Proximity Queries for Absolutely Continuous Parametric Curves

Arun Lakshmanan†, Andrew Patterson†, Venanzio Cichella‡ and Naira Hovakimyan†
†University of Illinois at Urbana-Champaign, ‡University of Iowa
lakshma2@illinois.edu, appatte2@illinois.edu, venanzio-cichella@uiowa.edu, nhovakim@illinois.edu

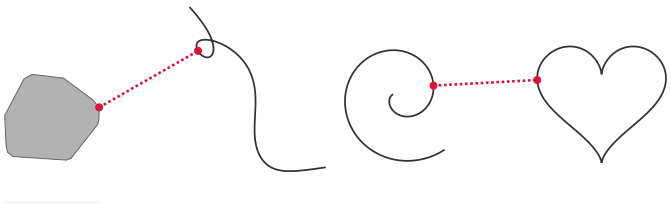
***Abstract*—In motion planning problems for autonomous robots, such as self-driving cars, the robot must ensure that its planned path is not in close proximity to obstacles in the environment. However, the problem of evaluating the proximity is generally non-convex and serves as a significant computational bottleneck for motion planning algorithms. In this paper, we present methods for a general class of absolutely continuous parametric curves to compute: (i) the minimum separating distance, (ii) tolerance verification, and (iii) collision detection. Our methods efficiently compute bounds on obstacle proximity by bounding the curve in a convex region. This bound is based on an upper bound on the curve arc length that can be expressed in closed form for a useful class of parametric curves including curves with trigonometric or polynomial bases. We demonstrate the computational efficiency and accuracy of our approach through numerical simulations[1] of several proximity problems.**

I. Introduction

Autonomous robots often operate in rapidly changing environments, and the ability to accurately and quickly predict future collisions is crucial for safely meeting task objectives. As proximity queries serve one of the major bottlenecks in motion planning frameworks [18], algorithms that efficiently assess the proximity of obstacles relative to the future states of the robot will greatly improve the run-time performance of the robot while guaranteeing safe operating procedures.

Fast methods for computing proximity queries between polyhedral objects have been widely studied and developed over the past few decades. The algorithms described in [13, 22] perform several different types of proximity queries between convex polytopes. Proximity between general polyhedral objects using hierarchical representations of convex bounding volumes are investigated in [21] and [14]. In [9], the authors evaluate the proximity queries for polyhedral objects using convex surface decomposition.

It is challenging to assess the proximity between objects in motion. Recent work has focused on developing methods for computing continuous collision detection (CCD), a type of query that evaluates the first time of contact between objects in motion. The most practical of these methods is known as conservative advancement, introduced in [39] and [33], and performs several static proximity queries between polyhedral or polygon-soup objects to accurately perform CCD queries. Conservative advancement has been applied in trajectory refinement algorithms [26] to obtain collision-free cubic B-spline trajectories. However, in the context of motion planning, such methods are computationally expensive and lack the ability to evaluate other proximity queries such as the minimum separating distance or tolerance verification between moving objects. As modern motion planning and trajectory optimization methods generate candidate paths as rectifiable parametric curves (*e.g.* Lagrange polynomials [31], Legendre polynomials [24], Bézier curves [7, 8, 29], B-splines [34], Dubins paths [35], Pythagorean Hodograph curves [28]), the proximity algorithms must be able to handle such representations.



(a) (b)

Fig. 1: Points closest between (a) a convex polygon and a Bézier curve, and (b) an involute of a circle and a heart-shaped curve.

Generally, proximity to parametric curves is handled by algberaic, interval analysis, or curve subdivision methods. Finding multiple intersections between parametric curves by implicitization and eigenvalue decomposition is discussed in [23]. In [10], the authors present an approach to compute the exact minimum separating distance between differentiably continuous freeform curves by solving a set of nonlinear equations. Computing the minimum separating distance between Bézier curves by sweeping a sphere along one of the curves and eliminating sections of the curve which lie outside of it through subdivision was introduced in [6]. A similar technique [25] was used to find the closest point on a free form curve to a point in free space. In [5], the authors present an efficient method for computing the minimum separating distances to Bézier curves using subdivision methods. The curves are recursively subdivided until the bounds on the minimum separating distance between the control polygons for each of the curves are within some prescribed accuracy. However, these methods do not hold in general for parametric curves and lack the computational efficiency to be used in motion planning algorithms.

In this paper, we introduce a family of algorithms to evaluate the (i) minimum separation distance, (ii) tolerance verification, and (iii) collision detection queries between absolutely continu-

[1]The implementation can be found at `https://github.com/arlk/CurveProximityQueries.jl`.

ous parametric curves and obstacles. The obstacles are defined as convex polytopes, parametric curves, or any other compact set to which minimum distances can be computed, Figure 1. A main feature of the proposed algorithms is their ability to provide proximity queries for a large class of parametric curves, more general than ones that have been considered previously [23, 10, 6, 25, 5]. Such queries are useful in scenarios when the motion planner's candidate path (i) incurs a distance-based penalty for approaching close to an obstacle, (ii) must keep a safe distance from an obstacle, or (iii) must not intersect with the obstacle's geometry.

Our main contributions are summarized as follows:

- Efficient computation of convex hulls for a general class of absolutely continuous parametric curves based on the arc length for any sub-interval in their domain.
- Fast procedures to evaluate the minimum separating distance, tolerance verification, and collision detection queries using interval branch-and-bound methods.

We present the problem formulation for the different query types in Section I. The analysis and results for the construction of convex hulls for parametric curves that provide bounds on the minimum separating distance is provided in Section III. In Section IV, the algorithms for each of the proximity queries are outlined. Finally, in Section V we present numerical examples and benchmarks to illustrate the efficacy of these methods.

II. Problem Formulation

The parametric equation of a curve is given by a function $\psi : \mathcal{I} \to \mathbb{R}^d$, where $\mathcal{I} \subset \mathbb{R}$ is a compact interval such that $|\mathcal{I}| > 0$, and $d \in \mathbb{N}$. We define the curve over the closed sub-interval $\mathcal{Q} \subseteq \mathcal{I}$ (such that $|\mathcal{Q}| > 0$) as the following compact set

$$\Psi_{\mathcal{Q}} = \{\psi(t) \in \mathbb{R}^d : t \in \mathcal{Q}\}. \tag{1}$$

We proceed to define the proximity query problems between a curve $\Psi_{\mathcal{I}}$ and an object $\mathcal{B}$ represented as a nonempty compact set in $\mathbb{R}^d$ under the following assumptions.

Assumption 1. The function ψ is absolutely continuous over its entire domain $\mathcal{I}$, *i.e.*, for every $\epsilon > 0$ there exists a $\delta > 0$, such that for each $n \in \mathbb{N}$, if the collection of mutually disjoint closed sub-intervals $\{[\alpha_i, \beta_i] \mid i = 1, \ldots, n\}$ satisfies $\sum_i |\alpha_i - \beta_i| < \delta$, then $\sum_i \|\psi(\alpha_i) - \psi(\beta_i)\| < \epsilon$.

Assumption 2. The function ψ is not a constant map over the entire domain $\mathcal{I}$, *i.e.*, $\psi(t) \neq y$ for all $t \in \mathcal{I}$ for some $y \in \mathbb{R}^d$.

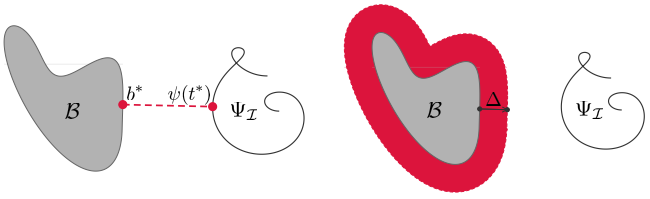
Problem 1 (Minimum Separating Distance)**.** The minimum separating distance between the parametric curve $\Psi_{\mathcal{I}}$ and the compact set $\mathcal{B}$ is defined as

$$d_{\min}(\Psi_{\mathcal{I}}, \mathcal{B}) = \min_{a \in \Psi_{\mathcal{I}}, b \in \mathcal{B}} \|a - b\|. \tag{2}$$

The points $\psi(t^*) \in \Psi_{\mathcal{I}}$ and $b^* \in \mathcal{B}$ that verify

$$d_{\min}(\Psi_{\mathcal{I}}, \mathcal{B}) = \|\psi(t^*) - b^*\|,$$

are the pair of points that lie closest to each other on the respective sets, as shown in Figure 2(a). In addition to considering the minimum separating distance over the entire curve, $\Psi_{\mathcal{I}}$, the definition can be applied to some continuous compact sub-interval, $\mathcal{Q} \subseteq \mathcal{I}$, of the curve as well. This section of the curve is referred to by $\Psi_{\mathcal{Q}}$.



(a) (b)

Fig. 2: (a) The red dashed line shows the segment connecting the pair of closest points between $\Psi_{\mathcal{I}}$ and $\mathcal{B}$ respectively. (b) The red shaded region specifies the Δ-tolerance afforded to $\mathcal{B}$ when computing the tolerance verification between the two objects.

Problem 2 (Tolerance Verification)**.** Tolerance verification is defined as a predicate function with three arguments; two objects, $\Psi_{\mathcal{I}}$ and $\mathcal{B}$, and a tolerance, $\Delta > 0$. The function evaluates the inequality

$$d_{\min}(\Psi_{\mathcal{I}}, \mathcal{B}) > \Delta \tag{3}$$

and returns either $true$ or $false$. Figure 2(b) depicts a scenario, when a curve is separated from a compact set by a distance greater than Δ.

Problem 3 (Collision Detection)**.** The collision detection function is defined as a predicate function with two arguments. These arguments are two objects, $\Psi_{\mathcal{I}}$ and $\mathcal{B}$. The function determines the truth of the following statement:

$$\Psi_{\mathcal{I}} \cap \mathcal{B} \neq \emptyset. \tag{4}$$

Remark. Notice that the solution of Problem 1 implies the solution of Problem 2, which in turn implies the solution of Problem 3. Nevertheless, defining each problem by itself is useful because, as we will see in Section IV, the numerical methods can be specifically tailored for each problem in order to improve its computational efficiency.

III. Bounding Methods and Analysis

The problems discussed in Section II are generally non-convex and difficult to solve[2]. Additionally, when a feasible solution is found it is still difficult to verify that the solution is indeed the global minimum. However, methods that rely on branch-and-bound and interval analysis [17] obtain a solution with a certificate of its optimality within some prescribed accuracy. This is achieved by successively *branching* the problem into smaller and smaller sub-problems over which *bounds* on the optimal solution are computed through relaxation. In this section we present the results and the analysis behind the relaxation of Problem 1 and its sub-problems.

[2]Non-convex problems are considered to be at least NP-hard.

A. Upper Bound on the Length of a Curve

As the presented methods require the computation of the arc length of parametric curves, we will only consider parametric curves that are rectifiable *i.e.* they possess a finite arc length over their domain. The absolute continuity of ψ over $\mathcal{I}$ is a sufficient condition [16] for the curve to be rectifiable[3], and the corresponding arc length function over the interval $\mathcal{Q} \subseteq \mathcal{I}$ is defined as

$$s_\psi(\mathcal{Q}) = \int_{\mathcal{Q}} \|\psi'(t)\| \ \mathrm{d}t, \tag{5}$$

where ψ has a finite derivative ψ' almost everywhere and is Lebesgue integrable.

As branch-and-bound techniques involve repeated calculations on the curve, for reasons having to do with computational efficiency, it is extremely desirable to have a closed-form expression for the antiderivative of the square root of the inner product of ψ'. However, the arc length of rectifiable parametric curves cannot be obtained in general. Even for simple curves described with polynomial or sinusoidal basis functions, one cannot express the integral in terms of elementary functions. In the following result, we introduce an upper bound on $s_\psi(\mathcal{Q})$ in Equation (5) that is better suited to provide a closed-form expression.

Lemma 1. Let ψ be an absolutely continuous function and have a derivative ψ' on the compact interval $\mathcal{I}$. For any closed $\mathcal{Q} \subseteq \mathcal{I}$ define the upper bound

$$u_\psi(\mathcal{Q}) = \sqrt{|\mathcal{Q}| \int_{\mathcal{Q}} \psi'(t)^\top \psi'(t) \, \mathrm{d}t}, \tag{6}$$

where $|\mathcal{Q}|$ is the length of the sub-interval. Then for the arc length function of the curve s_ψ (from Equation (5)) the following inequality holds

$$s_\psi(\mathcal{Q}) \leq u_\psi(\mathcal{Q}).$$

Proof: Consider the arc length function given in Equation (5). Expanding the Euclidean norm and scaling both sides yields

$$\frac{1}{|\mathcal{Q}|} s_\psi(\mathcal{Q}) = \frac{1}{|\mathcal{Q}|} \int_{\mathcal{Q}} \sqrt{\psi'(t)^\top \psi'(t)} \, \mathrm{d}t.$$

Note that $\psi'(t)^\top \psi'(t)$ is a strictly positive real-valued function and that the square root function is concave on the interval $[0, \infty)$. Using Jensen's inequality we have

$$\frac{1}{|\mathcal{Q}|} \int_{\mathcal{Q}} \sqrt{\psi'(t)^\top \psi'(t)} \, \mathrm{d}t \leq \sqrt{\frac{1}{|\mathcal{Q}|} \int_{\mathcal{Q}} \psi'(t)^\top \psi'(t) \, \mathrm{d}t}.$$

Multiplying by $|\mathcal{Q}|$ on both sides gives the result. ∎

For common parametric curves with polynomial or trigonometric basis functions, such as the ones typically employed in trajectory generation methods [34], the antiderivative for the inner product of ψ' is readily available, which greatly reduces the computation time. In Table I, notice the improvement in computation time when the antiderivative is known in closed-form.

Parametric Equations	**Median Time** (ns)	
	$s_\psi(\mathcal{Q})$	$u_\psi(\mathcal{Q})$
$(2\cos(t),\ \sin(t))$	1710.70	58.02
$((t^3+t),\ t)$	1673.80	44.69
$((t+1)^{-1},\ t)$	2102.11	38.43

TABLE I: Comparison between the median evaluation time for computing $s_\psi(\mathcal{Q})$ and $u_\psi(\mathcal{Q})$ for several different intervals $\mathcal{Q} \subseteq [0, 1]$. For each example presented in the table, the antiderivative of the integrand in Equation (5) is not available in closed form and must be numerically integrated, whereas the antiderivative of the integrand in Equation (6) is available in closed-form. The numerical integration uses the Gauss-Kronrod quadrature formula procedure over 15 points.

B. Convex Hull on Sub-intervals of a Curve

The following theorem establishes a convex hull for parametric curves on any sub-interval based on their upper bound of the arc length from Equation (6).

Theorem 1 (Convex Hull)**.** Let ψ be an absolutely continuous function defined on the compact interval $\mathcal{I}$. For any closed interval $[\alpha, \beta] \equiv \mathcal{Q} \subseteq \mathcal{I}$, define the convex compact set

$$\mathcal{U}_\mathcal{Q} = \{x \in \mathbb{R}^d : \|\psi(\alpha) - x\| + \|x - \psi(\beta)\| \leq u_\psi(\mathcal{Q})\}. \tag{7}$$

Then the curve defined on the interval $\mathcal{Q}$ satisfies

$$\Psi_\mathcal{Q} \subset \mathcal{U}_\mathcal{Q}. \tag{8}$$

Proof: It is obvious that the shortest distance between two points in Euclidean space is the length of the chord joining them. Therefore for all $t \in [\alpha, \beta]$ the following must hold

$$\|\psi(\alpha) - \psi(t)\| + \|\psi(t) - \psi(\beta)\| \leq s_\psi([\alpha, t]) + s_\psi([t, \beta]).$$

Then, from the definition of the arc length in Equation (5) we have that $s_\psi([\alpha, t]) + s_\psi([t, \beta])$ is the sum of integrals on adjacent intervals. Since ψ is rectifiable, the integrals are combined as the total arc length on $\mathcal{Q}$, implying that

$$\|\psi(\alpha) - \psi(t)\| + \|\psi(t) - \psi(\beta)\| \leq s_\psi(\mathcal{Q}).$$

Recall that since $s_\psi(\mathcal{Q}) \leq u_\psi(\mathcal{Q})$ from Lemma 1, then every point on the curve evaluated over the interval $\mathcal{Q}$ is an element of the set $\mathcal{U}_\mathcal{Q}$, *i.e.*

$$\psi(t) \in \mathcal{U}_\mathcal{Q}.$$

From the definition of the curve in Equation (1) it follows that $\Psi_\mathcal{Q} \subset \mathcal{U}_\mathcal{Q}$. ∎

The convex hull in Theorem 1 is a compact interval for curves in $\mathbb{R}$, and is also an ellipsoid for curves in higher dimensions. This geometric relationship to an ellipsoid is particularly useful because there are several methods [13, 30, 4] to cheaply compute the minimum separating distance to an ellipsoid. For a convex hull $\mathcal{U}_\mathcal{Q}$ of the function ψ evaluated over a sub-interval $[\alpha, \beta] \equiv \mathcal{Q} \subseteq \mathcal{I}$, the foci of the ellipse are at $\psi(\alpha)$ and $\psi(\beta)$. The major axis has a length of $u_\psi(\mathcal{Q})$, and the minor axes are of

[3]More generally, any function of bounded variation is rectifiable [20], however, the derivatives for such functions may not exist almost everywhere.

equal lengths $\sqrt{u_\psi(\mathcal{Q})^2 - \|\psi(\alpha) - \psi(\beta)\|^2}$. This construction is illustrated in Figure 3.

Remark. Fat arcs [32, 1] provide tighter bounds with cubic convergence, however, the non-convexity of the bounding region is problematic when computing proximity queries. Additionally, they are only useful in the context of planar Bézier curves and spirals. The convergence behavior of the ellipsoidal bounding region to the curve will be explored in the future.

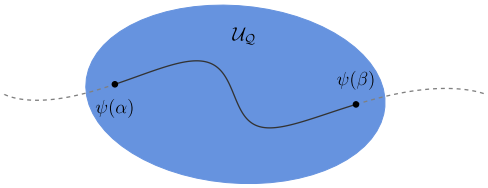


Fig. 3: The blue shaded region shows the convex set $\mathcal{U}_\mathcal{Q}$, which contains the function ψ evaluated over the sub-interval $[\alpha, \beta]$. The dashed-line is the evaluation of ψ over its entire domain.

C. Relaxation of Problem 1

Now that we have enclosed a parametric curve inside a convex hull, Problem 1 can be relaxed, and a more tractable problem can be solved by computing the minimum separating distance to the convex hull. The solution to this relaxed problem provides a lower bound to the optimal solution of the original problem. We define this lower bound on the minimum separating distance d_{lb} between $\Psi_\mathcal{Q}$ and a compact set B as

$$d_{\text{lb}}(\Psi_\mathcal{Q}, \mathcal{B}) = \min_{x \in \mathcal{U}_\mathcal{Q}, b \in \mathcal{B}} \|x - b\|, \tag{9}$$

where $\mathcal{U}_\mathcal{Q}$ is the convex hull corresponding to $\Psi_\mathcal{Q}$ from Equation (7). Figure 4(a) shows the lower bound between a convex polygon and a parametric curve evaluated over a sub-interval. As one might expect, if the object $\mathcal{B}$ is also a parametric curve, then d_{lb} is computed between the convex sets constructed from each of the curves (as shown in Figure 4(b)), as

$$d_{\text{lb}}(\Psi_\mathcal{Q}, \Phi_\mathcal{R}) = \min_{x \in \mathcal{U}_\mathcal{Q}, y \in \mathcal{V}_\mathcal{R}} \|x - y\|, \tag{10}$$

where $\Phi_\mathcal{R}$ is the parametric curve $\phi : \mathbb{R} \supset \mathcal{R} \to \mathbb{R}^d$, and $\mathcal{V}_\mathcal{R}$ is the convex hull of $\Phi_\mathcal{R}$.

In addition to the lower bound, an upper bound on the optimal solution must also be defined in order to bound the solution in a compact interval. The upper bound on the minimum separating distance d_{ub} between $\Psi_\mathcal{Q}$ and a compact set B is given by

$$d_{\text{ub}}(\Psi_\mathcal{Q}, B) = \min_{b \in \mathcal{B}} \|x'(\Psi_\mathcal{Q}) - b\|, \tag{11}$$

such that $x'(\Psi_\mathcal{Q}) \in \Psi_\mathcal{Q}$. Similarly, d_{ub} between the two parametric curves $\Psi_\mathcal{Q}$ and $\Phi_\mathcal{R}$ is

$$d_{\text{ub}}(\Psi_\mathcal{Q}, \Phi_\mathcal{R}) = \|x'(\Psi_\mathcal{Q}) - x''(\Phi_\mathcal{R})\|, \tag{12}$$

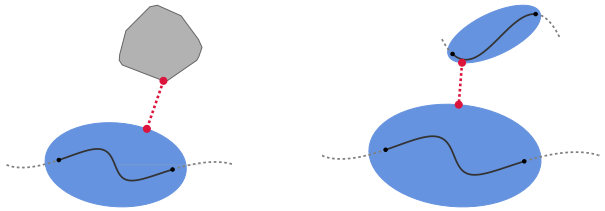


Fig. 4: Lower bound on minimum separating distance between (a) a parametric curve and a convex polygon, and (b) two parametric curves.

such that $x'(\Psi_\mathcal{Q}) \in \Psi_\mathcal{Q}$ and $x''(\Phi_\mathcal{R}) \in \Phi_\mathcal{R}$. In the numerical implementation as seen in Figure 5, the functions x' and x'' select the middle points of $\Psi_\mathcal{Q}$ and $\Phi_\mathcal{R}$ respectively. Heuristics to select points in the curve set based on the relative configuration of the objects rather than simply the midpoints may find tighter bounds, however, this is beyond the scope of the paper.

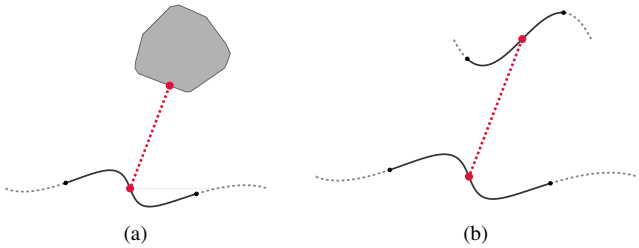


Fig. 5: Upper bound on minimum separating distance between (a) a parametric curve and a convex polygon, and (b) two parametric curves.

We present analysis on the functions d_{lb} and d_{ub}. We show for each interval $\mathcal{Q} \subseteq \mathcal{I}$ that indeed these functions bound the optimal solution on that interval, and that the difference in upper and lower bounds converges uniformly to zero as $|Q| \to 0$. The boundedness and uniform convergence results are necessary in order to guarantee the convergence of interval branch-and-bound algorithms in finite time.

Theorem 2 (Boundedness)**.** Let $\mathcal{B}$ be a compact set in $\mathbb{R}^d$, and $\Psi_\mathcal{I}$ be the curve $\psi : \mathcal{I} \to \mathbb{R}^d$. Then, for every closed $\mathcal{Q} \subseteq \mathcal{I}$, we have

$$d_{\text{lb}}(\Psi_\mathcal{Q}, \mathcal{B}) \leq d_{\min}(\Psi_\mathcal{Q}, \mathcal{B}) \leq d_{\text{ub}}(\Psi_\mathcal{Q}, \mathcal{B}). \tag{13}$$

Proof: It is clear that for all $t' \in \mathcal{Q} \subseteq \mathcal{I}$

$$\min_{t \in \mathcal{Q}, b \in \mathcal{B}} \|\psi(t) - b\| \leq \min_{b \in \mathcal{B}} \|\psi(t') - b\|,$$

which proves the result that $d_{\min}(\Psi_\mathcal{Q}, \mathcal{B}) \leq d_{\text{ub}}(\Psi_\mathcal{Q}, \mathcal{B})$. Recall from Theorem 1 that $\Psi_\mathcal{Q} \subset \mathcal{U}_\mathcal{Q}$, which implies

$$\min_{a \in \mathcal{U}_\mathcal{Q}, b \in \mathcal{B}} \|a - b\| \leq \min_{a \in \Psi_\mathcal{Q}, b \in \mathcal{B}} \|a - b\|.$$

Thus, $d_{\text{lb}}(\Psi_\mathcal{Q}, \mathcal{B}) \leq d_{\min}(\Psi_\mathcal{Q}, \mathcal{B})$, and we have the result. ■

Theorem 3 (Uniform Convergence)**.** Let $\mathcal{B}$ be a compact set in $\mathbb{R}^d$, and $\Psi_\mathcal{I}$ be the curve $\psi : \mathcal{I} \to \mathbb{R}^d$. For every $\epsilon > 0$,

there exists a $\delta > 0$ such that for all $\{\mathcal{Q} : \mathcal{Q} \subseteq \mathcal{I}, |\mathcal{Q}| < \delta\}$, we satisfy

$$d_{\text{ub}}(\Psi_{\mathcal{Q}}, \mathcal{B}) - d_{\text{lb}}(\Psi_{\mathcal{Q}}, \mathcal{B}) < \epsilon. \tag{14}$$

Proof: For any $\mathcal{Q} \subseteq \mathcal{I}$ recall that

$$d_{\text{ub}}(\Psi_{\mathcal{Q}}, \mathcal{B}) - d_{\text{lb}}(\Psi_{\mathcal{Q}}, \mathcal{B}) = \min_{b \in \mathcal{B}} \|x'(\Psi_{\mathcal{Q}}) - b\| - \|x^* - b^*\|,$$

such that $x^* \in \mathcal{U}_{\mathcal{Q}}$ and $b^* \in \mathcal{B}$ minimize Equation (9). Since $\min_{b \in \mathcal{B}} \|x'(\Psi_{\mathcal{Q}}) - b\|$ will be bounded from above by any other element in the set $\mathcal{B}$, we have the following inequality:

$$d_{\text{ub}}(\Psi_{\mathcal{Q}}, \mathcal{B}) - d_{\text{lb}}(\Psi_{\mathcal{Q}}, \mathcal{B}) \leq \|x'(\Psi_{\mathcal{Q}}) - b^*\| - \|x^* - b^*\|.$$

From the reverse triangle inequality we have

$$d_{\text{ub}}(\Psi_{\mathcal{Q}}, \mathcal{B}) - d_{\text{lb}}(\Psi_{\mathcal{Q}}, \mathcal{B}) \leq \|x'(\Psi_{\mathcal{Q}}) - x^*\|.$$

Then from Theorem 1, we know that the largest variation of elements in $\mathcal{U}_{\mathcal{Q}}$ is bounded by the upper bound of the arc length, $u_\psi(\mathcal{Q})$. And, since $x'(\Psi_{\mathcal{Q}}) \in \Psi_{\mathcal{Q}} \subset \mathcal{U}_{\mathcal{Q}}$ and $x^* \in \mathcal{U}_{\mathcal{Q}}$, we get

$$d_{\text{ub}}(\Psi_{\mathcal{Q}}, \mathcal{B}) - d_{\text{lb}}(\Psi_{\mathcal{Q}}, \mathcal{B}) \leq u_\psi(\mathcal{Q}).$$

From Equation (6), we expand $u_\psi(\mathcal{Q})$ with the expression

$$d_{\text{ub}}(\Psi_{\mathcal{Q}}, \mathcal{B}) - d_{\text{lb}}(\Psi_{\mathcal{Q}}, \mathcal{B}) \leq \sqrt{|\mathcal{Q}| \int_{\mathcal{Q}} \psi'(t)^\top \psi'(t) \, \mathrm{d}t}.$$

Since ψ is real valued, the inner product of ψ' is strictly positive. Thus, we further upper bound our expression by expanding the limits of integration with $\mathcal{I} \supseteq \mathcal{Q}$. Define $0 < k = \int_{\mathcal{I}} \psi'(t)^\top \psi'(t) \, \mathrm{d}t$. Then we have

$$d_{\text{ub}}(\Psi_{\mathcal{Q}}, \mathcal{B}) - d_{\text{lb}}(\Psi_{\mathcal{Q}}, \mathcal{B}) \leq \sqrt{|\mathcal{Q}|k}.$$

Thus, for every $\epsilon > 0$, choose $\delta = \frac{\epsilon^2}{k}$ so that for all $\mathcal{Q} \subseteq \mathcal{I}$ such that $|\mathcal{Q}| < \delta$ we get

$$d_{\text{ub}}(\Psi_{\mathcal{Q}}, \mathcal{B}) - d_{\text{lb}}(\Psi_{\mathcal{Q}}, \mathcal{B}) < \epsilon.$$

■

Remark. The minimization problems of Equations (9) and (11) may also be nonconvex depending on the geometry of $\mathcal{B}$. We refer readers to [13, 22] for convex polytopes and [14, 21] for more complex models.

IV. Proximity Queries

Given the boundedness results of Theorem 2 and the convergence results of Theorem 3, we present algorithms that evaluate the solutions to Problems 1 to 3 within some tolerance. Central to each of these algorithms is a process of interval subdivision and bounding. First, we present the methods to compute the minimum separating distance between a parametric curve and a compact set (or another parametric curve). Later, we highlight the variations of this algorithm for evaluating the tolerance verification and collision detection queries without computing the minimum separating distance between the objects.

Algorithm 1 Minimum Separating Distance $(\Psi_{\mathcal{I}}, B)$

$\mathcal{L} \leftarrow \{\mathcal{I}\}$
$\overline{d} \leftarrow d_{\text{ub}}(\Psi_{\mathcal{I}}, \mathcal{B})$
$\underline{d} \leftarrow d_{\text{lb}}(\Psi_{\mathcal{I}}, \mathcal{B})$
while $\overline{d} - \underline{d} > \epsilon$ **do**
  $\mathcal{X} \leftarrow \operatorname{argmin}_{\mathcal{Q} \in \mathcal{L}} d_{\text{lb}}(\Psi_{\mathcal{Q}}, \mathcal{B})$
  $\mathcal{X}_L, \mathcal{X}_R \leftarrow \texttt{split}(\mathcal{X})$
  $\mathcal{L} \leftarrow \mathcal{L} \setminus \{\mathcal{X}\}$
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{\mathcal{X}_L, \mathcal{X}_R\}$
  $\overline{d} \leftarrow \min_{\mathcal{Q} \in \mathcal{L}} d_{\text{ub}}(\Psi_{\mathcal{Q}}, \mathcal{B})$
  $\underline{d} \leftarrow \min_{\mathcal{Q} \in \mathcal{L}} d_{\text{lb}}(\Psi_{\mathcal{Q}}, \mathcal{B})$
end while
return $\underline{d}$

A. Minimum Separating Distance

Given a parametric function $\psi : \mathcal{I} \to \mathbb{R}^d$, Algorithm 1 computes the minimum separating distance between the curve $\Psi_{\mathcal{I}}$ and a compact set $\mathcal{B}$.

We proceed to describe the algorithm. The general structure of the algorithm closely matches that of a vanilla branch-and-bound method [3]. Consider computing the minimum separating distance between a curve and a convex polygon as shown in Figure 6(a). A collection $\mathcal{L}$, which stores the intervals over which the bounds on the optimal solution are computed, is initialized with a single element: the entire domain $\mathcal{I}$. The relaxed problem is solved over the domain $\mathcal{I}$ (Figure 6(b)), and the upper and lower bound values are stored in the states $\overline{d}$ and $\underline{d}$ respectively. If the difference between the bounds is above a prescribed tolerance, the algorithm proceeds to the iterative phase. $\mathcal{I}$ is split into sub-intervals $\mathcal{X}_R$ and $\mathcal{X}_L$, over which the relaxations of the problems on the new intervals are solved (Figure 6(c)). These sub-intervals are added to the collection $\mathcal{L}$, and the original domain $\mathcal{I}$ is removed from the collection. The least upper and lower bounds on the solution are kept track of by $\overline{d}$ and $\underline{d}$, respectively, on all the sub-intervals present in the collection $\mathcal{L}$. Each successive iteration sees the sub-interval with the least lower bound chosen for subdivision. Figures 6(d) and 6(e) show snapshots of the algorithm into the third and seventh iteration respectively. Notice that the algorithm preferentially selects sub-intervals closer to the polygon to subdivide because of the best-first search strategy.

Since the minimization problem of Equation (2) is over a continuous space, the branch-and-bound method will repeatedly subdivide infinitely many times (whenever the objects $\Psi_{\mathcal{I}}$ and $\mathcal{B}$ do not collide). However, by setting an absolute tolerance $\epsilon > 0$ on the solution, we obtain an ϵ-suboptimal solution in a finite number of iterations, *i.e.*

$$d_{\min}(\Psi_{\mathcal{I}}, \mathcal{B}) \in [\underline{d}, \underline{d} + \epsilon], \tag{15}$$

where $\underline{d}$ is the lower bound on the solution obtained from the minimum separating distance algorithm. The certificate proving that the global minimum lies in the interval shown in Equation (15) can be obtained by examining the collection $\mathcal{L}$. Figure 6(f) highlights the solution, when the bounds are separated by a magnitude of no more than ϵ. The convergence

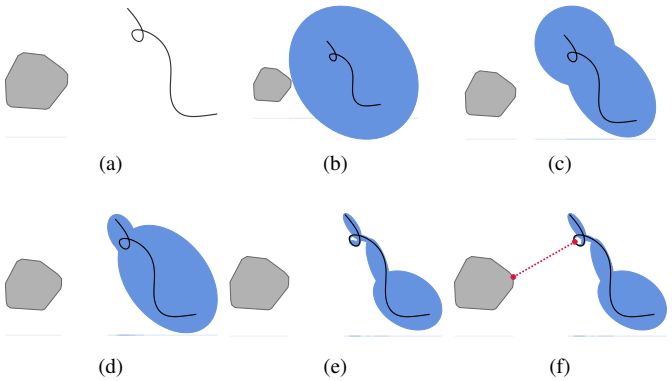


Fig. 6: The different stages of Algorithm 1 when computing the minimum distance between a 13th order Bézier curve and a convex polygon.

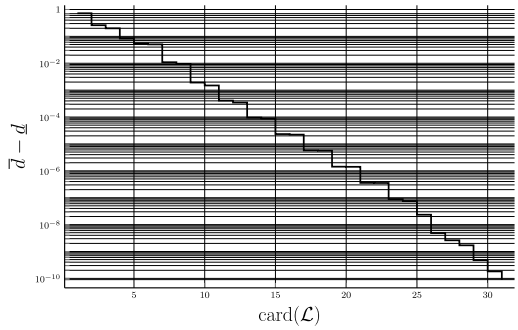


Fig. 7: Convergence of $\overline{d} - \underline{d}$ for the scenario presented in Figure 6 to $\epsilon = 10^{-10}$.

of $\overline{d} - \underline{d}$ to ϵ for the example presented in Figure 6 is shown in Figure 7.

Algorithm 2 describes the procedure to evaluate the minimum distance from $\Psi_{\mathcal{I}}$ to $\Phi_{\mathcal{J}}$, which is the trace of $\phi : \mathcal{J} \to \mathbb{R}^d$. As one might expect, structurally very little differs between Algorithms 1 and 2. The collection $\mathcal{L}$ stores pairs of sub-intervals over which the bounds on the solution will be evaluated. The lower and upper bounds are computed using Equations (10) and (12) respectively. Figure 8 shows the pair of closest points between Bézier curves computed using Algorithm 2.

Algorithm 2 Minimum Separating Distance $(\Psi_{\mathcal{I}}, \Phi_{\mathcal{J}})$

1: $\mathcal{L} \leftarrow \{\{\mathcal{I}, \mathcal{J}\}\}$
2: $\overline{d} \leftarrow d_{\text{ub}}(\Psi_{\mathcal{I}}, \Phi_{\mathcal{J}})$
3: $\underline{d} \leftarrow d_{\text{lb}}(\Psi_{\mathcal{I}}, \Phi_{\mathcal{J}})$
4: **while** $\overline{d} - \underline{d} > \epsilon$ **do**
5: $\quad \mathcal{X} \leftarrow \text{argmin}_{\{\mathcal{Q},\mathcal{R}\}\in\mathcal{L}}\, d_{\text{lb}}(\Psi_{\mathcal{Q}}, \Phi_{\mathcal{R}})$
6: $\quad \mathcal{X}_L, \mathcal{X}_R \leftarrow \texttt{split}(\mathcal{X})$
7: $\quad \mathcal{L} \leftarrow \mathcal{L} \setminus \{\mathcal{X}\}$
8: $\quad \mathcal{L} \leftarrow \mathcal{L} \cup \{\mathcal{X}_L, \mathcal{X}_R\}$
9: $\quad \overline{d} \leftarrow \min_{\{\mathcal{Q},\mathcal{R}\}\in\mathcal{L}} d_{\text{ub}}(\Psi_{\mathcal{Q}}, \Phi_{\mathcal{R}})$
10: $\quad \underline{d} \leftarrow \min_{\{\mathcal{Q},\mathcal{R}\}\in\mathcal{L}} d_{\text{lb}}(\Psi_{\mathcal{Q}}, \Phi_{\mathcal{R}})$
11: **end while**
12: **return** $\underline{d}$

Remark. The $\mathcal{X}_L, \mathcal{X}_R \leftarrow \texttt{split}(\mathcal{X})$ operation from (line 6) of Algorithms 1 and 2 returns two mutually disjoint sets such that $\mathcal{X}_L \cup \mathcal{X}_R = \mathcal{X}$. When $\mathcal{X}$ is an interval, the operation must ensure that both $\mathcal{X}_L$ and $\mathcal{X}_R$ are closed. And, when $\mathcal{X}$ is a collection of two intervals, the operation must only split the larger interval so as to preserve the uniform convergence property from Theorem 3. In the implementation[1], the $\texttt{split}(\mathcal{X})$ operation bisects the interval $\mathcal{X}$, however, uneven splitting techniques may lead to better performance as seen in [5].

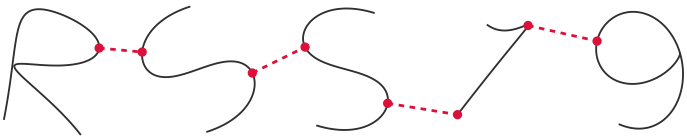


Fig. 8: Points closest between pairs of characters 'R', 'S', 'S', '1', and '9' respectively, that are represented as Bézier curves. The evaluation time for the four minimum separating distance queries was 1.14 ms.

B. Tolerance Verification

As discussed in Section I, in many applications an exact measurement on the minimum separating distance between objects is not required. A weaker method that verifies if two objects are separated by a distance greater than Δ may be preferred for computational reasons. Since the states $\overline{d}$ and $\underline{d}$ in Algorithms 1 and 2 keep track of the the bounds on the global optimum, the algorithm will terminate early if either $\overline{d}$ or $\underline{d}$ violate the Δ-tolerance, *i.e.* all the convex hulls formed from the curve evaluated at the sub-intervals in collection $\mathcal{L}$ are separated from the object $\mathcal{B}$ by a distance of at least Δ. Algorithm 3 highlights the differences from Algorithm 1.

Algorithm 3 Tolerance Verification $(\Psi_{\mathcal{I}}, B, \Delta)$

$\quad \vdots$
4: **while** $\overline{d} - \Delta > \epsilon$ **do**
$\quad \vdots$
11: $\quad$ **if** $\underline{d} > \Delta$ **then return** true
12: **end while**
13: **return** false

C. Collision Detection

Sometimes only the detection of intersection between two objects is required. This is a special case of Algorithm 3, where $\Delta = 0$. Algorithm 3 highlights the differences from Algorithm 1.

Algorithm 4 Collision Detection $(\Psi_{\mathcal{I}}, B)$

$\quad \vdots$
4: **while** $\overline{d} > \epsilon$ **do**
$\quad \vdots$
11: $\quad$ **if** $\underline{d} > 0$ **then return** false
12: **end while**
13: **return** true

V. Computational Results

We present numerical simulations of our approach for several different proximity query problems. In general, the classes of examples that we consider are categorized as the proximity query between a parametric curve and a point, a convex polygon, and another parametric curve. In these numerical problems we only consider proximity queries in $\mathbb{R}^2$ for ease of viewing, however, the algorithm will hold in general as long as there exist methods to compute d_{lb} and d_{ub} in the given dimension. Examples of proximity queries for parametric curves in $\mathbb{R}^3$ can be found in the online repository[1]. In every problem setting, the execution times for computing the minimum separating distance, tolerance verification and collision detection between two objects are benchmarked. Each benchmark result is computed as the median over 10,000 trials of the program. The implementation[4] uses double-precision arithmetic, and we chose the tolerance of $\epsilon = 10^{-10}$ to be used in the optimization procedure. For the tolerance verification queries presented in this section, we choose the value of Δ as half the minimum separating distance between the objects which they are querying.

A. Curve - Point Proximity

We show the performance of computing proximity queries between Bézier curves and a point in $\mathbb{R}^2$ using our approaches and the approach found in [5]. The control points of the curve are uniformly randomly placed in $[0, 1] \times [0, 1]$, and the point to which the proximity is computed is also randomly placed in the same unit square. Figure 9 shows a few examples of the problems that were considered.

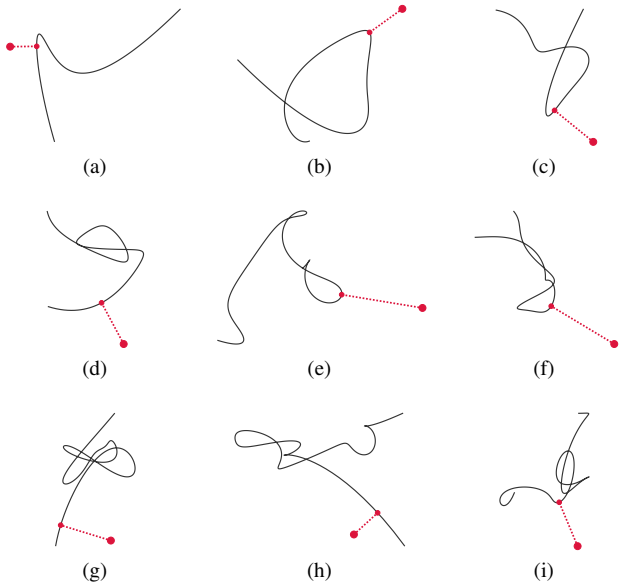
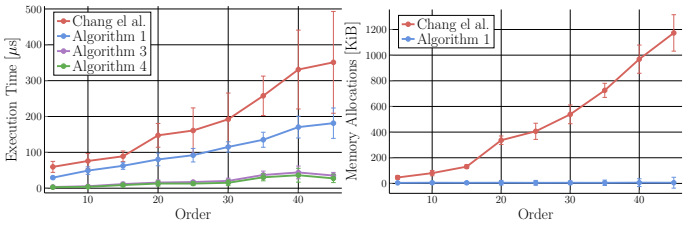


Fig. 9: The point on a (a) 5th, (b) 10th, (c) 15th, (d) 20th, (e) 25th, (f) 30th, (g) 35th, (h) 40th, and (h) 45th Bézier curve that is closest to another randomly chosen point.

[4]The algorithms are in implemented in Julia [2], and the benchmarks were conducted on a 2.3 GHz Intel Core i5 machine with 8 Gigabytes of RAM.

The numerical simulations compute the mean execution time over a range of different order Bézier curves as shown in Figure 10. Notice that the computational efficiency (Figure 10(a)) of Algorithm 1 is improved compared to that of the curve subdivision algorithm in [5]. This is in part because the construction of the bounding region in [5] is the control polygon of the subdivided curve that is obtained through the expensive De Casteljau's algorithm [11]. Furthermore, the parameterizations of the subdivided curves are stored in a queue, which is very memory expensive for higher order curves as seen in Figure 10(b). On the other hand, our methods are very memory efficient as the priority queue (expressed as the collection $\mathcal{L}$ in Algorithm 1 only contains information of the intervals. Additionally, computing proximity queries for Bézier curves using our approach is numerically robust as Equation (6) is also a Bézier curve and does not require any change of basis. Approaches to compute the closed-form expression of Equation (6) can be found in [12].

Both Algorithms 3 and 4 significantly outperform the other methods. This is a result of the fact that the algorithms are terminated as soon as the bounds are met on the Δ-tolerance, rather than proceeding to obtain an ϵ-suboptimal solution.



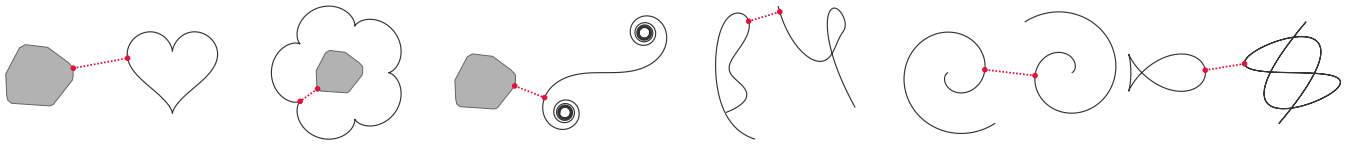
(a) (b)

Fig. 10: The average computational time (a) and allocated memory (b) to evaluate a proximity query between ten randomly generated Bézier curves and a point. The algorithm presented in [5] and Algorithm 1 both use a tolerance $\epsilon = 10^{-10}$. The error bars represent the standard deviation of the trial runs.

B. Curve - Convex Polygon Proximity

We proceed to discuss the performance of proximity queries between parametric curves and polygons through some examples in Figure 11. We consider the general class of absolutely continuous parametric curves, and, since not all the curves can be transformed[5] to a Bernstein basis, we do not provide numerical comparisons to [5] here. For the underlying routines that compute the bounds on the solution, *i.e.* d_{lb} and d_{ub}, we use the GJK-algorithm [13] to compute the distances between the polygons and the ellipsoidal convex hulls. Figures 11(a) to 11(c) show proximity of curves with a sinusoidal basis with convex polygons. It should be noted that although the Euler spiral is defined over $\mathbb{R}$, we consider its trace Figure 11(c) in the compact subset: $[-2\pi, 2\pi]$. In Table II, the median execution times for the different problems are enumerated. It should be highlighted that the reason for the longer run times of the Euler

[5]Even transformations between polynomial bases are numerically unstable for higher order polynomials [15].



(a) (b) (c) (d) (e) (f)

Fig. 11: Points closest between (a) a convex polygon and a heart-shaped curve [37], (b) a convex polygon and an epicycloid [38], (c) a convex polygon and an Euler spiral, (d) two 10^{th} order Bézier curves, (e) two involutes of a circle, and (f) a fish curve [36] and a Lissajous curve.

Query	**Median Time** (μs)					
	(a)	(b)	(c)	(d)	(e)	(f)
Alg. 1/2	208.22	234.52	150.03 $\times 10^3$	425.66	1.21 $\times 10^3$	3.74 $\times 10^3$
Alg. 3	10.52	80.90	29.70 $\times 10^3$	49.20	204.68	471.99
Alg. 4	10.38	59.19	17.08 $\times 10^3$	37.66	116.22	229.25

TABLE II: The median computational time to evaluate a proximity query between each of the objects in Figure 11.

spiral is due to the evaluation of the parametric function that requires numerical integration.

C. Curve - Curve Proximity

Similar to above, Table II also shows the performance for computing the proximity queries for the problems in Figures 11(d) to 11(f). As these problems require simultaneously searching across dual parameter spaces, it is natural that the proximity queries will have longer run times. This will be offset if a lower ϵ-tolerance is chosen in the procedures.

D. Trajectory Replanning Example

To demonstrate the benefit of fast collision detection, Figure 12 shows a common situation in path replanning problems. A vehicle is confronted with an obstacle and randomly samples a large number of possible trajectories from a distribution to plan a path around the obstacle [19]. For each of these trajectories, a certificate of collision avoidance and minimum safety distance are necessary. In this situation, 1000 quintic Bézier curves are generated and, using Algorithm 3, a tolerance verification query is performed against both the obstacles with a total run time of 11.61 milliseconds. In the simulation, 818 trajectories of the 1000 samples are in collision with at least one of the two obstacles, 116 trajectories are collision free but violate the minimum safety distance constraint, and only the remaining 66 trajectories are feasible. This ability to rapidly compute the feasibility of trajectories allow for a large sample size of trajectories to be validated in a very short amount of time. In addition, these methods prove beneficial for predicting collisions in dynamic environments wherein only probabilistic information of the obstacle behavior is available [27] by computing proximity queries with the boundary of the confidence region of an obstacle's trajectory.

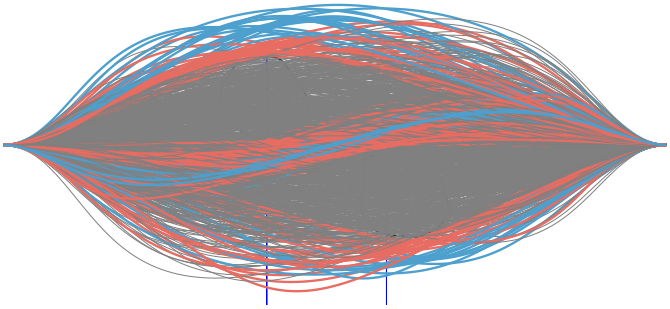


Fig. 12: An example scenario of a robot (starting on the left) attempting to replan its trajectory around obstacles: The blue trajectories represent feasible solutions, the red trajectories are collision-free, but infeasible because of their close proximity to the obstacles (*i.e.*$\leq \Delta$), and the grey trajectories are infeasible, because they collide with the obstacles in the environment.

VI. Conclusions

We have presented new algorithms for proximity queries evaluating minimum distance, tolerance verification and collision detection. To the authors' knowledge, this is the first proximity query method that works on such a large class of parametric curves. In addition, the methods show improved computational speed, even compared to the methods taking advantage of a known curve basis.

The presented proximity query algorithms are built around an interval branch-and-bound method that provide ϵ-suboptimal solutions. We present efficient methods to construct convex hulls that enclose a parametric curve based on the upper bound of its arc length. The convex hulls are used to compute the lower bound on the minimum separating distance to obstacles during the execution of the algorithm. Using this approach, computational results are shown for the evaluation of minimum distance, tolerance verification and collision detection between arbitrary absolutely continuous parametric curves.

Acknowledgments

This work was developed with financial support from the National Science Foundation's National Robotics Initiative (#1830639, #1528036), Office of Naval Research (N00014-19-1-2106), National Aeronautics and Space Administration (NASA), and the Air Force Office of Scientific Research.

References

[1] Michael Bartoň and Gershon Elber. Spiral fat arcs–Bounding regions with cubic convergence. *Graphical Models*, 73(2):50–57, 2011.

[2] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[3] Stephen Boyd and Jacob Mattingley. Branch and bound methods. 2007.

[4] Nilanjan Chakraborty, Jufeng Peng, Srinivas Akella, and John E Mitchell. Proximity queries between convex objects: An interior point approach for implicit surfaces. *IEEE Transactions on Robotics*, 24(1):211–220, 2008.

[5] Jung-Woo Chang, Yi-King Choi, Myung-Soo Kim, and Wenping Wang. Computation of the minimum distance between two Bézier curves/surfaces. *Computers & Graphics*, 35(3):677–684, 2011.

[6] Xiao-Diao Chen, Linqiang Chen, Yigang Wang, Gang Xu, Jun-Hai Yong, and Jean-Claude Paul. Computing the minimum distance between two Bézier curves. *Journal of Computational and Applied Mathematics*, 229(1):294–301, 2009.

[7] Venanzio Cichella, Isaac Kaminer, Claire Walton, and Naira Hovakimyan. Optimal motion planning for differentially flat systems using Bernstein approximation. *IEEE Control Systems Letters*, 2(1):181–186, 2018.

[8] Venanzio Cichella, Isaac Kaminer, Claire Walton, Naira Hovakimyan, and Antonio Pascoal. Bernstein approximation of optimal control problems. *arXiv preprint arXiv:1812.06132*, 2018.

[9] Stephen A Ehmann and Ming C Lin. Accurate and fast proximity queries between polyhedra using convex surface decomposition. In *Computer Graphics Forum*, volume 20, pages 500–511. Wiley Online Library, 2001.

[10] Gershon Elber and Tom Grandine. Hausdorff and minimal distances between parametric freeforms in $\mathbb{R}^2$ and $\mathbb{R}^3$. In *International Conference on Geometric Modeling and Processing*, pages 191–204. Springer, 2008.

[11] Gerald Farin and Dianne Hansford. *The essentials of CAGD*. AK Peters/CRC Press, 2000.

[12] Rida T Farouki. The Bernstein polynomial basis: A centennial retrospective. *Computer Aided Geometric Design*, 29(6):379–419, 2012.

[13] Elmer G Gilbert, Daniel W Johnson, and S Sathiya Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal on Robotics and Automation*, 4(2):193–203, 1988.

[14] Stefan Gottschalk, Ming C Lin, and Dinesh Manocha. OBBTree: A hierarchical structure for rapid interference detection. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 171–180. ACM, 1996.

[15] Thomas Hermann. On the stability of polynomial transformations between Taylor, Bernstein and Hermite forms. *Numerical algorithms*, 13(2):307–320, 1996.

[16] Kanesiroo Iseki. On certain properties of parametric curves. *Journal of the Mathematical Society of Japan*, 12(2):129–173, 1960.

[17] Luc Jaulin, Michel Kieffer, Olivier Didrit, and Eric Walter. *Applied interval analysis: With examples in parameter and state estimation, robust control and robotics*, volume 1. Springer Science & Business Media, 2001.

[18] Michal Kleinbort, Oren Salzman, and Dan Halperin. Collision detection or nearest-neighbor search? On the computational bottleneck in sampling-based motion planning. *arXiv preprint arXiv:1607.04800*, 2016.

[19] Marin Kobilarov. Cross-entropy motion planning. *The International Journal of Robotics Research*, 31(7):855–871, 2012.

[20] Andrey N Kolmogorov and Sergei V Fomin. *Elements of the Theory of Functions and Functional Analysis*. Dover, 1957.

[21] Eric Larsen, Stefan Gottschalk, Ming C Lin, and Dinesh Manocha. Fast proximity queries with swept sphere volumes. Technical report, TR99-018, Department of Computer Science, University of North Carolina, 1999.

[22] Ming C Lin and John F Canny. A fast algorithm for incremental distance calculation. In *IEEE International Conference on Robotics and Automation (ICRA), 1991*, pages 1008–1014. IEEE, 1991.

[23] Dinesh Manocha and James Demmel. Algorithms for intersecting parametric and algebraic curves I: Simple intersections. *ACM Transactions on Graphics (TOG)*, 13 (1):73–100, 1994.

[24] Daniel Mellinger, Alex Kushleyev, and Vijay Kumar. Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 477–483. IEEE, 2012.

[25] Young-Taek Oh, Yong-Joon Kim, Jieun Lee, Myung-Soo Kim, and Gershon Elber. Efficient point-projection to freeform curves and surfaces. *Computer Aided Geometric Design*, 29(5):242–254, 2012.

[26] Jia Pan, Liangjun Zhang, and Dinesh Manocha. Collision-free and smooth trajectory computation in cluttered environments. *The International Journal of Robotics Research*, 31(10):1155–1175, 2012.

[27] Andrew Patterson, Arun Lakshmanan, and Naira Hovakimyan. Intent-aware probabilistic trajectory estimation for collision prediction with uncertainty quantification. *arXiv preprint arXiv:1904.02765*, 2019.

[28] Javier Puig-Navarro, Naira Hovakimyan, Natalia Alexandrov, and Bonnie D Allen. Silhouette-informed trajectory generation through a wire maze for UAS. In *2018 Aviation Technology, Integration, and Operations Conference*, page 3845, 2018.

[29] Lorenzo A Ricciardi and Massimiliano Vasile. Direct transcription of optimal control problems with finite elements on Bernstein basis. *Journal of Guidance, Control, and Dynamics*, pages 1–15, 2018.

[30] Elon Rimon and Stephen P Boyd. Obstacle collision detection using best ellipsoid fit. *Journal of Intelligent and Robotic Systems*, 18(2):105–126, 1997.

[31] Isaac M Ross and Mark Karpenko. A review of pseudospectral optimal control: From theory to flight. *Annual Reviews in Control*, 36(2):182–197, 2012.

[32] Thomas W Sederberg, Scott C White, and Alan K Zundel. Fat arcs: A bounding region with cubic convergence. *Computer Aided Geometric Design*, 6(3):205–218, 1989.

[33] Min Tang, Young J Kim, and Dinesh Manocha. C^2A: Controlled conservative advancement for continuous collision detection of polygonal models. In *IEEE International Conference on Robotics and Automation (ICRA), 2009*, pages 849–854. IEEE, 2009.

[34] Wannes Van Loock, Goele Pipeleers, and Jan Swevers. Optimal motion planning for differentially flat systems with guaranteed constraint satisfaction. In *American Control Conference (ACC), 2015*, pages 4245–4250. IEEE, 2015.

[35] Petr Váňa and Jan Faigl. Optimal solution of the generalized Dubins interval problem. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018. doi: 10.15607/RSS.2018.XIV.035.

[36] Eric W Weisstein. Fish Curve. 2003.

[37] Eric W Weisstein. Heart Curve. 2003.

[38] Eric W Weisstein. Ranunculoid. 2003.

[39] Xinyu Zhang, Minkyoung Lee, and Young J Kim. Interactive continuous collision detection for non-convex polyhedra. *The Visual Computer*, 22(9-11):749–760, 2006.