\sigma_i + d^2\tau)$ [32, Thm. 5]. In the same bit-complexity we can decide if $\ell_{u_i}(\lambda)$ is non-negative in I. Summing for all i we get a total bit-complexity in $\widetilde{O}_B(d^6\tau)$.

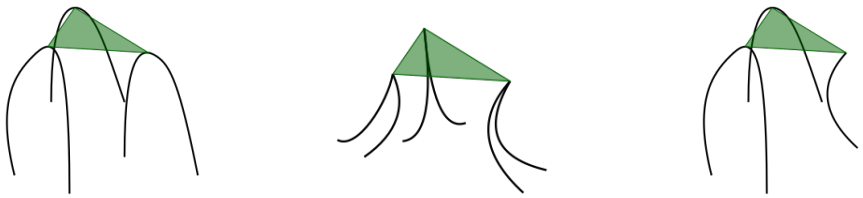
Step 4. To keep only the zero dimensional part of the solutions of the system in Eq. (6) we work as follows: Let $L(s,t,\lambda) = l_D(s,t)\lambda^D + \cdots + l_1(s,t)\lambda + l_0(s,t)$, where $D = O(d)$. For $i = 1, \ldots, D$, we compute $R_i^s(s) = \mathtt{res}_t(l_i(s,t), H(t))$ and $R_i^t(t) = \mathtt{res}_s(l_i(s,t), H(s))$ in $\widetilde{O}_B(d^5\tau)$ [29, Lem. 4]. We compute the gcd of $R_0^s(s), \ldots, R_D^s(s)$ in $\widetilde{O}_B(d^7 + d^6\tau)$ [24, Lem. 2] and then the gcd of the later with $H(s)$ in $\widetilde{O}_B(d^3\tau)$ [5, Lem. 4]. Let $\tilde{h}_s(s)$ the gcd-free part of $H(s)$. It has bitsize $O(d+\tau)$ [5, Lem. 4]. Analogously, we compute the gcd of $R_0^t(t), \ldots, R_D^t(t), H(t)$ and we let $\tilde{h}_t(t)$ be the gcd-free part of the last polynomial. Then, the system $\{\tilde{h}_s(s) = \tilde{h}_t(t) = L(s,t,\lambda) = 0\}$ is zero-dimensional and gives the isolated solutions of the system in Eq. (6). The bit-complexity of isolating its roots is $C(d, d + \tau, 2)$.

Examining the multiplicities of the roots, allows to find the pairs (s_0, t_0) for whom $L(s_0, t_0, \lambda)$ is sign-invariant in I, and therefore decide which segments are on the boundary of the convex hull. For the segments of type V, for all $a \in \mathrm{bd}(I)$, we compute the polynomial $L(a,t,\lambda)$ and then solve the system $\{H(t) = L(a,t,\lambda) = 0\}$. Since the number of boundary points is assumed to be in $O(1)$, this cost is $C(d,\tau,1)$. Step 4 requires also manipulating the output data structure; we perform $O(d^2)$ updates, each one in constant time.

So, it total the bit-complexity of the algorithm is in $\widetilde{O}_B(d^7 + d^6\tau) + C(d, d + \tau, 2)$.

4.2.3. Output of the algorithm

The output is circular doubly linked list $\mathcal{L}$; this data structure consists of a sequence of records, each one corresponding to a facet of the convex hull. Every record contains the data required to describe the facet and a link to a previous and a next record. The data is a parameter interval, say $[t_1, t_2]$, together with a parametrization; the image of the interval over the



(a) (b) (c)

Figure 5: In green (a) a tangent triangle, (b) a cuspidal triangle and (c) an endpoint triangle.

parametrization is an arc of $\phi(I)$ or a segment on the boundary of $\mathrm{conv}(\phi(I))$. The link to the previous record points to the neighboring facet containing $\phi(t_1)$, and the link to the next record points to the neighboring facet containing $\phi(t_2)$. Note that the intervals that are contained in the data of each record are all bounded sub-intervals of I, except from at most one which may be of the form $(-\infty, a] \cup [b, +\infty)$ (Rem. 1).

For the size of the output, we have that $N \in O(d^2)$ by taking into account the degrees of the systems entailed in the computation of segments of the different types and Assumption 4(ii).

Remark 7. *If the curve is smooth, without self-intersections, we have that $N \in O(d)$. [16]; this is since there are $O(d)$ inflection points and poles. If there is a bitangent connecting two smooth points $\phi(t_1)$, $\phi(t_2)$ with $t_1 < t_2$ then in the interval $[t_1, t_2]$, there exists at least one parameter that corresponds to an inflection point or a pole. Moreover, since there are $O(d)$ cusps, the number of Type II and III segments that are on the boundary of the convex hull is also in $O(d)$. The segments of Type IV, V, VI are already in $O(1)$ (Assumption 4(ii)).*

5. Convex hull of parametric curves in $\mathbb{R}^3$

Now $\phi(t) = (\phi_1(t), \phi_2(t), \phi_3(t)) = \left(\frac{p_1(t)}{q_1(t)}, \frac{p_2(t)}{q_2(t)}, \frac{p_3(t)}{q_3(t)}\right)$ is defined as in Eq. (1). We make again the simplifying Assumptions 4 of Sec. 4. The facets of the convex hull are triangles with vertices points on the curve. In addition, it includes one-dimensional families of segments with endpoints on the curve, which are one-dimensional faces. These families assemble to two-dimensional surface patches, to which we refer from now on as facets, with abuse of terminology. The next definition classifies the triangles on the boundary of the convex hull in several types (see Fig. 5 and Rem. 14).

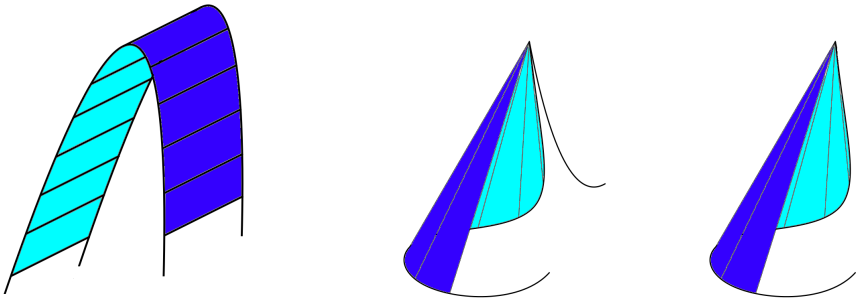
Definition 8. *We distinguish the following types of triangles on the boundary of $\mathrm{conv}(\phi(I))$ according to their vertices:*

I. tangent triangles: there is at least one vertex that is a smooth point of C and the triangle lies on a tangent plane to C at this point. The other two vertices of the triangle are either smooth points, at which the plane is also tangent, or cusps,

II. cuspidal triangles: the three vertices of the triangle are cusps,

III. endpoint triangles: there is at least one vertex of the triangle that is given by a parameter in $bd(I)$.

The one-dimensional families of segments on the boundary of the convex hull form developable surface patches, that can be also classified (see Fig. 6).



(a) (b) (c)

Figure 6: (a) A bitangent surface patch, (b) a cuspidal surface patch, (c) an endpoint surface patch.

Definition 9. *We distinguish the following types of surface patches according to the vertices of the spanning segments:*

I. bitangent surface patches: the segments' endpoints are smooth points of C and the segment lies on the plane tangent to the curve at these two points,

II. cuspidal surface patches: all segments have a common endpoint that is a cusp,

III. endpoint surface patches: all segments have a common endpoint that is given by a parameter in $bd(I)$.

The pairs of parameters that give these type of segments, are points of a plane curve. Let $E(s) := \prod_{a\in \mathrm{bd}(I)}(s-a)$. We define the *bisecant curve* as the zero set of the equation

$$G(s,t) := \frac{\det(\phi(s)-\phi(t),\phi'(s),\phi'(t))\cdot E(s)\cdot E(t)}{(s-t)^4} = 0\,. \tag{7}$$

We divided by $(s-t)^4$ since it is a factor of the numerator. We multiplied by both $E(t)$ and $E(s)$ to preserve the symmetry. A point $(s,t)\in\mathbb{R}^2$ satisfying Eq. (7) is such that:

- $\phi(s)-\phi(t),\phi'(t),\phi'(s)\neq\mathbf{0}$ and there is a bitangent plane to C at $\phi(s)$ and $\phi(t)$, or
- $\phi(s)=\phi(t)$ and/or $\phi'(t)=\mathbf{0}$ and/or $\phi(s)=\mathbf{0}$, i.e., $\phi(t)$ is a multiple point and/or $\phi(t)$ is a cusp and/or $\phi(s)$ is a cusp, or
- $s\in\mathrm{bd}(I)$ or $t\in\mathrm{bd}(I)$.

Now, we consider all the segments $\overline{\phi(s)\phi(t)}$, with s and t satisfying $G(s,t)=0$. This gives rise to the *bisecant surface* in $\mathbb{R}^3$:

$$\mathcal{B} = \Big\{T\,\phi(s)+(1-T)\,\phi(t)\,|\,T\in[0,1], G(s,t)=0\Big\}\,. \tag{8}$$

The bisecant surface is *ruled*, since through every point of the surface there is a straight-line segment that lies on it, and it is also *developable* [3, §5.1]. The surface patches of Def. 9 are part of this surface. Only certain parts of the bisecant surface are on the boundary of the convex hull. We will use the bisecant curve to find and describe these parts.

5.1. Algorithm

Step 1: Computing the triangles. We compute all triangles of types I to III for $\phi(I)$. Each triangle has a type (I to III) and is determined by three parameter values, say a, b and c, such that $\phi(a), \phi(b)$ and $\phi(c)$ are the triangle's vertices. We find the parameters that correspond to each type of triangles by solving a polynomial system and we denote by $\mathcal{S}$ the set of all such triples. We assume that for a given triple of parameters in $\mathcal{S}$ we can determine the type of the corresponding triangle in constant time. To find the different types of triangles, it is easier to factorize G (Eq. (7)) and consider each type separately. We know that cusps of C are given by parameters that are among the roots of $H(s)$ (Eq. (3)). So, in the case where there are cusps, the polynomial G can be factorised as $\hat{H}(s) \cdot \hat{H}(t) \cdot \hat{G}(s,t) \cdot E(s) \cdot E(t)$, where $\hat{H}$ corresponds to the parameters that give cusps. We consider the equations in $\mathbb{Z}(s,t,u)$:

$$\frac{\det(\phi(u) - \phi(t), \phi'(t), \phi'(s))}{(s-t)(t-u)^2} = 0\,, \tag{9}$$

$$\frac{\det(\phi'(s), \phi'(t), \phi'(u))}{(s-t)(t-u)(s-u)} = 0\,. \tag{10}$$

Let $(s,t) \in \mathbb{R}^2$ such that $\hat{G}(s,t) = 0$. Then, $\phi(s)$ and $\phi(t)$ are both smooth points, since we have excluded the factors that correspond to cusps. Let $\mathcal{P}$ be the common tangent plane passing from $\phi(s)$ and $\phi(t)$. Then, Eq. (9) expresses the fact that the plane $\mathcal{P}$ intersects the curve at $\phi(u)$. Eq. (10) expresses the condition that the three tangent vectors, $\phi'(s), \phi'(t)$, and $\phi'(u)$ lie on the same plane and so $\mathcal{P}$ is also tangent to C at $\phi(u)$, if it is smooth. Let $H_1, H_2 \in \mathbb{Z}[s,t,u]$ be the numerators of Eq. (9) and Eq. (10) respectively. From the isolated roots of the system $\{\hat{G} = H_1 = H_2 = 0\}$ we can obtain the tangent triangles with three or two smooth vertices (when $\phi(u)$ is a cusp). We consider the equations in $\mathbb{Z}(s,t,u)$:

$$\frac{\det(\phi(u) - \phi(s), \phi(u) - \phi(t), \phi'(u))}{(t-u)^2(s-u)^2(s-t)} = 0\,,$$

$$\frac{\det(\phi(u) - \phi(s), \phi(u) - \phi(t), \phi'(s))}{(t-u)(s-u)^2(s-t)^2} = 0\,.$$

Let $H_3, H_4 \in \mathbb{Z}[s,t,u]$ be their numerators respectively. For the tangent triangles with one smooth vertex and two cusps, we solve the system:

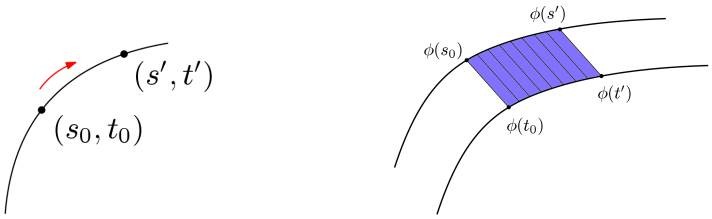
$$\begin{aligned} H(s) &= 0\,, \\ H(t) &= 0\,, \\ H_3(s,t,u) &= 0\,. \end{aligned} \tag{11}$$

The system has also as roots the parameters corresponding to cuspidal triangles, since $\phi(u)$ can be a cusp. For the endpoint triangles, for every $a \in \mathrm{bd}(I)$, to find the planes that go through it, we solve the system

$$\begin{aligned} H_1(s,t,a) &= 0\,, \\ H_4(s,t,a) &= 0\,. \end{aligned} \tag{12}$$

For any $a, b \in \mathrm{bd}(I)$, to find the planes that go through $\phi(a), \phi(b)$, we solve the equation

$$H_4(s,a,b) = 0\,. \tag{13}$$



(a) (b)

Figure 7: (a) An arc of the graph of the bisecant curve G and (b) the surface patch that corresponds to the part of the arc between (s_0, t_0) and (s', t') (in purple).

Step 2: Decomposition of the bisecant curve. We consider the graph of G and we compute a special purpose Cylindrical Algebraic Decomposition (CAD) of the s-axis. In a CAD we decompose the s-axis into a finite number of points and open intervals delimited by these points over which the graph of the bisecant curve G has a cylindrical structure, i.e., the number of branches of the graph of the curve is constant. We call the points defining the decomposition *special values*. Special fibers are the points on the curve above the special values. Regular fibers are the points on the curve above additional points between two special values.

In a CAD the special values are the projections of the s-critical points and the vertical asymptotes. We refine the decomposition by further subdividing the intervals at the projections of the t-critical points and at the parameters of all triples corresponding to the triangles computed in the previous step. Let (s_0, t_0) be a point on the graph of G, such that the line segment connecting $\phi(s_0)$ and $\phi(t_0)$ is on the boundary of the convex hull and is on a bisecant surface patch. If we move along the arc of the graph of G around the point (s_0, t_0) and we take a neighboring point (s', t') that is sufficiently close, then, the segment $\overline{\phi(s')\phi(t')}$ is also on the boundary of the convex hull, part of the same bisecant surface patch (see Fig. 7). In particular, by moving along the arc in both directions starting from (s_0, t_0), we get bitangent segments that are on the bisecant surface patch, and thus, obtain a way to trace it. We continue until we find point (s'', t'') that corresponds to a segment that is on the intersection of the bisecant surface patch with another facet. This point can be such that:

- $\phi(s'')$ and $\phi(t'')$ are vertices of triangle (so, s'' and t'' are both special values of the decomposition), or
- (s'', t'') is a self-intersection point of G, i.e., there are two surface patches that intersect at the segment with endpoints $\phi(s'')$ and $\phi(t'')$. So, in this case s'' is an s-critical point and therefore a special value of the decomposition.

All the previous discussion suggests that an arc of the decomposition of G (that is obtained by the decomposition of the s-axis) corresponds to a surface patch that is either on the boundary of the convex hull or in its interior.

Step 3: Computing the bisecant surface patches. We want to find the arcs of G that correspond to the bisecant surface patches on the boundary of the convex hull. The following lemma gives a criterion. It is a direct consequence of the discussion in Step 2.

Lemma 10. *We consider a smooth and monotonous arc on the graph of G, with $A = (s_1, t_1)$ and $B = (s_2, t_2)$ its endpoints. If there is no point on the arc that corresponds to a bitangent segment belonging on a triangle, then, if any interior point on the arc corresponds to a bitangent segment on the boundary of the convex hull then the entire arc corresponds to a bitangent surface patch on the boundary of the convex hull.*

Over every open interval of the decomposition of the s-axis there are several arcs of the graph of G, corresponding to bitangent surface patches. From Lem. 10, in order to determine if a patch is on the boundary it suffices to take an interior point of the arc and check if the corresponding tangent plane is supporting for $\phi(I)$, using the Support predicate.

For every (open) interval I_j of the decomposition, we work as follows: We pick any rational $u \in I_j$ and we solve $G(u, t) = 0$. For every v such that $G(u, v) = 0$, we have that $\phi(u)$ and $\phi(v)$ are always smooth points on C since the singular points are given by parameters that are among the points of decomposition of the s-axis. We check if the segment connecting $\phi(u)$ and $\phi(v)$ is on the boundary of $\text{conv}(\phi(I))$, which is the case if and only if there is a supporting plane of $\phi(I)$ passing from $\phi(u)$ and $\phi(v)$. But since both points are smooth, the only possible supporting plane is the tangent plane to C at $\phi(u)$ and $\phi(v)$. So we call the predicate Support for this plane. Its equation is $\det((x, y, z) - \phi(u), \phi'(u), \phi'(v)) = 0$. Let $\ell_{u,v}(\lambda) \in \mathbb{Q}[\lambda]$ be the numerator of the tangent line equation after substituting (x, y) with the parametrization $\phi(\lambda)$. We isolate the real roots of $\ell_{u,v}(\lambda)$ in order to determine its sign in I. If the plane is supporting to $\phi(I)$ then the arc of G over I_j, containing the point (u, v), corresponds to a bisecant surface patch that is on the boundary of $\text{conv}(\phi(I))$.

Step 4: Connecting the facets. Having determined the bisecant surface patches on the boundary at the previous step, we now describe how they connect with each other. In other words, we compute the triangles, among the ones found in the first step, that are on the boundary of the convex hull. Two neighboring facets intersect at a line segment with endpoints that are points on the curve. Let $\phi(s)$ and $\phi(t)$ be the endpoints of such a segment. Since this segment is on the boundary, there has to be a supporting plane to the curve containing it. If at least one point is smooth, then the supporting plane can only be the plane tangent to the curve at the smooth point and passing from both. If none of the points is smooth, then the supporting plane is not uniquely defined. So, for every arc of the graph of G corresponding to a bisecant surface patch on the boundary, we check each of its endpoints. Let (s, t) be one of them.

1. If at least one of $\phi(s)$ and $\phi(t)$ is a smooth point, say $\phi(s)$, then the plane that is tangent to C at $\phi(s)$ and passes from $\phi(t)$ is uniquely defined. This plane passes from a third point $\phi(u)$ on C, that was found in Step 1. So, there is only one possible plane facet (if there are no degeneracies) that is neighboring, and this is the triangle with vertices $\phi(s)$, $\phi(t)$ and $\phi(u)$. In a degenerate situation, there will be more than one u, corresponding to the multiple interections of the plane with the curve, and the different triangles that belong on the same plane will be considered multiple times (see Rem. 13).
2. If both $\phi(s)$ and $\phi(t)$ are cusps or endpoints of $\phi(I)$, then we need to check every possible combination in order to find the triangle containing this segment that is on boundary.
   For the cuspidal triangles, let $(s, t, u) \in \mathbb{R}^3$ be a triple of parameters corresponding to cusps. Then, $\langle(\phi(s) - \phi(u)) \times (\phi(s) - \phi(t)), \phi(s) - (x, y, z)\rangle = 0$ is the implicit equation of the plane in $\mathbb{R}^3$ that goes through $\phi(s), \phi(t)$ and $\phi(u)$. We substitute (x, y, z) with $\phi(\lambda)$ in the previous equation. Let $L(s, t, u, \lambda) \in \mathbb{Z}[s, t, u, \lambda]$ be the polynomial obtained after clearing

the denominators. Calling the predicate Support for all these planes, amounts to isolating the roots of the system

$$\begin{aligned} H(s) &= 0\,, \\ H(t) &= 0\,, \\ H(u) &= 0\,, \\ L(s,t,u,\lambda) &= 0\,. \end{aligned} \tag{14}$$

For the triangles with non-smooth vertices that involve an endpoint, we distinguish three cases for the system that we solve:

- If one vertex is an endpoint and the other two cusps, the system is of the form $\{H(s) = H(t) = L(s,t,a,\lambda) = 0\}$, where $a \in \mathrm{bd}(I)$,
- If two vertices are endpoints and the third one is a cusp, the system is of the form $\{H(s) = L(s,a,b,\lambda) = 0\}$, where $a, b \in \mathrm{bd}(I)$,
- If all the vertices are endpoints, then we just have to solve the univariate equation $L(a,b,c,\lambda) = 0$, where $a, b, c \in \mathrm{bd}(I)$.

So, for a surface patch that corresponds to an arc of the graph of G with endpoints (s_1, t_1) and (s_2, t_2), the neighboring facets from each side can be determined.

Complexity analysis. For the bit-complexity analysis of the algorithm we use $C(M, \Lambda, N)$ (Def. 3) to express it. The proof is given in detail in the next subsection.

Theorem 11. *Let C be a curve in $\mathbb{R}^3$ with a proper parametrization $\phi(t)$ as in Eq. (1), of size (d, τ). Let $I \subset \mathbb{R}$ such that $\phi(I)$ is compact in $\mathbb{R}^3$. There is an algorithm that computes the boundary of the convex hull of $\phi(I)$ in*

$$\widetilde{O}_B\left(d^{10} + d^9\tau\right) + C(d, d^6\tau, 1) + C(d, d+\tau, 3)\,.$$

The output of the algorithm is a doubly connected linked list describing the facets of conv$(\phi(I))$, which are $O(d^3)$.

Corollary 12. *Using the bit-complexity results of [23] for $C(d, d+\tau, 3)$ and [14] for $C(d, d^6\tau, 1)$, the bit-complexity of the previous theorem becomes $\widetilde{O}_B(d^{13} + d^{12}\tau)$.*

5.2. Proof of Theorem 11

5.2.1. Correctness

The correctness of the algorithm is based on Lem. 10 and the fact that the Support predicate correctly detects the bisecant surface patches that are on the boundary. On connecting the bisecant surface patches with triangles, we refer to the selection criterion of the gift-wrapping algorithm for the set of points $\phi(I)$. Given a segment that is on the boundary of the convex hull (and also on the boundary of a surface patch), the next point with whom it connects, is a point $q \in \phi(I)$ such that the plane that contains the segment and passes from q is strictly supporting for the convex hull. The connection phase in Step 4 respects this criterion.

Remark 13 (Degeneracies)**.** *In a degenerate situation, several triangles on the boundary of the convex hull can lie on the same supporting plane (e.g., Fig. 11) or several surface patches can be included in a bigger surface patch. Consequently, each boundary element will be accounted for multiple times. However, this problem can be resolved during a post-processing stage, where the systems' solutions from the first step are analyzed to identify such situations.*

Remark 14. *If we consider all the possible combinations of the three vertices of a triangle (smooth point, cusp, endpoint), we see that there exist ten different configurations. Among them, three correspond to tangent triangles, one to cuspidal triangle and six to endpoint triangles. We divide them in the three categories of Def. 8 due to the following reasons:*

- *To define uniquely a plane that passes from a smooth point of C and is tangent to the curve at this point, we need to specify another point belonging on the plane. Therefore, the plane in which the tangent triangle belongs is not affected by the smoothness (or not) of the two other vertices.*
- *The cuspidal and endpoint triangles can be obtained as solutions of a polynomial system of special structure, which is simpler comparing to the system corresponding to tangent triangles.*

5.2.2. Bit-complexity

Step 1. We construct the polynomials G, H_1, H_2, H_3, H_4 in $\widetilde{O}_B(d^3\tau)$ [44, Ch. 8]. We factorize G as $\hat{H}(s) \cdot \hat{H}(t) \cdot \hat{G}(s,t) \cdot E(s) \cdot E(t)$ in $\widetilde{O}_B(d^4 + d^3\tau)$ [14, Prop. 21], by considering $G(s,t)$ as a polynomial in s or in t and then finding the gcd of the coefficients.

To find the triangles with two or three smooth vertices we need to find the isolated roots of the system $\{\hat{G}(s,t) = H_1(s,t,u) = H_2(s,t,u) = 0\}$. In particular, we are only interested in the first two coordinates of the roots; if (a,b,c) is a solution to the system and $\phi(u)$ is smooth, then all the possible permutations of a, b, c are solutions as well. So, we can obtain the triples corresponding to tangent planes with smooth vertices just by the (s,t)-projections. We take the resultant $R_1(s,t) := \mathtt{res}_u(H_1, H_2) \in \mathbb{Z}[s,t]$. It is a polynomial of degree $O(d^2)$ in each variable and bitsize in $\widetilde{O}(d\tau)$ [2, Prop. 8.72]. It is computed in $\widetilde{O}_B(d^7\tau)$. Then, we consider the system $\{\hat{G} = R_1 = 0\}$. We take the resultant $R_2(s) := \mathtt{res}_t(\hat{G}, R_1) \in \mathbb{Z}[s]$. It is of size $\left(O(d^3), \widetilde{O}(d^2\tau)\right)$ [26, Cor. 5] and is computed in $\widetilde{O}_B(d^8\tau)$ [26, Lem. 6]. At last, we isolate the roots of $\{R_2 = \hat{G} = 0\}$ in $\widetilde{O}_B(d^9 + d^8\tau)$ [26] (if the system is not zero-dimensional we first compute the gcd and the gcd-free parts).

Tangent triangles with two smooth vertices can be found by computing the isolated roots of the system $\{\hat{G}(s,t) = H(u) = H_1(s,t,u) = 0\}$. In particular, we are only interested this time in the last two coordinates of the roots; if (a,b,c) is a solution to the system and $\phi(a), \phi(b)$ are smooth, then (b,a,c) is a solution as well. Thus, we can group the triples corresponding to the same triangle. The resultant $R_3(s,u) := \mathtt{res}_t(\hat{G}, H_1) \in \mathbb{Z}[s,u]$ is of size $(O(d^2), \widetilde{O}(d\tau))$ and is computed in $\widetilde{O}_B(d^7\tau)$. We isolate the roots of $\{H(u) = R_3(s,u) = 0\}$ in $\widetilde{O}_B(d^8 + d^7\tau)$.

For the triangles with one smooth point and the cuspidal triangles, we find the isolated roots of the system of Eq. (11) in $\widetilde{O}_B(d^7 + d^6\tau)$ in the same way as for the system in Eq. (6).

At last, for every $a \in \mathrm{bd}(I)$, we compute $H_1(s,t,a)$ and $H_4(s,t,a)$ by performing $O(d^2)$ evaluations at a; this costs $\widetilde{O}_B(d^3\tau)$ and the resulting polynomials have bitsize in $\widetilde{O}(d+\tau)$ [6, Lem.7]. Then, we solve the bivariate systems of Eq. (12) in $\widetilde{O}_B(d^6 + d^5\tau)$. For all $a, b \in \mathrm{bd}(I)$, we

compute $H_4(s,a,b)$ (Eq. (13)) in $\widetilde{O}_B(d^3\tau)$ [6, Lem.7]. These polynomials have bitsize in $\widetilde{O}(d+\tau)$ and so we isolate the roots in $\widetilde{O}_B(d^3+d^2\tau)$ [32, Thm. 5].

Step 2. The decomposition points of the s-axis are:

- s-projections of critical points of G, and thus roots of $\mathtt{res}_t(G, \partial G/\partial t)$. The graph of G is symmetric with respect to the line $s=t$, so we do not consider also $\mathtt{res}_t(G, \partial G/\partial s)$,
- The s-values where vertical asymptotes occur. These values are roots of the leading coefficient of $G(s,t)$, when considered as a polynomial in t,
- parameters corresponding to a triangle, and thus roots of

$$H(s)\cdot R_2(s)\prod_{a\in \mathrm{bd}(I)}\left((s-a)\cdot \mathtt{res}_t(H_1(s,t,a),H_4(s,t,a))\prod_{b\in \mathrm{bd}(I)}H_4(s,a,b)\right).$$

The product of these polynomials is of size $(O(d^3),\widetilde{O}(d^2\tau))$. We isolate its roots in $\widetilde{O}_B(d^9+d^8\tau)$ [32, Thm. 5]. Every endpoint of an isolating interval is a rational of size $\widetilde{O}(\sigma_i)$, and for all of them the bitsizes sum to $\widetilde{O}(d^5\tau)$.

Step 3. We take a rational point q_i inside every interval of the decomposition (e.g. the midpoint); it has bitsize $\widetilde{O}(\sigma_i)$. We construct the polynomial $G(q_i,t)$ by performing $O(d)$ evaluations of univariate polynomials at q_i. Then, calling the predicate Support for every intersection point on the graph of G, amounts to isolating the roots of the system

$$G(q_i,t)=0$$
$$H_1(t,q_i,\lambda)=0$$

This costs $C(d,d\sigma_i,1)$. Summing for all i, since C is linear in the bitsize, we obtain $C(d,d^6\tau,1)$.

Step 4. To keep only the zero dimensional part of the solutions of the system in Eq. (14) we work as follows: Let $L(s,t,u,\lambda)=l_D(s,t,u)\lambda^D+\cdots+l_1(s,t,u)\lambda+l_0(s,t,u)$, where $D=O(d)$. For $i=1,\ldots,D$, we compute

$$R_i^s(s)=\mathtt{res}_u(\mathtt{res}_t(l_i(s,t,u),H(t)),H(u))\in\mathbb{Z}[s]\,,$$
$$R_i^t(t)=\mathtt{res}_u(\mathtt{res}_s(l_i(s,t,u),H(s)),H(u))\in\mathbb{Z}[t]\,,$$
$$R_i^u(u)=\mathtt{res}_t(\mathtt{res}_s(l_i(s,t,u),H(s)),H(t))\in\mathbb{Z}[u]\,.$$

This is done in $\widetilde{O}_B(d^8\tau)$ [26, Lem. 6]. by successive resultant computations. They are polynomials of size $\left(O(d^3),\widetilde{O}(d^2\tau)\right)$ [2, Prop.8.72]. For $v=s,t,u$, we compute the gcd of $R_0^v(v),\ldots,R_D^v(v)$ in $\widetilde{O}_B(d^{10}+d^9\tau)$ [24, Lem. 2] and then the gcd of the later with $H(v)$ in $\widetilde{O}_B(d^6\tau)$ [5, Lem. 4]. Let $\tilde{h}_v(v)$ the gcd-free part of $H(v)$. It has bitsize $O(d+\tau)$ [5, Lem. 4]. Then, the system $\{\tilde{h}_s(s)=\tilde{h}_t(t)=\tilde{h}_u(u)=L(s,t,u,\lambda)=0\}$ is zero-dimensional and gives the isolated solutions of the system in Eq. (14). The bit-complexity of isolating its roots is $C(d,d+\tau,3)$. Computing the isolated roots of the other systems in this step is dominated by the previous computations.

So, the bit-complexity of the algorithm is in

$$\widetilde{O}_B\left(d^{10}+d^9\tau\right)+C(d,d^6\tau,1)+C(d,d+\tau,3)\,.$$

5.2.3. Output of the algorithm

We will use the half-edge data structure, or doubly connected edge list (DCEL), to represent the output [12, Ch.2.2]. This data structure is used to represent an embedding of a planar graph in the plane by maintaining the following records: vertices, edges and faces. By extension, it is widely used to describe the boundary of three-dimensional convex polyhedra and is also convenient to describe the boundary of the convex hull of $\phi(I)$. In our case, a face is a 2-dimensional facet of the convex hull, that can be either a triangle or a bisecant surface patch. An edge is intersection of two facets, i.e, a parametric arc or a segment connecting two points of the curve. A vertex is the intersection of two edges of the convex hull, i.e., a point on the curve.

The principle of this data structure is to 'decompose' every edge into two half-edges with opposite directions. All the records are associated to a half-edge and this allows to encode all the combinatorial information of the planar graph, or of the boundary of the convex hull in the present case. In particular:

- a vertex is associated to one (arbitrary) halfedge with the vertex as starting point,
- an edge is associated to one halfedge,
- a face is associated to some halfedge on its boundary,
- a halfedge is associated to the vertex that is its origin, to a "twin" halfedge, that is the halfedge with the opposite direction, and to the face that is incident to the edge on the left side, when we traverse it from the origin to its endpoint.

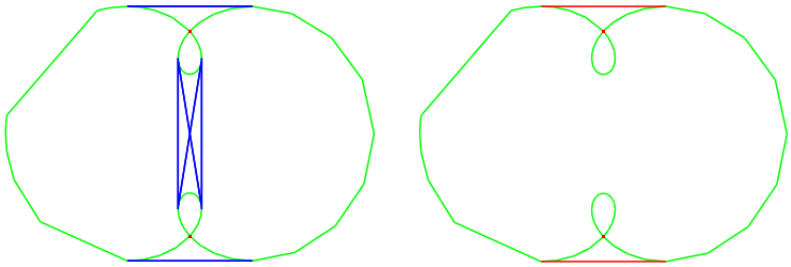
Although it is a combinatorial data structure, we can also store geometrical information on the records, by giving them extra attributes:

- For a vertex, since it a point on the curve, we store the corresponding parameter,
- For an edge, if it is a parametric arc we store the corresponding parameter interval. If it is a bitangent segment, it is described by the two parameters that correspond to the endpoints of the segment,
- A face can be either a triangle or a bisecant surface patch. In the first case, it is described by the parameters of the incident vertices. In the second case, it is described by the reference to its corresponding arc of the graph of G. An arc is described by its endpoints, its projection on the s-axis and its ordering with respect to other branches over the interval.

For the number of the facets on the boundary of the convex hull, we consider the degrees of the systems entailed in the computation of triangles and Assumption 4(ii).

6. Implementation and Examples

We provide some examples using our prototype implementation in Maple. It is built upon the real root isolation routines of Maple's RootFinding library and the `PTOPO` package [25], to compute the topology and visualize parametric curves in two and three dimensions.



(a) (b)

Figure 8: (a) The curve of Example 15 (in green) and the segments computed at the first step of the algorithm (in blue). (b) The segments on the boundary of the convex hull (in red).

Example 15. We consider a curve in $\mathbb{R}^2$ parametrized by

$$\phi(t) = \left(-\frac{2\left(12t^2+7t-12\right)\left(1679t^4+2688t^3-5074t^2-2688t+1679\right)}{15625\left(t^2+1\right)^3}, \right.$$
$$\left. \frac{(t-7)(7t+1)\left(2929t^4+2688t^3-2574t^2-2688t+2929\right)}{15625\left(t^2+1\right)^3} \right).$$

The parametrization is proper. We take $I = \mathbb{R}$. We compute the polynomials

$$\begin{aligned} H_1(s,t) =& 128092t^4s^4 + 86016t^3s^4 + 43008t^4s^3 - 159912t^2s^4 + 292544t^3s^3 - 25000t^4s^2 - \\ &- 43008t\,s^4 + 43008t^2s^3 + 129024t^3s^2 + 34364s^4 - 187456t\,s^3 + 265264t^2s^2 - \\ &- 187456t^3s + 34364t^4 - 129024t\,s^2 - 43008t^2s + 43008t^3 - 25000s^2 + \\ &+ 292544st - 159912t^2 - 43008s - 86016t + 128092\,, \\ H_2(s,t) =& -128092t^4s^4 - 43008t^3s^4 - 86016t^4s^3 + 25000t^2s^4 - 292544t^3s^3 + 159912t^4s^2 - \\ &- 129024t^2s^3 - 43008t^3s^2 + 43008t^4s - 34364s^4 + 187456t\,s^3 - 265264t^2s^2 + \\ &+ 187456t^3s - 34364t^4 - 43008s^3 + 43008t\,s^2 + 129024t^2s + 159912s^2 - \\ &- 292544st + 25000t^2 + 86016s + 43008t - 128092\,. \end{aligned}$$

We isolate the real roots of the system $\{H_1 = H_2 = 0\}$; they correspond to the blue segments in Fig. 8(a). Then $\mathbb{R}$ is decomposed at the s-projections of the roots (Fig. 9). The segments on the boundary of the convex hull are shown in red in Fig. 8(b) and the parameter intervals corresponding to the arcs on the boundary of $\mathrm{conv}(\phi(\mathbb{R}))$ are annotated in orange in Fig. 9.

Figure 9: Decomposition of the parameter interval in Example 15.

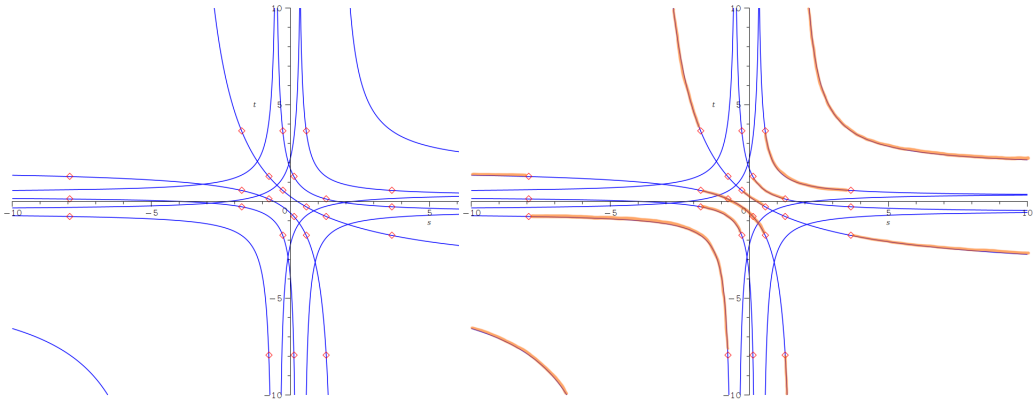
Example 16. We consider a curve in $\mathbb{R}^3$ parametrized by

$$\phi(t) = \left(\frac{-(t+3)(3t-1)}{5(t^2+1)}, \frac{2(t-2)(2t+1)}{5(t^2+1)}, \frac{8(t-2)(2t+1)(t+3)(3t-1)(7t^2+2t-7)(t^2-14t-1)}{625(t^2+1)^4} \right).$$

The parametrization is proper. We take $I = \mathbb{R}$. The bisecant curve is defined by:

$$\begin{aligned} G(s,t) =& 64\,(5s^2t^2 - s^2 + 12st - t^2 + 5)(17s^2t^2 + 62s^2t + 62st^2 - 17s^2 - 68st - 17t^2 - 62s - \\ & - 62t + 17)\cdot(31s^2t^2 - 34s^2t - 34st^2 - 31s^2 - 124st - 31t^2 + 34s + 34t + 31)\,. \end{aligned}$$

The curve has neither cusps nor endpoints. We compute H_1, H_2 and $R_1 = \mathtt{res}_u(H_1, H_2)$ and we solve the system $\{G = R_1 = 0\}$. There are 96 real roots. For simplicity, in Fig. 10(a) we show only 24 of them on the graph of G; they correspond to the boundary segments of the triangle facets on the boundary of $\mathrm{conv}(\phi(\mathbb{R}))$. Then, for the second step of the algorithm, we decompose the s-axis at the s-projections of these roots and of the critical points of G and at the values where G has a vertical asymptote; there are 46 decomposition points in total. The branches of the bisecant curve that correspond to bitangent surface patches on the boundary are annotated in Fig. 10(b). By sampling points on the graph of the bisecant curve, we construct the bisecant surface in Fig. 11(a). At last, again by sampling but only on the annotated branches of the bisecant curve we construct the convex hull in Fig. 11(b). It consists of 8 surface patches and triangle facets that assemble to two squares.



(a) (b)

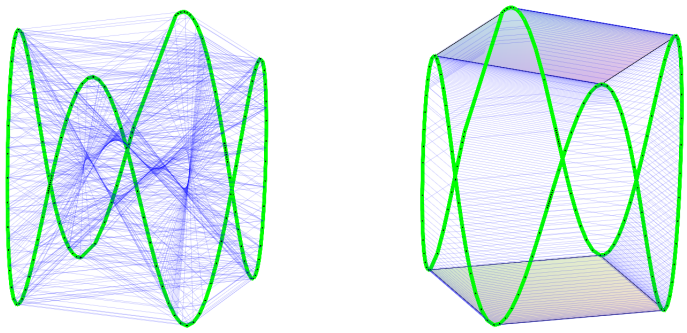
Figure 10: (a) The graph of the bisecant curve of Example 16. (b) The arcs of the bisecant curve that correspond to bisecant surface patches are shown in orange.

Example 17. We consider the curve in $\mathbb{R}^3$ parametrized by

$$\phi(t) = \left(-\frac{3t^2-1}{t^2+1}, -\frac{t\,(3t^2-1)}{(t^2+1)^2}, -\frac{(t^6-1)(3t^2-1)}{(t^2+1)^4}\right).$$

The parametrization is proper. We take $I = \mathbb{R}$. The bitangent curve is defined by:

$$\begin{aligned} G(s,t) =& -8(s+t)(63s^7t^7 + 81s^7t^5 - 33s^6t^6 + 81s^5t^7 + 45s^7t^3 - 159s^6t^4 - 201s^5t^5 - \\ & - 159s^4t^6 + 45s^3t^7 + 27s^7t + 13s^6t^2 + 195s^5t^3 + 295s^4t^4 + 195s^3t^5 + 13s^2t^6 + \\ & + 27s\,t^7 + 11s^6 + 93s^5t + 211s^4t^2 + 375s^3t^3 + 211s^2t^4 + 93s\,t^5 + 11t^6 + 13s^4 - \\ & - 31s^3t - 169s^2t^2 - 31s\,t^3 + 13t^4 + 17s^2 + 31st + 17t^2 + 15)\,. \end{aligned}$$



(a) (b)

Figure 11: (a) The bisecant surface and (b) the convex hull of the curve of Example 16.

The graph of G is shown in Fig. 12(a). There are no triangle facets on the boundary of the convex hull. The convex hull consists of two surface patches (Fig. 12(b)); the red one corresponds to the factor $(s + t)$ of G, and the blue one corresponds to the other factor.

References

[1] Chandrajit Bajaj and Myung-Soo Kim. Convex hulls of objects bounded by algebraic curves. *Algorithmica*, 6:533–553, 06 1991. https://doi.org/10.1007/BF01759058 doi:10.1007/BF01759058.

[2] S. Basu, R. Pollack, and M-F. Roy. *Algorithms in Real Algebraic Geometry*, volume 10 of *Algorithms and Computation in Mathematics*. Springer-Verlag, 2006.

[3] Paul A. Blaga. *Lectures on the Differential Geometry of Curves and Surfaces*. Napoca Press, Cluj-Napoca, Romania, 2005.

[4] Grigoriy Blekherman, Pablo A. Parrilo, and Rekha R. Thomas. *Semidefinite Optimization and Convex Algebraic Geometry*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2012. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611972290, https://doi.org/10.1137/1.9781611972290 doi:10.1137/1.9781611972290.

[5] Yacine Bouzidi, Sylvain Lazard, Guillaume Moroz, Marc Pouget, Fabrice Rouillier, and Michael Sagraloff. Improved algorithms for solving bivariate systems via rational univariate representations. *Research report, Inria*, 2015.

[6] Yacine Bouzidi, Sylvain Lazard, Marc Pouget, and Fabrice Rouillier. Separating linear forms and rational univariate representations of bivariate systems. *J. Symb. Comput.*, pages 84–119, 2015. URL: https://hal.inria.fr/hal-00977671.

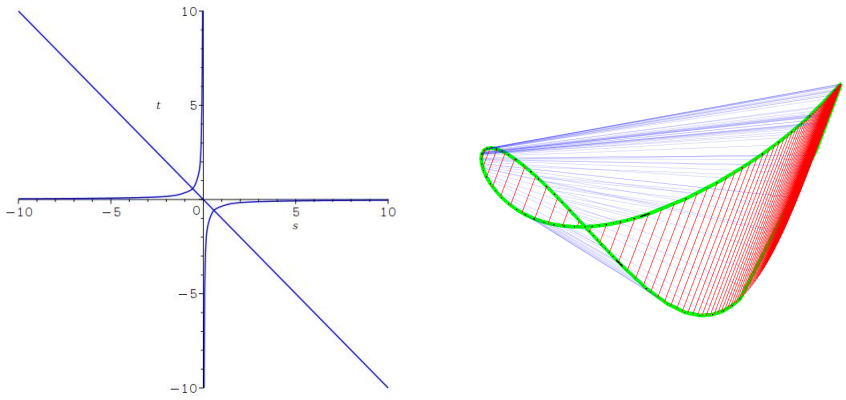
[7] Cornelius Brand and Michael Sagraloff. On the Complexity of Solving Zero-Dimensional Polynomial Systems via Projection. In *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation - ISSAC '16*, pages 151–158, Waterloo, ON, Canada, 2016. ACM Press. URL: http://dl.acm.org/citation.cfm?doid=2930889.2930934, https://doi.org/10.1145/2930889.2930934 doi:10.1145/2930889.2930934.

[8] Arne Brøndsted. *An Introduction to Convex Polytopes*. Graduate Texts in Mathematics. Springer, 1983.

[9] Jin-San Cheng, Kai Jin, Marc Pouget, Junyi Wen, and Bingwei Zhang. An Improved Complexity Bound for Computing the Topology of a Real Algebraic Space Curve. working paper or preprint, December 2021. URL: https://hal.inria.fr/hal-03469996.

[10] Daniel Ciripoi, Nidhi Kaihnsa, Andreas Löhne, and Bernd Sturmfels. Computing convex hulls of trajectories. *arXiv*, abs/1810.03547, 2018.

[11] Felipe Cucker and Steve Smale. On the mathematical foundations of learning. *BULLETIN*, 39, 11 2001. https://doi.org/10.1090/S0273-0979-01-00923-5 doi:10.1090/S0273-0979-01-00923-5.



(a) (b)

Figure 12: (a) The bisecant curve and (b) the convex hull of the curve in Example 17

[12] Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition*. Springer, 2008. URL: https://www.worldcat.org/oclc/227584184.

[13] Rui JP de Figueiredo and Hemant D Tagare. Curves and surfaces in computer vision. In *Curves and Surfaces in Computer Vision and Graphics*, volume 1251, pages 10–16. SPIE, 1990.

[14] Daouda Niang Diatta, Sény Diatta, Fabrice Rouillier, Marie-Françoise Roy, and Michael Sagraloff. Bounds for polynomials on algebraic numbers and application to curve topology. *Discrete & Computational Geometry*, Feb 2022. https://doi.org/10.1007/s00454-021-00353-w doi:10.1007/s00454-021-00353-w.

[15] David P. Dobkin and Diane L. Souvaine. Computational geometry in a curved world. *Algorithmica*, 5:421–457, 1990.

[16] Gershon Elber, Myung-Soo Kim, and Hee-Seok Heo. The convex hull of rational plane curves. *Graphical Models*, 63:151–162, 05 2001. https://doi.org/10.1006/gmod.2001.0546 doi:10.1006/gmod.2001.0546.

[17] M. Elkadi, B. Mourrain, and R. Piene. *Algebraic Geometry and Geometric Modeling*. Mathematics and Visualization. Springer Berlin Heidelberg, 2010. URL: https://books.google.fr/books?id=5yJ7cgAACAAJ.

[18] Laureano González-Vega, Ioana Necula, Sonia Pérez-Díaz, Juana Sendra, and Juan Sendra. Algebraic methods in computer aided geometric design: Theoretical and practical applications. *Geometric Computation*, 11, 03 2004. https://doi.org/10.1142/9789812794833$_0$001*doi* : 10.1142/9789812794833$_0$001.

[19] Didier Henrion. Semidefinite representation of convex hulls of rational varieties. *Acta applicandae mathematicae*, 115(3):319–327, 2011.

[20] Yan-Bin Jia and Huan Lin. On the convex hulls of parametric plane curves. *Technical report, www.cs.iastate.edu/jia*, 2005.

[21] J. K. Johnstone. A parametric solution to common tangents. In *Proceedings International Conference on Shape Modeling and Applications*, pages 240–249, May 2001. https://doi.org/10.1109/SMA.2001.923395 doi:10.1109/SMA.2001.923395.

[22] J. K. Johnstone. Giftwrapping a curve with the convex hull. In *Proceedings of the 42nd Annual Southeast Regional Conference*, ACM-SE 42, page 224–227, New York, NY, USA, 2004. Association for Computing Machinery. https://doi.org/10.1145/986537.986590 doi:10.1145/986537.986590.

[23] Christina Katsamaki and Fabrice Rouillier. On isolating roots in a multiple field extension. In *Proceedings of the 2023 International Symposium on Symbolic and Algebraic Computation*, ISSAC '23, page 363–371, New York, NY, USA, 2023. Association for Computing Machinery. https://doi.org/10.1145/3597066.3597107 doi:10.1145/3597066.3597107.

[24] Christina Katsamaki, Fabrice Rouillier, Elias Tsigaridas, and Zafeirakis Zafeirakopoulos. On the geometry and the topology of parametric curves. In *Proceedings of the 45th International Symposium on Symbolic and Algebraic Computation*, ISSAC '20, page 281–288, New York, NY, USA, 2020. Association for Computing Machinery. https://doi.org/10.1145/3373207.3404062 doi:10.1145/3373207.3404062.

[25] Christina Katsamaki, Fabrice Rouillier, Elias P. Tsigaridas, and Zafeirakis Zafeirakopoulos. PTOPO: A Maple

package for the topology of parametric curves. *ACM Commun. Comput. Algebra*, 54(2):49–52, 2020.

[26] Alexander Kobel and Michael Sagraloff. On the complexity of computing with planar algebraic curves. *Journal of Complexity*, 31(2):206–236, 2015. URL: https://www.sciencedirect.com/science/article/pii/S0885064X1400082X, https://doi.org/10.1016/j.jco.2014.08.002 doi:10.1016/j.jco.2014.08.002.

[27] Aleksei Kurbatskiĭ. Convex hulls of a curve in control theory. *Sbornik Mathematics - SB MATH*, 203:406–423, 03 2012. https://doi.org/10.1070/SM2012v203n03ABEH004228 doi:10.1070/SM2012v203n03ABEH004228.

[28] Jean B Lasserre. Convexity in semialgebraic geometry and polynomial optimization. *SIAM Journal on Optimization*, 19(4):1995–2014, 2009.

[29] Sylvain Lazard, Marc Pouget, and Fabrice Rouillier. Bivariate triangular decompositions in the presence of asymptotes. *Journal of Symbolic Computation*, 82:123–133, 2017. URL: https://www.sciencedirect.com/science/article/pii/S0747717117300111, https://doi.org/10.1016/j.jsc.2017.01.004 doi:10.1016/j.jsc.2017.01.004.

[30] In-Kwon Lee and Myung-Soo Kim. Primitive geometric operations on planar algebraic curves with gaussian approximation. In Tosiyasu L. Kunii, editor, *Visual Computing*, pages 449–468, Tokyo, 1992. Springer Japan.

[31] Dinesh Manocha and John F. Canny. Detecting cusps and inflection points in curves. *CAGD*, 9(1):1 – 24, 1992. URL: http://www.sciencedirect.com/science/article/pii/016783969290050Y, https://doi.org/10.1016/0167-8396(92)90050-Y doi:10.1016/0167-8396(92)90050-Y.

[32] Kurt Mehlhorn, Michael Sagraloff, and Pengming Wang. From approximate factorization to root isolation with application to cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 66:34 – 69, 2015. URL: http://www.sciencedirect.com/science/article/pii/S0747717114000200, https://doi.org/10.1016/j.jsc.2014.02.001 doi:10.1016/j.jsc.2014.02.001.

[33] Laxmi Parida and SP Mudur. Common tangents to planar parametric curves: a geometric solution. *Computer-Aided Design*, 27(1):41–47, 1995. URL: https://www.sciencedirect.com/science/article/pii/001044859590751Z, https://doi.org/10.1016/0010-4485(95)90751-Z doi:10.1016/0010-4485(95)90751-Z.

[34] Sonia Pérez-Díaz. On the problem of proper reparametrization for rational curves and surfaces. *CAGD*, 23(4):307–323, 2006.

[35] Kristian Ranestad and Bernd Sturmfels. The convex hull of a space curve. *Advances in Geometry*, 12, 12 2009. https://doi.org/10.1515/advgeom.2011.021 doi:10.1515/advgeom.2011.021.

[36] Kristian Ranestad and Bernd Sturmfels. The convex hull of a variety. *Notions of Positivity and the Geometry of Polynomials*, pages 331–344, 2011.

[37] Alejandro Schaffer and Christopher J Van Wyk. Convex hulls of piecewise-smooth jordan curves. *Journal of Algorithms*, 8:66–94, 03 1987. https://doi.org/10.1016/0196-6774(87)90028-9 doi:10.1016/0196-6774(87)90028-9.

[38] V. Sedykh. Structure of the convex hull of a space curve. *Journal of Mathematical Sciences*, 33:1140–1153, 05 1986. https://doi.org/10.1007/BF01086114 doi:10.1007/BF01086114.

[39] J Rafael Sendra, Franz Winkler, and Sonia Pérez-Díaz. Rational algebraic curves. *Algorithms and Computation in Mathematics*, 22, 2008.

[40] J.-K Seong, Gershon Elber, J. Johnstone, and Myung-Soo Kim. The convex hull of freeform surfaces. *Computing*, 72:171–183, 04 2004. https://doi.org/10.1007/s00607-003-0055-x doi:10.1007/s00607-003-0055-x.

[41] Suvrit Sra, Sebastian Nowozin, and Stephen J. Wright. *Optimization for Machine Learning*. The MIT Press, 2011.

[42] Guillaume Staerman, Pavlo Mozharovskyi, and S. Clémençon. The area of the convex hull of sampled curves: a robust functional statistical depth measure. In *AISTATS*, 2020.

[43] Cynthia Leslie Vinzant. *Real algebraic geometry in convex optimization*. University of California, Berkeley, 2011.

[44] Joachim von zur Gathen and Jürgen Gerhard. *Modern computer algebra*. Cambridge University Press, 3rd edition, 2013.

[45] Y Yang, Y.-C Liu, M.-Y Liu, and M.-Y Fu. A path planning algorithm based on convex hull for autonomous service robot. 31:54–58+63, 01 2011.

[46] F Zhou, Baoye Song, and Guohui Tian. Bézier curve based smooth path planning for mobile robot. *Journal of Information and Computational Science*, 8:2441–2450, 12 2011.