Differentiable Collision Detection for a Set of Convex Primitives

Kevin Tracy[1], Taylor A. Howell[2], and Zachary Manchester[1]

***Abstract*— Collision detection between objects is critical for simulation, control, and learning for robotic systems. However, existing collision detection routines are inherently non-differentiable, limiting their applications in gradient-based optimization tools. In this work, we propose DCOL: a fast and fully differentiable collision-detection framework that reasons about collisions between a set of composable and highly expressive convex primitive shapes. This is achieved by formulating the collision detection problem as a convex optimization problem that solves for the minimum uniform scaling applied to each primitive before they intersect. The optimization problem is fully differentiable with respect to the configurations of each primitive and is able to return a collision detection metric and contact points on each object, agnostic of interpenetration. We demonstrate the capabilities of DCOL on a range of robotics problems from trajectory optimization and contact physics, and have made an open-source implementation available.**

I. Introduction

Computing collisions is of great interest to the computer graphics, video game, and robotics communities. Popular algorithms for collision detection include the Gilbert, Johnson, and Keerthi (GJK) algorithm [1], its updated variant enhanced-GJK [2], and Minkowski Portal Refinement (MPR) [3], [4]. For objects that have interpenetration, the Expanding Polytope Algorithm (EPA) [5] is used to return a metric that describes the depth of penetration between two objects. These algorithms are implemented in the widely used Flexible Collision Library (FCL) [6], and are employed in most physics engines including Bullet [7], Drake [8], Dart [9], and MuJoCo [10]. While efficient and robust, all of these algorithms are inherently non-differentiable due to their logical control flow and pivoting.

This paper introduces DCOL, a differentiable collision-detection framework that computes closest points, minimum distance, and interpenetration depth between any pair of six convex primitive shapes: polytopes, capsules, cylinders, cones, ellipsoids, and padded polygons (Fig. 2). We do this by formulating a convex optimization problem that solves for the minimum uniform scaling that must be applied to the primitives for an intersection to occur. When primitives are not in contact, the minimum scaling for an intersection is greater than one, and when there is interpenetration between objects, the minimum scaling is less than one. The ability to return an informative collision metric in the presence of interpenetration is a key distinction between DCOL and GJK variants. In addition to the scaling parameter detecting a collision, the contact points on each object can be calculated from this solution as well.

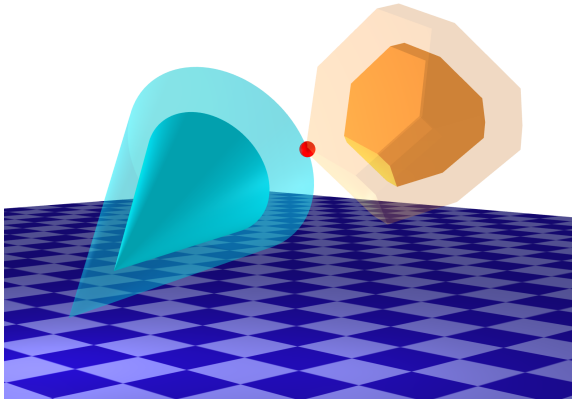

Fig. 1. Collision detection between a cone and a polytope. DCOL works by solving an optimization problem for the minimum scaling of each object that produces an intersection which, in this example, is greater than one, meaning there is no collision. The scaled objects are translucent and the intersection point between these scaled objects is shown in red.

The optimization problems produced by our formulation are bounded, feasible, and well-defined for all configurations of the primitives. Differentiable convex optimization allows the sensitivities of the solution to be calculated with respect to problem parameters with minimal added computation. This allows informative and smooth derivatives of the minimum scaling as well as the contact points with respect to the configurations of the primitives to be computed efficiently.

The ability to differentiate through our collision detection algorithm enables the inclusion of accurate collision information into gradient-based robotic simulation, control, and learning frameworks. We demonstrate this on several relevant robotics problems from trajectory optimization and contact physics.

Our specific contributions in this paper are the following:

- An optimization-based collision detection formulation between convex primitives that returns an informative collision metric even in the case of interpenetration
- Efficient differentiation of this optimization problem with respect to the configurations of each primitive
- A fast and efficient open-source implementation of these algorithms built on a custom primal-dual interior-point solver

The paper proceeds by providing background on standard convex conic optimization and differentiation of conic op-

[1]Kevin Tracy and Zachary Manchester are with The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA {ktracy,zacm}@cmu.edu

[2]Taylor Howell is with the Department of Mechanical Engineering, Stanford University, Stanford, CA 94305, USA thowell@stanford.edu

timization problems in Section II, a derivation of DCOL in Section III with the corresponding constraints for each of the six convex primitives shown in Fig. 2, example use cases in trajectory optimization and contact physics in Section IV, and our conclusions in Section V.

II. Background

The differentiable collision detection algorithm, DCOL, proposed in this paper is built on differentiable convex optimization. In this section, convex optimization with the relevant conic constraints is detailed, as well as a method for efficiently computing derivatives of these optimization problems with respect to problem parameters.

A. Conic Optimization

DCOL formulates collision detection problems as a convex optimization problem with conic constraints [11]. In standard form, these optimization problems have linear objectives and constraints of the following form:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & h - Gx \in \mathcal{K}, \end{aligned} \tag{1}$$

where $x \in \mathbf{R}^n$, $c \in \mathbf{R}^n$, $G \in \mathbf{R}^{m \times n}$, $h \in \mathbf{R}^m$, and $\mathcal{K} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_N$ is a Cartesian product of N proper convex cones. The optimality conditions for problem 1 are as follows:

$$c + G^T z = 0, \tag{2}$$

$$h - Gx \in \mathcal{K}, \tag{3}$$

$$z \in \mathcal{K}^*, \tag{4}$$

$$(h - Gx) \circ z = 0, \tag{5}$$

where a dual variable $z \in \mathbf{R}^m$ is introduced, $\mathcal{K}^*$ is the dual cone, and $\circ$ is a cone product specific to each cone [12].

The cones required for DCOL include the nonnegative orthant, denoted as $\mathbf{R}^m_+$, and the second-order cone, denoted as $\mathcal{Q}_m$. The nonnegative orthant contains any vector $s \in \mathbf{R}^m_+$ where $s \geq 0$, and the a second-order cone contains any vector $s \in \mathcal{Q}_m$ such that $\|s_{2:m}\|_2 \leq s_1$.

We develop a custom primal-dual interior-point solver for DCOL, with support for both of the relevant cones, based on the *conelp* solver from [12], with features taken from [13]–[16]. The memory for this custom solver is entirely stack-allocated and is optimized for the small problems that DCOL creates, dramatically outperforming off-the-shelf primal-dual interior-point conic solvers like ECOS [13] and Mosek [17].

B. Differentiating Through a Cone Program

Recent advances in differentiable convex optimization have enabled efficient differentiation through problems of the form (1) [18]–[20]. Solutions to (1) can be differentiated with respect to any parameters used in c, G, and h.

At the core of differentiable convex optimization is the implicit function theorem. An implicit function $g : \mathbf{R}^a \times \mathbf{R}^b \to \mathbf{R}^a$ is defined as:

$$g(z^*, \theta) = 0, \tag{6}$$

TABLE I
Average DCOL Computation Times

	polyt.	**caps.**	**cyl.**	**cone**	**ellips.**	**polyg.**
evaluate	5.9 μs	8.5 μs	8.4 μs	5.0 μs	6.8 μs	9.4 μs
differentiate	1.4 μs	1.4 μs	1.6 μs	1.3 μs	1.3 μs	1.7 μs

for an equilibrium point $z^* \in \mathbf{R}^a$, and problem parameters $\theta \in \mathbf{R}^b$. Approximating (6) with a first-order Taylor series results in:

$$\frac{\partial g}{\partial z}\delta z + \frac{\partial g}{\partial \theta}\delta\theta = 0, \tag{7}$$

which can be re-arranged to solve for the sensitivities of the solution with respect to the problem parameters:

$$\frac{\partial z}{\partial \theta} = -\left(\frac{\partial g}{\partial z}\right)^{-1}\frac{\partial g}{\partial \theta}. \tag{8}$$

By treating the optimality conditions in equations (2) and (5) as an implicit function at a primal-dual solution, the sensitivities of the solution with respect to the problem data can be computed. When the original optimization problem is solved using a primal-dual interior-point method as described in [12], these derivatives can be computed after the solve without any additional matrix factorizations [20]. This enables fast differentiation of conic programs that are fit for use in our differentiable collision detection algorithm.

III. The DCOL Algorithm

This section details how DCOL computes collision information between two convex primitives. The core part of this framework is an optimization problem that solves for a minimum uniform scaling $\alpha \in \mathbf{R}$ applied to both objects that result in an intersection. In the case where there is no collision between the two objects, the minimum scaling is greater than one, and when there is interpenetration, the minimum scaling is less than one. Because of this, we find the minimum scaling α is a better collision metric than the closest distance between the primitives, allowing for the straightforward description of collision constraints that are agnostic of interpenetration. All steps in the creation and solving of this optimization problem are fully differentiable, and average timing results for computing both solutions and derivatives are provided in Table I as an average over each primitive.

A. Optimization Problem

Scaled convex primitives are described as a set $S(\alpha)$, which is a specific instance of the primitive scaled by some α. A point $x \in \mathbf{R}^3$ is said to be in the set $x \in S(\alpha)$ if x is within the scaled primitive. This notation allows for the following formulation of the optimization problem:

$$\begin{aligned} \underset{x,\,\alpha}{\text{minimize}} \quad & \alpha \\ \text{subject to} \quad & x \in \mathcal{S}_1(\alpha), \\ & x \in \mathcal{S}_2(\alpha), \\ & \alpha \geq 0, \end{aligned} \tag{9}$$

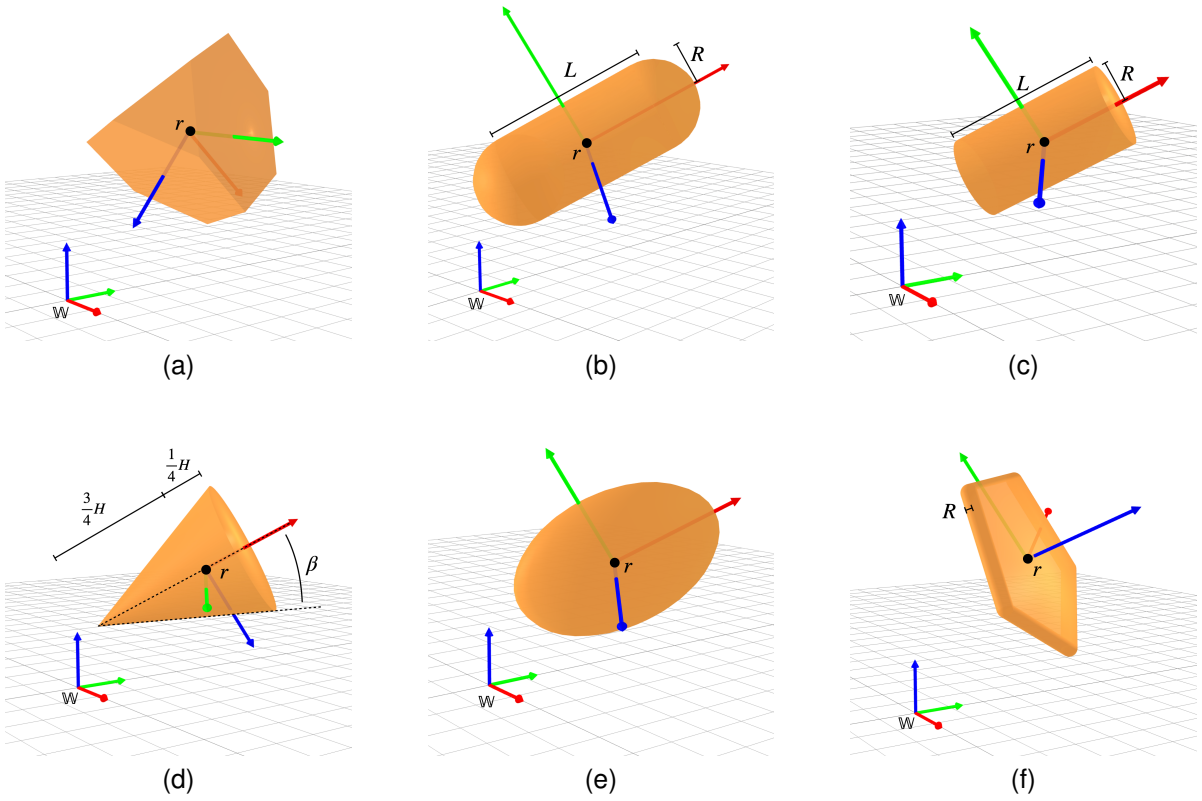

Fig. 2. Geometric descriptions of the six primitive shapes that are compatible with this differentiable collision detection algorithm. These shapes include a polytope (a), capsule (b), cylinder (c), cone (d), ellipsoid (e), and padded polygon (f). Collision information including the collision status as well as the contact points can be computed between any of two of these primitives using DCOL.

where the minimum scaling α is computed such that x is in the interior of both of the scaled primitives, making x an intersection point. This optimization problem is convex, bounded, and feasible for all of the primitives described in this paper. The boundedness comes from the constraint $\alpha \geq 0$, and the guarantee of feasibility comes from the fact that each object is uniformly scaled, so that in the limit $\alpha \to \infty$ each shape will encompass the entirety of $\mathbf{R}^3$, guaranteeing an intersection between objects. Another benefit to this problem formulation is that the only time the minimum scaling $\alpha = 0$ is when the origins of the two objects are coincident, in which the problem and its derivatives are still well defined.

B. Primitives

This section details the constraints that define set membership for each of the six scaled primitives. Each object is defined with an attached body reference frame $\mathbb{B}$ with an origin $r \in \mathbf{R}^3$ expressed in a world frame $\mathbb{W}$. The uniform scaling of these objects is always centered about this position r, and when the scaling parameter α is 0, the object is simply a point centered at r. The orientation of an object is defined by a rotation matrix ${}^{\mathbb{W}}Q^{\mathbb{B}} \in \mathbf{R}^{3\times3}$ relating the world frame to the object-fixed body frame, denoted as Q for shorthand. For each primitive, the constraints are also explicitly written in standard conic form for direct inclusion in our custom conic solver in the form of (1), where $h - Gx \in \mathcal{K}$.

1) Polytope: A polytope is a convex shape in $\mathbf{R}^3$ defined by a set of halfspace constraints, an example of which is shown in Fig. 2a. This polytope is described as the set of points $w \in \mathbf{R}^3$ such that $Aw \leq b$ for w expressed in $\mathbb{B}$, where $A \in \mathbf{R}^{m\times3}$ and $b \in \mathbf{R}^m$ represent the m halfspace constraints comprising the polytope. This polytope can be scaled by α, resulting in the following constraint for x to be inside the polytope:

$$AQ^T(x - r) \leq \alpha b. \tag{10}$$

The scaling parameter α scales the vector b, resulting in uniform scaling of all halfspace constraints and subsequent uniform scaling of the polytope. This constraint in standard form is the following:

$$\begin{bmatrix} AQ^T r \end{bmatrix} - \begin{bmatrix} AQ^T & -b \end{bmatrix} \begin{bmatrix} x \\ \alpha \end{bmatrix} \in \mathbf{R}_+. \tag{11}$$

2) Capsule: A capsule can be defined by the set of points within some radius R of a line segment, as shown in Fig. 2b. This internal line segment is along the x axis of the attached reference frame $\mathbb{B}$, and the end points of this line segment are some distance L apart. The scaled constraints for this primitive are that the point x must be within a scaled radius of the line segment, where the distance of the endpoints of

the line segment from r is also scaled:

$$\|x - (r + \gamma \hat{b}_x)\|_2 \leq \alpha R, \tag{12}$$

$$-\alpha \frac{L}{2} \leq \gamma \leq \alpha \frac{L}{2}, \tag{13}$$

where $\hat{b}_x = Q[1,0,0]^T$, and $\gamma \in \mathbf{R}$ is a slack variable. These constraints contain a linear inequality and one second-order cone constraint, shown here in standard form:

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0_{1\times 3} & -L/2 & 1 \\ 0_{1\times 3} & -L/2 & -1 \end{bmatrix} \begin{bmatrix} x \\ \alpha \\ \gamma \end{bmatrix} \in \mathbf{R}^2_+, \tag{14}$$

$$\begin{bmatrix} 0 \\ -r \end{bmatrix} - \begin{bmatrix} 0_{1\times 3} & -R & 0 \\ -I_3 & 0_{3\times 1} & \hat{b}_x \end{bmatrix} \begin{bmatrix} x \\ \alpha \\ \gamma \end{bmatrix} \in \mathcal{Q}_4. \tag{15}$$

3) Cylinder: The description of a cylinder is shown in Fig. 2c, with an orientation, a radius R, and a length L. The constraints for this primitive are the same as for the capsule in equations (12) and (13), with the introduction of two new scaled halfspace constraints that give the cylinder its flat ends:

$$[x - (r - \alpha \frac{L}{2} \hat{b}_x)]^T \hat{b}_x \geq 0, \tag{16}$$

$$[x - (r + \alpha \frac{L}{2} \hat{b}_x)]^T \hat{b}_x \leq 0. \tag{17}$$

These constraints in standard form include those shown in equations (14) and (15) with the following addition:

$$\begin{bmatrix} -\hat{b}_x^T r \\ \hat{b}_x^T r \end{bmatrix} - \begin{bmatrix} -\hat{b}_x^T & -L/2 & 0 \\ \hat{b}_x^T & -L/2 & 0 \end{bmatrix} \begin{bmatrix} x \\ \alpha \\ \gamma \end{bmatrix} \in \mathbf{R}^2_+. \tag{18}$$

4) Cone: As shown in Fig. 2d, a cone can be described with a height H, and a half angle β. The origin of the object-fixed frame r is one-quarter of the way from the flat face to the point of the cone, and α scales the distance of these two ends from the center point r:

$$\|\tilde{x}_{2:3}\|_2 \leq \tan(\beta)\tilde{x}_1, \tag{19}$$

$$(x - r - \alpha \frac{H}{4} \hat{b}_x)^T \hat{b}_x \leq 0, \tag{20}$$

where $\tilde{x} = Q^T(x - r + \alpha \frac{3H}{4} \hat{b}_x)$. These constraints in standard form are:

$$\begin{bmatrix} \hat{b}_x^T r \end{bmatrix} - \begin{bmatrix} \hat{b}_x^T & -H/4 \end{bmatrix} \begin{bmatrix} x \\ \alpha \end{bmatrix} \in \mathbf{R}^m_+, \tag{21}$$

$$\begin{bmatrix} -EQ^T r \end{bmatrix} - \begin{bmatrix} -EQ^T & v \end{bmatrix} \begin{bmatrix} x \\ \alpha \end{bmatrix} \in \mathcal{Q}_3, \tag{22}$$

where $E = \mathrm{diag}(\tan\beta, 1, 1)$ and $v = (-\frac{3H}{4}\tan\beta, 0, 0)$.

5) Ellipsoid: An ellipsoid, shown in Fig. 2e, can be described by a quadratic inequality $x^T P x \leq 1$, where $P \in \mathbf{S}^n_{++}$ is strictly positive definite and has an upper-triangular Cholesky factor $U \in \mathbf{R}^{n\times n}$ [11]. From here, a scaled ellipsoid with arbitrary position and orientation can be expressed in the following way:

$$\|UQ^T(x - r)\|_2 \leq \alpha, \tag{23}$$

where a sphere of radius R is just a special case of an ellipsoid with $P = I/R^2$. These constraints can be written in standard form as:

$$\begin{bmatrix} 0 \\ -UQ^T r \end{bmatrix} - \begin{bmatrix} 0_{1\times 3} & -1 \\ -UQ^T & 0_{3\times 1} \end{bmatrix} \begin{bmatrix} x \\ \alpha \end{bmatrix} \in \mathcal{Q}_4. \tag{24}$$

6) Padded Polygon: A "padded" polygon is defined as the set of points within some radius R of a two-dimensional polygon. Shown in Fig. 2f, the first two basis vectors of $\mathbb{B}$ span the polygon, and the polygon itself is defined with a slack variable $y \in \mathbf{R}^2$, and $Cy \leq \alpha d$, where $C \in \mathbf{R}^{m\times 2}$, and $d \in \mathbf{R}^m$, describe the m halfspace constraints for the polygon. This polygon is scaled in the same fashion as the polytope, and results in the following constraints:

$$\|x - (r + \tilde{Q}y)\|_2 \leq \alpha R, \tag{25}$$

$$Cy \leq \alpha d, \tag{26}$$

where $\tilde{Q} \in \mathbf{R}^{3\times 2}$ is the first two columns of Q. These constraints can be represented in standard form as the following:

$$\begin{bmatrix} 0_m \end{bmatrix} - \begin{bmatrix} 0_{m\times 3} & -d & C \end{bmatrix} \begin{bmatrix} x \\ \alpha \\ y \end{bmatrix} \in \mathbf{R}^m_+, \tag{27}$$

$$\begin{bmatrix} 0 \\ -r \end{bmatrix} - \begin{bmatrix} 0_{1\times 3} & -R & 0_{1\times 2} \\ -I_3 & 0_{3\times 1} & \tilde{Q} \end{bmatrix} \begin{bmatrix} x \\ \alpha \end{bmatrix} \in \mathcal{Q}_4. \tag{28}$$

C. Contact Points and Minimum Distance

While the computation of contact points and minimum distance between primitives is not needed for any of the examples in Section IV, they are easy to compute with DCOL if desired. The intersection point on the two scaled primitives is referred to as x^*, but unless $\alpha^* = 1$, this point does not exist on the surface of the primitives. The corresponding contact point for primitive i, $p_i \in \mathbf{R}^3$, is calculated using the optimal x^* and α^* from (9) as

$$p_i = r_i + \frac{x^* - r_i}{\alpha^*}, \tag{29}$$

where the intersection point between the scaled primitives is simply scaled back to each unscaled primitive. The distance between these points can also be calculated as follows:

$$\|d\|_2 = \|p_1 - p_2\|_2 = \|r_1 - r_2 + \frac{r_2 - r_1}{\alpha}\|_2. \tag{30}$$

Both of these operations are fully differentiable given the derivatives from DCOL, allowing for the calculation of the sensitivities of the contact points with respect to the configurations of the primitives.

IV. Examples

In this section, we demonstrate the utility of differentiable collision detection in trajectory optimization problems where contact is to be avoided, and in physics simulation with contact where exact and differentiable collision information is required. In both of these applications, a collision constraint $\alpha \geq 1$ is used to enforce no interpenetration between each pair of primitives, where α is the minimum scaling from DCOL.

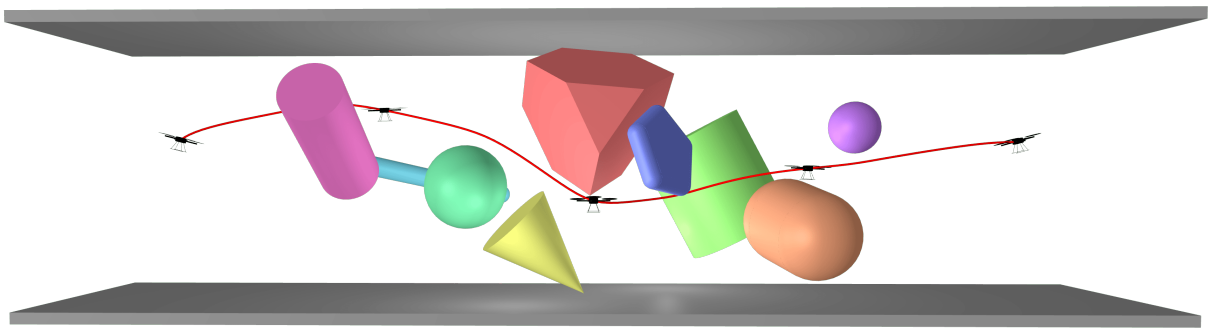

Fig. 3. Trajectory optimization for a 6-DOF quadrotor as it moves from left to right through a cluttered hallway. The collision constraints were represented with DCOL, and the trajectory optimizer was initialized with a static hover at the initial condition.

A. Trajectory Optimization

Trajectory optimization is a powerful tool in motion planning and control, where a numerical optimization problem is formulated to solve for a constrained trajectory that minimizes a cost function. A generic trajectory optimization problem with collision avoidance constraints from DCOL is as follows:

$$\begin{aligned} \underset{x_{1:N},u_{1:N-1}}{\text{minimize}} \quad & \ell_N(x_N) + \textstyle\sum_{k=1}^{N-1} \ell_k(x_k, u_k) \\ \text{subject to} \quad & x_{k+1} = f_k(x_k, u_k), \\ & h_k(x_k, u_k) \leq 0, \\ & g_k(x_k, u_k) = 0, \\ & \alpha_k(x_k) \geq 1, \end{aligned} \tag{31}$$

where k is the time step, x_k and u_k are the state and control inputs, ℓ_k and ℓ_N are the stage and terminal costs, $f(x_k, u_k)$ is the discrete dynamics function, $h_k(x_k, u_k)$ and $g_k(x_k, u_k)$ are inequality and equality constraints, and $\alpha_k(x_k)$ are the collision avoidance constraints from DCOL. Problems of this form can be solved with general purpose nonlinear program solvers like SNOPT [21], and Ipopt [22], or more specialized solvers like ALTRO [23], [24].

A key requirement for any gradient-based solver used to solve (31) is the ability to differentiate all of the cost and constraint functions with respect to the state and control inputs. This requirement has made collision-avoidance constraints difficult to incorporate into trajectory optimization frameworks because traditional collision detection methods are non-differentiable. In this section, DCOL is used to formulate collision-avoidance constraints in trajectory optimization problems to solve for collision-free trajectories.

1) "Piano Mover" Problem: The first problem we will look at is a variant of the *"Piano Mover"* problem, where a piano must maneuver around a 90-degree turn in a hallway [25], [26]. The walls are 1 meter apart, and the "piano" (a line segment) is 2.6 meters long, making the path around the corner nontrivial. This problem is solved with trajectory optimization and collision constraints, where the piano is parameterized as a cylindrical rigid body in two dimensions, with a position and orientation, and the hallway is modeled with polytopes. The solution to this problem is shown in Fig. 4, where the piano successfully maneuvers around the tight corner and reaches the goal state without traveling through any of the walls. The trajectory optimizer was initialized with the piano in a static pose at the initial condition. [25] [26]

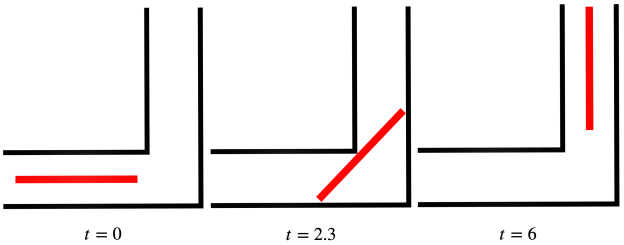

Fig. 4. The *"Piano Movers"* problem, where a "piano" (red rectangle) has to make a turn down a hallway, is solved with trajectory optimization. The piano and the walls are modeled as rectangular prisms. DCOL was used to represent all of the collision avoidance constraints that ensure the piano cannot travel through the wall, and the trajectory optimizer was able to converge on a feasible trajectory to deliver the piano to the goal state.

2) Quadrotor: Motion planning for quadrotors has received significant attention in recent years [27]–[29], with collision avoidance featured in many of these works [30]–[32]. With DCOL, we are able to directly and exactly incorporate collision avoidance constraints into a quadrotor motion planner to solve for trajectories through cluttered environments. In this example, we use trajectory optimization for a classic 6-DOF quadrotor model from [27], [33] to solve for a trajectory that traverses a cluttered hallway with 12 objects in it, shown in Fig. 3. The solver was initialized with the quadrotor hovering at the initial condition, and a spherical outer approximation of the quadrotor geometry was used to compute collisions. Despite this naive guess, the solver was able to quickly converge on a collision-free trajectory through the obstacles.

3) Cone Through an Opening: This example demonstrates how trajectory optimization with DCOL can route a cone through a square hole in a wall, as shown in Fig. 5. The dynamics of the cone are modeled as a rigid body with full translational and rotational control, and the wall is comprised of four rectangular prisms, making a rectangular opening in the wall. The trajectory optimizer converged on a solution where the cone successfully passes through the opening in the wall, requiring that the cone slewed its orientation and "squeezed through" the opening. This example demonstrates the importance of the differentiability of the

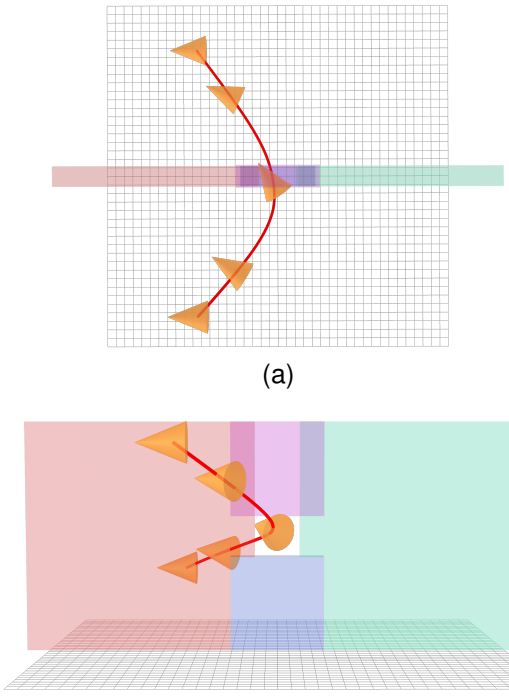

Fig. 5. Trajectory optimization for a cone (orange) with translation and attitude control as it travels through a square opening in a wall. Top-down and side views are shown in (a) and (b), respectively. The cone is forced to slew to an attitude that allows for the passing of the cone through the opening before returning to the initial attitude. The trajectory optimizer was simply initialized with the static initial condition.

collision avoidance constraints, as the optimizer was forced to leverage both translational and rotational manipulation of the cone in order to successfully pass through the opening. As with the previous two examples, there was no expert initial guess provided to the trajectory optimizer, just a static initial condition.

B. Contact Physics

Another application of differentiable collision detection in robotics is contact physics for simulation. Rigid-body mechanics with inelastic collisions can be simulated using complementarity-based time-stepping schemes [34], where stationary points of a discretized action integral are solved for subject to contact constraints [35]. Normally these constraints are limited to traditionally differentiable ones like those between fixed contact points and a floor. The differentiability of DCOL enables these same methods to be extended for simulating contact between convex primitives, as shown in Fig. (6) where twelve primitives collide. In terms of computation times, using DCOL for contact physics is reasonable given each constraint evaluation and differentiation are usually less than 10 μs as shown in Table I.

V. Conclusion

We have presented DCOL, a fast differentiable collision detection algorithm capable of computing useful collision information and derivatives for pairs of any of six convex primitives. By formulating the collision-detection problem

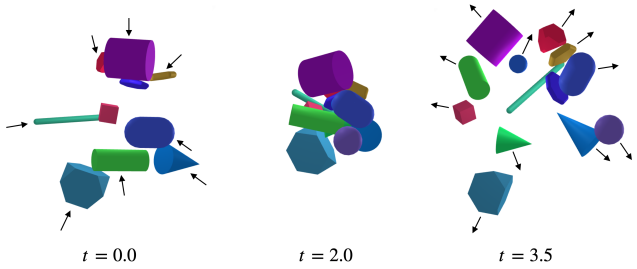

Fig. 6. Contact physics with differentiable collision constraints embedded in a complementarity-based time-stepping scheme, simulated at 100 Hz. Twelve convex objects are started at random positions with velocities pointing towards the origin at $t = 0$. The objects impact each other at $t = 2$ and spread out again by $t = 3.5$. Despite the complexity of the simulation, the collision constraints can be enforced to machine precision and the integration is stable.

as an optimization problem that solves for the minimum uniform scaling that must be applied to each primitive before an intersection occurs, a surrogate proximity value is returned that is informative for primitives with or without a collision. Using differentiable convex optimization and a primal-dual interior-point conic solver, smooth derivatives of this optimization problem are returned after convergence with very little additional computation. The utility of DCOL is demonstrated in a wide variety of robotics applications, including motion planning and contact physics, where collision derivatives are required. Future work includes methods for convex decompositions of complex shapes as well as the incorporation of DCOL into existing physics engines. Our open-source Julia implementation of DCOL is available at `https://github.com/kevin-tracy/DCOL.jl`.

References

[1] E. Gilbert, D. Johnson, and S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal on Robotics and Automation*, vol. 4, no. 2, pp. 193–203, Apr. 1988.

[2] S. Cameron, "Enhancing GJK: Computing minimum and penetration distances between convex polyhedra," in *Proceedings of International Conference on Robotics and Automation*, vol. 4, Albuquerque, NM, USA: IEEE, 1997, pp. 3112–3117.

[3] G. Snethen, "XenoCollide: Complex Collision Made Simple," *undefined*, 2008.

[4] J. Newth, "Minkowski Portal Refinement and Speculative Contacts in Box2D," Master of Science, San Jose State University, San Jose, CA, USA, Apr. 2013.

[5] G. Van Den Bergen, "Proximity Queries and Penetration Depth Computation on 3D GAme Objects," 2001.

[6] J. Pan, S. Chitta, and D. Manocha, "FCL: A general purpose library for collision and proximity queries," in *2012 IEEE International Conference on Robotics and Automation*, St Paul, MN, USA: IEEE, May 2012, pp. 3859–3866.

[7] E. Coumans, "Bullet Physics Simulation," in *SIGGRAPH*, Los Angeles: ACM, 2015.

[8] R. Tedrake and The Drake Development Team, "Drake: Model-based design and verification for robotics," 2019.

[9] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman, and C. Karen Liu, "DART: Dynamic Animation and Robotics Toolkit," *The Journal of Open Source Software*, vol. 3, no. 22, p. 500, Feb. 2018.

[10] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[11] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[12] L. Vandenberghe, "The CVXOPT linear and quadratic cone program solvers," p. 30,

[13] A. Domahidi, E. Chu, and S. Boyd, "ECOS: An SOCP solver for embedded systems," in *2013 European Control Conference (ECC)*, Zurich: IEEE, Jul. 2013, pp. 3071–3076.

[14] Y. E. Nesterov and M. J. Todd, "Self-Scaled Barriers and Interior-Point Methods for Convex Programming," *Mathematics of Operations Research*, vol. 22, no. 1, pp. 1–42, Feb. 1997.

[15] E. Andersen, C. Roos, and T. Terlaky, "On implementing a primal-dual interior-point method for conic quadratic optimization," *Mathematical Programming*, vol. 95, no. 2, pp. 249–277, Feb. 2003.

[16] Y. E. Nesterov and M. J. Todd, "Primal-Dual Interior-Point Methods for Self-Scaled Cones," *SIAM Journal on Optimization*, vol. 8, no. 2, pp. 324–364, May 1998.

[17] Mosek ApS, "The MOSEK optimization software," Tech. Rep., 2014.

[18] A. Agrawal, S. Barratt, S. Boyd, W. M. Moursi, and E. Busseti, "Differentiating Through a Conic Program," *Journal of Applied and Numerical Optimization*, vol. 1, no. 2, pp. 107–115, 2019.

[19] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and Z. Kolter, "Differentiable Convex Optimization Layers," *Advances in Neural Information Processing Systems*, pp. 9558–9570, 2019.

[20] B. Amos and J. Z. Kolter, "OptNet: Differentiable Optimization as a Layer in Neural Networks," *arXiv:1703.00443 [cs, math, stat]*, Oct. 2019.

[21] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," *SIAM Review*, vol. 47, no. 1, pp. 99–131, Jan. 2005.

[22] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, Mar. 2006.

[23] T. A. Howell, B. E. Jackson, and Z. Manchester, "ALTRO: A Fast Solver for Constrained Trajectory Optimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Macau, China, Nov. 2019.

[24] B. E. Jackson, T. Punnoose, D. Neamati, K. Tracy, and R. Jitosho, "ALTRO-C: A Fast Solver for Conic Model-Predictive Control," in *International Conference on Robotics and Automation (ICRA)*, Xi'an, China, 2021, p. 8.

[25] D. Wilson, J. H. Davenport, M. England, and R. Bradford, "A "Piano Movers" Problem Reformulated," in *2013 15th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, Sep. 2013, pp. 53–60.

[26] J. T. Schwartz and M. Sharir, "On the "piano movers"' problem I. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers," *Communications on Pure and Applied Mathematics*, vol. 36, no. 3, pp. 345–398, May 1983.

[27] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation (ICRA)*, May 2011, pp. 2520–2525.

[28] D. Mellinger, N. Michael, and V. Kumar, "Trajectory Generation and Control for Precise Aggressive Maneuvers with Quadrotors," p. 13,

[29] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza, "A Comparative Study of Nonlinear MPC and Differential-Flatness-Based Control for Quadrotor Agile Flight," *IEEE Transactions on Robotics*, pp. 1–17, 2022.

[30] D. Falanga, K. Kleber, and D. Scaramuzza, "Dynamic obstacle avoidance for quadrotors with event cameras," *Science Robotics*, vol. 5, no. 40, eaaz9712, Mar. 2020.

[31] R. Penicka, Y. Song, E. Kaufmann, and D. Scaramuzza, "Learning Minimum-Time Flight in Cluttered Environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7209–7216, Jul. 2022.

[32] H. Shraim, A. Awada, and R. Youness, "A survey on quadrotors: Configurations, modeling and identification, control, collision avoidance, fault diagnosis and tolerant control," *IEEE Aerospace and Electronic Systems Magazine*, vol. 33, no. 7, pp. 14–33, Jul. 2018.

[33] B. Jackson, T. Howell, K. Shah, M. Schwager, and Z. Manchester, "Scalable Cooperative Transport of Cable-Suspended Loads with UAVs using Distributed Trajectory Optimization," in *International Conference on Robotics and Automation*, Paris, France, Jun. 2020, p. 8.

[34] T. A. Howell, S. Le Cleac', J. Z. Kolter, M. Schwager, and Z. Manchester, "Dojo: A Differentiable Simulator for Robotics," 2022.

[35] J. E. Marsden and M. West, "Discrete Mechanics and Variational Integrators," *Acta Numerica*, vol. 10, pp. 357–514, 2001.