Article

The International Journal of Robotics Research
2023, Vol. 42(6) 433–458

DOI: 10.1177/02783649231170897
journals.sagepub.com/home/ijr
Sage

Autogenerated manipulation primitives

Eric Huang, Xianyi Cheng, Yuemin Mao*, Arnav Gupta* and Matthew T Mason

Abstract

The central theme in robotic manipulation is that of the robot interacting with the world through physical contact. We tend to describe that physical contact using specific words that capture the nature of the contact and the action, such as grasp, roll, pivot, push, pull, tilt, close, open etc. We refer to these situation-specific actions as manipulation primitives. Due to the nonlinear and nonsmooth nature of physical interaction, roboticists have devoted significant efforts towards studying individual manipulation primitives. However, studying individual primitives one by one is an inherently limited process, due engineering costs, overfitting to specific tasks, and lack of robustness to unforeseen variations. These limitations motivate the main contribution of this paper: a complete and general framework to autogenerate manipulation primitives. To do so, we develop the theory and computation of contact modes as a means to classify and enumerate manipulation primitives. The contact modes form a graph, specifically a lattice. Our algorithm to autogenerate manipulation primitives (AMP) performs graph-based optimization on the contact mode lattice and solves a linear program to generate each primitive. We designed several experiments to validate our approach. We benchmarked a wide range of contact scenarios and our pipeline's runtime was consistently in the 10 s of milliseconds. In simulation, we planned manipulation sequences using AMP. In the real-world, we showcased the robustness of our approach to real-world modeling errors. We hope that our contributions will lead to more general and robust approaches for robotic manipulation.

Keywords

Contact modeling, dexterous manipulation, kinematics, dynamics

1. Introduction

Let us assume, for the purposes of a thought experiment, that everyone in the world owns a futuristic home-service robot. Suppose robots A, B, and C are all tasked with picking up packages recently delivered to their respective owners. Robot A walks out the front door and locates the package which is a medium-sized cardboard box. The robot slips its hands into the box's hand-holds, lifts the box into the air, and walks back home. Robot B is greeted with a shrink-wrapped case of beverages outside the front door. Since there are no hand-holds, the robot first attempts to lift the case of beverages by compressing opposing sides of the case with its hands. However, the weight of the beverages cause the hand contacts to slip and the case to thump back onto the ground. Robot B is forced to devise a different strategy. Using its hands on the left and right sides, the robot tilts the case onto the left edge and slides its right hand underneath the right edge. Next, the robot tilts the case forward onto the front edge and slides the left hand underneath. Now that both hands are underneath with no chance of slippage, robot B lifts the package and returns indoors. Robot C is greeted with a rolled up rug leaning against the doorway. The rug is too heavy to pick up and the robot must pivot this column-shaped item until it is resting on the ground. Through a long sequence of pushes and pulls, the robot incrementally maneuvers the rug until the short end is facing the door. Finally, robot C drags the rug indoors.

The above vignettes about robots A, B, and C illustrate the need for a library of robotic manipulation primitives which can generalize across objects and scenarios. In this work, we define a manipulation primitive as a basic robot action which can serve as a building block for more complex behaviors. Examples of manipulation primitives include pushing, pulling, grasping, lifting, pivoting, throwing, etc. Robot A represented an ideal case where the robot could use a single manipulation primitive to accomplish its task. Robot B encountered a significant model error during execution. The additional weight of the beverage case caused the bimanual compressive grasp

Carnegie Mellon University, Pittsburgh, PA, USA

Yuemin Mao and Arnav Gupta contributed equally

Corresponding author:
Xianyi Cheng, Department of Mechanical Engineering, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213-3890, USA.
Email: xianyic@andrew.cmu.edu

to fail. However, the robot was able to robustly recover from the model failure by using a different set of primitives from its library. Robot C illustrated how long sequences of manipulation primitives are required to accomplish difficult tasks. We can plan such a sequence using search-based algorithms and a library of manipulation primitives. The goal of this paper is to introduce a method for automatically generating a library of robotic manipulation primitives.

This special issue publication extends and revises the material first presented in Huang et al. (2020). The previous publication presented the first-known theoretical and computational framework for contact mode enumeration in 3D environments. In a system of rigid bodies with friction, *contact modes* are an interpretable, but not quite semantic, description of rigid body motion which classify the relative motion at each contact point as *contacting* or *separating* and *sliding* or *sticking*. The contact mode encodes the velocity and, therefore, the dynamics of the system at instantaneous point in time. This framework forms the basis for our efforts towards the types of dexterous manipulation exemplified by robots A, B, and C. As a first contribution, this paper improves the computational complexity of sliding-sticking contact mode enumeration by a d-th-root factor compared to the previously state-of-the-art algorithm published in Huang et al. (2020). For our main contribution, this paper extends the theoretical and computational framework of contact modes into a principled method to autogenerate robotic manipulation primitives (AMP). The AMP method generates a library of manipulation primitives by enumerating the contact modes of a target object and selecting the contact modes, or rather, manipulation primitives, which are quasi-statically feasible. In this work, we use contact modes to parameterize the set of possible manipulation primitives. This parameterization is based on the following idea. First, the contact modes cover the entire tangent space of the object. Therefore, the set of contact modes represents all the ways an object can be moved. Second, under a polyhedral friction model, each contact mode parameterizes a convex region of the state space with smooth dynamics. Therefore, given a contact mode and a convex cost function, we can quickly and easily find a sequence of controls to move the object in a desired direction i.e., the manipulation primitive, using a direct single shooting method (Kelly 2017). Because the dynamics are convex within each contact mode, the AMP method can generate an entire library of manipulation primitives in tens of milliseconds.

The rest of this paragraph outlines the remainder of this paper and highlights new additions with respect to Huang et al. (2020). Section 2 discusses related work directly related to this paper's technical contributions. Next, contact modes are formally introduced in Section 3. This section is the theoretical foundation for the rest of the paper because it enables us to view contact modes as kinematic, geometric, and combinatorial structures. Sections 4 and 5 are concerned with the enumeration of contact modes. The latter also describes a new partial hyperplane arrangement algorithm for sliding-sticking mode enumeration. To prepare for autogeneration of manipulation primitives, Section 6 interprets contact-modes from the perspective of frictional contact dynamics. It is shown that the non-convex structure of linearized frictional contact dynamics can be precisely decomposed into piece-wise convex components i.e., into contact modes. Section 7 introduces an algorithm to autogenerate manipulation primitive (AMP). Section 8 examines the validity of our approach through a series of carefully designed experiments. Section 9 discusses limitations and future work. Finally, Section 10 presents concluding remarks.

2. Related work

2.1. Enumeration of contact modes

This section discusses related work specific to contact mode enumeration. Mason (2001) sketched an algorithm for contact mode enumeration in 2D for a single rigid body that intersects the positive (negative) rotation centers on the positive (negative) oriented plane and intersects the rotation centers at infinity on the equator. Though Mason (2001) upper-bounded the number of modes at $O(n^2)$, by our analysis, the algorithm's runtime is actually $O(n \log n)$ and the correct number of modes is $\Theta(n)$. Unfortunately, the oriented plane technique does not generalize to contact mode enumeration in 3D. Later, Haas-Heger et al. (2018) independently published an algorithm for partial contact mode enumeration in 2D. There, they interpret the feasible modes as the regions of an arrangement of hyperplanes in 3D. However, Haas-Heger et al. (2018)'s algorithm is at least $\Omega(n^4)$ and does not enumerate separating modes. Disregarding these issues, Haas-Heger et al. (2018)'s work inspired us to investigate hyperplane arrangements in higher dimensions for our algorithm. Xiao and Ji (2001) created an algorithm for contacting-separating mode (cs-mode) enumeration in 3D for the specific case of convex-convex polyhedra with an explicit limit on the number of allowed contacts. The authors do not provide a runtime complexity of the algorithm[1]. However, their work does not enumerate sliding-sticking modes which are important for generating dynamically feasible motions. Neither does their work support non-convex polyhedra, which our work does. Our work in Huang et al. (2020) and in this paper represents the most efficient and general method for contact mode enumeration in 3D to date.

It is well known that frictional contact problems can be modeled as a complementarity problem or equivalently, a variational inequality Facchinei and Pang (2007). Within that theory, it is known that the normal manifold (which is a linear hyperplane arrangement) divides the solution space of an affine variational inequality Facchinei and Pang (2007). Not surprisingly, we found related papers in other fields containing problems that can be modeled as variational inequalities Geyer et al. (2010); Potočnik et al. (2004).

For example, in the study of digital controllers and power electronics, Geyer et al. (2010) proposed a mode enumeration algorithm for compositional hybrid systems based on the reverse search technique of Avis and Fukuda (1996). The contribution of our work (and theirs) is in presenting the theory in an understandable manner for our field and optimizing the relevant algorithms for our specific problem formulation.

The existence of efficient contact mode enumeration algorithms which are polynomial in the number of contacts is not widely appreciated in the literature. For instance, Greenfield et al. (2005) used the exponential time algorithm for contact mode enumeration in 2D. The work in Li et al. (2015) proposed a 3D contact mode state estimation pipeline that generates the contact modes offline using the naive exponential enumeration method. Johnson and Koditschek (2013) published an optimization-based technique for legged robot leaping which searched along the contacting-separating mode graph for a sequence of foot takeoffs. Their application was sufficiently simple that they could enumerate the modes by hand.

2.2. Autogeneration of manipulation primitives

Our work is based on the linear complementarity formulation of frictional contact. Horak and Trinkle (2019) compared various popular frictional contact models used in rigid body simulation, including the linear complementarity formulation and the convex relaxation of Todorov (2014), and concluded that, apart from some minor artifacts, the qualitative behaviors of the different models were very similar. Past works have modeled contact dynamics as a complementarity problem (Stewart and Trinkle 1996). Several methods exist for solving complementarity problems, which we categorize as iterative or direct solution methods. A popular iterative method is projected Gauss-Seidel (PGS) (Murty and Yu 1988). Stewart and Trinkle (1996) used a direct pivoting method, Lemke's algorithm, for solving contacts modeled as linear complementarity problems.

The theory and algorithms introduced in this paper were also motivated by the issue of multi-modal constrained planning for robotic manipulation. Of the families of methods for constrained motion planning reviewed by Kingston et al. (2018), our approach falls into the families of tangent space and atlas methods. Our publications present the first general tangent space method for sampling-based robotic manipulation planning in 2D (Cheng et al., 2021) and 3D (this paper). Prior work in robotic manipulation planning often focused on specific manipulation primitives. In the early work of in-hand regrasping Leveroni (1997), grasp gaits are obtained from searching on a grasp map developed from the force closure model. For nonprehensile two-palm manipulation, Erdmann (1998) generated the set of possible motions by partitioning the configuration space through primitive operations under different contact modes, and generate motion plans by searching. More recently, Chavan-Dafle et al. (2018) combined high-level sampling-based planning with the motion cones to achieve prehensible in-hand manipulation with external pushes. Hou et al. (2018) proposed a fast planning framework using two reorientation motion primitives for object reorientation problems. Quite often, researchers end up designing and implementing their own manipulation primitives (Michelman and Allen 1994; Barry et al., 2013; King et al., 2015, 2016; Hogan et al., 2020). Our work in autogenerating manipulation primitives can greatly reduce the engineering effort required for robotic manipulation planning systems.

Our work is closely related to contact-implicit trajectory optimization (CITO) (Posa et al., 2014). The output of a CITO algorithm is a single trajectory which has automatically determined the contact modes at each timestep. One major issue in CITO is how to solve the non-linear complementarity constraints generated by frictional contact within an optimization framework such as SNOPT, IPOPT, mixed integer convex programming (MICP), differential dynamic programming (DDP), etc. (Önol et al., 2018; Posa et al., 2014; Tassa et al., 2012; Kelly 2017; Landry et al., 2019; Manchester et al., 2019). Whereas CITO views complementarity constraints as a problem to be solved within the trajectory optimizer, our work takes the novel viewpoint that the complementarity constraints should be solved outside of the trajectory optimizer. This approach enables us to enumerate the solutions to the complementarity constraints in tens of milliseconds, and then perform fixed mode trajectory generation at a later stage. In addition, a library of manipulation primitives adds a level of robustness through action diversity which is difficult to obtain in a principled manner through CITO.

3. Theory of contact modes

This section reviews the background required to understand contact modes as kinematic, geometric, and combinatorial objects. We introduce the normal and tangent velocity constraints generated by rigid contacts and describe how they partition the space of generalized velocities into discrete contact modes. These constraints form geometric objects known as polyhedra and partial hyperplane arrangements. Furthermore, we demonstrate that contact modes exhibit a partial order. In particular, they form a geometric lattice. The concepts introduced in this section will be used later on in developing enumeration and autogeneration algorithms. We highly recommend working through Figure 1 while reading this section. In addition, Multimedia Extension one provides an animated overview of the kinematics, geometry, and combinatorics of contact modes.

3.1. Contact normal velocity

In a rigid body model of the world, two rigid bodies in collision cannot penetrate one another. To a first-order approximation, this generates a linear constraint on their relative velocities with respect to the contact normal.

This normal velocity constraint can be derived as follows. Let α and β be two rigid bodies in collision at a point $c \in \mathbb{R}^3$. Let $g_{wc} \in \mathrm{SE}(3)$ be the contact frame at a contact point c with z-axis pointing along β's surface normal towards α. Let $\phi \in \mathbb{R}_{\geq 0}$ be the distance between the two bodies. Taking the derivative of ϕ, we observe that at the time of contact, the velocity of the two rigid bodies must satisfy the constraint $0 \leq \dot{\phi}$. This suggests that the relative motion at the contact point is non-negative in direction of the contact normal. We can rewrite this in terms of the generalized (system) velocity $\dot{q} \in \mathbb{R}^d$

$$0 \leq n^T \left(J^b_{ac} + J^b_{\beta c} \right) \dot{q} \tag{1}$$

where d is the degrees of freedom of the system, n is the vector $[0, 0, 1, 0, 0, 0]^T$, $J^b_{ac} \in \mathbb{R}^{6 \times d}$ is the body Jacobian of α to the contact frame, and $J^b_{\beta c} \in \mathbb{R}^{6 \times d}$ is the body Jacobian of β to the contact frame. We define a contact to be *separating* if (1) is positive and *contacting* if (1) is equal to zero. We refer to this classification as the *contacting-separating mode* (cs-mode) at that contact. For a system with n contacts, we can concatenate (1) into a set of linear inequalities

$$0 \leq N\dot{q}, N \in \mathbb{R}^{n \times d} \tag{2}$$

where the i-th row of N is

$$N_{i,:} = n_i^T \left(J^b_{\alpha_i c_i} + J^b_{\beta_i c_i} \right) \dot{q} \tag{3}$$

3.2. Convex polyhedra

Convex polyhedra describe the region of space enclosed by normal velocity constraints. A *hyperplane* $h = (a, z)$ is the set $h = (a, z) = \{x \in \mathbb{R}^d : a^T x = z\}$. Associated with each hyperplane h are the positive and negative *halfspaces*, h^+ and h^-. An $\mathcal{H}$ *polyhedron* $P \subseteq \mathbb{R}^d$ is the intersection of closed positive halfspaces in the form $P = \mathcal{H}(A, z) = \{x \in \mathbb{R}^d : Ax \geq z\}$, for some $A \in \mathbb{R}^{n \times d}, z \in \mathbb{R}^n$. A $\mathcal{V}$ *polytope* $P \subseteq \mathbb{R}^d$ is a convex combination of points, i.e. $P = \mathcal{V}(A) = \{At : t \geq 0, \sum t = 1\}$ for some $A \in \mathbb{R}^{n \times d}$. A *face* F of a polyhedron P is any set of the form $F = \{x \in P : c^T x = c_0\}$ with $c \in \mathbb{R}^d, c_0 \in \mathbb{R}$, and $c^T x \geq c_0$ is true for all $x \in P$. Figure 2 depicts example polytope faces. The dimension of a face is defined as the dimension of its affine hull. The faces of dimensions $0, 1, d-2$, and $d-1$ are called *vertices*, *edges*, *ridges*, and *facets*, respectively. The *sign* of a face $F \subseteq \mathbb{R}^d$ with respect to a hyperplane h of P is defined as

$$\mathrm{sign}_h(F) = \begin{cases} + & \text{if} \quad F \subseteq h^+ \\ 0 & \text{if} \quad F \subseteq h \\ - & \text{if} \quad F \subseteq h^- \end{cases} \tag{4}$$

and the *signed vector* of a face F with respect to P is the vector

$$\mathrm{sign}(F) = \begin{bmatrix} \mathrm{sign}_{h_1}(F) & \cdots & \mathrm{sign}_{h_n}(F) \end{bmatrix} \tag{5}$$

where h_i is the i-th hyperplane of P. The relative interior of a face is the set

$$\mathrm{relint}(F) = \{x \in F : \exists \epsilon \text{s.t.} B_\epsilon(x) \cap \mathrm{aff}(F) \subseteq F\} \tag{6}$$

For a more complete introduction, one can refer to Ziegler (1995).

3.3. Contacting-separating modes

3.3.1. Theorem. For a system at a given state, let $N \in \mathbb{R}^{n \times d}$ be its normal velocity constraints. The contacting-

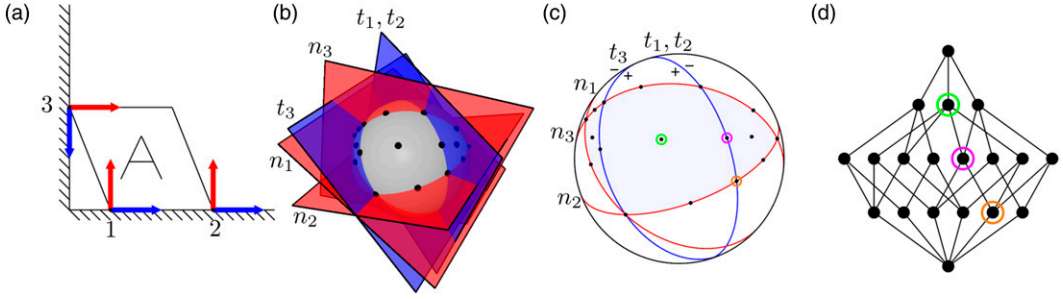

Figure 1. From velocity constraints to contact modes (left to right): (a) Input 2D scene with a parallelogram against the wall. The parallelogram's coordinate frame is at the geometric center of 'A' with x-axis to pointed right and y-axis pointed up. The normals and tangents are depicted in red and blue, respectively. (b) The normal and tangent velocity constraints are visualized as a hyperplane arrangement $\mathcal{A}([N; T])$ in the velocity space of the parallelogram, $[\dot{x}, \dot{y}, \dot{\theta}] \in \mathbb{R}^3$. The positive $\dot{x}$ axis is the ray along the visible intersection of n_1, n_2, t_3. The positive $\dot{y}$ axis is the ray along the visible intersection of t_1, t_2, n_3. The positive $\dot{\theta}$ axis is along the cross-product of the $\dot{x}$ axis and $\dot{y}$ axis. The gray sphere serves as a visual aid. Each region of space carved out by the hyperplanes corresponds to a face of the hyperplane arrangement. The faces which represent feasible (non-penetrating) velocities are marked with a black dot. The faces correspond exactly with the contact modes of the system and the black dots are example velocities which achieve that contact mode. (c) For clarity, the hyperplanes are visualized again as their projected geodesics on the visible hemisphere. The partial hyperplane arrangement $\mathcal{A}([N; T]) \cap \mathcal{H}(N, 0)$ is shaded light gray-blue. The light gray-blue region represents the non-penetrating velocities. (d) The face lattice $\mathcal{L}$ over the feasible faces in the partial hyperplane arrangement. This lattice captures the structure of the contacting-separating and sliding-sticking modes. The black dots in (c) highlighted with green, purple and orange correspond to the contact modes highlighted with their respective colors in the lattice in (d).

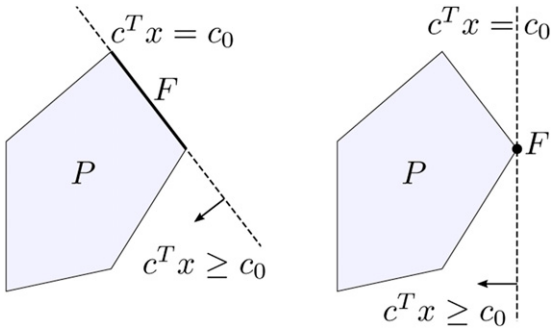

Figure 2. Faces of a two-dimensional polytope. A one dimensional face is an *edge* (left). A zero dimensional face is a *vertex* (right).

separating modes of this system are the faces of the convex polyhedra $\mathcal{H}(N, 0)$.

Proof. Let m_{cs} be a contacting-separating mode and let $\dot{q}_{cs}$ be a velocity which realizes this mode. Since $\mathcal{H}(N, 0)$ is the disjoint union of the relative interiors of its faces, $\dot{q}_{cs}$ must be contained in the relative interior of a single face. The sign vector of this face is unique and it is m_{cs}.

3.4. Contact tangent velocity

Coulomb friction is a simple model of dry friction which characterizes frictional forces based on the relative velocity of the contact point. The tangent velocity constraints are a set of hyperplanes that help us classify the relative velocity of the contact point.

We approximate the infinite tangent velocity directions by dividing the tangent plane into sectors of equal angles. Let k be the number of dividing planes (which generate $2k$ sectors). We define a basis matrix D such that its i-th column equals $[\cos i\pi/k, \sin i\pi/k, 0, 0, 0, 0]^T$. Given a generalized velocity $\dot{q}$, we can use the following equation to determine the discretized tangent velocity direction

$$D^T \left(J^b_{ac} + J^b_{\beta c}\right)\dot{q} \tag{7}$$

where J^b_{ac} and $J^b_{\beta c}$ are the body Jacobians from before. We define a contact to be *right-sliding* with respect to a tangent direction if its row in (6) is positive, *left-sliding* if its row is negative, and *sticking* if all rows are zero. We refer to this classification as the *sliding-sticking mode* (ss-mode) at that contact. For a system with n contacts, we can concatenate (6) into the tangent velocity constraints

$$T\dot{q}, T \in \mathbb{R}^{(nk)\times d} \tag{8}$$

3.5. Partial hyperplane arrangements

Hyperplane arrangements describe the regions of space partitioned by tangent velocity constraints. A *hyperplane arrangement* $\mathcal{A}(H)$ is a set of hyperplanes $H = (A, z)$, for some $A \in \mathbb{R}^{n\times d}, z \in \mathbb{R}^n$ which dissect $\mathbb{R}^d$ into different regions of space. The arrangement is *linear* when $z = 0$. The *signed vector* of a point p with respect to a hyperplane arrangement $\mathcal{A}(H)$ is $\text{sign}_{\mathcal{A}}(p) = \begin{bmatrix}\text{sign}_{h_1}(p) & \cdots & \text{sign}_{h_n}(p)\end{bmatrix}$ See Figure 3 for examples of hyperplane arrangements. The signed vectors of the points in $\mathbb{R}^d$ define equivalence classes known as the *faces* of $\mathcal{A}$. That is, given a signed vector $s \in \{+,0,-\}^n$, the associated face is $F = \{p \in \mathbb{R}^d : \text{sign}_{\mathcal{A}}(p) = s\}$ For a more complete introduction to hyperplane arrangements, one can refer to Edelsbrunner (2005). A *partial hyperplane arrangement* is a hyperplane arrangement with certain hyperplanes restricted to be one-sided i.e., to be halfspaces. We can express it as $\mathcal{A}(H)\cap\mathcal{H}(H', 0)$ where H' is a subset of hyperplanes in H which are one-sided.

3.6. Sliding-sticking modes

3.6.1. Theorem. For a system at a given state with cs-mode m_{cs}, let $N_c \in \mathbb{R}^{n_c\times d}$ be the normal velocity constraints which are maintaining contact, let $N_s \in \mathbb{R}^{n_s\times d}$ be the normal velocity constraints which are separating, and let $T_c \in \mathbb{R}^{(n_c k)\times d}$ be the active tangent velocity constraints. The sliding-sticking modes are the faces of the hyperplane arrangement $\mathcal{A}(T_c)$ which intersect the relative interior of the convex polyhedra $H_{cs} = \{N_c\dot{q} = 0, N_s\dot{q} \geq 0\}$.

Proof. Let m_{ss} be a sliding-sticking mode for a given contacting-separating mode m_{cs} and let $\dot{q}_{ss}$ be a velocity which realizes these modes. Since $\mathbb{R}^d$ is the disjoint union of the relative interiors of the faces of $\mathcal{A}(T_c)$, we know that the relative interior of H_{cs} can be written as the disjoint union of the non-empty intersections of the faces of $\mathcal{A}(T_c)$ and H_{cs}. As before, $\dot{q}_{ss}$ must be contained in an unique face of $\mathcal{A}(T_c)$ with sign vector m_{ss}.

3.6.2. Corollary. For a system at a given state, let $N \in \mathbb{R}^{n\times d}$, $\phi \in \mathbb{R}^n$ be its normal velocity constraints and let $T \in \mathbb{R}^{(nk)\times d}$ be its tangent velocity constraints. The sliding-sticking modes of this system are contained in the faces of the hyperplane arrangement $\mathcal{A}(T)$ which intersect the boundary of the convex polyhedra $\mathcal{H}(N, \phi)$.

3.7. Contact mode lattice

The *face lattice* describes the hierarchical structure of the faces of convex polyhedra and hyperplane arrangements. Consequently, it describes the hierarchical structure of contact modes and their adjacencies, i.e. it is the contact mode lattice. A *partially ordered set* is a set L and a binary relation $\leq$ such that for all u, v, $w \in L$

$$\begin{aligned} &u \leq u && \text{(reflexivity)} \\ &u \leq v \wedge v \leq w \Rightarrow u \leq w && \text{(transitivity)} \\ &u \leq v \wedge v \leq w \Rightarrow u \leq w && \text{(anti-symmetry)} \end{aligned} \tag{9}$$

Moreover, a pair of elements u, v are comparable if $u \leq v$ or $v \leq u$; otherwise, they are incomparable. The faces of a

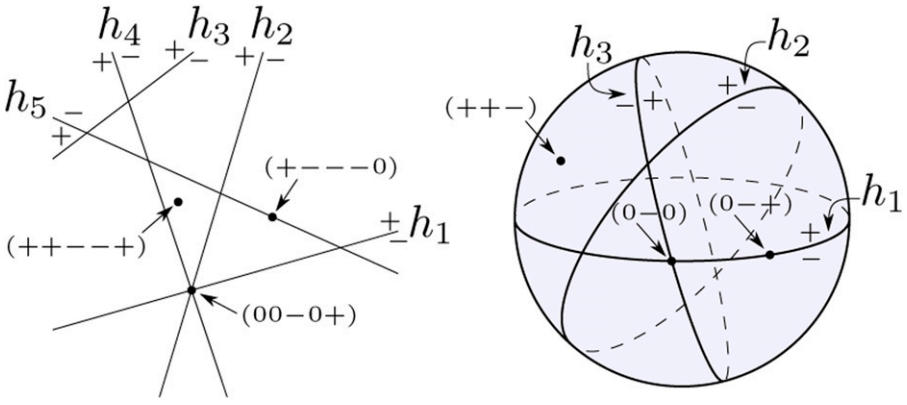

Figure 3. Examples of a hyperplane arrangement in $\mathbb{R}^2$ (left) and a linear hyperplane arrangement in $\mathbb{R}^3$ (right, visualized on a sphere). Selected faces are labeled with their signed vectors. For example, in the left, the signed vector $(+---0)$ means that the face lies on the $+$ side of h_1, $-$ side of h_2, $-$ side of h_3, $-$ side of h_4, and on h_5. The signs of sides are marked with $+$ and $-$ by the hyperplanes.

polyhedra or arrangement form a partially ordered set over their signed vectors. Signed vectors u, v satisfy $u \leq v$ if and only if $u_i \leq v_i$, for all indices i, where individual signs are compared according to $0 < +$, $0 < -$, and $+$ and $-$ incomparable. A *lattice* is a partially ordered set $\mathcal{L}$ such that for any elements $x, y \in \mathcal{L}$, there exists elements $x \wedge y$ and $x \vee y$ in $\mathcal{L}$ satisfying

$$(x \wedge y \geq x, y) \text{ and } (z \in \mathcal{L}, z \geq x, y \Rightarrow z \geq x \wedge y)$$
$$(x \vee y \leq x, y) \text{ and } (z \in \mathcal{L}, z \leq x, y \Rightarrow z \leq x \vee y) \quad (10)$$

The faces of a polyhedra or arrangement in $\mathcal{R}^d$ form a lattice $\mathcal{L}$ known as the *face lattice*. Figure 4 illustrates the face lattice of a cube. The face lattice is bounded by unique minimal and maximal elements which we denote $\{0\}$ and $\{1\}$, respectively. The face lattice $\mathcal{L}$ contains $d+1$ proper *ranks*. The k-th rank contains the faces of dimension k.

4. Enumeration of contacting-separating modes

The first step in autogenerating manipulation primitives is to enumerate contacting-separating modes. Once these modes are enumerated, we can then enumerate the family of sliding-sticking modes which belong to each contacting-separating mode. Geometrically, the contacting-separating modes are the faces of the polyhedra generated by the contact normal velocity constraints. This enumeration problem lies within the domain of computational geometry; The goal of this section is to reduce the problem into one we know how to efficiently solve using existing computational geometry algorithms.

4.1. Polar polytopes

A common technique is to transform the input into a combinatorially equivalent geometric object, apply some algorithm, and map the output back into the input space. The algorithm drives the choice of transformation, and in this problem, the convex hull algorithm will allow us to enumerate the faces. Therefore, the inputs must be transformed from halfspaces into points.

Recall that the contact normal velocity constraints form an $\mathcal{H}$ polyhedra. Moreover, the polyhedra is a polyhedral convex cone which contains the origin. The cone is combinatorially equivalent to the polytope obtained by taking its vertex-figure (Ziegler 1995). This allows us to use the following transformation on the normal velocity constraints. The *polar transformation* allows us to easily convert between combinatorially equivalent $\mathcal{H}$ polytopes and $\mathcal{V}$ polytopes. Without loss of generality, let $P \subseteq \mathbb{R}^d$ be a polytope with $0 \in P$. Its *polar polytope* $P^\triangle \subseteq \mathbb{R}^d$ is the set

$$P^\triangle = \left\{ c \in \mathbb{R}^d : c^T x \leq 1, \forall x \in P \right\} \quad (11)$$

The polar polytope can be specified in closed form for $\mathcal{H}$ and $\mathcal{V}$ polytopes. If P is a $\mathcal{V}$ polytope with $0 \in \text{int}(P)$ and $P = \mathcal{V}(A)$ then

$$P^\triangle = \mathcal{H}(A, 1) = \{x : Ax \leq 1\} \quad (12)$$

If P is a $\mathcal{H}$ polytope with $0 \in \text{int}(P)$ and $P = \mathcal{H}(A, 1)$ then

$$P^\triangle = \mathcal{V}(A^T) = \left\{ A^T t : t \geq 0, \sum t = 1 \right\} \quad (13)$$

Figure 5 depicts a 2D $\mathcal{H}$ polytope and its polar $\mathcal{V}$ polytope. Polar polytopes are useful because P and P^Δ share the same combinatorial structure. Specifically, the face lattice of the polar polytope $P^\triangle$ is the opposite of the face lattice of P

$$L(P^\triangle) = L(P)^{op} \quad (14)$$

and there is a bijection between the faces

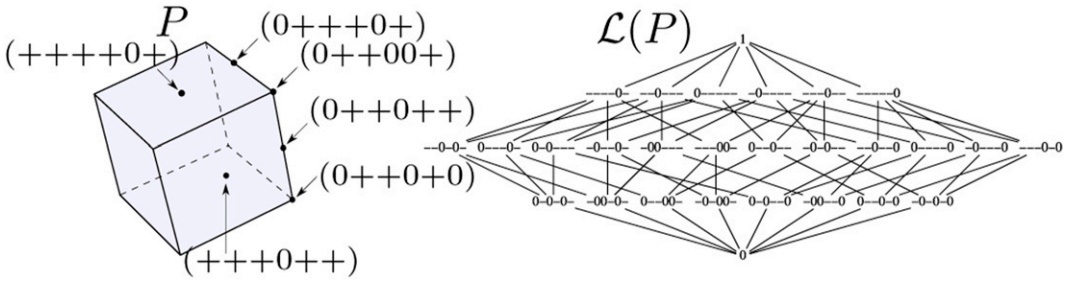

Figure 4. Left: Some signed vectors of a cube. Right: The face lattice of the cube. Each element in the face lattice corresponds to a face of a cube. Except for {1} and {0}, there are three levels of elements in the hierarchy. From top to bottom, three levels of elements respectively correspond to the 2-dimensional faces, edges, and vertices of the cube.

$$\begin{array}{c} \varnothing \leftrightarrow P \\ \textit{vertices} \leftrightarrow \textit{facets} \\ \textit{edges} \leftrightarrow \textit{ridges} \\ \cdots \leftrightarrow \cdots \end{array} \tag{15}$$

$$\begin{aligned} &\min_{r,c} \begin{bmatrix} 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r \\ c \end{bmatrix} \\ &\text{s.t.}\, Nr + c \geq 0, c \leq 0 \\ &\|r\|_\infty \leq 1, \end{aligned} \tag{16}$$

4.2. Convex hull enumeration method

Algorithm 1. CS Mode Enumeration with Convex Hull

```
function CS-Mode-Enumerate (N)
    r ←Relative-Interior-Point(N)
    if Any(Nr = 0) then
        N_c, N_s ←Partition(N, r)
        N ← N_s ·Null(N_c)
        r ← r ·Null(N_c)
    N̂ ← Preprocess (N)
    z ← − N̂r
    N̂[i, :] ← N̂[i, :]/z[i]
    M ← ConvexHull(N̂^T)
    L_cs ← FaceLattice(M)
    return L_cs
```

The contacting/separating mode enumeration algorithm, or CS-Mode-Enumerate, takes as input the normal velocity constraint equations $N \in \mathbb{R}^{n \times d}$ (see Section 3) and generates a list of valid contacting/separating sign vectors of the form $m \in \{0,+\}^n$. The algorithm is based on taking the convex hull in the polar form of the polytope associated with the normal velocity constraints. The pseudo-code is listed in Algorithm 1 and we provide explanations for each of the steps below.

Find an interior point: The polar form P^Δ of a polyhedron P is defined only when $0 \in \textit{relint}(P)$. However, 0 is on the boundary of the polyhedral cone $\mathcal{H}(N, 0)$ defined by our normal velocity constraints. Therefore, our first step is to find a point $r \in \text{relint}(\mathcal{H}(N, 0))$. This is a classical problem in linear programming, and for our implementation, it amounts to solving the following linear program

where $\|r\|_\infty \leq 1$ constrains r to be within the hypercube in $\mathbb{R}^d$. Note that if the solution to the linear program is $r = 0$, then the only valid mode is all-contacting and the algorithm can terminate early. The above method was adapted from the linear program in the documentation of QHull Library (2020) to work on pointed cones.

Project to contacting hyperplanes: If the interior point is on the boundary for a subset of the normal velocity constraints, then that subset of contact points must always be in contact (for example, a box sandwiched between two walls). Let N_c be the set of normal velocity constraints which must always be in contact and let N_s be the set of normal velocity constraints which could separate. Then we map N_s into the nullspace coordinates of N_c, like so $N_s = N_s \cdot \text{Null} N_c$, to reduce the dimension of the problem. We also express the interior point as coordinates in the null space.

Preprocess hyperplanes: The affine dimension of the polar polytope dim ($\textit{aff}P^\Delta$) may not necessarily be equal to the dimension of the ambient space $\mathbb{R}^d$. In this situation, we project P^Δ into its affine subspace $\textit{aff}P^\Delta = \{N^T v\colon 1^T v = 1\}$ and further reduce the dimensionality of the convex hull. Recall that an affine space can also be expressed as a linear space plus a translate, i.e., $\textit{aff}P^\Delta = \{Vx + z\}$ for some V and z. If $0 \notin \textit{aff}P^\Delta$, then $z \neq 0$ and we translate the affine space until it contains the origin. Now that $0 \in \textit{aff}P^\Delta$, $\textit{aff}P^\Delta$ is a linear subspace and we project each point (column vector) in N^T to coordinates on the column space of N^T.

Convert to polar form: Given a strictly interior point r, we translate the origin to r, resulting in the new $\mathcal{H}$ polyhedron $P = \mathcal{H}(\widehat{N}, -\widehat{N}r)$. Next we normalize inequalities so that $P = \mathcal{H}(\widehat{N}, 1)$ and obtain the polar polytope $P^\Delta = \mathcal{V}(\widehat{N}^T)$.

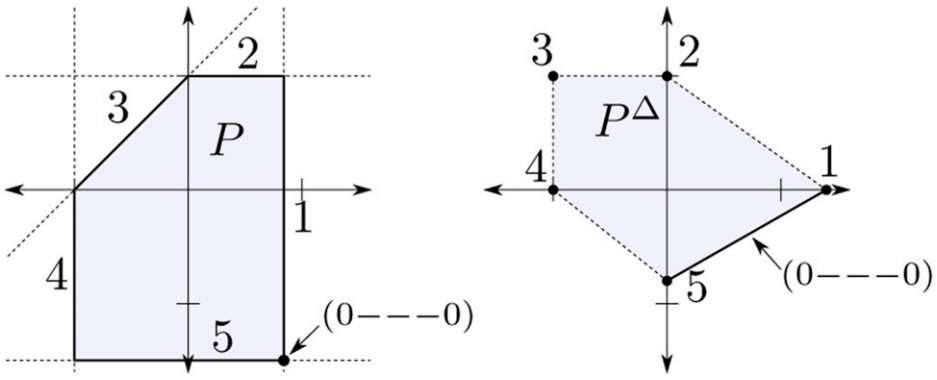

Figure 5. Example of a 2D $\mathcal{H}$ polytope P (left) and its polar $\mathcal{V}$ polytope $P^\triangle$ (right). There is a bijection between their faces. The edges marked with numbers in the $\mathcal{H}$ polytope P correspond to the vertices marked with the same numbers in its polar $\mathcal{V}$ polytope $P^\triangle$. The vertices of the $\mathcal{H}$ polytope P correspond to the edges of its polar $\mathcal{V}$ polytope $P^\triangle$.

Compute convex hull: Next, the algorithm constructs the vertex-facet incidence matrix M of $P^\Delta = \mathcal{V}(\widehat{N}^T)$ by using a convex-hull algorithm. The vertex-facet incidence matrix is a matrix $M \in \{0,1\}^{n_v \times n_f}$, where n_v and n_f are the number of vertices and facets, respectively. We associate the vertices and facets with the index sets $I_V = \{1, \ldots, n_v\}$ and $I_F = \{1, \ldots, n_f\}$, so that $m_{vf} = 1$ if facet f contains v and $m_{fv} = 0$ otherwise. The vertex-facet incidence matrix is a standard return value from convex hull algorithms such as Qhull Barber et al. (1996).

Build face lattice: Given the facet-vertex incidence matrix M of P^Δ, we can construct the face lattice $L(P^\Delta)$ using the algorithm of Kaibel and Pfetsch (2002). Their method is based on finding the closed sets (= faces) with respect to a closure map defined over vertex sets. Each face is uniquely represented by its vertex set.

4.3. Complexity of convex hull method

4.3.1. Theorem. For a set of n contacts in a system of colliding bodies with d degrees of freedom, Algorithm 1 enumerates the possible contacting and separating modes in $O(d \cdot n^{\lfloor d/2 \rfloor + 2} + l(n, d))$ time.

Proof. We analyze correctness first before complexity. The proof is simple and relies on the combinatorial equivalences between

$$\text{CS-MODES} \leftrightarrow L(\mathcal{H}(N,0)) \leftrightarrow L\left(\mathcal{H}\left(\widehat{N},1\right)\right) \leftrightarrow L\left(\mathcal{V}\left(\widehat{N}^T\right)\right) \tag{17}$$

First, we show that $\mathcal{H}(N,0)$ and $\mathcal{H}(\widehat{N},1)$ are affinely isomorphic and thus, combinatorially equivalent Ziegler (1995). Two polytopes P and Q are *affinely isomorphic* if there exists an affine map $f : \mathbb{R}^d \to \mathbb{R}^e$ that is a bijection between the vertices of the two polytopes. By inspection, the operations $P \cap aff(P)$ and $P + r$ preserve the extremal points (vertices). Finally, re-scaling the inequalities does not affect the underlying polytope.

For this next paragraph, let us define $P = \mathcal{H}(\widehat{N},1)$ and $P^\Delta = \mathcal{V}(\widehat{N}^T)$. Our aim is to show the first and third bijections in (18). Let $F \in L(P^\Delta)$ be identified by its vertex set $V(F) = \{a: a \cap F \neq \emptyset, a \in vert(P^\Delta)\}$ and recall that $\text{vert}(P^\Delta) \subseteq \text{col}(\widehat{N}^T) = \text{row}(\widehat{N})$. (That is, each vertex of P^Δ corresponds to a facet of P, i.e. a normal velocity constraint.) Then by Corollary 2.13 of Ziegler (1995), there is a bijection $L(P^\Delta) \leftrightarrow L(P)$ from F to $F^\Diamond$ such that

$$F^\Diamond = \left\{x : \widehat{N}x \leq 1, ax = 1, \forall a \in V(F)\right\} \tag{18}$$

is a non-empty face of P. Because face lattice of a polytope is coatomic, we can uniquely specify its proper elements as meets (intersections) $n_1 \wedge \ldots \wedge n_k$ of its coatoms (facets). Therefore, for each $F \in L(P^\Delta)$, the vertex set $V(F)$ maps bijectively to a valid contacting/separating mode string, and $L(P^\Delta)$ enumerates the set of all valid contacting/separating modes.

The normal velocity constraint matrix N can be constructed in $O(n \cdot d)$ time. The orthonormal basis and null space can be computed in $O(\min\{n \cdot d^2, n^2 \cdot d\})$ using SVD. An interior point can be computed in time $O(l(n,d))$, where $l(n,d)$ is the cost of linear programming. For a balanced problem like this one (every input point is extremal), quick hull runs in $O(n^{d/2})$. The number of faces in $L(P)$ is bound by $O(n^{\lfloor d/2 \rfloor})$ (Edelsbrunner 2005). The combinatorial face enumeration algorithm runs in time $O(d \cdot n^2 \cdot n^{\lfloor d/2 \rfloor}) = O(d \cdot n^{\lfloor d/2 \rfloor + 2})$ (Kaibel and Pfetsch 2002). The complexity reported in Kaibel and Pfetsch (2002) uses an $\alpha = n \cdot m$ term, where m is the number of facets. For this analysis, we assume that the number of facets is on the same order as the number of vertices. In practice, we have found this assumption to be reasonable. Therefore, the total runtime is $O(d \cdot n^{\lfloor d/2 \rfloor + 2} + l(n,d))$.

5. Enumeration of sliding-sticking modes

This section derives two algorithms for sliding-sticking mode enumeration. Recall that the sliding-sticking modes are contained in the faces of the partial hyperplane

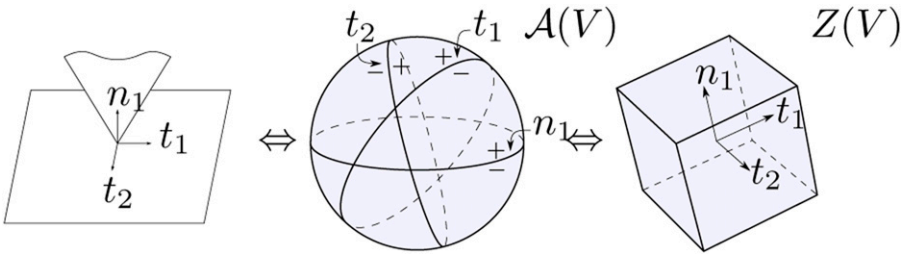

Figure 6. Combinatorially equivalent representations of sliding-sticking modes. (Left) A pointed finger which is maintaining contact. (Middle) Hyperplane arrangement in velocity space $[\dot{x}, \dot{y}, \dot{\theta}] \in \mathbb{R}^3$, where x, y are along t_2, t_1 and θ is the rotation about n_1. (Right) Zonotope in the polar space generated by t_1, t_2, n_1. Normals in $\mathcal{A}(V)$ map to points in $Z(V)$ and vice versa.

arrangement generated by the normal and tangent velocity equations. Therefore, the first algorithmic design is the combinatorially equivalent geometric object to base the algorithm on. The two choices, zonotopes or hyperplane arrangements, used in this work are illustrated in Figure 6. The zonotope algorithm is based on computing the zonotope using an iterative Minkowski sum. The partial hyperplane arrangement algorithm iteratively builds the arrangement directly in the space of hyperplanes. We begin with the former algorithm.

5.1. Zonotopes

Zonotopes are a special type of convex polytope which are combinatorially equivalent to linear hyperplane arrangements. Recall that the *Minkowski sum* of sets X and Y is given by $X \oplus Y = \{x + y : x \in X, y \in Y\}$. We can define a zonotope as the Minkowski sum of a set of line segments

$$Z(V) = [-v_1, v_1] \oplus \ldots \oplus [-v_k, v_k] \tag{19}$$

where $V = [v_1, \ldots, v_k] \in \mathbb{R}^{d \times k}$. We can map each face F of a zonotope to an unique signed vector. Let $p = \sum \lambda_i v_i \in \operatorname{int} F$ be an interior point of F. Then the signed vector with respect to v_i is

$$\operatorname{sign}_{v_i}(F) = \begin{cases} + & \text{if} \quad \lambda_i = +1 \\ 0 & \text{if} \; -1 < \lambda_i < 1 \\ - & \text{if} \quad \lambda_i = -1, \end{cases} \tag{20}$$

and the signed vector is $\operatorname{sign}_Z(F) = \left[\operatorname{sign}_{v_1}(F), \ldots, \operatorname{sign}_{v_k}(F)\right]$. From Corollary 7.17 in Ziegler (1995), there is a bijection between the signed vectors of $\mathcal{A}(V)$ and $Z(V)$. We have the identification of face lattices

$$L(Z(V)) \leftrightarrow L(Z(V)^{\triangle}) \leftrightarrow L(\mathcal{A}(V)) \tag{21}$$

For example, there is a correspondence between the facets of $Z(V)$, the vertices of $Z(V)^{\triangle}$, and the rays (unbounded edges) of $\mathcal{A}(V)$. Figure 7 maps the equivalencies between faces and vertices on a hyperplane arrangement to vertices and faces on a zonotope (and its polar zonotope).

5.2. Zonotope/Minkowski-sum enumeration method

Algorithm 2. Zonotope/Minkowski Sum Method

function SS-MODE-ENUMERATE-ZONO (N, T, m_{cs})
  N_c, N_s, T_c ←Partition(N, T, m_{cs})
  $H \leftarrow [N_s; T_c]$
  $H \leftarrow H \cdot \text{Null}(N_c)$
  V, S ←GetZonotopeVertices(H)
  L← FACE − LATTICE $\mathcal{V}(V)$
  $\mathcal{F} = \{F \in L(V) : \text{sign}_{N_s}(F) = [+, \ldots, +]\}$
  M_{ss}← *GetSignVector*($\mathcal{F}, S$)
  return M_{ss}
function Get-Zonotope-Vertices (H)
  V, $S \leftarrow [0], \varnothing$
  for $h \in$ Rows(H) **do**
    V', $S' \leftarrow [\,], \varnothing$
    for $v \in V$ **do**
    V' ←Add-Points(v+h, v−h)
    s_v ←GetSignVector(v, s)
    $S' \leftarrow S' \cup \{(s_v, +), (s_v, -)\}$
  V ←ConvexHullExtremalPoints(V')
  S ←GetSignVector(V, S')
return V, S

The algorithm, SS-MODE-ENUMERATE-ZONO takes as input the normal velocity constraint matrix $N \in \mathbb{R}^{n \times d}$, the tangent velocity constraint matrix $T \in \mathbb{R}^{nk \times d}$, and the contacting separating mode string m_{cs} and produces a list of sliding/sticking sign vectors of the form $m_{ss} \in \{-,0,+\}^{nk}$. As before we provide explanations for each of the steps below.

Partition the hyperplanes: The first step in our algorithm is to partition the input normal velocity constraint matrix and tangent velocity constraint matrix into active equality and inequality constraints. We read off the input contacting-separating mode m_{cs} and partition in the input matrices into the normal velocity constraint matrix N_c which must

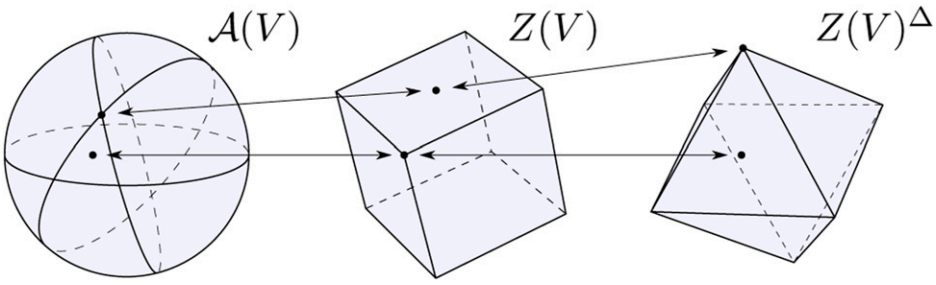

Figure 7. Combinatorially equivalent faces in a linear hyperplane arrangement, zonotope, and polar zonotope.

maintain contact, the normal velocity constraint matrix N_s which can separate, and the active tangent velocity constraint matrix T_c which corresponds to the maintained contacts. We then form the set of hyper-planes $H = [N_s; T_c]$. The algorithm will compute the combinatorial structure of the arrangement $\mathcal{A}(H)$.

Project to contact planes: We project H into null space coordinates of N_c to reduce dimensionality and thereby speed up computation. This is step 4 in Algorithm 2.

Construct the zonotope: From the set of hyperplanes H, we can identify the vertices of its associated zonotope $Z(H)$. From equation (19), zonotopes can be represented by the minkowski sum of line segments, which in this case are $[h_i, -h_i]$ for $h_i \in H$. Algorithm 2 Function GET-ZONOTOPE-VERTICES obtains zonotope vertices V through computing the minkowski sum iteratively for every $h_i \in H$. We initialize the vertex set $V = [0]$, and at the i-th iteration we update V to be the external points of the convex hull of $\cup_{v \in V}\{v + h_i, v - h_i\}$. Simultaneously, we track the sign vectors for every vertex and update the sign vectors using this rule at every iteration: sign$(v + h_i) = (v), +]$, sign $(v - h_i) =$ [sign$(v), -$]. To decrease the zonotope construction cost, one may omit some separating hyperplanes from H at the expense of allowing some invalid sliding/sticking modes.

Build the face lattice: Using the same method as describe in Section 4.2, we construct the face lattice $L(V)$ from the vertices V. Not every $F \in L(V)$ corresponds to a valid sliding/sticking contact mode. Only the faces that have positive signs + with respect to normal velocity constraints for all separating contacts are valid sliding/sticking contact modes. After building the face lattice $L(V)$, valid faces $\mathcal{F}$ are selected by ensuring their sign vectors with respect to N_s are all +'s

$$\mathcal{F} = \left\{F \in L(V) : \text{sign}_{N_s}(F) = [+, \ldots, +]\right\} \tag{22}$$

The sign vectors of all faces in $\mathcal{F}$ with respect to H represent all valid sliding/sticking contact modes for the given contact/separating mode.

5.3. Complexity of Zonotope method

5.3.1. Theorem. For a set of n contacts (modeled with k tangent planes) in a system of colliding bodies with d degrees of freedom, Algorithm 2 enumerates the possible sliding/sticking modes in $O(d(kn)^{d^2/2+2d})$ time for a given contacting/separating mode.

Proof. As before, we first proceed with a proof of correctness. For a given contacting/separating mode string m_{cs}, let $H = [h_{s_1}, \ldots, h_{s_k}, h_{t_1}, \ldots, h_{t_m}]$ be the input hyperplanes to our zonotope construction algorithm, where k is the number of separating hyperplanes and m is the number of tangent hyperplanes. We incrementally construct the zonotope by using the fact that the Minkowski sum of two polytopes is the convex hull of the sums of their vertices Delos and Teissandier (2015). By Corollary 7.18 of Ziegler (1995), the face lattice of the zonotope constructed above is the opposite of the face lattice of the hyperplane arrangement.

Next, we analyze the complexity of our algorithm. The maximum number of hyperplanes is kn. The number of vertices, i.e., f_0, for a d-zonotope that is the projection of a p-cube is of the order $O\ (p^{d-1})$ Edelsbrunner (2005). Therefore, our zonotope construction algorithm takes time

$$O\left(\sum_{p=1}^{kn}\left(p^{d-1}\right)^{\frac{d+1}{2}}\right) \approx O\left(\sum_{p=1}^{kn} p^{\frac{d^2}{2}}\right) \approx O\left((kn)^{\frac{d^2}{2}}\right) \tag{23}$$

We use Kaibel and Pfetsch (2002) to construct the face lattice of the resulting zonotope. As before, their algorithm runs in

$$O(kn \cdot d \cdot \alpha \cdot (kn)^{d-1}) \approx O(d(kn)^{2d}) \tag{24}$$

where $\alpha = kn \cdot (kn)^{d-1}$ in the worst case (when the zonotope is simple) Ziegler (1995). The full complexity of SS-MODE-ENUMERATE-ZONO is therefore $O((kn)^{d^2/2+2d})$.

The zonotope algorithm's strength is its ease of implementation.

However, it has two major drawbacks.

First, the iterative Minkowski sum is very inefficient. It requires d orders of magnitude more computation than there are faces in $\mathcal{A}$.

Second, the algorithm computes the full arrangement, including invalid modes (faces) that have $-$ signs with respect to the separating normals.

In the next section, we will address both those of those issues and derive the partial hyperplane arrangement

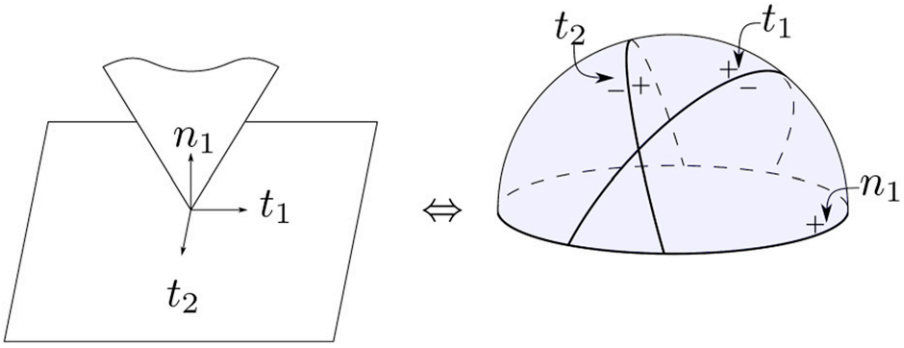

Figure 8. (Left) A point finger is contacting a surface. (Right) The partial hyperplane arrangement generated by the normal and tangent velocity hyperplanes at that point contact.

algorithm that has runtime nearly proportional to the number of valid sliding-sticking modes.

5.4. Partial hyperplane arrangement enumeration method

Algorithm 3. Partial Hyperplane Arrangement Method

```
1: function SS-MODE-ENUMERATE (N, T, m_cs)
2:     N_c, N_s, T_c ←Partition(N, T, m_cs)
3:     [N_s; T_c] ←ProjectNullspace(N_c, [N_s; T_c])
4:     [N_s; T_c] ←Preprocess([N_s; T_c])
5:     ifRank(N_s) = Rank ([N_s; T_c]) then
6:         L_ss← HalfspaceIntersection(N_s)
7:     else
8:         L_ss← InitPartialArrangement([N_s; T_c])
9:     for h ∈ Rows([N_s; T_c]), h ∉ L_ss do
10:        L_ss← IncrementArrangement(h, L_ss)
11:    return L_ss
```

This section describes the new sliding-sticking mode enumeration algorithm proposed by this paper. To avoid enumerating infeasible modes, our algorithm builds the face lattice of the partial hyperplane arrangement $\partial\mathcal{A}$ for a given contacting-separating mode m_{cs}

$$\partial\mathcal{A} \equiv \text{Null}(N_c) \cap \mathcal{H}(N_s, 0) \cap \mathcal{A}(T_c) \tag{25}$$

where N_c are the contacting normal velocity hyperplanes, N_s are the separating normal velocity hyperplanes, and T_c are the active tangent velocity hyperplanes. Figure 8 depicts the partial hyperplane arrangement generated by a single point contact. Nearly half of the tangent space can be discarded due to non-penetration constraints, allowing us to speed up the enumeration. This approach improves upon the zonotope method covered in Section 5.2 (Huang et al., 2020), which enumerates both feasible and infeasible modes.

The Edelsbrunner (2005) incremental hyperplane arrangement algorithm is a natural choice for this method as it achieves runtime proportional to the number of faces in the arrangement i.e., $O(n^d)$. We further improve the method with a partial hyperplane arrangement technique based on the following observation. The normal velocity equations define one-sided constraints, and as such, the algorithm should avoid enumerating modes representing invalid, penetrating velocities. Figure 8 illustrates the key idea. The ideal location to enforce one-sided hyperplanes is in the initialization phase of Edelsbrunner's algorithm. During initialization, the partial variant of the algorithm either initializes using a convex hull of the one-sided hyperplanes or a mixed initialization of d_{eff} one-sided and two-sided hyperplanes, where d_{eff} is the effective dimension of the arrangement. (The mixed method is used when a convex hull of rank d_{eff} is not possible.)

The rest of this section describes in more detail each step of the pseudo-code, Algorithm 3, SS-MODE-ENUMERATE, for the partial hyperplane arrangement method. The algorithm takes normals, separating tangent planes, and contact/separating mode as inputs, and produces the lattice of partial hyperplanes corresponding to feasible sliding/sticking contact modes.

Preprocess hyperplanes based on cs-mode: Based on the input cs-mode m_{cs}, we partition N and T into N_c, N_s, and T_c and project $[N_s; T_c]$ into the nullspace coordinates of N_c. Next we preprocess the hyperplanes using Algorithm 6.

Initialize partial hyperplane arrangement: Our goal is to initialize the partial hyperplane arrangement with the region of space bounded by $\mathcal{H}(N, 0)$. Let d_s be the rank of N_s. Our algorithm divides the intialization into two cases:

1. If there are enough linearly independent hyperplanes in N_s, i.e. $d_s = d$, we compute $\mathcal{H}(N_s, 0)$ using a polar convex hull.
2. Otherwise, we generate the initial partial arrangement from the first d linearly independent hyperplanes in $[N_s; T_c]$. This can be accomplished by lexicographically enumerating the $2^{d_s}3^{d-d_s}$ faces (signed vectors) in the initial partial arrangement.

Incrementally add remaining hyperplanes: For each hyperplane (row) h in $[N_s; T_c]$, we add it to $\mathcal{L}_{ss}$ using Edelsbrunner's method (Procedures 7.3–7.5 of

Edelsbrunner (2005)). This method colors the vertex and edge nodes of $\mathcal{L}_{ss}$ with respect to the incoming hyperplane and updates the rest of the lattice-based on that coloring.

5.5. Complexity of partial arrangement method

5.5.1. Theorem. The partial hyperplane arrangement computes the sliding-sticking modes in $O(n^d)$ time.

We present a sketch of proof. To recap, we are using an incremental hyperplane arrangement algorithm on a partial hyperplane arrangement instead of a full hyperplane arrangement. First, we argue the correctness of the algorithm. The key idea is that faces of the partial hyperplane arrangement are faces in the full hyperplane arrangement. The correctness of Edelsbrunner's algorithm is solely based on the properties of faces outlined in Table 7.3, Observation 7.2, Observation 7.3, and Observation 7.4 Edelsbrunner (2005). It is straightforward to verify each property holds for the faces in the partial hyperplane arrangement using proof by contradiction.

The runtime analysis is based on Section 7.6 of Edelsbrunner (2005). The key result is that a new hyperplane h can be inserted into an existing arrangement with time proportional to the number of faces properly intersecting h. However, it is outside the scope of this work to derive bounds on the number of k-faces intersecting h, as is done in Theorem 5.4 of Edelsbrunner (2005). A conservative bound on the runtime is $O(n^d)$ which is the cost of constructing the full hyperplane arrangement.

6. Dynamics of contact modes

This section explains the relationship between contact modes and dynamics. The first two sections review the Coulomb friction model and its linearization, the polyhedral friction model. These models can be expressed as sets of complementarity constraints. Next, it is shown that the nonconvex structure of frictional contact dynamics can be precisely decomposed into piece-wise convex components i.e., into contact modes. Lastly, we show how the polyhedral friction model fits into the dynamic equations of motion.

6.1. Coulomb friction model

Coulomb friction is a friction model with the property that the frictional force is both proportional in magnitude to the normal force and anti-parallel to the sliding direction. The Coulomb model of friction can be represented using the friction cone and the principle of maximal dissipation. The friction cone is defined to be the convex set

$$\mathcal{F} = \{\lambda_n, \lambda_x, \lambda_y : \lambda_n \geq 0, \mu\lambda_n^2 - \lambda_x^2 - \lambda_y^2 \geq 0\} \tag{26}$$

where μ is the coefficient of friction, λ_n is the component of the frictional force in the normal direction n, and λ_x, λ_y are the components of the frictional force in the t_x and t_y tangent directions (see Figure 9). The principle of maximal dissipation states that for a tangential sliding velocity $v = [v_x; v_y]$, the direction of the frictional force maximizes the dissipated energy, i.e.

$$\lambda_n^\star, \lambda_x^\star, \lambda_y^\star \in \arg\max_{\mathcal{F}} -\lambda_x v_x - \lambda_y v_y \tag{27}$$

Note, when $v = 0$, any $\lambda_x, \lambda_y \in \mathcal{F}$ satisfies equation (27), but when $v \neq 0$, $[\lambda_x; \lambda_y]$ must be anti-parallel to v.

We can express Coulomb friction as the solution to a system of equations comprised of nonlinear complementarity constraints. This technique is used to encode Coulomb friction into a form suitable for simulation and optimization.

6.1.1. Definition. A *complementarity constraint* is a constraint on two variables $a, b \in \mathbb{R}$ such that

$$a \geq 0 \tag{28}$$

$$b \geq 0 \tag{29}$$

$$ab = 0 \tag{30}$$

Under this constraint, either $a > 0$ and $b = 0$, or $b > 0$ and $a = 0$, or $a = b = 0$. We also write the constraint as $0 \leq a \perp b \geq 0$.

Let ϕ be a function which returns the minimum separating distance between two colliding surfaces. The contact normal force is zero if the distance is trending positive $\phi \geq 0$. We can model this as the complementarity constraint

$$0 \leq \lambda_n \perp \phi + \dot{\phi} \geq 0 \tag{31}$$

where $\dot{\phi} = d\phi/dt$ is the separating velocity. Next, we show how to extend this complementarity formulation to Coulomb friction. We can encode the friction cone and principle of maximum dissipation as a set of equality and complementarity constraints

$$0 = \mu\lambda_n v_x + \lambda_x \sigma \tag{32}$$

$$0 = \mu\lambda_n v_y + \lambda_y \sigma \tag{33}$$

$$0 \leq \sigma \perp \mu^2\lambda_n^2 - \lambda_x^2 - \lambda_y^2 \geq 0 \tag{34}$$

where σ is a slack variable. When the sliding velocity is nonzero, σ^2 evaluates to $v_x^2 + v_y^2$ and $[\lambda_x; \lambda_y]$ is constrained to be anti-parallel to v. Note, the friction cone is a second-order cone but the principle of maximal dissipation introduces bilinear constraints.

6.2. Polyhedral friction model

Because equations (32)–(34) are nonlinear, researchers have also developed linearized models of Coulomb friction (Stewart and Trinkle 1996; Horak and Trinkle 2019). Undoubtedly, using a linearized model is an engineering choice. Within the context of this paper, we believe that

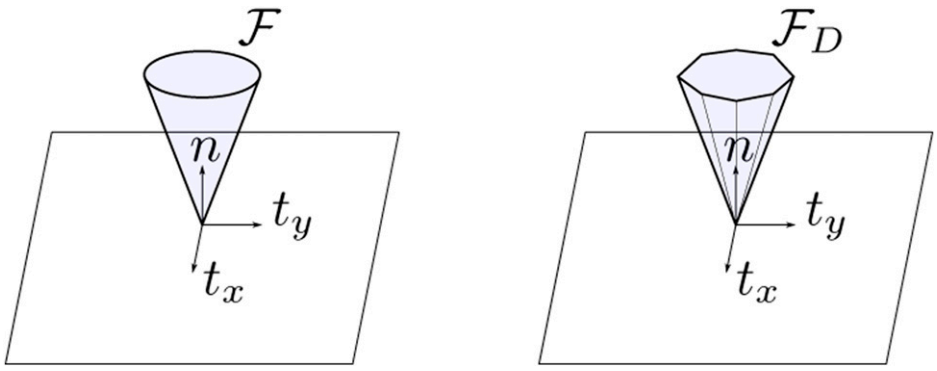

Figure 9. (Left) Coulomb friction cone. (Right) polyhedral friction cone with eight sides.

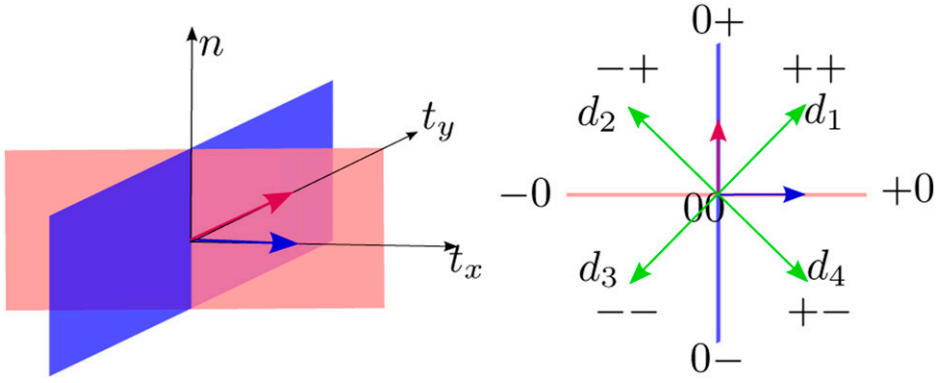

Figure 10. Right: the contact frame of one contact and two tangent dividing planes (shaded with pink and blue) with their normals. Left: top view of the contact tangent plane, marked with all feasible sliding modes and frictional force directions $\{d_1, d_2, d_3, d_4\}$ (green vectors) for a 4-sided friction cone.

low-resolution friction models are the right choice (for a discussion see Section 7).

We can linearize the friction cone by approximating it with a polyhedral convex cone (Figures 9 and 11)

$$\mathcal{F}_\Delta = \{\lambda_n n + \Delta\lambda_f : \mu\lambda_n - e^T\lambda_f \geq 0, \lambda_f \geq 0\} \tag{35}$$

where the frictional force directions $\Delta = [d_1, \ldots, d_{n_d}] \in \mathbb{R}^{3\times n_d}$ is a matrix whose columns are the generators of the polyhedral convex cone projected into the plane, $\lambda_f \in \mathbb{R}^{n_d}$ are the components of frictional force for each direction d_i, and $e = [1, \ldots, 1]^T \in \mathbb{R}^{n_d}$ is a vector of ones. Figure 10 visualizes d_i for a 4-sided friction cone. The principle of maximal dissipation is rewritten as

$$D\lambda_f^\star = \arg \max_{D\lambda_f \in \mathcal{F}_D} -v^T D\lambda_f \tag{36}$$

where $v = [v_x, v_y, 0]^T$ is the sliding velocity. When the sliding velocity is non-zero, the resultant frictional force is located on the ray most anti-parallel to v. As opposed to the

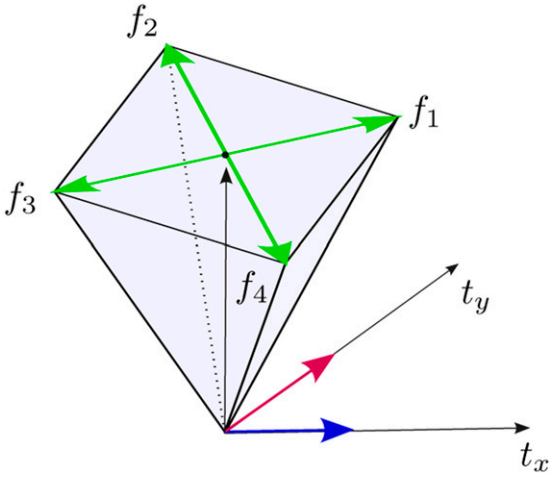

Figure 11. A 4-sided friction cone derived from two tangent sliding planes (normals pink and blue), with frictional force generators $\{f_1, f_2, f_3, f_4\}$, which are the sum of the normal vector and μ times the frictional force directions.

Coulomb friction model, the polyhedral friction model can be expressed as linear complementarity constraints

$$0 \leq \lambda_f \perp D^T v + e\sigma \geq 0 \tag{37}$$

$$0 \leq \sigma \perp \mu\lambda_n - e^T\lambda_f \geq 0 \tag{38}$$

where the inequalities are evaluated element-wise. This time, $\sigma = -\max -v^T d_i$ when v is non-zero.

6.3. Contact mode friction model

In this section, we show that the linear complementarity representation of the polyhedral friction model can be precisely decomposed into piece-wise convex components represented by contact modes. In order to do so, we will first introduce the concept of a frictional force generator.

Figure 10 shows the contact frame for a single contact with two tangent directions. On the right side of Figure 10, we show the top view of the sliding plane. Each sliding-sticking mode defines a sliding velocity cone. We mark all sliding-sticking modes beside their corresponding sliding velocity cones, and visualize the frictional force directions $\{d_1, d_2, d_3, d_4\}$. Figure 11 shows the corresponding 4-sided polyhedral friction cone $\mathcal{F}_D$. The *frictional force generators* are the generators (rays) of the polyhedral convex cone $\mathcal{F}_D$. The i-th generator is of the form $f_i = n + \mu d_i$. This allows us to write elements in $\mathcal{F}_D$ in a simple manner, as conical combinations of the frictional force generators. With n_t tangent directions, there will be a $2n_t$-sided polyhedral friction cone. As n_t approaches infinity, the original Coulomb friction cone is recovered.

Next, we derive the force and motion models for each contact mode based on the sliding velocity and frictional force generators mentioned above. We use one contact two tangent directions as an example, but note that the same applies with more tangent directions. There are two contacting-separating modes for this single contact, $\{+, 0\}$, and nine sliding-sticking modes with two tangent dividing planes, $\{0+, 0+, -+, -0, --, 0-, +-, +0, 00\}$. For each contact mode, we can write down its corresponding part of the polyhedral friction model. Let λ_i be the weights of the frictional force generators f_i. At a high level there are three cases to consider, when a contact is separating, sticking, or sliding.

Separating contact: The contacting-separating mode of a separating contact is +, which means that the contact normal velocity v_n is positive. There is no sliding-sticking mode for a separating contact. There is no contact force for a separating contact. The contact dynamics for a separating contact can be written as

$$v_n > 0$$

$$\lambda_i = 0 \tag{39}$$

Sticking contact: For a fixed contact, its contacting-separating mode is 0 and its sliding-sticking mode is 00. Thus, the contact velocity is 0 and the friction force lies inside the polyhedral friction cone $\mathcal{F}_D$ shown in Figure 11

$$v_n, v_x, v_y = 0$$

$$\lambda_i \geq 0 \tag{40}$$

where $[v_x; v_y] \in R^2$ is the sliding velocity.

Sliding contact: A sliding contact has a cs-mode 0 and non-zero components in its ss-mode. Each ss-mode describes a sliding velocity cone $\mathcal{V}_{ss} \subseteq \mathbb{R}^2$ in the contact tangent plane. The sliding velocity cones can be one-dimensional (a single vector) or two-dimensional (the convex cone of two vectors). Figure 12 shows the sliding velocity cones for the ss-mode ++ (2D cone) and 0 + (1D cone). From the force-velocity complementary constraints in equation (37), we get the tangent friction force direction to be the one among the friction generators that has the smallest dot product with the sliding velocity. As a result, for a sliding velocity in the 2D cone (left, Figure 12), the tangent friction force direction is the same as the one force generator that is in the opposite cone of the sliding velocity cone. For a sliding velocity in the 1D cone (right, Figure 12), there are two frictional force generators that are of the same smallest dot product with the sliding velocity. Thus, the tangent force should be the positive linear combination of these two friction force generators. The contact dynamics for sliding contacts can be written as

$$v_n = 0$$

$$[v_x; v_y] \in \mathcal{V}_{ss}$$

$$\lambda \in \mathcal{F}_{ss} \tag{41}$$

where $\mathcal{F}_{ss}$ is the polyhedral convex cone of active frictional force generator weights for the given sliding velocity cone $\mathcal{V}_{ss}$. These constraints can be represented using linear equality and inequality constraints.

6.4. Dynamic equations of motion

Next, we combine the contact mode friction model with rigid body dynamics to derive the dynamic equations of motion at a given contact mode. As a motivating example, consider a system consisting of an object and a manipulator. This system is modeled with a 4-sided friction cone.

First, we define the variables required at each contact (Table 1). In a system with n contacts, let $c_i \in \mathrm{SE}(3)$ refer to the i-th contact frame. The origin of c_i is at the point of contact and its z-axis is pointed along the surface normal into the object. Several variables are shared amongst all contacts because they exist in the contact frame. They are the normal direction $n = [0, 0, 1, 0, 0, 0] \in \mathbb{R}^6$, the tangent basis matrix $D \in \mathbb{R}^{6\times 4}$, and the frictional force directions $\Delta \in \mathbb{R}^{6\times 4}$. The coefficient of friction $\mu_i \in \mathbb{R}$ is specified per contact and as a result, the frictional force generators $F_i \in \mathbb{R}^{6\times 4}$ are also specified per contact. Finally, we also

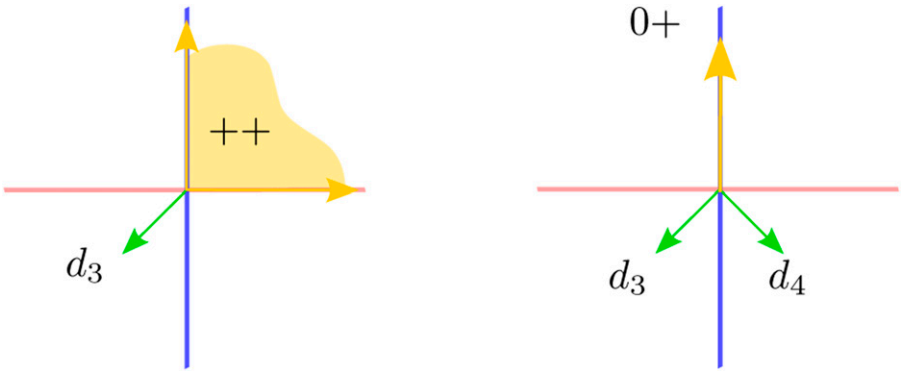

Figure 12. The sliding velocity cones (yellow) and tangent frictional force directions (green) for ss-modes (++) and (0+) respectively.

Table 1. Variables and parameters at the i-th contact.

$ci \in SE(3)$	Contact frame
$\mu_i \in \mathbb{R}$	Coefficient of friction
$n \in \mathbb{R}^6$	Vector of $[0,0,1,0,0,0]$
$D \in \mathbb{R}^{6\times 4}$	Tangent basis matrix
$\Delta \in \mathbb{R}^{6\times 4}$	Frictional force directions
$F_i \in \mathbb{R}^{6\times 4}$	Frictional force generators
$J_{oc_i} \in \mathbb{R}^{6\times d}$	Object velocity to contact twist
$J_{rc_i} \in \mathbb{R}^{6\times d}$	Robot velocity to contact twist

require the Jacobians $J_{oc_i}, J_{rc_i} \in \mathbb{R}^{6\times d}$ mapping object and robot velocities, respectively, to body twists in the contact frame. If the object/robot is not involved in the contact, then their Jacobian is comprised of 0s.

Next we show how to construct the dynamic equations of motion at a given contact mode using the variables defined in Table 2. Let the cs-mode be $m_{cs} \in \{0,+\}^n$ and ss-mode be $m_{ss} \in \{-,0,+\}^{2c}$, where c is the number of contacting contacts i.e., $m_{cs}(i) = 0$. At each contact, we have the following kinematic values. Let $J_{c_i} = J_{oc_i} - J_{rc_i}$. Then the normal and tangent velocity constraints are given by

$$N = \begin{bmatrix} n^T J_{c_1} \\ \vdots \\ n^T J_{c_n} \end{bmatrix}, T = \begin{bmatrix} D^T J_{c_1} \\ \vdots \\ D^T J_{c_n} \end{bmatrix}$$

Using the contact mode, we can partition N into the contacting normal velocity constraints $N_c \in \mathbb{R}^{c\times d}$ and the separating normal velocity constraints $N_s \in \mathbb{R}^{s\times d}$, and we can partition T into the sticking tangent velocity constraints $T_z \in \mathbb{R}^{z\times d}$ and the sliding tangent velocity constraints $T_\sigma \in \mathbb{R}^{\sigma\times d}$. At each contact, we also have the following dynamic values. The mapping between contact forces and generalized torques is given by

$$B^T = \begin{bmatrix} J_{c_1}^T F_1 & \cdots & J_{c_n}^T F_n \end{bmatrix} \tag{43}$$

The forces at each frictional force generator are parameterized by $\lambda \in \mathbb{R}^{4n}$. Using the contact mode, we can

Table 2. Variable and parameters for the dynamic equations of motion.

$m_{cs} \in \{0,+\}^n$	Contacting-separating mode
$m_{ss} \in \{-,0,+\}^{2c}$	Sliding-sticking mode
$q, \dot{q} \in \mathbb{R}^d$	Generalized position and velocity
$q_0, \dot{q}_0 \in \mathbb{R}^d$	Initial position and velocity
$M \in \mathbb{R}^{d\times d}$	Mass matrix
$C, g, \tau \in \mathbb{R}^{d\times 1}$	Coriolis/potential/actuator forces
$B \in \mathbb{R}^{4n\times d}$	Contact force generators
$\lambda \in \mathbb{R}^{4n}$	Contact forces
$\lambda_{z\sigma} \in \mathbb{R}^{z+\sigma}$	Active contact forces
$\lambda_0 \in \mathbb{R}^{4n-z-\sigma}$	Inactive contact forces
$N_c \in \mathbb{R}^{c\times d}$	Contacting normal constraints
$N_s \in \mathbb{R}^{s\times d}$	Separating normal constraints
$T_z \in \mathbb{R}^{z\times d}$	Sticking tangent constraints
$T_\sigma \in \mathbb{R}^{\sigma\times d}$	Sliding tangent constraints
$h \in \mathbb{R}_+$	Timestep

partition λ into the active contact forces $\lambda_{z\sigma} \in \mathbb{R}^{z+\sigma}$ and the inactive contact forces $\lambda_0 \in \mathbb{R}^{4n-z-\sigma}$.

We can now present the dynamic equations of motions at a given contact mode. The dynamics are based on the implicit velocity-impulse formulation of Stewart and Trinkle (1996). Let $q_0, \dot{q}_0 \in \mathbb{R}^d$ be the initial position and velocity of the system and let $q, \dot{q} \in \mathbb{R}^d$ be the position and velocity at the next timestep. Let $M \in \mathbb{R}^{d\times d}$ be the mass matrix, $C \in \mathbb{R}^d$ be the Coriolis forces, $g \in \mathbb{R}^d$ be the potential forces, and $\tau \in \mathbb{R}^d$ be the actuator forces. With $h \in \mathbb{R}$ as the timestep, the dynamics at a given contact mode are given by

$$M(\dot{q} - \dot{q}_0) = h\left(B^T\lambda + C + g + \tau\right) \tag{44}$$

$$q = q_0 + h\dot{q} \tag{45}$$

$$0 = N_c\dot{q} \tag{46}$$

$$0 \le N_s\dot{q} \tag{47}$$

$$0 = T_z\dot{q} \tag{48}$$

$$0 \leq T_\sigma \dot{q} \tag{49}$$

$$0 \leq \lambda_{z\sigma} \tag{50}$$

$$0 = \lambda_0 \tag{51}$$

The constraints state that the

(46) contacting normal velocities maintain contact,
(47) separating normal velocities do not penetrate,
(48) sticking tangent velocities are zero,
(49) sliding tangent velocities are in the correct direction,
(50) active contact forces are non-negative,
(51) inactive contact forces are zero.

The important thing to note is that with contact mode specified, contact and friction force models are expressed as simple equality and inequality constraints. This remains true when we eliminate the inertial terms to produce a quasi-static model in the next section.

7. Autogeneration of manipulation primitives

The term "motion primitive" can be very inclusive. In sampling-based planning and control, a motion primitive can be a pre-design robot motion, a continuous/discrete state transfer function (LaValle 2006), a family of trajectories, or a more complicated motion policy (Ratliff et al., 2018). In task and motion planning, a primitive is a smallest symbolic unit which may require further geometric planning but not more discrete symbolic planning (Kaelbling and Lozano-Pérez 2010). The representations of a motion primitive can be an abstract symbolic task (e.g. turn on the dishwasher), a trajectory of motion, a state transfer function, a control policy, etc. In general, a motion primitive enables us to generate smooth motions inside the primitive and perform high-leveling discrete planning on top.

Manipulation is featured by the acyclic contact interactions among the robot, the manipulated objects, and the environment. In comparison, the design motion primitives for wheeled or aerial robots often do not need to consider the discrete changes of dynamics introduced by contact interactions as the trajectories for them are designed to avoid collisions. We focus on motion primitives for manipulation (manipulation primitives). To design manipulation primitives, one must consider the contact-rich nature of manipulation.

While previously studied manipulation primitives are manually assigned by researchers, we define manipulation primitives in a more mathematically precise way. We consider a contact mode as a manipulation primitive. A contact mode maintains continuous and smooth kinematic and dynamic constraints, allowing easy and smooth trajectory generation. This is consistent with the general definition of motion primitives, and not losing the generality of representations — we can easily derive state transfer functions, perform geometric planning, or dynamic trajectory generation inside a contact mode. As enumerated contact modes effectively capture the transitions of discrete states of contacts, this representation is also capable of generating diverse motions when a series of contact modes are sequenced together.

This section introduces the AMP algorithm, a technique for autogenerating manipulation primitives. The AMP algorithm automatically creates a library of motion primitives (contact modes) for a specific system configuration, and performs fast selection and filtering over these motion primitives. The auto-generated library can be used to efficiently generate trajectories within a motion primitive, or to plan a sequence of motion primitives. While the trajectory generation and planning motion sequence are not covered by this paper, there are many existing works addressing these two problems: a sampling-based framework for sequencing and generating motions (Cheng et al., 2021; Kingston et al., 2018), optimization methods for trajectory generation within one mode or a selected sequence of modes (Kelly 2017), etc.

We believe this algorithm has a wide range of applications within robotic manipulation. First, AMP remove the burden of writing task-specific manipulation primitives. Second, AMP adds robustness to modeling errors and failures. For instance if the object is too heavy to pick up, AMP can provide alternatives which are better adapted to heavy objects such as pushing or dragging the object. Lastly, AMP is fast and can be used within planning frameworks to generate sequences of manipulation primitives. We envision AMP having applications in dexterous manipulation ranging from real-time replanning to closed-loop control and more.

The algorithm requires as input the object-environment contact modes, the hand-object contacts, and the quasi-static equations of motion. Within this section, contact mode implies object-environment contact modes. The hand-object contacts are assumed to be sticking for two reasons: (1) sliding hand contacts are challenging to control Doshi et al. (2022), so we filter them out manually; and (2) we deliberately choose sticking hand contacts in order to reduce the number of contact modes. Nonetheless, the AMP algorithm does allow any mode of hand contacts.

The AMP algorithm is based on two key observations. First, specifying the contact mode reduces contact and friction force constraints into equality and inequality constraints (Section 6). This in turn allows us to solve for quasi-static feasibility using linear programming. The generated manipulation primitives are uniquely specified by their contact mode. Second, the contact mode lattice is effectively a graph data structure. This allows us to use graph search techniques when designing our algorithm. Our algorithm walks along the contact mode lattice and solves a linear program at each visited node. When applicable, the algorithm propogates solution information up and down the lattice to prevent unnecessary calls to the linear program solver.

The definition of a primitive varies. One planner's primitive is another planner's sequence. For our purposes,

a primitive means contact mode, maintained over some motion for some amount of time to be determined by the planner. We consider the AMP problem to have been solved once the feasible contact modes have been determined, i.e., the library has been generated. The equivalency between contact modes and manipulation primitives is based on following idea. Under a polyhedral friction model, each contact mode parameterizes a convex region of the state space with smooth dynamics. Therefore, given a primitive (contact mode), the user can apply any fixed-mode trajectory optimizer to create a sequence of controls for that primitive. In this work, we use direct single shooting, identical to our previous work in Cheng et al. (2021).

This work is intended to be general and allow the user to determine which primitives to use based on their own criteria. In our prior work on sample-based planning with autogenerated manipulation primitives in 2D, we extended each feasible primitive from the nearest neighbor state to the sampled state and kept the primitives which made strictly positive progress (Cheng et al., 2021). However, there are other methods for filtering the primitives. As a by-product of the contact mode enumeration, each contact mode is equipped with an interior point i.e., a velocity, which is a certificate for the contact mode. The dot product of the target object velocity with the object velocity component of the certificate can be used to rank primitives. The Manhattan distance between the target object velocity's signed vector and the contact mode could also be used to rank primitives. The next section also provides additional details about the structure of contact modes which can assist in manipulation primitive selection.

7.1. Classification of primitives

In past research, manipulation primitives are named by the researchers studying them. For instance, we can identify papers which study pushing (Lynch and Mason 1996; Mason 1986), pivoting (Aiyama et al., 1993; Holladay et al., 2015; Hou et al., 2018), tumbling (Maeda et al., 2004), and whole-body manipulation (Salisbury 1988). However, the labels assigned by researchers, while evocative, are ultimately imprecise. To automatically generate manipulation primitives, we first need a mathematically precise method for classifying manipulation primitives. Otherwise, we would not be able to distinguish between different manipulation primitives. This work uses contact modes (Section 3) as labels for classifying manipulation primitives. There are several benefits from the contact mode based classification.

1. Contact modes are partially ordered. This allows us to make statements about the primitives such as less than, greater than, equals, or incomparable. If one primitive fails, trying an incomparable primitive could be a good way to ensure the robot explores the space of possible actions.
2. Contact modes have a geometric lattice structure. Each lattice rank contains the primitives with a specific number of degrees of freedom. Selecting a low-rank primitive can provide additional robustness as the motion has fewer degrees of freedom.
3. Contact modes are complete in the sense that every possible primitive can be described in this framework. Grasping, pushing, toppling, pivoting, peg-in-hole, etc. can all be distinctly labeled according to their contact mode.
4. Contact modes provide a description of the motion at each contact point. While these descriptions are not as compact as semantic labels, they are still easily interpreted and visualized.

7.2. Quasi-static feasibility test

Many ideas in this paper were developed with the idea that, "All models are wrong, but some are useful." We believe that, in the real world, manipulation primitives can fail due to all sorts of modeling errors ranging from bad friction coefficients to incorrect geometry (Rodriguez 2021). Moreover, execution errors compound over time. Therefore, the ability to resense, replan, and regenerate relevant manipulation primitives should be prioritized over the ability to create an optimal manipulation trajectory using techniques such as CITO. For this reason, we have chosen to use a quasi-static dynamics model. Though it is less accurate, it simplifies the optimization problem significantly and allows us to quickly find the manipulation primitive which is useful despite its wrongness.

The core routine within the AMP algorithm is determining the quasi-static feasibility of each contact mode. In quasi-static dynamics, we assume that the inertial forces are negligible (Mason, 2001). We use the generator form of the contact dynamics to formulate quasi-static feasibility as the following linear program.

7.2.1. Problem. The quasi-static feasibility of a given contact mode is determined by the existence of a solution to three constraints. A simple practical technique is to solve the following linear program

$$\min_{\lambda} \; 0 \tag{52}$$

$$\text{s.t.} \quad 0 = B^T\lambda + g + \tau \tag{53}$$

$$0 \leq \lambda_{z\sigma} \tag{54}$$

$$0 = \lambda_0 \tag{55}$$

where the B is the contact force generators and λ are the contact forces. The forces λ are further partitioned into the active contact forces $\lambda_{z\sigma}$ and the inactive contact forces λ_0 based on the contact mode. Because inertial forces are assumed to be zero, the quasi-static feasibility of a contact

mode does not depend on the velocity. In other words, the velocity $\dot{q}$ and the contact forces λ are separable. The constraint equations state that

(53) the system is in quasi-static force balance,
(54) the active contact forces are non-negative,
(55) the inactive contact forces are zero.

In our implementation, we used Gurobi Optimization (2016) to solve the above feasibility program. We set the solver method to both primal and dual simplex to encourage solution reuse.

Quasi-static dynamics do not meaningfully restrict the space of possible manipulation primitives. The types of manipulation primitives which satisfy quasi-static dynamics are those which can be paused at anytime in a statically stable configuration. (Note the reverse is not true.) Examples of quasi-statically stable primitives include pushing, grasping, pulling, and pivoting. Counter-examples include toppling, dropping[2], and throwing. In the future, we plan to extend our models to include quasi-dynamic motions i.e., where the dynamics contain non-zero inertial terms but each timestep is integrated with the assumption that the previous velocity was zero (Mason 2001). This will allow us to generate additional primitives such as toppling and dropping but not throwing.

7.3. Full mode algorithm

Algorithm 4. The Full Mode Algorithm

```
1: function FULL-MODE (ℒ)
2:   for u ∈ ℒ do
3:     f ←Solve-Feasibility (u)
4:     Set-Feasible (u, f)
5:   return
```

As a baseline for comparison, we implemented an algorithm which computes the quasi-static feasibility for each contact mode. FULL-MODE takes as input the partial hyperplane arrangement lattice and adds a feasibility flag to each element of the lattice. This is in essence our previously published algorithm in Cheng et al. (2021) except that the expensive quadratic program has been replaced with a linear program and the problem is now in 3D. The pseudo-code is listed in Algorithm 4.

7.4. AMP algorithm

Algorithm 5. The AMP Algorithm

```
1: function AMP-CS(ℒ_cs)
2:   cs Q←Atoms(ℒ_cs)
3:   while Not-Empty(Q) do
4:     u ←Pop(Q)
5:     if Not-Feasible(u) then
6:       continue
```

(continued)

(continued)

```
7:     AMP-SS(u)
8:     if Feasible(u) then
9:       for v ∈ Super-Modes(u) do
10:        if Not-Visited(v) then
11:          Push-Back(Q, v)
12:          Set-Visited(v), True
13:    else
14:      Set-Parents-As-Infeasible(u)
15: function AMP-SS(m_cs)
16:   ℒ_ss← Enum-ss-Modes(m_cs)
17:   Q← Atoms(ℒ_ss)
18:   Set-Feasible(m_cs), False
19:   while Not-Empty (Q) do
20:     u ←Pop(Q)
21:   if Unknown-Feasibility(u) then
22:     f ←Solve-Feasibility(u)
23:     Set-Feasible(u, f)
24:   if Feasible(u) then
25:     Set-Feasible(m_cs), True
26:     for v ∈ Super-Modes(u) do
27:       if Not-Visited(v) then
28:       Push-Front(Q, v)
29:       Set-Visited(v), True
30:     Set-Children-As-Feasible(u)
31:   else
32:     Set-Parents-As-Infeasible(u)
33: function SET-PARENTS-AS-INFEASIBLE(u)
34:   Q ← u
35:   while Not-Empty(Q) do
36:     u ←Pop(Q)
37:     for v ∈ Sub-Modes(u) do
38:       if Unknown-Feasibility(v) then
39:         Set-Feasible(v), False
40:         Push-Back(Q, v)
41: function SET-CHILDREN-AS-FEASIBLE(u)
42:   Q ← u
43:   while Not-Empty(Q) do
44:     u ←Pop(Q)
45:     for v ∈ Super-Modes(u) do
46:       if Unknown-Feasibility(v) then
47:         Set-Feasible(v), True
48:         Push-Back(Q, v)
```

The AMP algorithm improves upon the full mode algorithm by propogating feasibility information up and down the lattice. This strategy reduces the number of calls to the linear program solver i.e., the most computationally expensive part of the algorithm. The partial ordering on the contact modes implies that the active frictional force bases of the parent are strictly a subset of the child's active frictional force bases.

7.4.1. Definition. A contacting-separating mode u is feasible if and only if there exists a sliding-sticking mode at u which is feasible.

7.4.2. Proposition. If a contact mode u is feasible, then all contact modes v such that $v \leq u$ are feasible. If a contact

mode u is infeasible, then all contact modes v such that $u \leq v$ are infeasible. This statement holds for both contacting-separating modes and sliding-sticking modes.

Proof. We begin by showing the statement holds for sliding-sticking modes. At a single contact point, the possible sliding-sticking modes have the following partial order

Where a line between the left mode u and the right mode v implies that $u \leq v$. Let Λ_u and Λ_v be the active set of frictional force generators for u and v, respectively. By inspection, if $u \leq v$ then $\Lambda_v \subseteq \Lambda_u$ (see Section 6). Clearly, this relationship holds for sliding-sticking modes of multiple contact points. The proposition follows from contradiction. If u is infeasible, there cannot be a feasible v such that $u \leq v$ because $\Lambda_v \subseteq \Lambda_u$ implies u would be feasible. If u is feasible, there cannot be an infeasible v such that $v \leq u$ because $\Lambda_u \subseteq \Lambda_v$ implies v is feasible.

Next we show that the proposition holds for contacting-separating modes. Let u and v be two contacting-separating modes. Observe that w is a feasible sliding-sticking mode at u if and only if 0 is also a feasible sliding-sticking mode at u, because $0 \leq w$ and 0 is always a valid contact mode. Let Λ_u and Λ_v be the active frictional force generators for 0 at u and 0 at v, respectively. If $u \leq v$, then clearly $\Lambda_v \subseteq \Lambda_u$. The proposition follows from the same proof by contradiction as before.

The AMP pseudo-code in Algorithm 5 describes a method to incorporate Proposition 7.3 and the lattice structure into the feasibility tests at each contact mode. AMP-CS takes the contact-separating mode lattice as input, and the output is the revised lattice. The algorithm traverses the lattice from bottom up in a depth-first manner. The traversal queue Q is initialized with the set of elements $u \in \mathcal{L}_{cs}$ such that $u \leq v$ for all comparable $v \in \mathcal{L}_{cs}$. These elements are known as *atoms*. We provide a proof of correctness.

7.4.3. Theorem. Algorithm 5 computes the feasibility of each sliding-sticking mode at every contacting-separating mode.

Proof. We begin by arguing the correctness of AMP-CS, which is the entry-point of the AMP algorithm. This function traverses the input contacting-separating mode lattice $\mathcal{L}_{cs}$ from the bottom up and determines the feasibility of each element $u \in \mathcal{L}_{cs}$. We initiate a traversal queue Q with the atoms of $\mathcal{L}_{cs}$. We assume that all elements in $\mathcal{L}_{cs}$ are initialized with feasibility set to unknown. We claim that the while loop in AMP-CS calls AMP-SS for all feasible u. First we observe that the set of feasible elements is connected because for all feasible $u \in \mathcal{L}_{cs}$ the element $0. \leq u$. Next we observe that for every feasible $u \in Q$, the while loop calls AMP-SS and adds the parents of u to Q. This shows that AMP-SS is called for every feasible u. As an additional time-saving measure, when the function encounters an infeasible u, it marks all v such that $u \leq v$ as infeasible.

The next step is to argue the correctness of AMP-SS. Let $\mathcal{L}_{ss}$ be the sliding-sticking mode lattice enumerated at the input contacting-separating mode. Similar to before, the set of feasible elements in $\mathcal{L}_{ss}$ is connected and the while-loop ensures that every feasible element is pushed into and popped from Q. This function also includes another run-time improvement based on Proposition 7.3. Whenever a feasible element u is encountered, every element v such that $v \leq u$ is also set to feasible.

We will conclude this section with a discussion of the computational complexity of the AMP algorithm. The number of elements and edges in the contacting-separating mode lattice is $O(n^{\lfloor d/2 \rfloor})$ (Edelsbrunner 2005). In the worst case scenario, the AMP-CS function calls AMP-SS on every element and traverses every edge. This scenario occurs when the entire lattice is feasible. The number of elements and edges in a sliding-sticking mode lattice is $O(n^d)$ (Edelsbrunner 2005). We observe that the AMP-SS function traverses the lattice in a depth-first manner and can touch every edge at most twice. Let k be the number of feasible coatoms in $\mathcal{L}_{ss}$, i.e. feasible u such that for all comparable v, $v \leq u$. In the worst case, the function must call d linear program solves for every coatom because d is the maximal chain length in $\mathcal{L}_{ss}$. Therefore the number of linear program solves is $O(n^{\lfloor d/2 \rfloor} dk)$ and the number of edges traversed is $O(n^{\lfloor 3d/2 \rfloor})$. From a performance perspective, the linear program solver is the bottleneck in this algorithm. Moreover, the number of contact modes is typically fairly small for a single object, on the order of 1000 sliding-sticking modes over all contacting-separating modes (see Section 8).

8. Experiments

This section presents several experimental results and benchmarks which support our approach towards dexterous robotic manipulation. In the first section, we demonstrate the viability of contact mode enumeration for a single object. We report results on computation time and mode complexity, an often misunderstood subject. The second section reports computational results for the autogeneration of manipulation primitives. In the final section, we showcase the usefulness of autogenerated manipulation primitives in simulated planning problems and real-world experiments.

We created a number of manipulation-specific test cases for the experiments in this section. These test cases were chosen to represent a range of possible contact scenarios for a single object. The number of tangent planes was set to two i.e., a 4-sided friction cone. We included open environments like a cup on the table-top and constrained environments like a book sandwiched between two other books. We generated 32 test environments in total. The experiments were run on a computer with an Intel i7-7820x CPU (3.5 MHz, 16 threads).

8.1. Enumeration of contact modes

This section reports the results of running our enumeration algorithms on the test cases. The runtime results are graphed

in Figure 13. The convex-hull method is able to enumerate contacting-separating modes in less than 1 ms. The time for enumerating sliding-sticking modes ranges from 1 ms to 63 ms. Moreover, the partial hyperplane arrangement method is consistently faster than the zonotope method, and on certain problems, it is up to 10x faster. The most challenging enumeration problems for the zonotope method are the contacting-separating modes which have a large number of contacting contacts i.e., have many 0s. In the zonotope method, normal velocity hyperplanes are treated as two-sided hyperplanes. In the partial hyperplane arrangement method, normal velocity hyperplanes are treated as one-side hyperplanes. As a result, the zonotope method does twice the work for each 0 in the contacting-separating mode.

Figure 14 visualizes the number of contacting-separating modes and sliding-sticking modes against the number of contact points. The sliding-sticking mode count was taken as the total number of sliding-sticking modes across all contacting-separating modes. We observed that the number of contacting-separating modes is almost always under 50. The outlier at 136 belongs to the test case where a book is flush against a corner in the bookshelf. Interestingly, there are exactly 136 sliding-sticking modes too for that test case. We observed that the number of sliding-sticking modes to be around 1000 or less typically. The larger sliding-sticking mode counts can be attributed to the bowl-within-a-bowl test cases. For a single object with (simplified) geometry, the key take-away message is that the number of contact modes is within 1000 and the number of visually distinctive ways to make and break contacts is 50. This may appear to be at odds with the computational complexity of $O(n^d)$ we derived for the enumeration algorithms. One expects a single object with 12 contacts to have $12^6 = 2{,}985{,}984$ contact modes. This discrepancy is caused by several factors that reduce the dimensionality of problem. First, contacts often generate degenerate hyperplanes. In the test case of a cup on a table, the contacts are all coplanar and share the same normal. The normal velocity inequalities generated by these contacts have an affine dimension of two instead of 6. Second, a contacting-separating mode u restricts a number of contacts to be contacting i.e., $n^T v = 0$ where n is a normal velocity constraint and v is the velocity. We use this constraint to futher reduce the dimensionality before enumerating the sliding-sticking modes at u. Finally, each additional contact point restricts the valid object motions. Therefore, as the number of contacts increase, the dimensionality often decreases.

8.2. Autogeneration of manipulation primitives

We added additional hand contact points to each test case and computed the feasible manipulation primitives. We constrained the hand contact points to be sticking. The computation times for the baseline and the AMP algorithm

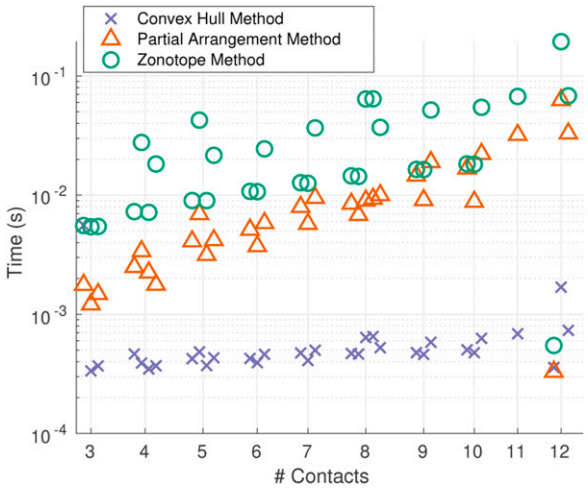

Figure 13. Plot of runtime for convex hull, partial arrangement, and zonotope methods versus the # of contacts. The *y*-axis is log-scale.

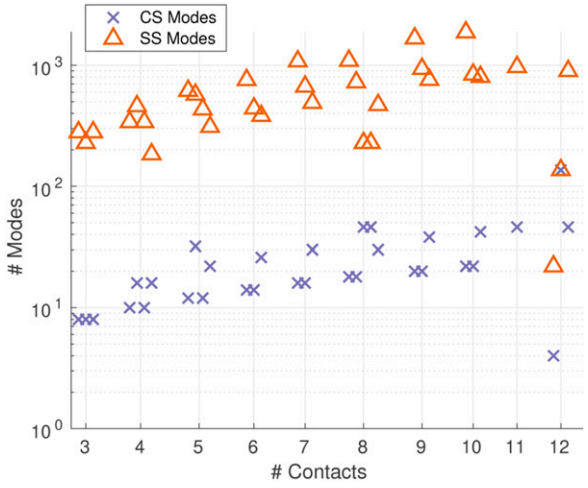

Figure 14. Plot of # of contact modes versus the # of contacts.

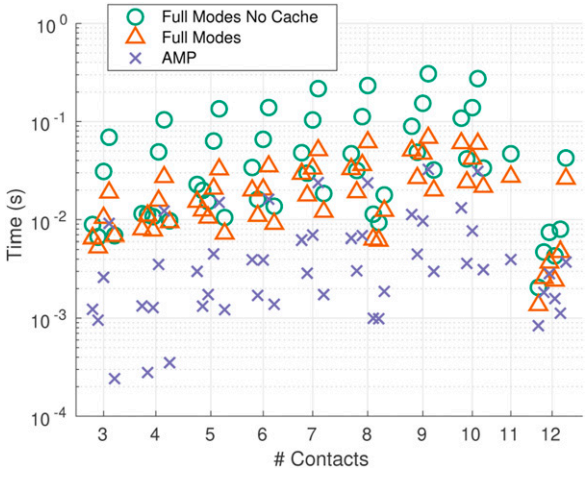

Figure 15. Figure showing the autogeneration time versus # of contacts.

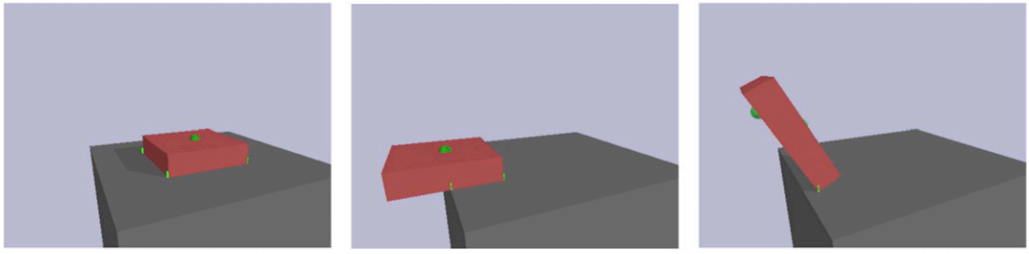

Figure 16. The generated motion sequence for the test case "pick up a book" (use case: sequencing manipulation primitives). Left to right: sliding the object, making a contact from the bottom surface, and levering up.

are plotted in Figure 15. The "no cache" version of the full modes algorithm resets the linear programming solver after each feasibility test. Between solves, the only changes to the feasibility problem defined in Problem 7.1 are to the upper bounds on contact forces. The cached version reuses solution basis and the matrix factorization from the previous solve to speed up computation. On average, we observed that the cached full modes algorithm results in a 10x speed-up over the uncached version. By propagating feasibility information throughout the lattice, the AMP algorithm runs on average 6x faster than the (cached) full-modes algorithm.

8.3. Sequencing manipulation primitives

One use case is an intensively studied yet not solved problem: planning through contacts Posa et al. (2014); Manchester and Kuindersma (2020). The main difficulty of this problem is the nonlinearity and nonconvexity introduced by frictional contacts dynamics. With the algorithms in this paper, we can use the framework of contact modes to decompose the search space into multiples subspaces and guide the search process along contact-mode based manipulation primitives. Figures 16 and 17 shows two test cases in 3D: "pick up a book" and "retrieve a bottle". Both test cases need dexterous maneuvers to achieve the goals. In the "pick up a book" task, the object is too thin to be picked up directly. In Figure 16, a contact mode guided sampling-based planner from Cheng et al. (2021, 2022) uses autogenerated manipulation primitives to help it find a solution of first sliding the object to the edge of the table, then making contact on the object bottom surface to pick the object up. In the "retrieve a bottle" task, the object is occluded, and there is no space to form a grasp. As shown in Figure 17, the planner comes up with pivoting the object first to create some space to form a grasp. This generated strategy is similar to "simultaneous levering out and grasp formation" observed in human grasping by Nakamura et al. (2017). The generated trajectories can be viewed in Multimedia Extension 2.

8.4. Robustness to model uncertainty

We designed three real-world experiments that highlight how a library of manipulation primitives adds robustness to model uncertainty. These experiments can be viewed in Multimedia Extension 3. The first experiment in Figure 18 demonstrates how AMP could provide robustness in the presence of an incorrect mass model. Suppose a robot was tasked with moving a box. In the top sequence, the robot grasped and lifted the box which was empty. In the middle sequence, the robot attempted to grasp and lift the same box, but this time the box was filled with chess pieces and it slipped out of the grasp. In the bottom sequence, robot successfully press-pulled the filled box to the right, despite the additional weight. The experiment in Figure 19 demonstrates how AMP could provide robustness in the presence of incorrect model geometry. Suppose a robot was tasked with rotating a rectangular block onto its side. In the top sequence, the robot made contact with the top-right of the block, pivoted it on the right edge, and squeezed the block from the left to finish the rotation. In the middle sequence, the robot attempted to pivot the block, but this time the block was replaced with a rhomboid. The obtuse top-right edge caused the finger contact to slip and the robot failed to finish the rotation. In the bottom sequence, the robot approached the rhomboid from a different orientation. This primitive could sampled by running AMP on a mirrored set of contact points. During this execution, the acute top-right edge enabled the pivot motion to complete and the robot successfully rotated the rhomboid. The experiment in Figure 20 demonstrates how AMP could provide robustness in the presence of incorrect or unknown friction coefficients. Suppose a robot was tasked with moving a block from left to right. In the top sequence, the robot successfully press-pulled the block because the friction coefficient between the block and the hand was greater than the friction coefficient between the surface and the block. We created this scenario by gluing high-friction felt to the top of the block. In the middle sequence, the robot attempted to press-pull the block, but this time the high-friction felt was on the bottom of the block. The sliding motion was no longer dynamically feasible and the block toppled over in failure. In the bottom sequence, the robot successfully pushed the block. The high-friction felt-surface contact had no impact on the pushing primitive, unlike on the press-pull primitive.

9. Limitations & future work

In our implementation, we observed that numerical errors in the inputs to the sliding sticking mode algorithm can cause the algorithm to return an incorrect enumeration. As our

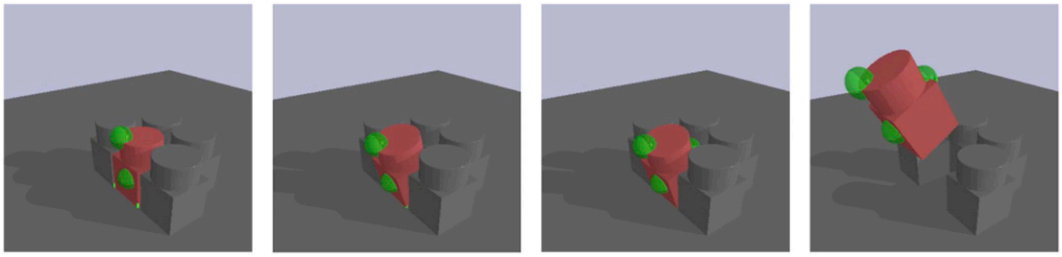

Figure 17. The generated motion sequence for the test case "retrieve a bottle" (use case: sequencing manipulation primitives). Left to right: pivoting, forming a grasp, and picking up.

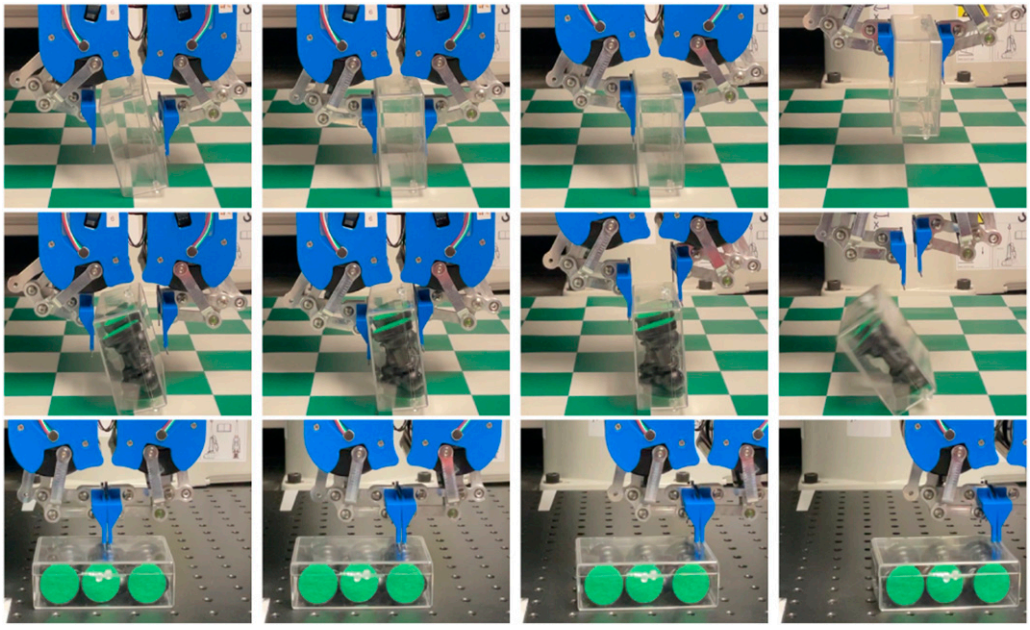

Figure 18. (Top) successful grasp of an empty box. (Middle) failed grasp of a weighted box. (Bottom) successful press-pull of weighted box.

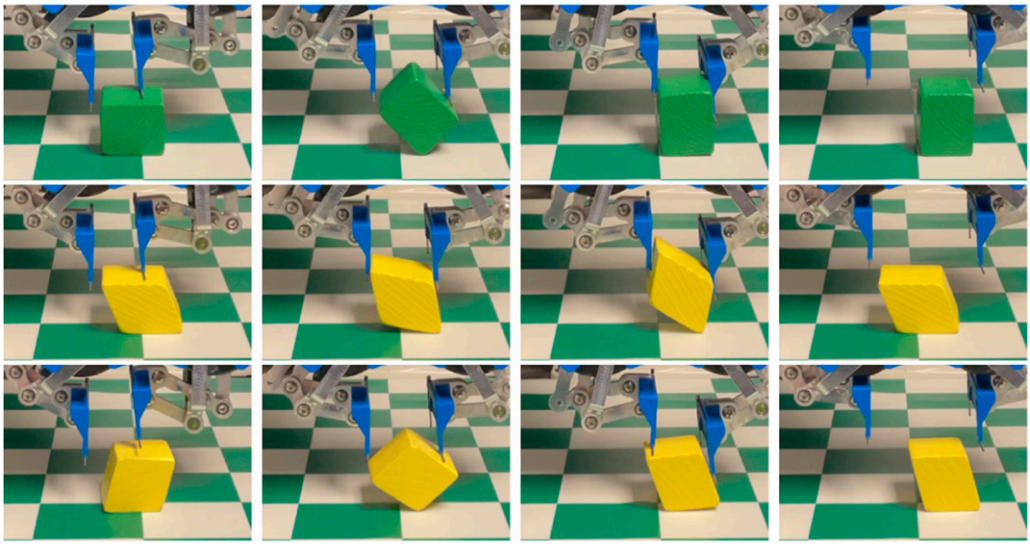

Figure 19. (Top) successful pivot of a rectangle. (Middle) failed pivot when applied to a rhomboid. (Bottom) successful pivot when applied to rhomboid at a different orientation.

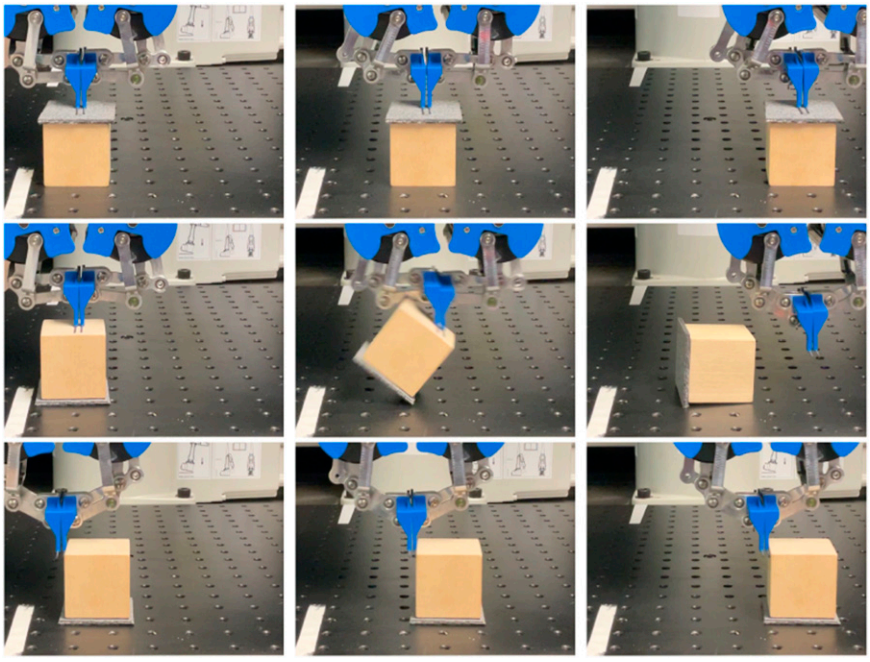

Figure 20. (Top) successful press-pull with a high-friction hand contact. (Middle) failed press-pull with a low-friction hand contact. (Bottom) successful push despite high-friction surface contact.

algorithm tries to satisfy the input constraints exactly, large enough errors can cause the algorithm to decide that no motion is possible. Typically, this causes the autogenerated manipulation primitives to be missing a small number of primitives. In practice, we have observed that the generated library is still sufficient for planning purposes (Cheng et al., 2021).

In our experiments, we assumed sticking contacts between the hand and the object for two reasons. First, controlling sliding modes between the hand and the object on robotic hardware experiments was outside the scope of this work. Second, the contact modes between the object-environment and between the object-hand are independent. The independence implies that the number of contact modes in the system are product of each contact subsystem. For future work, we hope to improve the performance of the AMP algorithm to also reason about the object-hand contact modes.

Algorithm 6. Preprocess hyperplanes

1: **function** PREPROCESS (A^T)
2: $C \leftarrow$Column-Space(A)
3: $A \leftarrow C^T A$
4: $A \leftarrow$Remove-Zeros(A)
5: $A\,[:, i] \leftarrow A\,[:, i]/\|A\,[:, i]\|$
6: $A \leftarrow$Remove-Parallel (A)
7: **return** A^T

10. Conclusion

This paper extended the theory and computation of contact modes introduced in Huang et al. (2020) into a principled method for autogenerating manipulation primitives. The main contributions are a d-th root complexity improvement in sliding-sticking mode enumeration and the AMP algorithm for autogenerating manipulation primitives. We validated our algorithms on a number of experiments. The computational experiments demonstrated the efficacy of contact mode enumeration and AMP on 3D manipulation problems. The entire pipeline runs in 10s of milliseconds. We also demonstrated the usefulness of having libraries of manipulation primitives for planning dexterous manipulation and for robustness to model errors. We hope that our contributions will lead to more general and more robust approaches for robotic manipulation.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the National Science Foundation; IIS-1637908; National Science Foundation; IIS-1813920 and National Science Foundation; IIS-1909021.

ORCID iDs

Eric Huang https://orcid.org/0000-0002-2182-2770
Xianyi Cheng https://orcid.org/0000-0001-8342-9459

Supplemental Material

Supplemental material for this article is available online.

Notes

1. Our interpretation of their work is that the runtime of their algorithm is at least O (num-cs-modes × n^2 × work-per-cs-mode-hypothesis) where n is the number of potentially adjacent contacts. It is unclear to us what the complexity of testing each cs-mode hypothesis is in their algorithm.
2. Both toppling and dropping can be accomplished with a mix of other quasi-static primitives, like pivoting and placing.

References

Aiyama Y, Inaba M and Inoue H (1993) Pivoting: A new method of graspless manipulation of object by robot fingers. In Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 93), Tokyo, Japan, July 26–30, 1993. vol. 1: 136–143

Avis D and Fukuda K (1996) Reverse search for enumeration. *Discrete Applied Mathematics* 65(1): 21–46.

Barber CB, Dobkin DP and Huhdanpaa H (1996) The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software* 22(4): 469–483.

Barry J, Hsiao K, Kaelbling LP, et al. (2013) Manipulation with multiple action types. In *Experimental Robotics*. Berlin: Springer. 531–545.

Bjorner A, Björner A, Las Vergnas M, et al. (1999) *Oriented Matroids*. Cambridge: Cambridge University Press.

Chavan-Dafle N, Holladay R and Rodriguez A (2018) Planar in-hand manipulation via motion cones.*The International Journal of Robotics Research* 39, 163–182.

Cheng X, Huang E, Hou Y., et al. (2021) Contact mode guided sampling-based planning for quasistatic dexterous manipulation in 2d. In Robotics and Automation (ICRA), 2021 IEEE International Conference on, Xi'an, China, May 30–June 5, 2021. IEEE.

Cheng X, Huang E, Hou Y, et al. (2022) Contact mode guided motion planning for quasidynamic dexterous manipulation in 3d. In 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, USA, May 23–27, 2022. IEEE. 2730–2736.

Delos V and Teissandier D (2015) Minkowski sum of polytopes defined by their vertices. *Journal of Applied Mathematics and Physics* 3(1): 62–67.

Doshi N, Taylor O and Rodriguez A (2022) 'Manipulation of unknown objects via contact configuration regulation'. (2022). In IEEE International Conference on Robotics and Automation (ICRA), Philadelphia, USA, May 23–27, 2022.

Edelsbrunner H (2005) *Algorithms in Combinatorial Geometry*. Berlin: Springer-Verlag.

Erdmann M (1998) 'An exploration of nonprehensile two-palm manipulation' *The International Journal of Robotics Research* 17: 485–503

Facchinei F and Pang JS (2007) *Finite-dimensional Variational Inequalities and Complementarity Problems*. Berlin: Springer Science & Business Media

Geyer T, Torrisi FD and Morari M (2010) Efficient mode enumeration of compositional hybrid systems. *International Journal of Control* 83(2): 313–329

Greenfield A, Saranli U and Rizzi AA (2005) Solving models of controlled dynamic planar rigid-body systems with frictional contact. *The International Journal of Robotics Research* 24(11): 911–931

Gurobi Optimization I (2016) *Gurobi Optimizer Reference Manual*. https://www.gurobi.com

Haas-Heger M, Papadimitriou C, Yannakakis M, et al. (2018) Passive static equilibrium with frictional contacts and application to grasp stability analysis. In Proceedings of Robotics: Science and Systems. Pittsburgh, PA, June 26–30, 2018.

Hogan FR, Ballester J, Dong S, et al. (2020). Tactile dexterity: Manipulation primitives with tactile feedback. In 2020 IEEE International Conference On Robotics And Automation (ICRA), Palais des Congrès de Paris in Paris, France, May 31–June 4, 2020. IEEE. 8863–8869.

Holladay A, Paolini R and Mason MT (2015), A general framework for open-loop pivoting. In 2015 IEEE International Conference on Robotics and Automation (ICRA), Washington State Convention Center in Seattle, Washington, USA, May 26–30, 2015. IEEE. 3675–3681.

Horak PC and Trinkle JC (2019) On the similarities and differences among contact models in robot simulation. *IEEE Robotics and Automation Letters* 4(2): 493–499.

Hou Y, Jia Z and Mason MT (2018) Fast planning for 3d any-pose-reorienting using pivoting. In 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane Convention & Exhibition Centre in Brisbane, Australia, May 21–25, 2018. IEEE. 1631–1638.

Huang E, Cheng X and Mason MT (2020) Efficient contact mode enumeration in 3d. In Workshop on the Algorithmic Foundations of Robotics, Sokos Hotel Eden, Oulu, Finland, June 21–23, 2021.

Johnson AM and Koditschek DE (2013) Toward a vocabulary of legged leaping. In 2013 IEEE International Conference on Robotics and Automation, Germany, Karlsruhe, May 6–10, 2013: 2568–2575

Kaelbling LP and Lozano-Pérez T (2010) Hierarchical planning in the now. In Workshops at the Twenty-Fourth AAAI Conference on Artificial Intelligence, Westin Peachtree Plaza in Atlanta, Georgia, USA, July 11–15.

Kaibel V and Pfetsch ME (2002) Computing the Face Lattice of a Polytope from Its Vertex-Facet Incidences. *Computational Geometry* Elsevier. 23 (3), 281–290.

Kelly M (2017) An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review* 59(4): 849–904.

King J, Haustein J, Srinivasa S, et al. (2015) Nonprehensile whole arm rearrangement planning with physics manifolds. In IEEE International Conference on Robotics and Automation

King J, Cognetti M and Srinivasa S (2016) Rearrangement planning using object-centric and robot-centric action spaces. In IEEE International Conference on Robotics and Automation, Stockholm Waterfront Congress Centre in Stockholm, Sweden, May 16–21, 2016.

Kingston ZK, Moll M and Kavraki LE (2018) Sampling-based methods for motion planning with constraints. *Annual Review of Control, Robotics, and Autonomous Systems* 1: 159–185.

Landry B, Lorenzetti J, Manchester Z, et al. (2019) Bilevel optimization for planning through contact: A semidirect method. In Proceedings of International Symposium on Robotics Research (ISRR 19), Hanoi, Vietnam, October 6–10, 2019.

LaValle SM (2006) *Planning Algorithms*. Cambridge: Cambridge University Press.

Leveroni SR (1997) *Grasp Gaits for Planar Object Manipulation*. PhD Thesis. Cambridge, MA: Massachusetts Institute of Technology.

Li S, Lyu S and Trinkle J (2015) State estimation for dynamic systems with intermittent contact. In 2015 IEEE International Conference on Robotics and Automation (ICRA), Washington State Convention Center in Seattle, Washington, USA, May 26–30, 2015. IEEE. 3709–3715.

Lynch KM and Mason MT (1996) Stable pushing: Mechanics, controllability, and planning. *The International Journal of Robotics Research* 15(6): 533–556.

Maeda Y, Nakamura T and Arai T (2004). Motion planning of robot fingertips for graspless manipulation. In IEEE International Conference on Robotics and Automation, 2004 Proceedings ICRA 04 2004, Hilton New Orleans Riverside, New Orleans, LA, USA, April 26–May 1, 2004. Vol. 3. IEEE: 2951–2956.

Manchester Z, Doshi N, Wood RJ, et al. (2019) Contact-implicit trajectory optimization using variational integrators. *The International Journal of Robotics Research* 38(12): 1463–1476.

Manchester Z and Kuindersma S (2020) Variational contact-implicit trajectory optimization. Robotics Research: The 18th International Symposium ISRR, Puerto Varas, Chile, December 11–14, 2017. Springer. 985–1000.

Mason MT (1986) Mechanics and planning of manipulator pushing operations. *The International Journal of Robotics Research* 5(3): 53–71.

Mason MT (2001) *Mechanics of Robotic Manipulation*, Cambridge, MA: MIT Press.

Michelman P and Allen PK (1994) Forming complex dextrous manipulations from task primitives. Proceedings of the 1994 IEEE International Conference on Robotics and Automation, San Diego (CA), USA, May 8–13, 1994. vol. 4: 3383–3388.

Murty KG and Yu FT (1988) *Linear Complementarity, Linear And Nonlinear Programming* Citeseer. 3

Nakamura YC, Troniak DM, Rodriguez A., et al. (2017) The complexities of grasping in the wild. In 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids), The REP Theatre in Birmingham, UK, November 15–17, 2017. IEEE. 233–240.

Önol AÖ, Long P and Padir T (2018) *Contact-implicit Trajectory Optimization Based on a Variable Smooth Contact Model and Successive Convexification*. In 2019 International Conference on Robotics and Automation (ICRA), Montreal, Canada, May 20, 2019. 2447–2453

Oxley JG (2006) Matroid theory. Oxford: Oxford University Press

Posa M, Cantu C and Tedrake R (2014). A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research* 33(1): 69–81.

Potočnik B, Mušič G and Zupančič B (2004). A new technique for translating discrete hybrid automata into piecewise affine systems. *Mathematical and Computer Modelling of Dynamical Systems* 10(1): 41–57

QHull Library (2020) QHalf notes. https://www.qhull.org/html/qhalf.htm

Ratliff ND, Issac J, Kappler D, et al. (2018) Riemannian Motion Policies. *arXiv preprint arXiv:1801.02854*

Rodriguez A (2021) The unstable queen: Uncertainty, mechanics, and tactile feedback. *Science Robotics* 6: eabi4667

Salisbury K (1988) Whole arm manipulation, In Proceedings of the 4th international symposium on Robotics Research, Cambridge, MA, October 7–1988. MIT Press. 183–189

Stewart DE and Trinkle JC (1996) An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal for Numerical Methods in Engineering* 39(15): 2673–2691.

Tassa Y, Erez T and Todorov E (2012) Synthesis and stabilization of complex behaviors through online trajectory optimization. In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura, Algarve, Portugal, October 7–12, 2012. IEEE. 4906–4913

Todorov E (2014) Convex and analytically-invertible dynamics with contacts and constraints: Theory and implementation in Mujoco. In 2014 IEEE International Conference on Robotics and Automation (ICRA), The Hong Kong Convention and Exhibition Center, Hong Kong, China, May 31–June 5, 2014. IEEE. 6054–6061.

Xiao J and Ji X (2001) Automatic generation of high-level contact state space. *The International Journal of Robotics Research* 20(7): 584–606.

Ziegler GM (1995) Lectures on polytopes. *Graduate Texts in Mathematics*, New York: Springer-Verlag

Appendix

A.1. Preprocessing hyperplanes for contact mode enumeration

A.1.1. Matroid theory. This section provides the pertinent details of matroid theory which will help us preprocess the input normal and tangent velocity constraints. Specifically, it will help us remove degenerate hyperplanes and reduce problem dimensionality. A *matroid* is the pair $(E, \mathcal{I})$ of a finite set E and a collection $\mathcal{I}$ of subsets of E such that

(I1) $\varnothing \in \mathcal{I}$
(I2) If $I \in \mathcal{I}$ and $I' \subseteq I$, then $I' \in \mathcal{I}$.
(I3) If I_1 and I_2 are in $\mathcal{I}$ and $|I_1| < |I_2|$, then there is an element e of $I_2 - I_1$ such that $I_1 \cup e \in \mathcal{I}$.

Let $A \in \mathbb{R}^{d \times n}$ be a matrix. Let E be the set of column vector indices of A and $\mathcal{I}$ the collection of subsets $I \subseteq E$

such that the corresponding vectors in A are linearly independent. This matroid $M[A]$ is known as a *vector matroid*. The *signed covectors* of $M[A]$ are the elements of the set $\mathcal{V}^*(A) = \{\text{sign}(c^T A) : c \in \mathbb{R}^d\}$ The signed covectors are one of several sets of data which uniquely define a matroid. If we interpret c as a point in $\mathbb{R}^d$, then the signed covectors of $M[A]$ are the signed vectors of the arrangement $\mathcal{A}(A^T)$.

The matroid $M[A]$ is unchanged if one performs any of the following operations on A.

(E1) Interchange two rows.
(E2) Multiply a row by a non-zero member of R.
(E3) Replace a row by the sum of that row and another.
(E4) Adjoin or remove a zero row.
(E5) Interchange two columns (moving their labels with their columns).
(E6) Multiply a column by a non-zero member of R.
(E7) Replace each matrix entry by its image under some automorphism of $\mathbb{R}$

Using operations **(E1)-(E5)**, one can always reduce a matrix A into the form $[I|D]$. The only automorphism of the real numbers is the identity map. Therefore, ignoring **(E7)**, we arrive at the following notion of *projective equivalence*. Matrices $A_1, A_2 \in \mathbb{R}^{d \times n}$ are *projectively equivalent* representations of a matroid if and only if there exists a non-singular matrix $X \in \mathbb{R}^{d \times d}$ and non-singular diagonal matrix $Y \in \mathbb{R}^{n \times n}$ such that $A_2 = XA_1Y$. For further reading, the books by Oxley and Bjorner are excellent references Oxley (2006); Bjorner et al. (1999).

A.1.2. Preprocessing Hyperplanes. The preprocessing routine is an important step which removes degeneracies from the input hyperplanes and reduces problem dimensionality. Recall from Section A.1.1 that a linear hyperplane arrangement $\mathcal{A}(A^T)$ is combinatorially equivalent to the vector matroid $M[A]$. This section outlines a matroid-based preprocessing routine for reducing the set of normal and tangent velocity hyperplanes into a minimal set of projectively equivalent hyperplanes. For simplicity, we first present our preprocessing routine for the zero offset case, i.e. $N\dot{q} \leq 0$. Afterwards, we will describe how to extend this test to non-zero offsets.

Let $A = [N; T]^T \in \mathbb{R}^{d \times m}$ be the input hyperplane normals in column vector form. We want to find invertible matrices X and Y, with Y diagonal, such that $XAY = A' = [N'; T']^T \in \mathbb{R}^{d' \times m'}$ has minimal dimensions. We can accomplish this in two steps.

1. Reduce dimension: To minimize the dimension d, we can choose $X = [C^T; L^T]$ where $C \in \mathbb{R}^{d' \times d}$ is the an orthonormal basis for the column space of A and $L \in \mathbb{R}^{(d-d') \times d}$ is a basis of the left nullspace of A. After multiplying X and A, we can remove the bottom $d - d'$ rows (**E4**). Note that this operation does not change the orientation of the hyperplanes.

2. Remove duplicate/zero hyperplanes: First, we remove any hyperplanes with zero magnitude. Next, we use Y to normalize each hyperplane and remove any hyperplanes that are parallel to a previous column in A. This operation maintains projective equivalence because $M[A]$ is defined over the *set* of column vectors in A.

We can extend the above test to handle non-zero offsets by lifting the hyperplanes into dimension $d + 1$. Given a hyperplane $h = \{x \in \mathbb{R}^d : ax = b\}$, observe that h is the orthogonal projection of a $d + 1$ dimensional linear hyperplane onto the unit vertical plane

$$\{y : [a, -b]y = 0\} \cap \{y : y_{d+1} = 1\}, y \in \mathbb{R}^{d+1} \tag{56}$$

Therefore, we can extend this test to the non-zero offset case by changing the inputs to $N' = [N, \phi]$ and $T' = [T, 0]$, where ϕ is the offset.