Efficient Incremental Penetration Depth Estimation between Convex Geometries

Wei Gao

***Abstract*— Penetration depth (PD) is essential for robotics due to its extensive applications in dynamic simulation, motion planning, haptic rendering, etc. The Expanding Polytope Algorithm (EPA) is the de facto standard for this problem, which estimates PD by expanding an inner polyhedral approximation of an implicit set. In this paper, we propose a novel optimization-based algorithm that incrementally estimates minimum penetration depth and its direction. One major advantage of our method is the capability to be warm-started by leveraging the spatial and temporal coherence. This coherence emerges naturally in many robotic applications (e.g., the temporal coherence between adjacent simulation time knots). As a result, our algorithm achieves substantial speedup — we demonstrate it is 5-30x faster than EPA on several benchmarks. Moreover, our approach is built upon the same implicit geometry representation as EPA, which enables easy integration into existing software stacks. The code and supplemental document are available on: https://github.com/weigao95/mind-fcl.**

I. Introduction

Penetration depth (PD) is a distance measure that characterizes how much two overlapping shapes penetrate into each other. PD is of significant importance to various robotic applications. For instance, 1) in dynamic simulations, PD is used to calculate the contact force [24], [16], [22] in almost all rigid body contact models; 2) in motion planning, many planners [23], [20], [10] are designed to regulate (minimize) the PD to alleviate and avoid collisions; and 3) in haptic rendering [7], [14], [12], PD is used to resolve the interactions between objects. PD estimation is usually coupled with binary collision detection, which provides overlapping shape pairs as its input [2], [19]. A common strategy to accelerate collision detection and PD estimation is to split the computation into two phases [4]. The first is the "broad phase" which eliminates shape pairs that are too far away. The second "narrow phase" checks if the shape pairs passed the broad phase are really colliding, and computes the PD for the colliding pairs. This paper focuses on PD estimation in the narrow phase, as detailed below.

Problem Formulation. We investigate two closed convex shapes $A_1, A_2 \subset R^n$. Usually, the space dimension $n = 2$ or $n = 3$, corresponds to 2-dimensional (2D) or 3-dimensional (3D) setups. Non-convex shapes can be handled by computing their convex-hull [15] or performing convex decomposition [13]. Two shapes A_1, A_2 collide with each other if $A_1 \cap A_2 \neq \emptyset$. For two shapes A_1, A_2 that are in collision, the penetration depth $\text{PD}(A_1, A_2)$ is defined as

$$\begin{aligned} \text{PD}(A_1, A_2) = \ & \min_{d_{1,2} \in R^n} \quad ||d_{1,2}||_2 \\ & \text{subject to:} \quad \text{interior}(A_1 + d_{1,2}) \cap A_2 = \emptyset \end{aligned} \tag{1}$$

where $\|\cdot\|_2$ is L2-norm; $A_1 + d_{1,2}$ is the set obtained by applying a translation $d_{1,2} \in R^n$ to each point in A_1. The displacement $d^*_{1,2} \in R^n$ that is an optimal solution to Problem (1) is denoted as *minimum penetration displacement*. The unit vector $n^*_{1,2} = \text{normalized}(d^*_{1,2})$ is usually denoted as *minimum penetration direction*. Obviously, $d^*_{1,2} = n^*_{1,2} \times \text{PD}(A_1, A_2)$. An illustration is shown in Fig 1.

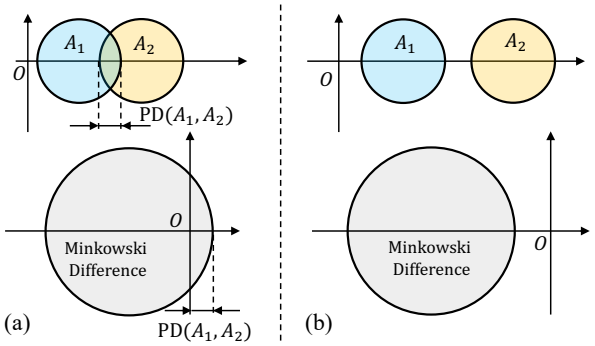

Fig. 1: An illustration of penetration depth and Minkowski Difference. (a) For two colliding shapes, their Minkowski Difference contains the origin O. The penetration depth $\text{PD}(A_1, A_2)$ is also the minimum distance from the origin O to the boundary of their Minkowski Difference. (b) For two non-overlapping shapes, the origin O is outside their Minkowski Difference and PD is undefined.

Related Works. The most widely used method for PD estimation is the EPA [25], which is coupled with the Gilbert-Johnson-Keerthi (GJK) algorithm [6] for collision detection. Both GJK and EPA operate on *Minkowski Difference* [6], which is a convex set constructed from original shapes A_1 and A_2 (a detailed explanation of Minkowski Difference is in Sec. II). Intuitively, EPA algorithm estimates the PD by building an inner polyhedral approximation of the Minkowski Difference. EPA algorithm requires a tetrahedron inscribed to the Minkowski Difference as the input, which is produced by GJK algorithm for each colliding shape pair. Built upon the Minkowski Difference formulation, GJK and EPA can handle arbitrary convex shapes such as convex polyhedra and basic primitives (*i.e.,* spheres, boxes etc.). These desirable properties make EPA the de facto standard for PD estimation. However, an accurate PD estimation using EPA may require an inner polyhedral approximation with a lot of vertices and faces. In this paper, we take a different

Mech-Mind Robotics. Email: gaowei19951004@hotmail.com

approach that incrementally estimates PD by solving an optimization problem. A key advantage of our method is to leverage spatial and temporal coherence in many robotic applications to warm-start the optimization. For instance, in dynamic simulation the object pairs tend to have very similar penetration direction and depth in consecutive time steps. Initializing our incremental PD algorithm by the information of the previous time knot can lead to substantial speed-up.

The idea of incrementally estimating the PD has also been explored in DEEP [11], which seeks a locally optimal solution of PD by walking on vertices of Minkowski Difference. Compared with [11], our method can handle arbitrary convex shapes, while [11] is restricted to convex polyhedra. Specifically, the proposed algorithm is capable of penetration computation between two primitive shapes or a primitive-polyhedra pair, while [11] does not support it. Moreover, experimental results in Sec. IV demonstrate the proposed method is about 2x faster than [11]. In the broader context of penetration computation between non-convex geometries, several contributions [9], [21], [17] proposed to iteratively compute the tangent space of Minkowski Difference and project onto it, which is conceptually related to our method. Compared with them, the key technical distinction of our method is: 1) a novel method to construct the tangent space of Minkowski Difference for convex geometries (Sec. III-B); and 2) an early termination mechanism to reduce the iterations and improve the performance (Sec. III-C). Non-convex penetration algorithms [9], [21], [17] and convex ones [25], [11] have different trade-off on problem complexity and computational performance. These two types of algorithms complement each other in practical applications. Researchers have also proposed to explicitly construct the Minkowski Difference [3] for PD estimation. However, the construction procedures tend to be computationally expensive.

Contributions. Built upon the seminal works [6], [25], [8], this paper proposes a novel incremental PD estimation algorithm between general convex shapes. In particular,

- We formulate PD estimation as a "Difference-of-Convex" problem, which is usually solved by Sequential Quadratic Programming (SQP) [1]. The major challenge of applying SQP to PD estimation is the implicit geometry representation (the Minkowski Difference and *Support Function* explained in Sec. II-B). To address it, we propose a novel instantiation of SQP that utilizes a modified Minkowski Portal Refinement (MPR) algorithm [8] as a subroutine.
- We propose a novel shortcut mechanism that reduces the computation of the vanilla SQP procedure for PD estimation. Furthermore, we demonstrate that the algorithm can still converge to locally optimal solutions despite the shortcut mechanism.
- We experimentally evaluate our method on several benchmarks. Our method demonstrates a 5-30x speedup over EPA at a comparable accuracy.
- Our method can be easily integrated into existing software stacks and we provide an open-source implementation to facilitate its usage in robotic applications.

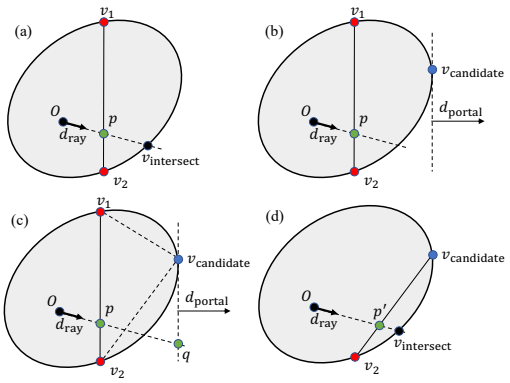

Fig. 2: An illustration of a modified MPR algorithm to compute the intersection point (the $v_{\text{intersect}}$ in (a)) of an origin ray with the boundary of the Minkowski Difference. The algorithm takes the ray direction d_{ray} as the input. In each iteration, the algorithm maintains and updates a "portal", which is a segment for 2D (or a triangle for 3D) that is inscribed to the Minkowski Difference. The illustrations from (a) to (c) show one MPR iteration, as the portal (v_1, v_2) in (a) is updated to a new portal $(v_2, v_{\text{candidate}})$ in (c).

This paper is organized as follows: in Sec. II presents the preliminaries. Sec. III explains the PD algorithm. Sec. IV shows the results. Sec. V concludes.

II. Preliminary

A. Minkowski Difference and Penetration Depth

As mentioned in Sec. I, we investigate two closed, convex shapes $A_1, A_2 \subset R^n$. Usually the dimension $n = 2$ or $n = 3$. The Minkowski Difference $\mathscr{D}_{1,2}$ for A_1 and A_2 is defined as

$$\mathscr{D}_{1,2} = A_1 - A_2 = \{v = v_1 - v_2 | v_1 \in A_1, v_2 \in A_2\} \quad (2)$$

The following properties hold for $\mathscr{D}_{1,2}$ as proved in [25]:

- $\mathscr{D}_{1,2}$ is a closed convex set.
- A_1 and A_2 overlap if the origin $O \in \mathscr{D}_{1,2}$.
- For two overlapping shapes A_1 and A_2, their penetration depth $\text{PD}(A_1, A_2)$ defined in Problem (1) corresponds to the minimum distance from O to the *boundary* of $\mathscr{D}_{1,2}$.

An illustration is provided in Fig. 1. Using these properties, we can reformulate Problem (1) in Sec. I into:

$$\begin{aligned} \text{PD}(A_1, A_2) = \ & \min_{v \in R^n} \quad ||v||_2 \\ & \text{subject to:} \quad v \in \text{boundary}(\mathscr{D}_{1,2}) \end{aligned} \quad (3)$$

Let the point v^* on the boundary of $\mathscr{D}_{1,2}$ be one optimal solution to Problem (3). The minimum penetration displacement $d^*_{1,2}$ in Sec. I can be computed as $d^*_{1,2} = -v^*$ (in other words $\text{interior}(A_1 - v^*) \cap A_2 = \emptyset$). The minimum penetration direction in Sec. I can be computed as $n^*_{1,2} = \text{normalized}(-v^*)$.

B. Support Function for Convex Shapes

GJK/EPA [6], [25] uses an implicit, functional geometry representation called *support function*. For a given convex set A, we can define its support function $\text{supp}_A(\cdot)$ as

$$\text{supp}_A(d) = \text{argmin}_{x \in A} \ \text{dot}(x, d) \ \subset A \quad (4)$$

where $\text{dot}(\cdot,\cdot)$ is dot product, and d is a unit vector denoted as *support direction*. Intuitively, the support function computes points in A that are the farthest along the support direction. For some d, there can be more than one point that satisfies Equ. (4). Following existing works, we assume $\text{supp}_A(\cdot)$ produces one point for each d, which can be arbitrarily selected from points that satisfy Equ. (4). The support function is closely related to the *supporting hyperplane* [1] of a convex set. For a given direction d and $v = \text{supp}_A(d)$, the plane $\{x|\text{dot}(d, x-v) = 0\}$ is a supporting plane for the convex shape A at the point $v \in A$.

As shown in [6], we can use $\text{supp}_{A_1}(\cdot)$ and $\text{supp}_{A_2}(\cdot)$ of two shapes A_1, A_2 to construct the support function of their Minkowski Difference $\text{supp}_{\mathscr{D}_{1,2}}(\cdot)$. In particular,

$$\text{supp}_{\mathscr{D}_{1,2}}(d) = \text{supp}_{A_1}(d) - \text{supp}_{A_2}(-d) \tag{5}$$

In other words, we can get a point from $\text{supp}_{\mathscr{D}_{1,2}}(d)$ using $\text{supp}_{A_1}(d)$ and $\text{supp}_{A_2}(-d)$. Algorithms in this work will access the Minkowski Difference through its support function.

C. Origin Ray and Minkowski Portal Refinement

We investigate two overlapping shapes A_1 and A_2, thus their Minkowski Difference $\mathscr{D}_{1,2}$ contains the origin. Given a direction d_{ray} as the input, we define the *origin ray* that starts from the origin O and extends along d_{ray} as

$$\text{origin\_ray}(d_{\text{ray}}) = \{x|x = t\ d_{\text{ray}},\ \text{for } t \geq 0\} \tag{6}$$

The MPR [8] algorithm was originally proposed for binary collision checking. In this subsection, we present a modified MPR that computes the intersection point of the origin ray with the boundary of $\mathscr{D}_{1,2}$:

$$v_{\text{intersect}}(d_{\text{ray}}) = \text{origin\_ray}(d_{\text{ray}}) \cap \text{boundary}(\mathscr{D}_{1,2}) \tag{7}$$

As illustrated in Fig. 2 (a), MPR maintains and updates a *portal*, which is a n-simplex inscribed to the Minkowski Difference $\mathscr{D}_{1,2} \subset R^n$ that intersects with the origin ray. For 2D setup, the portal is a segment inscribed to $\mathscr{D}_{1,2}$. For example, the segment (v_1, v_2) in Fig. 2 (a) is a portal that intersects the origin ray at p. In 3D setup, the portal is a triangle. The portal is initialized using a 'find_portal" procedure [8], as shown in Algorithm 1.

In each iteration, MPR computes the portal normal d_{portal}. Then, the support function $\text{supp}_A(\cdot)$ is invoked with d_{portal} to get a point $v_{\text{candidate}}$. As the origin ray intersects with portal (v_1, v_2), it must intersect with one of the segment $(v_1, v_{\text{candidate}})$ and $(v_2, v_{\text{candidate}})$. The intersecting one would be the new portal for the next iteration. This is the update_portal in Algorithm 1. The iterations continue until p is a good approximation of $v_{\text{intersect}}$, up to some tolerance Δ in the portal normal direction, as shown in Fig. 2.

Algorithm 1 MPR for Origin Ray Intersection in Sec. II-C

Require: $\mathscr{D}_{1,2}$ with support function $\text{supp}_{\mathscr{D}_{1,2}}(\cdot)$
Require: ray direction d_{ray}
Require: tolerance Δ
  $\text{portal}_0 \leftarrow \text{find\_portal}(\mathscr{D}_{1,2})$
  while $k = 0, 1, 2, \ldots$ **do**
    $d_{\text{portal\_}k} \leftarrow \text{portal\_normal}(\text{portal}_k)$
    $v_{\text{candidate}} \leftarrow \text{supp}_{\mathscr{D}_{1,2}}(\text{portal}_k)$
    $p_k \leftarrow \text{intersect}(\text{origin\_ray}(d_{\text{ray}}), d_{\text{portal\_}k})$
    ▷ A plane is defined by its normal and one point on it
    $q_k \leftarrow \text{intersect}(\text{origin\_ray}(d_{\text{ray}}), \text{Plane}(d_{\text{portal\_}k}, v_{\text{candidate}}))$
    if $|\text{dot}(q_k - p_k, \text{normalized}(d_{\text{portal\_}k}))| \leq \Delta$ **then**
      return (p_k, $v_{\text{candidate}}$, $d_{\text{portal\_}k}$)
    end if
    $\text{portal}_{k+1} \leftarrow \text{update\_portal}(\text{portal}_k,\ v_{\text{candidate}})$
  end while

III. Penetration Depth Estimation Algorithm

As shown in Sec. II, PD estimation can be expressed as

$$\begin{aligned} &\min_{v \in R^n} \quad ||v||_2 \\ &\text{subject to:} \quad v \in \text{boundary}(\mathscr{D}) \end{aligned} \tag{8}$$

where $\mathscr{D}$ is the Minkowski Difference that contains the origin. The subscript of $\mathscr{D}_{1,2}$ is dropped for notational simplicity. As shown in Sec. II-A, we can only access the convex set $\mathscr{D}$ through its support function $\text{supp}_{\mathscr{D}}(\cdot)$. This implicit geometric representation implies:

1) For a given point $v \in \text{boundary}(\mathscr{D})$, it is hard to compute a supporting hyperplane of $\mathscr{D}$ (or equivalently the hyperplane *normal*) that passes through v. This operation is required for many optimization procedures, such as the SQP in Sec. III-A.
2) The support function $\text{supp}_{\mathscr{D}}(\cdot)$ is actually an "inverse" of the operation in 1): for a given normal direction d, we can compute a point $v = \text{supp}_{\mathscr{D}}(d)$. This point, combined with the normal d, is able to define a supporting hyperplane for $\mathscr{D}$.

In this section we will address this challenge and propose an optimization-based PD estimation algorithm.

To make a clear presentation, we first introduce two "conceptual" algorithms in Sec. III-A and III-B. The first one highlights the SQP procedure assuming we can compute supporting hyperplanes for $v \in \text{boundary}(\mathscr{D})$. As mentioned above, this assumption is not true for our problem. To address it, we propose a novel instantiation of SQP that utilizes a modified MPR algorithm as a subroutine. This is the second "conceptual" algorithm in Sec. III-B. Finally, we present the actual PD algorithm with a shortcut mechanism, which reduces the computation and leads to substantial speed-up.

A. Formulation as a Difference-of-Convex Problem

In this subsection, we investigate the following optimization problem, which is very similar to Problem (8):

$$\begin{aligned} &\min_{v \in R^n} \quad ||v||_2 \\ &\text{subject to:} \quad v \in \text{boundary}(\mathscr{E}) \end{aligned} \tag{9}$$

where $\mathscr{E}$ is also a closed convex set in R^n that contains the origin. Different from Problem (8), we assume an explicit representation of $\mathscr{E}$ is available, such that we can compute a supporting hyperplane for each point $v \in \text{boundary}(\mathscr{E})$.

Algorithm 2 SQP for Problem 10

Require: $\mathscr{E}$ that supports compute_supporting_hyperplane$(\cdot)$
Require: $v_{\text{init}} \in \text{boundary}(\mathscr{E})$
  $v_0 \leftarrow v_{\text{init}}$
  while $k = 0, 1, 2, \ldots$ **do**
    $n_{\text{plane\_}k} \leftarrow$ compute_supporting_hyperplane$(\mathscr{E},\ v_k)$
    ▷ Plane defined by normal $n_{\text{plane\_}k}$ and a point v_k on it
    $z_k \leftarrow$ project_origin_to_plane(Plane$(v_k,\ n_{\text{plane_}k})$)
    $v_{k+1} =$ boundary_intersection$(O\text{_to_}z_k,\ \text{boundary}(\mathscr{E}))$
  end while
  return v_k

Since this assumption does not hold for the Minkowski Difference $\mathscr{D}$, the PD algorithm presented in this subsection is "conceptual" and will be instantiated in Sec. III-B.

As a concrete example, the convex shape $\mathscr{E}$ might be represented as the level-set of a continuous convex function $g(v)$. In other words, $\mathscr{E} = \{v | g(v) \leq 0\}$ and the boundary is $\{v | g(v) = 0\}$. For a point $v \in \text{boundary}(\mathscr{E})$, we can compute a supporting hyperplane whose normal is $\nabla g(v)$, where ∇ is the subgradient operator.

The domain for the decision variable in Problem (9) only includes the boundary of $\mathscr{E}$. As the convex set $\mathscr{E}$ contains the origin, we can enlarge the input domain to everywhere except the inner of $\mathscr{E}$ and rewrite the optimization as

$$\begin{aligned} &\min_{v \in R^n} \quad ||v||_2 \\ &\text{subject to:} \quad v \in (R^n \setminus \text{inner}(\mathscr{E})) \end{aligned} \tag{10}$$

where $R^n \setminus \text{inner}(\mathscr{E})$ is the complement of inner$(\mathscr{E})$. Problem (10) can be solved using the SQP [1] algorithm. Moreover, Problem (10) is a Difference-of-Convex problem. Thus, additional properties can be used to simplify the SQP:

- SQP usually requires a *trust region*. This trust region can be infinity for Difference-of-Convex problems [1].
- The linearization of the constraint $v \in (R^n \setminus \text{inner}(\mathscr{E}))$ is a halfspace outside the supporting hyperplane. Thus, the inner QP step of SQP can be implemented by projecting the origin onto the supporting hyperplane.

The SQP algorithm is presented in Algorithm 2, which alternates between: 1) compute the supporting hyperplane; 2) project the origin onto the hyperplane; and 3) compute the intersection of the projection line with the boundary. Detailed derivation is provided in the Supplemental Material.

Algorithm 2 cannot be directly applied to Minkowski Difference $\mathscr{D}$, as it requires explicit geometric representation to compute a supporting hyperplane for boundary points. This challenge will be addressed in Sec. III-B.

B. Instantiation using MPR as a Subroutine

In this subsection, we adopt the SQP procedure in Algorithm 2 to the Minkowski Difference $\mathscr{D}$, which is represented by its support function $\text{supp}_{\mathscr{D}}(\cdot)$. For each point $v \in \text{boundary}(\mathscr{D})$, we can compute its direction $d_v = \text{normalized}(v)$. We propose to use d_v as the decision variable

Algorithm 3 SQP using Support Function

Require: support function $\text{supp}_{\mathscr{D}}(\cdot)$
Require: initial direction d_{init}
Require: tolerance Δ for MPR subprocedure
  $d_0 \leftarrow d_{\text{init}}$
  while $k = 0, 1, 2, \ldots$ **do**
    mpr_output $\leftarrow$ mpr$(\text{supp}_{\mathscr{D}},\ d_k,\ \Delta)$ ▷ Algorithm 1
    $(p_k,\ v_{\text{candidate}},\ d_{\text{portal\_}k}) \leftarrow$ mpr_output
    support_plane $\leftarrow$ Plane$(v_{\text{candidate}},\ d_{\text{portal_}k})$
    $z_k \leftarrow$ project_origin_to_plane(support_plane)
    if should_terminate$(p_k,\ z_k,\ d_{\text{portal_}k})$ **then**
      return intersect(origin_ray(d_k), support_plane)
    end if
    $d_{k+1} \leftarrow d_{\text{portal\_}k}$
  end while

and represent v in Problem (10) implicitly by d_v, based on the following observations:

1) The point $v \in \text{boundary}(\mathscr{D})$ can be estimated from d_v (with accuracy up on some tolerance Δ), using the MPR algorithm in Sec. II-C as a subroutine.
2) More importantly, the supporting hyperplane normal at v can be approximated by the portal normal d_{portal} upon the convergence of the MPR algorithm. It is emphasized that usually $d_{\text{portal}} \neq d_v$.

With these observations, we propose Algorithm 3 for the Minkowski Difference $\mathscr{D}$, which is transformed from Algorithm 2. As this algorithm will be superseded in Sec. III-C, we only provide an intuitive analysis in this subsection.

A critical property of Algorithm 3 is approximating a supporting hyperplane normal by $d_{\text{portal\_}k}$, which is further illustrated in Fig. 3. Upon the convergence of MPR (Algorithm 1) subroutine with tolerance Δ, the ground-truth intersecting point v for direction d_v must lie in the between of the portal (segment $(v_{\text{portal\_1}}, v_{\text{portal\_2}})$ in Fig. 3) and the supporting hyperplane at $v_{\text{candidate}}$. The portal and supporting hyperplane are parallel to each other (they share the same normal d_{portal}), and the distance between these two planes is no more than Δ as the MPR terminates in this iteration. Algorithm 1 uses p to approximate v and uses d_{portal} to approximate a supporting hyperplane normal at v. This approximation is equivalent to removing points from $\mathscr{D}$ that are separated from the origin by the portal, as illustrated in Fig. 3 (b). This is because p is the exact intersecting point for the shape in Fig. 3 (b), while d_{portal} is the exact supporting hyperplane at that intersecting point. The removed part is a convex region whose "thickness" along d_{portal} is no more than Δ. Thus, the approximation becomes more accurate as the tolerance Δ is set smaller.

Algorithm 3 might work in practice. However, there can be many iterations in the MPR subroutine, which is computationally expensive. This issue will be addressed in Sec. III-C.

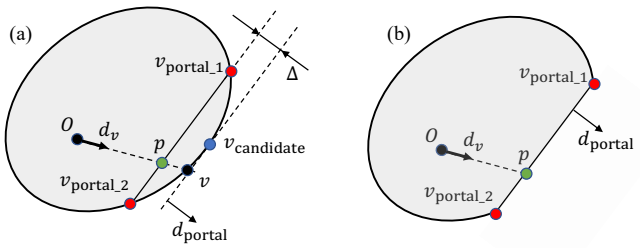

Fig. 3: Approximation scheme used in Sec. III-B and Algorithm 3. (a) shows the configuration of the portal and supporting hyperplane upon the convergence (with tolerance Δ) of MPR, when it is used as a subroutine in Algorithm 3. Intuitively, the approximation in Sec. III-B corresponds to removing all points "outside" the final portal, as shown in (b). A detailed explanation is in Sec. III-B.

C. Penetration Depth Estimation with Shortcut

In existing software stacks [19], [2], MPR is typically used with a rather small tolerance for accurate binary collision checking. However, the primary functionality for the MPR subroutine in Algorithm 3 is to produce a search direction d_{k+1} via the supporting hyperplane normal d_{portal}. Thus, we propose to reduce the MPR iterations by an early termination mechanism (shortcut) once a "good" d_{portal} is discovered.

Fig. 4 shows the configuration of the portal and supporting hyperplane in the MPR subroutine. The distance between the portal and the supporting hyperplane might be (much) larger than the tolerance Δ, as it is not necessarily the final iteration. Let the point p be the intersection of origin_ray(d_v) with the portal. MPR subroutine uses p as the current estimation of v (the intersection with the boundary of $\mathscr{D}$), and $|p| \leq |v|$. In other words, if we continue the MPR subroutine with direction d_v, the PD we would find is *lower* bounded by $|p|$.

Besides, we can project origin O to the supporting hyperplane to get z, as shown in Fig 4. Let v_z be the intersection of segment (O,z) with the boundary of $\mathscr{D}$. Obviously $|v_z| \leq |z|$. If we switch to a new search direction $d_z = \text{normalized}(z)$, the PD we would find is *upper* bounded by $|z|$.

From the analysis above, in each MPR subroutine iteration, we have: 1) a lower bound of PD $|p|$ along the current search direction d_v; and 2) a new search direction d_z candidate with an upper bound on PD $|z|$ along it. We propose to switch to the new search direction d_z once the upper bound z along d_z is smaller than the lower bound along the original direction d_v, as shown in Fig. 4.

With the analysis above, the overall PD estimation algorithm is summarized in Algorithm 4, which uses a subroutine in Algorithm 5. The SQP procedure is almost identical to Algorithm 3, except that the MPR subroutine is changed to Algorithm 5. The MPR subroutine would be terminated early with the abovementioned condition, as shown in Algorithm 5. Despite the new shortcut mechanism, this SQP is guaranteed to converge to a locally optimal solution. Please refer to the Supplemental Material for a detailed analysis.

D. Implementation Details: Initialization

The Algorithm 4 requires an initial direction d_{init}. This is a guess of the smallest displacement direction that could move shape A_1 away from colliding with shape A_2. For applications that exhibit high spatial or temporal coherence, there might be a good application-specific guess of the minimum penetration direction. For instance, in dynamic simulation [24], [16], [22] colliding object pairs tend to have very similar penetration direction and depth in consecutive simulation steps. Similarly, in optimization-based motion planning [23], [20], [10] we might use the penetration direction from the previous optimization iteration as d_{init}, especially for planning algorithms with trust-region mechanisms to ensure a small difference between consecutive iterations. If application-specific prior about penetration direction is unavailable, one alternative method is to use the centroid difference between two shapes as d_{init} as suggested by [11].

Algorithm 4 Proposed SQP Algorithm for PD Estimation

Require: support function $\text{supp}_{\mathscr{D}}(\cdot)$
Require: initial direction d_{init}
Replaces the invoked MPR subroutine in Algorithm 3 by mpr_shortcut in Algorithm 5. The tolerance Δ in Algorithm 3 is no longer necessary.

Algorithm 5 MPR Subroutine with Shortcut in Sec. III-C

Require: $\mathscr{D}_{1,2}$ with support function $\text{supp}_{\mathscr{D}_{1,2}}(\cdot)$
Require: ray direction d_{ray}
  $\text{portal}_0 \leftarrow \text{find\_portal}(\mathscr{D}_{1,2})$
  while $k = 0, 1, 2, \ldots$ **do**
    $d_{\text{portal\_}k} \leftarrow \text{portal\_normal}(\text{portal}_k)$
    $v_{\text{candidate}} \leftarrow \text{supp}_{\mathscr{D}_{1,2}}(\text{portal}_k)$
    $p_k \leftarrow \text{intersect}(\text{origin\_ray}(d_{\text{ray}}), \text{portal}_k)$
    ▷ A plane is defined by its normal and one point on it
    $z_k \leftarrow \text{project\_origin\_to\_plane}(\text{Plane}(v_{\text{candidate}},\ d_{\text{portal\_}k}))$
    if $|z_k| \leq |p_k|$ **then** ▷ Shortcut in Sec. III-C
      return p_k, $v_{\text{candidate}}$, $d_{\text{portal\_}k}$
    end if
    $\text{portal}_{k+1} \leftarrow \text{update\_portal}(\text{portal}_k,\ v_{\text{candidate}})$
  end while

IV. Results

In this section, we experimentally investigate the robustness, accuracy and efficiency of the proposed method. We first compare our method with EPA [25] and DEEP [11]. Then, an ablation study is conducted to highlight the shortcut mechanism introduced in Sec. III-C. Parameters are tuned such that all algorithms have roughly the same accuracy. The source code and supplemental document are available on this link.

A. Comparison with EPA

Our method is compared with the EPA [25] algorithm implemented in libccd [5], a popular library used in various robotics applications [19], [22]. The evaluation is performed on two sets of shapes. The first one consists of simple primitives, and the second set is convex polyhedra with different numbers of vertices.

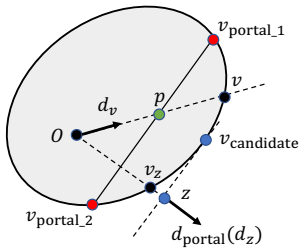

Fig. 4: An illustration of the shortcut mechanism in Sec. III-C. The penetration depth along d_v is lower bounded by $|p|$, while the penetration depth along $d_{\text{portal}}(d_z)$ is upper bounded by z. We proposed to switch to d_z as the new search direction when $|p| \geq |z|$, as explained in Sec. III-C.

Primitive Shapes. We consider the penetration depth between three types of shape pairs:

- Sphere collides with sphere (sphere vs. sphere)
- Capsule collides with capsule (capsule vs. capsule)
- Sphere collides with capsule (sphere vs. capsule)

For these simple primitive shapes, we use hand-written PD algorithms as the ground truth. The shapes have characteristic dimensions (such as sphere diameter) of 1 meter in this experiment. All of the statistical results are the averaged value of 10000 independent runs with randomly generated poses for the shape pair. Please refer to supplemental materials for detailed setup.

Our method requires an initial penetration direction guess. For the purpose of benchmarking, we use the following method to obtain the initial guess. Given the ground truth penetration direction from the hand-written algorithm, we apply a rotation of n degrees with respect to a random axis that is perpendicular to the ground-truth direction. Then, the rotated direction is used as the initial guess.

Table. I and II shows the accuracy of our method under different rotation perturbation angles n, where n ranges from 5° to 45°. Table. I presents the distance error while Table. II shows the directional error. We report the average deviation of 10000 independent runs. Both EPA and our method can accurately estimate the penetration. The proposed algorithm is rather accurate despite the large initialization error.

Table. III summarizes the performance of our method. Our method achieves 20x-30x speed up compared to EPA. Moreover, the performance improvement is consistent for different shape types and deviations of initial guess direction.

Convex polyhedra. The second experiment compares our method with EPA on convex polyhedra. We use the similar sphere-capsule setup, but replace the sphere with its convex polyhedral approximation at different resolutions. Higher resolution with more vertices and faces improves the approximation accuracy at the cost of computational complexity.

Fig. 5 shows the performance of the proposed algorithm with respect to the number of vertices in the convex polyhedra. The proposed method consistently outperforms EPA baselines with about 6-10x speedup.

	Ours (5°)	Ours (25°)	Ours (45°)	EPA
Sphere vs. Sphere	0.909	1.018	1.03	1.58
Capsule vs. Capsule	1.092	1.12	1.12	0.39
Sphere vs. Capsule	1.255	1.31	1.30	1.72

TABLE I: The penetration depth error in micrometers (10^{-6} meter) of various algorithms compared with the ground truth. The result errors are the average of 10000 independent runs. Our method is evaluated with different angular deviations (of initial penetration direction) at 5°, 25°, and 45°. Geometries have characteristic dimensions (such as the diameter of the sphere) of 1 meter.

	Ours (5°)	Ours (25°)	Ours (45°)	EPA
Sphere vs. Sphere	1.82	1.38	1.93	8.84
Capsule vs. Capsule	3.19	3.42	3.23	10.39
Sphere vs. Capsule	2.25	2.31	2.30	10.72

TABLE II: The penetration direction error in milliradian (10^{-3} radian) of various algorithms compared with the ground truth. The result errors are the average of 10000 independent runs. Our method is evaluated with different angular deviations (of initial penetration direction) at 5°, 25°, and 45°.

B. Comparison with DEEP [11]

The proposed method is compared with [11], which computes penetration distance incrementally by walking on the surface of Minkowski Difference. We implement the baseline algorithm in C++, as well as the internal convex polygon intersection algorithm [18]. We use the suggested initialization scheme: given an initial guess of penetration direction, compute the vertex hub pair by taking the extremal vertex of each shape along that direction. As [11] can only handle convex polyhedra, we use a sphere-sphere setup similar to Sec. IV-A but replace the sphere with its convex polyhedral approximation.

Fig. 6 shows the performance comparison between the proposed algorithm and the baseline. The initial direction deviation is fixed as 5° and the sphere shape is discretized at different resolutions. The proposed method outperforms the baseline [11] with about 2x speedup.

	Ours (5°)	Ours (25°)	Ours (45°)	EPA
Sphere vs. Sphere	1.9	2.1	2.3	85.1
Capsule vs. Capsule	1.6	1.9	2.0	39.5
Sphere vs. Capsule	2.0	2.2	2.4	61.5

TABLE III: Performance of the proposed method under different angular deviations of the initial direction guess. Results are the averaged time in microseconds from 10000 independent runs. The proposed method is 20x-30x times faster than EPA.

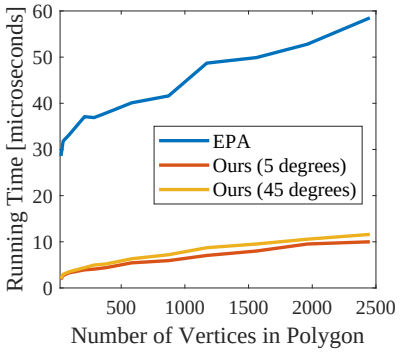

Fig. 5: Performance comparison between our method and EPA [25] under different number of vertices in convex polyhedron. Results are the averaged time in microseconds computed from 10000 independent runs. Our method is evaluated with different angular deviations (of initial penetration direction) at 5° and 45°. The proposed method achieves 6-10x speedup compared to the EPA baseline.

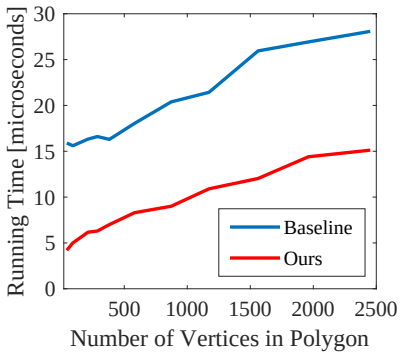

Fig. 6: Performance comparison between our method and DEEP [11] under different number of vertices in convex polyhedron. Results are the averaged time in microseconds. Both methods are evaluated with angular deviation (of initial penetration direction) at 5°. The proposed method is about 2x faster compared to the baseline.

C. Ablation Study on Shortcut Mechanism

In Sec. III-C, a shortcut mechanism is introduced to improve the performance by reducing the number of MPR iterations. To highlight its benefit, we conduct an ablation study by removing the shortcut mechanism of the proposed algorithm. We use the same convex polyhedra setup as Sec. IV-A. The initial direction deviation is fixed as 45°.

The result is shown in Fig. 7. The number of support function invocations and running time are used to characterize the performance. From the figure, the proposed shortcut mechanism can halve the required support function invocations and lead to about 2-3x speedup.

V. Conclusion

This paper presents a novel algorithm to estimate the PD between convex shapes. To achieve this, we formulate the PD estimation as a Difference-of-Convex optimization. Then, we propose a novel instantiation of SQP using a modified MPR subroutine that solves the optimization-based PD estimation. We further present a shortcut mechanism that significantly

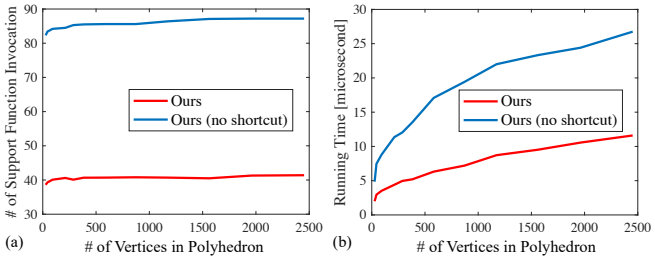

Fig. 7: An ablation study is conducted to highlight the benefit of the proposed shortcut in Sec. III-C. In the figure, (a) shows the performance measured by the number of support function invocations, while (b) shows the performance in microseconds. The proposed shortcut mechanism can halve the required support function invocations and lead to about 2-3x speedup.

reduces the computation. Through various experiments, we show that the proposed method achieves a 5-30x speedup over the well-known EPA algorithm at comparable accuracy.

VI. Acknowledgments

This work was conducted during the author's employment at Mech-Mind Robotics. The views expressed in this paper are those of the authors themselves and are not endorsed by the supporting agencies.

References

[1] S. Boyd, S. P. Boyd, and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.
[2] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. 2016.
[3] D. Dobkin, J. Hershberger, D. Kirkpatrick, and S. Suri. Computing the intersection-depth of polyhedra. *Algorithmica*, 9(6):518–533, 1993.
[4] C. Ericson. *Real-time collision detection*. Crc Press, 2004.
[5] D. Fiser. Libccd: Library for collision detection between two convex shapes, 2015.
[6] 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 Journal on Robotics and Automation*, 4:193–203, 1988.
[7] A. Gregory, A. Mascarenhas, S. Ehmann, M. Lin, and D. Manocha. Six degree-of-freedom haptic display of polygonal models. In *Proceedings Visualization 2000.*, pages 139–146. IEEE, 2000.
[8] S. Jacobs. *Game programming gems 7*. 2008.
[9] C. Je, M. Tang, Y. Lee, M. Lee, and Y. J. Kim. Polydepth: Real-time penetration depth computation using iterative contact-space projection. *ACM Transactions on Graphics (TOG)*, 31(1):1–14, 2012.
[10] D. Kappler, F. Meier, J. Issac, J. Mainprice, C. G. Cifuentes, M. Wüthrich, V. Berenz, S. Schaal, N. Ratliff, and J. Bohg. Real-time perception meets reactive motion generation. *IEEE Robotics and Automation Letters*, 3(3):1864–1871, 2018.
[11] Y. J. Kim, M. C. Lin, and D. Manocha. Incremental penetration depth estimation between convex polytopes using dual-space expansion. *IEEE transactions on visualization and computer graphics*, 10(2):152–163, 2004.
[12] V. Kumar and E. Todorov. Mujoco haptix: A virtual reality system for hand manipulation. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 657–663. IEEE, 2015.
[13] K. Mamou and F. Ghorbel. A simple and efficient approach for 3d mesh approximate convex decomposition. In *2009 16th IEEE international conference on image processing (ICIP)*, pages 3501–3504. IEEE, 2009.
[14] W. A. McNeely, K. D. Puterbaugh, and J. J. Troy. Six degree-of-freedom haptic rendering using voxel sampling. In *ACM SIGGRAPH 2005 Courses*, pages 42–es. 2005.
[15] A. A. Melkman. On-line construction of the convex hull of a simple polyline. *Information Processing Letters*, 25(1):11–12, 1987.

[16] B. Mirtich. Rigid body contact: Collision detection to force computation. In *Workshop on Contact Analysis and Simulation, IEEE Intl. Conference on Robotics and Automation*, 1998.

[17] G. Nawratil, H. Pottmann, and B. Ravani. Generalized penetration depth computation based on kinematical geometry. *Computer Aided Geometric Design*, 26(4):425–443, 2009.

[18] J. O'Rourke, C.-B. Chien, T. Olson, and D. Naddor. A new linear algorithm for intersecting convex polygons. *Computer graphics and image processing*, 19(4):384–391, 1982.

[19] J. Pan, S. Chitta, and D. Manocha. Fcl: A general purpose library for collision and proximity queries. In *IEEE International Conference on Robotics and Automation*, pages 3859–3866. IEEE, 2012.

[20] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization. Citeseer.

[21] M. Tang and Y. J. Kim. Interactive generalized penetration depth computation for rigid and articulated models using object norm. *ACM Transactions on Graphics (TOG)*, 33(1):1–15, 2014.

[22] E. Todorov. Convex and analytically-invertible dynamics with contacts and constraints: Theory and implementation in mujoco. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6054–6061. IEEE, 2014.

[23] M. Toussaint, K. Allen, K. A. Smith, and J. B. Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning.

[24] J. C. Trinkle, J.-S. Pang, S. Sudarsky, and G. Lo. On dynamic multi-rigid-body contact problems with coulomb friction. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 77(4):267–279, 1997.

[25] G. Van Den Bergen. Proximity queries and penetration depth computation on 3d game objects. In *Game developers conference*, volume 170, 2001.