Determining Contact Data for Time Stepping Rigid Body Simulations with Convex Polyhedral Geometries

Bjoern Cheng Yi[1] and Evan M. Drumwright[2]

***Abstract*— Dynamic simulation of multi-rigid bodies is a key tool in robotics and many other domains. Over the past two decades, such simulations have moved from modeling primariliy contact-free motion to simulating contact-based interactions like locomotion and grasping. Contact data—points of contact, surface normals, and signed distance—have proven straightforward to compute for multi-bodies modeled using primitive shapes, but there has been no accepted procedure for determining such data for bodies with convex polyhedral bodies. This paper investigates techniques that can be used to determine such data and pays special attention to numerical issues. We describe a unit test and provide two verification benchmarks for assessing correctness.**

I. INTRODUCTION

Multi-rigid body simulation is a key tool used for modeling the behavior of many robotic systems. Importing kinematic and dynamic parameters is somewhat tedious, but can be accomplished for, e.g., typical legged and manipulator robots, within a few hours. Importing geometry for governing contact interactions is much more challenging.

One of the more common representations is that of polygon-based geometries, which are ubiquitous in geometric modeling for both computer animation and computer aided engineering. In the former context, triangle meshes, also known as "triangle soups", provide the ability to approximate shapes, both solid and otherwise, to geometric precision that scales with the number of triangles. In the latter context, geometries comprised of interconnected tetrahedra or hexahedra, among other possible polyhedra, are used to compute stresses on materials and predict their deformation; these collections of interconnected polyhedra are readily exported into a boundary representation for a rigid link. Since polygon-based geometries are supported by many 3D modeling and engineering tools and are sufficiently flexible to approximate any shape, effective incorporation of this representation into rigid body simulations is critical.

The community of physics-based simulation developers focused on computer games has developed many approaches for working with polygon-based geometries. These approaches typically consider polyhedra and triangle mesh representations separately, for good reason: since triangle meshes need not describe a solid body, existing rigid body simulation techniques are generally unable to correct the inevitable interpenetration in a plausible manner (even as the integration step tends to zero). In any case, we characterize without prejudice that community's approaches as "better always fast than always correct" because it is often possible for game developers to implement ad hoc solutions to work around artifacts. Indeed, recent work [22] found that multiple open source multi-body simulators exhibit significant artifacts when using convex polyhedral geometric representations in the context of simulating robotic grasping. We will indeed demonstrate that a library used by such simulators to compute interpenetration depth generally fails to work correctly. We investigated that particular library after noting the experiences of others in locating or attempting to develop robust implementations of its underlying algorithms (GJK and EPA); we failed to develop a robust implementation also.

This paper therefore focuses on correct, numerically robust techniques for rigid body dynamics with rigid contact. Speed is a secondary priority. This paper works to define "correctness" since the use of both floating point arithmetic (which can admit interpenetration for bodies in kissing configurations even when constraints are solved to numerical precision) and popular and fast rigid body dynamics time stepping techniques (which explicitly allow interpenetration) preclude a simple definition.

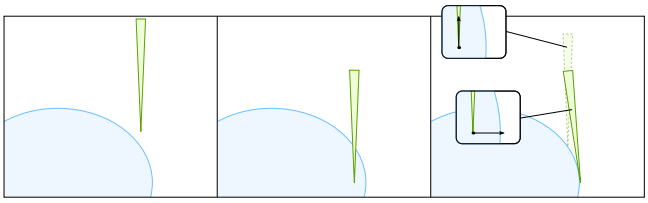


Fig. 1. Interpenetration violates an invariant of rigid bodies. Nevertheless, preventing interpenetration is technically challenging and constraint stabilization approaches like [2] use the notion of signed distance to minimize interpenetration. Every metric, including *minimum translational distance* [5], can cause bodies to be pushed apart in a different direction than they took while interpenetrating, as this illustration shows.

II. BACKGROUND

This section discusses time stepping methods for simulating multi-rigid body dynamics with contact (Section II-A); the relationship between vertices, edges, and faces in polyhedra (Section II-B); and the minimum translational distance metric (Section II-C).

A. Multi-body dynamics with contact

We now describe several topics pertinent to simulating multi-rigid body dynamics with contact, namely the rigid contact model, contact constraint stabilization, and time stepping methods.

[1] George Washington University, Washington, DC 20052, USA
[2] Toyota Research Institute, Palo Alto, CA 94306, USA

1) Rigid body dynamics with rigid contact: Rigid contact can be modeled as differential variational inequalities (DVI) [18] or as piecewise differential algebraic equations. Rigid contact is conceptually straightforward; one need not consider undeformed and deformed shapes, nor attempt to approximate the interconnected nodes in a finite element simulation with a computationally fast model. Literature on the rigid contact model uses the distance function, often denoted $\phi(.)$, extensively. Note that while ϕ *should* always be non-negative for rigid contact, time stepping methods like [19], [2] generally allow the distance between bodies to become negative—to be corrected using constraint stabilization, as will be discussed below—for computational efficiency. In such cases, the output of $\phi(.)$ is required to be positive if two rigid bodies are separated, zero if they are in a kissing configuration, and negative if they are interpenetrating. In addition, $\phi(.)$ should decrease as the Euclidean distance between two disjoint bodies decreases, as disjoint bodies come into contact or intersect, or as intersecting bodies interpenetrate to a greater degree (we leave the meaning of "degree" nebulous at this point in the discussion). *For the purposes of this paper, we focus upon such time stepping methods, which precludes tracking the direction that the bodies have interpenetrated.* Whether the ability of time stepping methods to ignore exact event time finding compensates for the additional computation that is required to compute the direction to separate interpenetrating bodies will be left for future work.

2) Contact constraint stabilization: One challenge with time stepping approaches is how to separate interpenetrating bodies; although the rigid body model precludes interpenetration, existing approaches for modeling rigid body dynamics are generally unable to maintain this invariant (a separate issue is whether it is better to avoid attempting to maintain this invariant, as alluded to in Section II-A.1). Standard time stepping based simulation approaches must use *constraint stabilization* methods to correct interpenetration. Convergence proofs for DVI-based time stepping approaches with constraint stabilization typically assume the existence of a continuously differentiable signed distance function (see, e.g., [2]).

3) Time stepping equations: Typical time stepping implements formulate the problem of simulating rigid body dynamics with rigid contact and Coulomb friction as a solvable *mixed linear complementarity problem* (MLCP) [6]:

$$\begin{bmatrix} \mathbf{M} & -\mathbf{N}^{\mathsf{T}} & -\mathbf{D}^{\mathsf{T}} & \mathbf{0} \\ \mathbf{N} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{D} & \mathbf{0} & \mathbf{0} & \mathbf{E} \\ \mathbf{0} & \mu & -\mathbf{E}^{\mathsf{T}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{v}(t+\Delta t) \\ \boldsymbol{f}_N \\ \boldsymbol{f}_D \\ \boldsymbol{\lambda} \end{bmatrix} + \begin{bmatrix} -\boldsymbol{k} \\ \mathbf{1}\frac{\phi}{\Delta t} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{w}_N \\ \boldsymbol{w}_D \\ \boldsymbol{w}_\lambda \end{bmatrix} \tag{1}$$

$$\boldsymbol{f}_N \geq \mathbf{0}, \boldsymbol{w}_N \geq \mathbf{0}, {\boldsymbol{f}_N}^{\mathsf{T}} \boldsymbol{w}_N = 0 \tag{2}$$

$$\boldsymbol{f}_D \geq \mathbf{0}, \boldsymbol{w}_D \geq \mathbf{0}, {\boldsymbol{f}_D}^{\mathsf{T}} \boldsymbol{w}_D = 0 \tag{3}$$

$$\boldsymbol{f}_\lambda \geq \boldsymbol{\lambda}, \boldsymbol{w}_\lambda \geq \mathbf{0}, {\boldsymbol{f}_\lambda}^{\mathsf{T}} \boldsymbol{w}_\lambda = 0 \tag{4}$$

where Δt is the time increment to step forward, $\boldsymbol{v}$ is the vector of generalized velocities, $\boldsymbol{k} \equiv \mathbf{M}\boldsymbol{v}(t) + \Delta t \boldsymbol{f}_{\text{ext}}(t)$, $\boldsymbol{f}_N \geq \mathbf{0}$ is the vector of compressive contact forces (applied along the contact normals), $\boldsymbol{f}_D$ is the vector of frictional contact forces, $\boldsymbol{\lambda}$ acts similarly to slack variables in linear programming, $\mathbf{M} \in \mathbb{R}^{\ell \times \ell}$ is the generalized inertia matrix (ℓ is the dimension of generalized coordinates), $\mathbf{N} \in \mathbb{R}^{r \times \ell}$ is the Jacobian matrix that transforms generalized velocities to relative speeds along the normals of the r contact points, $\mathbf{D} \in \mathbb{R}^{rq \times \ell}$ is the Jacobian matrix that transforms generalized velocities to relative speeds along q spanning vectors of each of the r contact points, $\mu \in \mathbb{R}^{r \times r}$ is a diagonal matrix of Coulomb friction coefficients, and $\mathbf{E} \in \mathbb{R}^{rq \times \ell}$ is a binary matrix. Finally, the $\boldsymbol{w}_N$, $\boldsymbol{w}_D$, and $\boldsymbol{w}_\lambda$ variables are required to formulate the above MLCP, but these variables do not affect our focus and can be ignored in this paper. The formulation above serves only to integrate velocity variables to $t + \Delta t$ (all other values are evaluated at time t). Integrating position (coordinate) variables requires a more sophisticated treatment due to the singularities inherent in minimal representations of 3D orientation; [13] is one work that succinctly describes how to address this problem.

Finally, we refer reader to other sources for better understanding of time stepping equations and approaches, including [21], [19], [20], [12], [11]. We present the time stepping equations above to allow the reader to make the connection between the r points of contact (used to form $\mathbf{N}$ and $\mathbf{D}$) that our present work focuses upon determining and the value of the signed distance function $\phi(.)$ at time t.

B. Polyhedral features

The Descartes-Euler polyhedral formula provides a relationship between the number of vertices (V), edges (E), and faces (F) of a polyhedron.

$$V + F - E = 2 \tag{5}$$

This paper leverages this relationship to reason about asymptotic time complexity of polyhedral *features*, rather than considering numbers of vertices, edges, and faces separately. The formula applies to all convex polyhedra and many non-convex polyhedra as well. We assume each polyhedron input to any algorithm referenced in this paper contains n features.

C. Minimum translational distance

There exist a number of metrics for degree of rigid body interpenetration. One of the most popular, judging by citations in literature, is *minimum translational distance* (MTLD). MTLD is the minimum Euclidean distance a pair of polyhedra must be translated in Cartesian space so that the polyhedra are kissing (i.e., the polyhedra are intersecting, but no feature of either polyhedron lies strictly inside the other). The fastest algorithms for computing MTLD on convex polyhedra currently run in quadratic time in the best and worst cases [7]. Practical implementations based on the Separating Axis Theorem and the Minkowski "difference" exhibit slightly higher asymptotic complexity ($O(n^2 \lg n)$ and $O(n^2 \lg n^2)$, respectively).

The Expanding Polytope Algorithm (EPA), a third algorithm that operates on the Minkowski-difference, uses the

output of the GJK algorithm [8] to compute the closest point on the Minkowski difference polytope to the origin. Such closest points can be used to determine both the MTLD and the direction of translation that yields MTLD as byproducts [23]. It is not clear how, or even if, the algorithm improves on the time complexity of explicitly constructing the Minkowski difference polytope ($O(n^2 \lg n^2)$, as stated above). We are unaware of a peer reviewed source that establishes time complexity, proof of termination, or correctness for EPA. Web searches readily uncover the challenge of finding or developing robust GJK and EPA implementations.

III. QUESTIONS

A. How should contact points be selected?

Existing algorithms for computing contact forces presume a number of discrete point samples are selected from the contact manifold (in the case when the contact manifold does not correspond to a single point). As we will discuss, *the use of point samples does not necessarily imply loss of solution accuracy*. We assume for now that the surface normal is known.

The combination of the rigid contact model and the Coulomb friction model has long been known to be subject to indeterminism [14] for rigid bodies contacting simultaneously at many points. It is currently unknown whether these indeterminate configurations—for which multiple, equally valid possible contact wrenches exist—can result in non-unique accelerations. If the accelerations are unique, then providing an increasingly dense sampling of the contact manifold would not generally yield an increase in solution accuracy. While unique accelerations implies additional sampling would be inefficient, non-unique solutions would point to larger problems: solutions can not be efficiently enumerated ($2^{O(n)}$ possible solutions to a complementarity problem with n variables might exist [6], [3]), so one might be resigned to selecting an arbitrary solution.

With that caveat in mind, selecting an insufficient number of samples can be problematic for simulation. Using a single point to represent the surface between two cubes in face/face contact could possibly lead to interpenetration. Therefore, we suggest requiring that **the point samples be chosen such that, if the projections of the bodies' relative velocity along the contact normal ($\hat{n}$) are non-negative at all point contact samples, the time derivative of the signed distance between the bodies will be non-negative**. Here, as elsewhere in the paper, we use the convention that a positive value of the projection of the relative velocity onto the contact normal indicates impending separation (and therefore that a negative value indicates impending interpenetration). The sign aspects will follow from the convention, to be discussed in Section III-B, that the contact normal points from Body B to Body A (these body labels will be used further below). For the ensuing discussion, we also assume that the rigid bodies are in "kissing" contact, meaning that the signed distance between the two bodies is zero. Representing the statement in boldface above mathematically yields:

$$\dot{\phi} \geq 0 \text{ if } \hat{\boldsymbol{n}}^{\mathsf{T}} \boldsymbol{v}_{\boldsymbol{p}_i} \geq 0, \forall i \tag{6}$$

where

$$\boldsymbol{v}_{\boldsymbol{p}_i} \equiv \dot{\boldsymbol{x}}_A - \dot{\boldsymbol{x}}_B + \boldsymbol{\omega}_A \times (\boldsymbol{p}_i - \boldsymbol{x}_A) - \boldsymbol{\omega}_B \times (\boldsymbol{p}_i - \boldsymbol{x}_B) \tag{7}$$

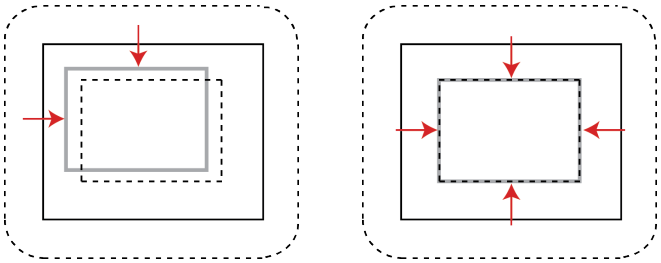
and $\boldsymbol{p}_i$ is the i^{th} point of contact, $\dot{\boldsymbol{x}}$ is the linear velocity of a body's center-of-mass, and $\boldsymbol{\omega}$ is a body's angular velocity. For convex polyhedra, the boldfaced requirement is straightforward to meet. The intersection between the two polyhedra will be convex [16], so—by convexity—if the relative velocities at all vertices of the intersection, projected along the surface normal, are all non-negative, the relative velocity at *any* point of the intersection projected along the surface normal must also be non-negative. In other words, the boldface requirement above can be met for convex polyhedra by selecting contact samples from the vertices of the intersection.

For rigid contact, point samples should only be generated for rigid bodies that are in kissing contact (i.e., $\phi = 0$). Floating point arithmetic makes identifying such a condition exactly generally infeasible. Instead, we consider the case of approximately contacting bodies, i.e., $|\phi| \leq \epsilon$ (where $\epsilon \ll 1$), meaning that bodies are either disjoint or interpenetrating. In both cases, we must translate, rotate, or translate and rotate one or both bodies until they are in kissing contact. Contact points can then be sampled as already described above. Exactly how the bodies should be translated or rotated is not clear, particularly as the target kissing configuration would be unknown (implied by Figure 1). There is no obvious metric that should be optimized either: for example, moving the bodies along the path that minimizes squared work is not generally correct, which can be seen by examining the case where the velocities of the rigid bodies are initially zero (which would allow *any* ending configuration to be chosen, as long the velocities were to remain zero). Since no path is necessarily "right", we use the surface normal (with its determination described in the following section) to translate the bodies apart/together. We hereafter denote $-\delta$ (and δ) to be the non-negative distance that the bodies should be translated apart (and together, respectively) to yield a kissing configuration.

B. How should the surface normal be selected?

This section will refer collectively to the vector between two closest features on a pair of disjoint bodies, the vector normal to the intersecting surface between kissing bodies, and the vector pointing along the direction that two intersecting bodies (with non-zero intersecting volume) should be pushed apart as the *surface normal*. We assume that this vector points toward Body A.

Following the discussion in Section III-A, there is not a "correct" surface normal for bodies that are not kissing but are sufficiently close (or are intersecting) to be considered to be contacting. Even when polyhedral bodies are kissing and thus cases of disjoint and intersecting configurations



(a) configuration that drives the bodies to the local minimum

(b) configuration at the local minimum

Fig. 2. An example of how improper surface normal selection can fail to give the desired result. Each diagram shows two rigid bodies, both boxes. Using the procedure described in [10], the Minkowski sum of a circle and each line segment) from the larger box are computed and depicted in dotted stroke. [10] indicates that the surface normals are selected as shown with the red arrows. Selecting the surface normals in this way causes the boxes in (a) to move to an incorrect, interlocking configuration (b) from which separation would not be possible: the arrows cause all relative movement to be cancelled. Without lack of generality, we use 2D for illustrative purposes and omit the Minkowski sum for the smaller box.

can be ignored, the contact normal will not be uniquely defined in cases of vertex-vertex, vertex-edge, or parallel edge-edge contact. This particular issue is described further and addressed in [24], though it appears that obtaining robust, fast solutions to such scenarios remains an open problem. This paper and our software implementations avoid this issue for bodies in kissing contact by returning no result in that case; in other words, we allow (generally small) interpenetration to occur in these expectedly rare cases, and rely upon constraint stabilization techniques to minimize that interpenetration.

Challenges aside, the selection of a "good" surface normal is important to keep simulations from exhibiting unrealistic behavior, like bodies passing through one another or becoming locked together. We can specify two necessary conditions for selecting the surface normal for contacting rigid bodies.

Necessary condition: The surface normal must point in a direction $\hat{\boldsymbol{n}}$, such that in the limit of translation $s \to 0$ along $s\hat{\boldsymbol{n}}$ and $-s\hat{\boldsymbol{n}}$, respectively, the time derivative of signed distance between the two bodies is positive.

Informally, a small translation along $\hat{\boldsymbol{n}}$ and $-\hat{\boldsymbol{n}}$ must increase the signed distance between the two bodies; alternatively, it should be evident that for any sufficiently large translation, the signed distance between two bodies will be positive. Procedures that do not respect this condition are prone to trapping the bodies in local minima; for example [10] can generate the problem depicted in Figure 2 because it does not use a global measure of interpenetration.

For bodies with convex geometries, the necessary condition points to a simple possible choice of surface normal selection: the vector from the center-of-mass of Body B to the center-of-mass of Body A. This selection appears physically plausible (a qualitative goal advocated by Barzel et al. [4]) but may result in the appearance of artifacts when a contacting body is particularly long, as Figure 3 depicts.

For disjoint bodies, a candidate for the surface normal that is nearly as fast to compute is the vector pointing along the closest points between the two closest features. For sake of precision, we define a pair of closest features as the most general pair of { vertices, edges, faces }; for example, if two polyhedra are equally close at multiple points along two edges, and one pair of elements from this set includes two vertices, the pair of closest features should be *edge-edge*. As will be discussed in Section IV-A, computing closest features for disjoint convex polyhedra can be performed in "near" constant time [14]. For intersecting bodies, an analogous choice for the surface normal is the direction of minimum translational distance. The asymptotic time complexity for computing MTLD is much greater: the current best algorithm runs in time $O(m^{\frac{3}{4}+\epsilon}n^{\frac{3}{4}+\epsilon} + m^{1+\epsilon} + n^{1+\epsilon})$ for some $\epsilon > 0$ [1]. Consistent with the discussion in Section III-A, we again note that there is no "right" choice of surface normal selection for disjoint or interpenetrating bodies with rigid contact.

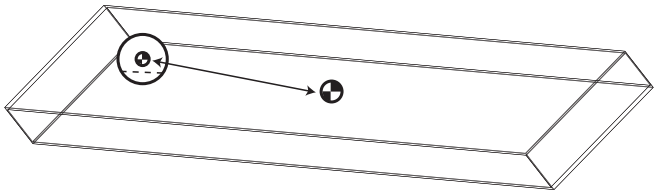


Fig. 3. A small "ball" (spherical truncated icosahedron) interpenetrating a long box. The points of intersection with the sphere on the box are drawn with dotted stroke. Centers-of-mass of both objects are depicted. If the surface normal is chosen as described in Section III-B—pushing the centers-of-mass apart, as depicted—the sphere will counterintuitively not be pushed in a direction normal to the face of the box that the sphere is intersecting.

Necessary condition: The surface normal should be piecewise continuous with respect to the bodies' configurations.

While this continuity requirement is weaker than that desired for proving time stepping convergence in [9], better than piecewise continuity is not achievable since the geometries are polyhedral: surface normals may change discontinuously as the contact surface between bodies transitions from one face to another. Accurate solutions *to the differential algebraic equations for rigid body dynamics with rigid contact* can be obtained only by switching constraints as the surface normal changes; such accurate solutions will not be possible if the surface normal is not piecewise continuous.

IV. TECHNICAL DETAILS

A. Querying whether polyhedra are intersecting

While it is possible to use one algorithm to compute both Euclidean distance and minimum translational distance, it is computationally more efficient to use separate algorithms for the two cases. For convex polyhedra, we use V-CLIP [15], which takes advantage of convexity. V-CLIP runs in worst-case linear time of the total features of the queried polyhedra and can leverage temporal coherence between simulation steps to attain near constant time performance [15].

B. Computing closest features for disjoint polyhedra

V-CLIP computes a pair of closest features for disjoint polyhedra (V-CLIP's reported "closest features" are mean-

ingless for intersecting polyhedra). V-CLIP can return vertex/vertex, vertex/edge, vertex/face, and edge/edge types; edge/face and face/face cases are not optional return types (the features returned correspond to the Voronoi region iterates [15]: V-CLIP does not return the most general pair of features). For disjoint polyhedra, two pieces of information can be determined by examining the closest features: $\boldsymbol{d}$, a vector from the feature on the second polyhedron to the feature on the first polyhedron, and $\delta = ||\boldsymbol{d}||$, i.e., the Euclidean distance.

Algorithm 1 PROJECT$(P, \hat{\boldsymbol{d}})$ Determines the two vertices from polyhedron P that yield the minimum and maximal distance along direction $\hat{\boldsymbol{d}}$. Returns the minimum and maximum projections (dot_{min} and dot_{max}) as well as the vertices that yield these extrema ($\boldsymbol{p}^{\text{min}}$ and $\boldsymbol{p}^{\text{max}}$).

$\boldsymbol{p}^{\text{min}} \leftarrow$ NIL
$\boldsymbol{p}^{\text{max}} \leftarrow$ NIL
$\text{dot}_{\text{min}} \leftarrow \infty$
$\text{dot}_{\text{max}} \leftarrow -\infty$
$\mathcal{V} \leftarrow$ all vertices from P
for $\boldsymbol{v}_i \in \mathcal{V}$ **do**
  $y \leftarrow \boldsymbol{v}_i^{\mathsf{T}} \hat{\boldsymbol{d}}$
  if $y < \text{dot}_{\text{min}}$ **then**
    $\boldsymbol{p}^{\text{min}} \leftarrow \boldsymbol{v}_i$
    $\text{dot}_{\text{min}} \leftarrow y$
  if $y > \text{dot}_{\text{max}}$ **then**
    $\boldsymbol{p}^{\text{max}} \leftarrow \boldsymbol{v}_i$
    $\text{dot}_{\text{max}} \leftarrow y$
return $\{\text{dot}_{\text{min}}, \text{dot}_{\text{max}}, \boldsymbol{p}^{\text{min}}, \boldsymbol{p}^{\text{max}}\}$

C. Computing MTLD (for intersecting bodies)

For computing the minimum translational distance between bodies with convex geometries, we use the separating axis theorem (SAT), which requires computing the projection of the bodies onto a quadratic number of axes (a normal from each face of both polyhedra, and the cross products of each edge from one polyhedron and each edge from the other). $O(n^2 \lg n)$ operations are required, if using a Dobkin-Kirkpatrick hierarchy data structure to speed extremal distance queries ($O(n^2)$ queries, each at a cost of $O(\lg n)$). The axis that yields the minimum absolute overlap of the projections of the two polyhedra's vertices is the direction that provides the MTLD, and the absolute overlap is the MTLD, $-\delta$. The SAT-based process is described in Algorithms 1 and 2. Multiple candidate normals that do not point in the same direction indicate that an indeterminate normal has been located (refer back to §III-B).

D. Computing the contact plane (for kissing bodies)

For a pair of kissing rigid bodies, we define the *contact plane* as the plane defined by the surface normal and a point inside or on both polyhedra. For disjoint polyhedra, the plane will be defined arbitrarily by a point halfway through points on closest features (it would be equally "correct" to define the plane by a point on the closest feature of one of the polyhedra). For intersecting polyhedra, the plane will be defined by vertices that yield maximum/minimum projections ($\boldsymbol{p}_A$ and $\boldsymbol{p}_B$ from Algorithm 2).

Algorithm 2 FIND-MTLD Uses the Separating Axis Theorem to find the minimum translational distance between two intersecting convex polyhedra, A and B. Returns the *negated* minimum translational distance σ (i.e., a non-positive number), the vector that yields this minimum distance ($\hat{\boldsymbol{n}}$), and scalar d from the equation of the contact plane $\boldsymbol{x}^{\mathsf{T}}\hat{\boldsymbol{n}} = d$. This algorithm does not attempt to identify indeterminate surface normals.

$\hat{N}_A \leftarrow$ set of face normals from A
$\hat{N}_B \leftarrow$ set of face normals from B
$\hat{N}_E \leftarrow$ cross-products of all edges from A with all edges from B ▷ *Products must be normalized (or removed if* $\mathbf{0}$*)*
$\mathcal{V} \leftarrow \hat{N}_A \cup \hat{N}_B \cup \hat{N}_E$
$\text{min}_{\text{overlap}} \leftarrow \infty$
$\text{min}_{\text{axis}} = \begin{bmatrix}0 & 0 & 0\end{bmatrix}^{\mathsf{T}}$
$\boldsymbol{p}_A \leftarrow$ NIL
$\boldsymbol{p}_B \leftarrow$ NIL
for $\boldsymbol{v} \in \mathcal{V}$ **do**
  $\{\text{min}_A, \text{max}_A, \boldsymbol{p}_A^{\text{min}}, \boldsymbol{p}_A^{\text{max}}\} \leftarrow$ PROJECT$(A, \boldsymbol{v})$
  $\{\text{min}_B, \text{max}_B, \boldsymbol{p}_B^{\text{min}}, \boldsymbol{p}_B^{\text{max}}\} \leftarrow$ PROJECT$(B, \boldsymbol{v})$
  $o_1 \leftarrow \text{max}_A - \text{min}_B$
  $o_2 \leftarrow \text{max}_B - \text{min}_A$
  if $o_1 < 0$ **or** $o_2 < 0$ **then**
    return { ePolyhedraDisjoint, NIL, NIL, NIL }
  if $\min(o_1, o_2) > \text{min}_{\text{overlap}}$ **then**
    continue
  if $o_1 < o_2$ **then**
    $\text{min}_{\text{overlap}} \leftarrow o_1$
    $\text{min}_{\text{axis}} \leftarrow -\boldsymbol{v}$
    $\boldsymbol{p}_A \leftarrow \boldsymbol{p}_A^{\text{max}}$
    $\boldsymbol{p}_B \leftarrow \boldsymbol{p}_B^{\text{min}}$
  else
    $\text{min}_{\text{overlap}} \leftarrow o_2$
    $\text{min}_{\text{axis}} \leftarrow \boldsymbol{v}$
    $\boldsymbol{p}_A \leftarrow \boldsymbol{p}_A^{\text{min}}$
    $\boldsymbol{p}_B \leftarrow \boldsymbol{p}_B^{\text{max}}$
$\hat{\boldsymbol{n}} \leftarrow \text{min}_{\text{axis}}$
$\sigma \leftarrow -\text{min}_{\text{overlap}}$
$d \leftarrow \frac{1}{2}\hat{\boldsymbol{n}}(\boldsymbol{p}_A + \boldsymbol{p}_B)$
return $\{\text{eSuccess}, \hat{\boldsymbol{n}}, \sigma, d\}$

E. Computing the contact manifold

After the two bodies have been translated along $\boldsymbol{d}$ by distance δ, the bodies will now be in a kissing configuration and contact features can be computed. The set of vertices from each polyhedron touching the contact plane can now be computed using an $O(\lg n)$ operation, again using extremal distance queries with a Dobkin-Kirkpatrick hierarchy. Each set of vertices (from each polyhedron) lying on the contact

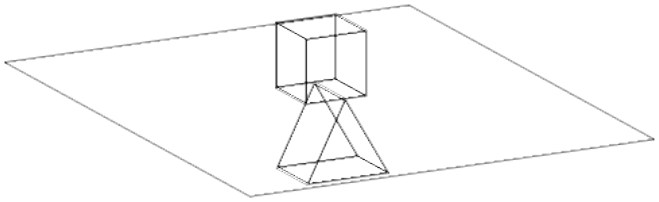


Fig. 4. Depiction of the contact plane for two rigid bodies in a kissing contact configuration. The contact plane is coplanar with the bottom face of the box; the top edge of the prism also lies on the plane.

plane can represent a single point (a 0-simplex), an edge (a 1-simplex), or a convex polygon (a simplicial 2-complex, which we henceforth refer to for brevity as a simplex). The intersection between the two simplices represents the contact manifold. We now describe how we address two numerical challenges.

1) Numerical issue: none of a polyhedron's vertices lie on the contact plane after the polyhedron has been translated: We know that at least one of the polyhedron's vertices should lie on the contact plane. The solution to this problem is to reset the floating point zero tolerance to the distance of the vertex closest to the contact plane and redetermine the set of points from the polyhedron that lie within this distance.

2) Numerical issue: there is no intersection between the polyhedral features determined to be lying on the contact plane: It is conceivable that the geometric intersection of the two siimplices results in the empty set. If the bodies were disjoint, it is evident that translating them to a kissing configuration along the vector between their closest points results in a non-empty intersection, at least in theory. If bodies are intersecting, then translating them to a kissing configuration along the vector of minimum translational distance should also result in a non-empty intersection. Our numerical approach thus first attempts to intersect the two simplices. If the intersection is empty, we select a point halfway between the closest points on the two simplices.

3) Implementation details: For polygon-polygon intersections, we project all points to the contact plane and then use floating-point versions of algorithms from [17]. For all other operations, we use implementations from `http://www.geometrictools.com`.

V. EXPERIMENTS

Our experiments can be reproduced using code in `Moby`, available from `https://github.com/PositronicsLab/Moby`, commit 5837e22. Our implementation of SAT does not use the Dobkin-Kirkpatrick hierarchy: asymptotic time complexity will thus be $O(n^3)$ (rather than the $O(n^2 \lg n)$ reported above). We have successfully tested the algorithms in this paper on numerous virtual interactions between convex polyhedra, including cases anecdotally considered to be challenging (like stacking identical boxes) in addition to experiments described below.

A. Unit tests

We developed unit tests for our software using two randomly placed convex polyhedra. The signed distance used in the unit test was computed using the Minkowski difference: we computed the minimum distance from the origin to one of the faces of the explicitly constructed Minkowski polyhedron. If the origin was found to be inside the Minkowski polyhedron, we negated this distance. We compared this distance to the separating axis-test based approach and to `libccd`, a stable (in a software development sense) library that implements the GJK and EPA algorithms. For two $2 \times 2 \times 2$ boxes, we noted the results in Table I.

B. Verification-based tests

We also propose two benchmark problems for multi-rigid body verification that stress contact between convex polyhedra. The first benchmark is a sphere spinning on the face of an immobile box. The second benchmark is a sphere rolling on top of another, immobile sphere. 2D versions of these benchmarks are depicted in Figure 5. The benchmarks possess the following advantages: (1) the solution can be computed readily and reliably for the limiting case of zero area tessellation (i.e., true spheres), for any size spheres; (2) the geometry dimensions can be scaled arbitrarily, stressing numerical tolerances; and (3) inaccurate solutions cause energy loss, an easily perceived phenomenon.

We tested the spinning sphere example using 50-vertex and 100-vertex spheres. The 50-vertex sphere abruptly stops spinning after approximately 8 ms of virtual time. The 100-vertex sphere maintains kinetic energy perfectly through at least five seconds of virtual time. We note that setting the time step size correctly ($\Delta t = 0.001$ or so) is key to getting this example to work properly. With $\Delta t = 0.01$, even a 2,000-vertex sphere does not conserve energy.

We tested the two-spheres scenario with unit radius tessellated spheres and were unable to attain a solution that conserves energy. We attempted spheres with 50, 100, 200, and 1,000 vertices; using somewhat more vertices (2,000) caused the simulation to run intractably slowly. We present this particular benchmark to the multi-rigid body simulation community for further study.

C. Profiling

We profiled CPU operations to identify timing "hot spots" on a box spinning on another box, the spinning sphere-box, and the two tessellated spheres. For the spinning box, the simulation spent 16.2% of time on contact determination and 47.7% of time on V-CLIP (the majority of remaining time went to solving the MLCP). For the spinning sphere/box scenario (the sphere contained 200 vertices), the simulation spent 75.2% of time on contact determination (primarily SAT) and 26.8% of time on V-CLIP. For the two spheres scenario (200 vertex spheres), the simulation spent 94.8% of time on contact determination (primarily SAT) and 1.7% of time on V-CLIP. Table II details raw simulation times.

VI. CONCLUSION

We have suggested necessary conditions for contact normals and contact point sample selection, such that numerical correctness to the prescribed models can be attained. While

TABLE I

ERROR IN SIGNED DISTANCE FROM THAT REPORTED BY THE MINKOWSKI-DIFFERENCE APPROACH ON TWO $2 \times 2 \times 2$ BOXES (THE UNIT TEST)

Algorithm	Minimum error	Maximum error	Mean error	Error standard deviation
GJK+EPA (`libccd`)	2.693×10^{-6}	0.608937	0.0518467	0.1311
SAT (`Moby`)	0.000000	2.66454×10^{-15}	3.353×10^{-16}	4.413×10^{-16}

TABLE II

MEAN TIME TO COMPUTE A SIMULATION ITERATION (2.7 GHZ MACBOOK PRO)

Scenario	Pairwise vertices	Time per iteration
Spinning box	8×8	1.3 ms
Spinning sphere	8×200	18 ms
Rolling spheres	200×200	370 ms

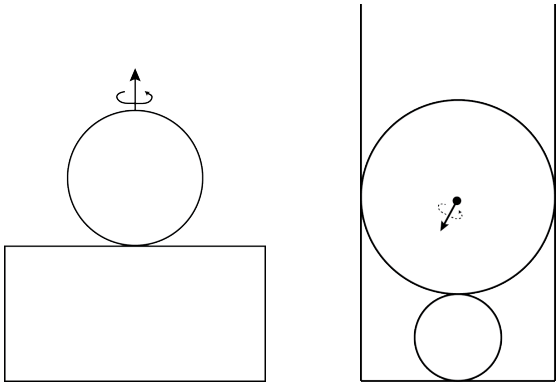


Fig. 5. 2D depictions of the 3D spinning sphere (left) and rolling sphere (right) scenarios. The height of the box and radius of the bottom sphere are variable to facilitate testing with extreme differences in scale. In the spinning sphere example, the box is immobile. In the rolling sphere scenario, both the sides of the open-top box and the bottom sphere are immobile. A no-slip frictional model is applied between the sphere and the box (left) and between the two spheres (right), and frictionless contact is modeled between the top sphere and the sides of the box (right). Each simulation begins with the mobile sphere rotating with some angular velocity. In the limiting case of idealized spheres, energy should be conserved.

we have shown that our approach using the SAT produces answers identical to those produced by the Minkowski difference, we have not proven that the direction yielding MTLD is piecewise continuous; it should be evident that translating polyhedra instantaneously along this direction yields a positive change in signed distance between the polyhedra. Our approach is numerically robust but not fast. We have provided a unit test and two verification benchmarks so that future research can test the speed of accurate approaches.

ACKNOWLEDGMENT

We thank Sean Curtis and Michael Sherman at TRI for providing significant constructive feedback. This work was funded by a GWU Undergraduate Research Fellowship for Bjoern Cheng Yi and ARO grant W911NF-16-1-0118.

REFERENCES

[1] P. K. Agarwal, L. J. Guibas, S. Har-Peled, A. Rabinovitch, and M. Sharir. Penetration depth of two convex polytopes in 3D. *Nord. J. Comput.*, 7(3):227–240, 2000.

[2] M. Anitescu and G. D. Hart. A constraint-stabilized time-stepping approach for rigid multibody dynamics with joints, contacts, and friction. *Intl. Journal for Numerical Methods in Engineering*, 60(14):2335–2371, 2004.

[3] D. Baraff. Fast contact force computation for nonpenetrating rigid bodies. In *Proc. of SIGGRAPH*, Orlando, FL, July 1994.

[4] R. Barzel, J. F. Hughes, and D. N. Wood. Plausible motion simulation for computer graphics animation. In R. Boulic and G. Hégron, editors, *Computer Animation and Simulation (Proc. Eurographics Workshop)*, pages 183–197, 1996.

[5] S. A. Cameron and R. Culley. Determining the minimum translational distance between two convex polyhedra. In *Proc. IEEE Intl. Conf. Robotics Automation (ICRA)*, pages 591–596, 1986.

[6] R. W. Cottle, J.-S. Pang, and R. Stone. *The Linear Complementarity Problem*. Academic Press, Boston, 1992.

[7] D. Dobkin, J. Hershberger, D. Kirkpatrick, and S. Suri. Computing the intersection-depth of polyhedra. *Algorithmica*, 9:518–533, 1993.

[8] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE J. of Robotics and Automation*, 4(2):193–203, April 1988.

[9] S. Hart, R. Grupen, and D. Jensen. A relational representation for generalized knowledge in robotic tasks. Technical Report 04-101, Computer Science Dept, Univ. of Massachusetts Amherst, 2004.

[10] K. Hauser. Robust contact generation for robot simulation with unstructured meshes. In *Proc. Intl. Symp. Robotics Research (ISRR)*, 2013.

[11] C. Lacoursière. Regularized, stabilized, variational methods for multibodies. In *Proc. of Scandinavian Conf. on Simulation and Modeling (SIMS)*, Oct, 2007.

[12] J. E. Lloyd. Fast implementation of Lemke's algorithm for rigid body contact simulation. In *Proc. of the IEEE Conf. on Robotics and Automation (ICRA)*, pages 4538–4543, 2005.

[13] A. T. Miller and H. I. Christensen. Implementation of multi-rigid-body dynamics within a robotic grasping simulator. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 2262–2268, Sept 2003.

[14] B. Mirtich. *Impulse-based Dynamic Simulation of Rigid Body Systems*. PhD thesis, University of California, Berkeley, 1996.

[15] B. Mirtich. V-Clip: fast and robust polyhedral collision detection. *ACM Trans. on Graphics*, 17(3):177–208, 1998.

[16] D. E. Muller and F. P. Preparata. Finding the intersection of two convex polyhedra. *Theoretical Computer Science*, 7:217–236, 1978.

[17] J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, second edition, 2001.

[18] J.-S. Pang and D. E. Stewart. Differential variational inequalities. *Math. Program., Ser. A*, 113:345–424, 2008.

[19] D. Stewart and J. C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with Coulomb friction. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, San Francisco, CA, April 2000.

[20] D. E. Stewart. Rigid-body dynamics with friction and impact. *SIAM Review*, 42(1):3–39, Mar 2000.

[21] D. E. Stewart and J. C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and Coulomb friction. *Intl. J. Numerical Methods in Engineering*, 39(15):2673–2691, 1996.

[22] J. R. Taylor, E. M. Drumwright, and J. Hsu. Analysis of grasping failures in multi-rigid body simulations. In *Proc. Intl. Conf. on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, San Francisco, CA, 2016.

[23] G. van den Bergen. Proximity queries and penetration depth computation on 3D game objects. In *Proc. Game Developer's Conference*, 2001.

[24] J. Williams, Y. Lu, and J. Trinkle. A geometrically accurate contact model for polytopes in multi-rigid-body simulation. *ASME J. Computational and Nonlinear Dynamics*, 2016.