Penetration Depth Between Two Convex Polyhedra: An Efficient Stochastic Global Optimization Approach

Mark A. Abramson* Griffin D. Kent† Gavin W. Smith‡

***Abstract*—During the detailed design phase of an aerospace program, one of the most important consistency checks is to ensure that no two distinct objects occupy the same physical space. Since exact geometrical modeling is usually intractable, geometry models are discretized, which often introduces small interferences not present in the fully detailed model. In this paper, we focus on computing the depth of the interference, so that these false positive interferences can be removed, and attention can be properly focused on the actual design. Specifically, we focus on efficiently computing the penetration depth between two polyhedra, which is a well-studied problem in the computer graphics community. We formulate the problem as a constrained five-variable global optimization problem, and then derive an equivalent unconstrained, 2-variable nonsmooth problem. To solve the optimization problem, we apply a popular stochastic multistart optimization algorithm in a novel way, which exploits the advantages of each problem formulation simultaneously. Numerical results for the algorithm, applied to 14 randomly generated pairs of penetrating polytopes, illustrate both the effectiveness and efficiency of the method.**

***Index Terms*—Constrained optimization, global optimization, nonlinear optimization, penetration depth, geometric modelling**

I. Introduction

In the detailed phase of any aerospace program, one of the most important consistency checks is to ensure that no two distinct objects occupy the same physical space. This is complicated by the terabytes of data required for geometric modeling of an airplane. In order to make design problems tractable, discretization of the geometry models is a normal practice. However, this introduces small errors that may cause interferences between objects, which were not part of the fully detailed model. These interferences are artifacts of the discretization, not the underlying design. To handle this problem, we can compute the depth of interference and use it to distinguish between these false positives and genuine inconsistencies in the design.

We focus here on computing penetration depth between two convex polyhedra or polytopes. Specifically, we want to detect if two polytopes overlap, and if so, what the penetration depth is; *i.e.*, the shortest distance that one polytope must move to eliminate the overlap.

* Utah Valley University, 800 West University Parkway, Orem, UT 84058 USA, mark.abramson@uvu.edu

† Utah Valley University, 800 West University Parkway, Orem, UT 84058 USA, 10764442@my.uvu.edu

‡ Intel Corporation, 2501 NE Century Blvd, Hillsboro, OR 97124 USA, gavin.w.smith@intel.com

The penetration depth problem has been well-studied (*e.g.*, see [1]–[7]), and several algorithms have been proposed for solving the problem. However, most of the approaches described in the literature involve some type of enumeration technique, which we believe is too inefficient for the application that motivates us – one in which polytopes may have on the order of 20-30 vertices, and for which very quick and accurate computations are needed because they will be performed potentially hundreds of thousands of times.

In this paper, we formulate the problem as a constrained optimization problem with five variables, based on the idea of separating hyperplanes [7], [8]. We also introduce a two-variable nonsmooth reformulation, which we will exploit in a new algorithm we introduce for solving this problem. We believe this algorithm is the most efficient to date for solving the penetration depth problem.

The rest of this paper is laid out as follows. In Section II, we give the mathematical description of the problem, including the two formulations as optimization problems. In Section III, we describe our algorithm, which is based on a well-known stochastic multistart optimization algorithm with strong convergence properties. In Section IV, we present numerical results based on tests of some randomly generated pairs of intersecting polytopes. Section V offers some concluding remarks, including a note on dealing with nonconvex polytopes.

II. Problem Formulation

Given two convex polytopes P and Q with respective vertices of $\{p_1, p_2, \ldots, p_{n_p}\}$ and $\{q_1, q_2, \ldots, q_{n_q}\}$ in $\mathbb{R}^3$, the overlap or penetration is given by $\text{int}(P) \cap \text{int}(Q)$. The penetration depth is defined as the minimal translation distance (in any direction) of Q required to eliminate the penetration.

The mathematical formulation of the depth penetration problem is based on the notion of separating hyperplanes. The general (linear) equation for a hyperplane is given by

$$a^T v - c = 0, \tag{1}$$

where $a \in \mathbb{R}^3$ is a unit vector normal to the hyperplane, and $c \in \mathbb{R}$ is a constant representing the offset of the plane from the origin. A hyperplane divides the domain into two half-spaces, each defined by an inequality version of (1). This means that

there is no penetration between two polytopes if and only if, for some v and c, we have

$$\begin{aligned} p_i^T v - c &\leq 0, \quad i = 1, 2, \ldots, n_p, \\ q_j^T v - c &\geq 0, \quad j = 1, 2, \ldots, n_q. \end{aligned}$$

Then, for two intersecting polytopes, moving one while holding the other fixed w ill s eparate t hem, a nd penetration depth $\alpha > 0$ can be computed as the solution to the optimization problem,

$$\begin{aligned} \min_{v,\alpha,c} \quad & \alpha \\ \text{subject to} \quad & q_j^T v - c + \alpha \geq 0, \quad j = 1, 2, \ldots, n_q \\ & p_i^T v - c \leq 0, \qquad i = 1, 2, \ldots, n_p \\ & v^T v = 1, \\ & \alpha \geq 0, \end{aligned} \tag{2}$$

where the vector $v \in \mathbb{R}^3$ in (2) becomes the direction of the translation. If there exists a feasible solution to (2) with $\alpha = 0$, then there is no penetration between the polytopes.

Note that the objective function and all constraints in (2) are linear, except for the quadratic equality constraint $v^T v = 1$. However, this also makes the feasible region S nonconvex, which means that the problem will have multiple local solutions, and if we naively apply a derivative-based optimization algorithm, we are only ensured of finding a local solution to (2). This is not unexpected, since the optimal translation for separating the polytopes can be proved to occur when the vector v is orthogonal to a face, edge, or vertex of one of the polytopes. For this reason, the majority of algorithms found in the literature (e.g., [3], [4], [6], [7]) for solving (2) focus on the enumeration of facets, edges, and vertices. For polytopes with more than just a small number of vertices, the convex hull of the intersection may have a larger number of vertices, and enumerating all possible combinations of faces, edges, and vertices and then computing distances becomes cost-prohibitive very quickly. We give an example of this later in the paper.

It turns out that there is another equivalent formulation of (2), based on an observation in [1]. To show this, we first rewrite the two sets of plane constraints as

$$\begin{aligned} \alpha - c &\geq -q_j^T v, \quad j = 1, 2, \ldots, n_q, \\ c &\geq p_i^T v, \quad i = 1, 2, \ldots, n_p. \end{aligned}$$

By taking the sum of the two constraints and noting that we are minimizing $\alpha \geq 0$, we can express the optimal solution as

$$\begin{aligned} \alpha^* &= \max\{0, \max_i (p_i^T v) - \min_j (q_j^T v)\} \\ &= \max_{i,j} \{0, (p_i + q_j)^T v\}, \end{aligned}$$

which is a function of v. Then, to enforce the quadratic equality constraint in (2), we can represent v on the unit sphere in terms of angles $\theta \in [0, 2\pi)$ and $\phi \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$, using a standard transformation to spherical coordinates (but with radius 1); namely,

$$v = v(\theta, \phi) = (\cos\theta \cos\phi, \sin\theta \cos\phi, \sin\phi)^T. \tag{3}$$

The result is a bound-constrained, nonsmooth minimization problem in two variables; namely,

$$\min_{(\theta,\phi) \in S} \alpha(\theta, \phi) = \max_{i,j} \{0, (p_i + q_j)^T v(\theta, \phi)\}, \tag{4}$$

where $v(\theta, \phi)$ is given in (3), and

$$S = \{(\theta, \phi) : 0 \leq \theta \leq 2\pi, \ -\tfrac{\pi}{2} \leq \phi \leq \tfrac{\pi}{2}\}. \tag{5}$$

We shall refer to (2) and (4) as the 5-variable and 2-variable problems, respectively.

The reduction to two variables also allows us to plot the nonsmooth objective function, as shown in Figure 1 for a pair of randomly generated penetrating polytopes. We can see in the figure that the global minimum is at a point of nonsmoothness, which means that the gradient of the objective function is not zero (in fact, it does not exist), and we cannot successfully apply a derivative-based solver.

Note that bounds on θ and ϕ given in (5) are optional. This is because $\alpha(\theta, \phi)$ is periodic in both variables and we have no particular interest in the angles at optimality – only the resulting penetration depth. Furthermore, applying these bounds to the 2-variable problem adds artificial local optima on the boundary, which are almost never global optima. However, since the algorithm we use requires finite bounds on all variables, we will use them.

As an alternative to enumeration algorithms, Agarwal et al. [1] proposed approximating the penetration depth by performing a grid search in the space of (θ, ϕ). This idea was extended in [9] by starting local optimizations from the grid points by means of a pattern search algorithm [10], [11], a derivative-free method with a suitable (but weaker) convergence theory for nonsmooth problems [12], [13]. However, the nonsmoothness often caused premature termination of the algorithm, leading to somewhat inconsistent results, in the sense that increasing the number of grid points did not always produce a better result.

III. Solving the Optimization Problem

We propose a new algorithm, which appears to be significantly more effective in quickly solving the penetration depth problem. It is constructed from an existing multistart stochastic global optimization algorithm, but applied in a novel way. Specifically, we apply the multi-level single linkage (MLSL) algorithm [14]–[16] to the 2-variable problem, but each time we perform a local optimization, we convert the 2-variable starting point to the 5-variable problem domain, apply a more powerful Newton-based local solver to the 5-variable problem, and then map the resulting 5-variable solution back to 2-variables.

In describing MLSL, we use the following nomenclature:

- f: function to be optimized
- n: number of variables
- N: number of samples per iteration
- X: current set of sample points
- M: number of local optimizations per iteration
- X_0: current set of local optimization start points
- $\mu(A)$: measure (or hypervolume) of the set A
- τ: desired percentage of local optima found

σ: critical distance parameter
X^*: set of local optima found
w: number of local optima found
$\bar{w}$: expected number of local optima

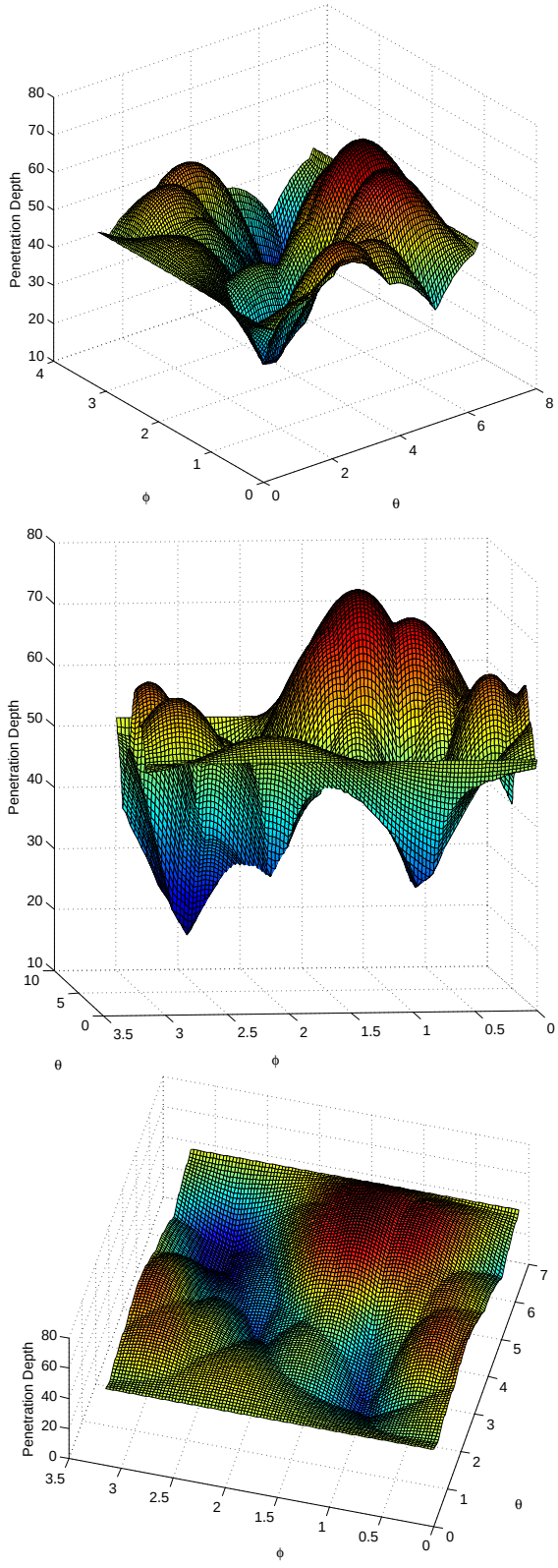

Fig. 1. 3 views of a surface plot of $\alpha(\theta, \phi)$

The MLSL algorithm, which is summarized in Algorithm 1, consists primarily of a local inner loop nested inside of a global outer loop. Steps 4-5 generate uniformly distributed

Algorithm 1 Multi-level Single Linkage (MLSL)

Input: $\sigma > 4$, $N > 0$, $M \in \{1, 2, \ldots, N\}$, $\tau \in [0, 1)$.
Output: X^*

1: $k \leftarrow 0$, $w \leftarrow 0$, $\bar{w} \leftarrow \infty$, $X^* \leftarrow \emptyset$
2: **while** $w/\bar{w} < \tau$ **do**
3: $k \leftarrow k + 1$
4: $r_k \leftarrow \frac{1}{\sqrt{\pi}} \left[\Gamma\left(1 + \frac{n}{2}\right) \mu(S) \sigma \frac{\ln kN}{kN} \right]^{1/n}$
5: Randomly sample (uniformly) N points X that satisfy $\|x - y\|_2 > r_k$ for all $x, y \in X$.
6: Evaluate $f(x)$ for all $x \in X$, and set $X_0 \subseteq X$ as the subset of M points with lowest function values.
7: **for all** $x_0 \in X_0$ **do**
8: Perform local optimization starting at x_0, returning optimal point x^*.
9: **if** $x^* \notin X^*$ **then**
10: $X^* \leftarrow X^* \cup \{x^*\}$
11: $w \leftarrow w + 1$
12: **end if**
13: **if** $kM > w + 2$ **then**
14: $\bar{w} \leftarrow \frac{w(kM - 1)}{kM - w - 2}$
15: **end if**
16: **end for**
17: **end while**
18: **return** X^*

random sample points in the compact domain S (see (5)), which are sufficiently far away from each other, as defined by the distance threshold r_k. This parameter is constructed from mathematical observations given in [15], and it decreases to zero as the iteration number k increases. In Step 6, we evaluate the objective function at each sampled point and then choose the best M sampled points as starting points for the local optimizations performed by the inner loop (Steps 7-17). Inside the loop, Steps 9-12 are for processing each new local optima found in Step 8, and Steps 13-15 update the (Bayesian) expected number of local optima $\bar{\omega}$ (also shown in [15]), where ω denotes the number of local optima found thus far. Termination occurs when the expected percentage of local optima found reaches a specified threshold τ (see Step 2).

Rinnooy Kan and Timmer [16] proved the following properties for Algorithm 1:

- The total number of local optimizations started by Algorithm 1 is finite with probability 1.
- Any local minimum will be found by Algorithm 1 in a finite number of iterations with probability 1.
- Algorithm 1 can be implemented in such a way that the expected running time up to iteration k is $\mathcal{O}(k)$.

For the penetration depth problem, $\mu(S) = 2\pi^2$ (see (5)). To implement the MLSL algorithm effectively, we modify the inner loop of Algorithm 1 by replacing the single Step 8 in Algorithm 1 with all the steps given in Algorithm 2.

Algorithm 2 Modified Step 8 of MLSL Algorithm

Input: $x_0 \in \mathbb{R}^2$

Output: $x^* \in \mathbb{R}^2$

1: $(\theta_0, \phi_0) \leftarrow x_0$

2: $v_0 \leftarrow (\cos\theta_0 \cos\phi_0,\ \sin\theta_0 \cos\phi_0,\ \sin\phi_0)$

3: $c_0 \leftarrow \max_i p_i^T v_0$

4: $\alpha_0 \leftarrow c_0 - \min_j q_j^T v_0$.

5: Perform local optimization of the 5-variable problem from initial point (v_0, c_0, α_0), returning optimal point (v, c, α).

6: $$\theta^* \leftarrow \begin{cases} \pi/2, & v_1 = 0,\ v_2 = 0 \\ (\pi/2)\,\mathrm{sign}(v_2), & v_1 = 0,\ v_2 \neq 0 \\ \arctan\left(\dfrac{v_2}{v_1}\right), & v_1 \neq 0 \end{cases}$$

7: $$\phi^* \leftarrow \begin{cases} \pi/2, & v_3 = 0 \\ \arctan\left(\dfrac{\sqrt{v_1^2 + v_2^2}}{v_3}\right), & v_3 \neq 0 \end{cases}$$

8: $x^* \leftarrow (\theta^*, \phi^*)$

In Algorithm 2, Steps 1-4 convert the 2-variable starting point x_0 into the 5-variable starting point (v_0, c_0, α_0). Following the local optimization in Step 5, Steps 6-8 convert the 5-variable local optimum (v, c, α) into the 2-variable x^*. In Steps 6 and 7, the expressions with $\frac{\pi}{2}$ are only for handling divisions by zero, since the values of $\pm\frac{\pi}{2}$ represent the limits of $\arctan x$ as $x \to \pm\infty$.

Note that the first case in Step 6 is for handling the indeterminate case, which only happens at the pole of the spherical coordinate system. In this case, θ^* can be chosen arbitrarily. However, unless we consistently choose the same value (we chose $\frac{\pi}{2}$) whenever this occurs, the algorithm may map the same solution to different points in the 2-variable domain. This would make Algorithm 1 less efficient, since its termination condition may take longer to achieve.

IV. Numerical Testing

Algorithms 1 and 2 were coded into both Matlab and Python and tested on 14 randomly generated pairs of penetrating polytopes with n_p and n_q vertices. In the results that follow, we compared MLSL with three implementations of a simple grid search (GS), as proposed in [1], with 10, 40, and 180 points along each axis. The value of 180 was chosen to match exactly one degree in each angle θ and ϕ. Local optimizations were performed using three Matlab Optimization Toolbox algorithms and two SciPy optimization algorithms. In particular, the following solvers were tested:

1) Matlab interior-point (IP) [17]–[19]
2) Matlab sequential quadratic programming (SQP) [20]–[22]
3) Matlab active-set (AS) [23]–[28]
4) SciPy SLSQP [29]
5) SciPy trust-region (TR) [18], [30]

The parameters used by MLSL were set as follows: $N = 10$, $M = 3$, $\sigma = 4.01$, and $\tau = 0.90$. Termination tolerances for local solvers were all set to 10^{-8}, and all other parameters were set to their defaults. Outputs of interest were the actual penetration depth α, and the required number of function evaluations and CPU time to find α.

We also compare our results against a GS-SQP algorithm, in which we keep the local SQP solver, but replace the MLSL framework with a grid search. In particular, the grid is formed using only 3 points along each axis (θ and ϕ), and local optimizations are initialized from all of them.

For each of the numbered problems, Table I gives computed penetration depths using the different algorithms. All five local solver options were grouped together in the MLSL column because they all converged to the same values. Table II gives the number of function evaluations for each problem and algorithm, along with the totals for function evaluations and CPU time across all problems.

From Tables I and II, we make the following general observations:

- Regardless of the local solver, MLSL finds better solutions than any grid search at a much lower computational cost. The only exception is the coarsest grid search, which is not particularly useful in practice, since the best penetration depths found by this grid search are significantly worse than what MLSL found.
- In a relative sense, the chosen local solvers perform equally well, except for the SciPy trust region algorithm, which requires significantly more CPU time.
- The GS-SQP algorithm achieves everything that the MLSL-SQP algorithm does, but at a fraction of the computational cost. However, it comes with the price of not having the strong global convergence properties that MLSL has. Our purpose in showing the GS-SQP result is to show how conservative MLSL is with respect to its percentage of expected local optimizers found. In reality, while the optimization problems we are solving are nonconvex, they do not have many local optima (as can be seen in Figure 1). Therefore, it is safe to say that MLSL-SQP finds all the local optima quickly, and spends the rest of its time converging to the same optima from different starting points, so that the percentage τ^* can be reached, in which case, there is sufficient confidence that the best solution found is indeed the global solution.
- With a Newton-based local solver, MLSL and GS-SQP find better solutions at no worse computational cost than the grid-based pattern-search approach employed in [9].

In addition to these runs, we also tried varying the MLSL parameters σ, N, and M, and we found that the algorithm is not sensitive to reasonable changes to these parameter values. More specifically,

- No changes to MLSL parameters resulted in any change to the computed penetration depth. This is not surprising, given our results for GS-SQP, in which we achieve the same penetration depths from a total of only 9 local optimization starting points.
- While changes to MLSL parameter values can significantly alter the number of MLSL *iterations*, only modest changes to the number of function evaluations and CPU time were observed, and they were of the same order of

TABLE I
COMPUTED PENETRATION DEPTH FOR 14 RANDOMLY GENERATED POLYTOPE PAIRS

Problem #: (n_p, n_q)	GS-10	GS-40	GS-180	MLSL	GS-SQP
1: (8, 8)	1.000	1.000	1.000	1.000	1.000
2: (6, 9)	22.038	19.496	19.390	19.314	19.314
3: (14, 5)	35.224	32.537	31.877	31.842	31.842
4: (8, 10)	32.658	30.112	29.095	28.911	28.911
5: (9, 7)	27.500	23.263	22.291	22.147	22.147
6: (10, 5)	22.315	21.614	21.413	21.314	21.314
7: (13, 6)	14.719	11.970	11.500	11.479	11.479
8: (9, 8)	30.310	27.288	26.587	26.406	26.406
9: (15, 9)	35.854	34.875	34.148	34.128	34.128
10: (15, 11)	35.402	34.004	33.651	33.644	33.644
11: (8, 11)	37.936	35.370	35.124	34.971	34.971
12: (10, 6)	27.104	25.372	24.754	24.446	24.446
13: (10, 8)	21.729	19.854	19.866	19.710	19.710
14: (103, 93)	16977.000	16977.000	16977.000	16970.909	16970.909

TABLE II
FUNCTION EVALUATIONS FOR 14 RANDOMLY GENERATED POLYTOPE PAIRS

Problem #: (n_p, n_q)	GS-10	GS-40	GS-180	IP	SQP	AS	SLSQP	TR	GS-SQP
1: (8, 8)	220	3280	65160	353	345	318	381	1007	30
2: (6, 9)	220	3280	65160	483	713	508	644	660	50
3: (14, 5)	220	3280	65160	920	937	811	751	894	44
4: (8, 10)	220	3280	65160	1058	1531	1385	2094	1894	51
5: (9, 7)	220	3280	65160	560	871	733	1191	1046	66
6: (10, 5)	220	3280	65160	889	1406	1157	1371	1274	50
7: (13, 6)	220	3280	65160	475	702	593	1033	1276	46
8: (9, 8)	220	3280	65160	1263	1711	1477	1265	952	56
9: (15, 9)	220	3280	65160	932	1403	1178	1776	1321	48
10: (15, 11)	220	3280	65160	867	1286	1091	1737	1508	56
11: (8, 11)	220	3280	65160	994	1562	1184	1491	1581	56
12: (10, 6)	220	3280	65160	978	825	699	1915	933	50
13: (10, 8)	220	3280	65160	849	937	764	1134	1104	58
14: (103, 93)	220	3280	65160	1108	1479	835	2316	1700	42
Function Evaluations:	3080	45920	912240	11729	15891	12983	19099	17150	704
CPU Time (sec):	0.051	0.36	6.29	14.38	5.66	8.37	6.16	46.91	0.97

magnitude as those shown in Table II. This is because the MLSL termination condition is dependent on the number of local optimizations performed. A larger value of N, results in fewer iterations before the condition is satisfied, and vice versa.

- Changes to the local optimization termination tolerance only changes the penetration depth when care is not taken to ensure that this tolerance is smaller than the one used to identify distinct local optima in MLSL. Furthermore, a tighter local optimization tolerance is not costly at all, due to the superlinear convergence rate of the local solvers.

Admittedly, we did not compare against any enumeration algorithms. The reason for this can be illustrated by considering Problem 14, whose two polytopes have 103 and 93 vertices. Enumeration algorithms require the formation of $P-Q$, often referred to as the Minkowski sum of the two polytopes, which is akin to the Cartesian product of two vectors. The resulting polytope can have up to $(103)(93) = 9579$ vertices. The convex hull for this test problem turns out to have 2274 vertices. The penetration depth becomes the shortest distance from the origin to one of its faces. Since each face is defined by 3 vertices, the (worst case) number of distance computations (or function evaluations) would be

$$\begin{pmatrix} 2274 \\ 3 \end{pmatrix} = 1,957,253,024.$$

which is significantly more function evaluations than any of the other methods we have tested, and which makes it intractable for our application.

V. CONCLUDING REMARKS

We have introduced here a new approach for finding global solutions to the penetration depth problem, by combining the MLSL stochastic global optimization algorithm with a Newton-based local solver in a novel way that exploits the 2-variable and 5-variable problem formulations to full advantage. The Newton-based local solvers we have tested are all provably convergent and have at least a superlinear rate of convergence to each local solution. The MLSL algorithm is provably convergent to a global solution in a finite number of iterations. Numerical testing shows that our modified MLSL algorithm is superior to grid searches and much more efficient than an enumeration algorithm would be, without sacrificing the quality of the solution. Furthermore, we have shown that even a multistart local solver that selects starting points from a fixed grid can achieve the same solutions as MLSL in a fraction of the cost, but without the convergence guarantee.

Finally, in the application that has motivated our work, we can certainly encounter a mixture of convex and nonconvex polytopes. However, we are not concerned with actually separating the polytopes; we only wish to compute penetration

depth. Thus, we do not wish to redefine penetration depth to include rotations, as has been suggested as an option in the literature (*e.g.*, see [31]). Instead, we prefer a simpler and more naive approach. Since a nonconvex polytope can be decomposed into the union of convex polytopes, we can perform a decomposition (see, *e.g.*, [32], [33]), apply our algorithm, in turn, to each pair of convex pieces (one piece per object), and compute the maximum penetration depth among the pairs. Of course, this is more computationally costly than the convex case (and there may be other issues than those described here), but it is less costly than the majority of algorithms proposed in the literature for the nonconvex case.

ACKNOWLEDGEMENT

The authors thank Tom Grandine of The Boeing Company for introducing us to this problem and for many valuable conversations.

REFERENCES

[1] P. K. Agarwal, L. J. Guibas, S. Har-peled, A. Rabinovitch, and M. Sharir, "Penetration depth of two convex polytopes in 3D," *Nordic Journal of Computing*, vol. 7, pp. 227–240, 2000.

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

[3] S. A. Cameron and R. K. Culley, "Determining the minimum translational distance between two convex polyhedra," in *International Conference on Robotics and Automation*, San Francisco, California, 1986, pp. 591–596.

[4] D. Dobkin, J. Hershberger, D. Kirkpatrick, and S. Suri, "Computing the intersection-depth of polyhedra," *Algorithmica*, vol. 9, no. 6, pp. 518–533, 1993.

[5] Y. J. Kim, M. C. Lin, and D. Manocha, "DEEP: Dual-space expansion for estimating penetration depth between convex polytopes," in *IEEE Conference on Robotics and Automation*, 2002, pp. 921–926.

[6] ——, "Incremental penetration depth estimation between convex polytopes using dual space expansion," *IEEE Transactions on Visualization and Computer Graphics*, vol. 10, no. 2, pp. 152–163, 2004.

[7] C. J. Ong and E. G. Gilbert, "Growth distances: New measures for object separation and penetration," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 6, pp. 888–903, 1996.

[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 Journal on Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 1988.

[9] M. A. Abramson, T. A. Grandine, W. D. McGarry, and G. W. Smith, "Computing the penetration depth of two polytopes," The Boeing Company, Seattle, Tech. Rep. S & A-TECH-14-004, September 2014.

[10] R. M. Lewis and V. Torczon, "Pattern search algorithms for bound constrained minimization," *SIAM Journal on Optimization*, vol. 9, no. 4, pp. 1082–1099, 1999.

[11] T. G. Kolda, R. M. Lewis, and V. Torczon, "Optimization by direct search: new perspectives on some classical and modern methods," *SIAM Review*, vol. 45, no. 3, pp. 385–482, 2003.

[12] M. A. Abramson, "Second-order behavior of pattern search," *SIAM Journal on Optimization*, vol. 16, no. 2, pp. 315–330, 2005.

[13] C. Audet and J. E. Dennis, Jr., "Analysis of generalized pattern searches," *SIAM Journal on Optimization*, vol. 13, no. 3, pp. 889–903, 2003.

[14] A. H. G. R. Kan and G. T. Timmer, *A stochastic approach to global optimization*. Philadelphia: SIAM, 1984, pp. 245–262.

[15] ——, "Stochastic global optimization methods, part 1: clustering methods," *Mathematical Programming*, vol. 39, no. 1, pp. 27–56, 1987.

[16] ——, "Stochastic global optimization methods, part 2: multi-level methods," *Mathematical Programming*, vol. 39, no. 1, pp. 57–78, 1987.

[17] R. H. Byrd, J. C. Gilbert, and J. Nocedal, "A trust region method based on interior point techniques for nonlinear programming," *Mathematical Programming*, vol. 89, no. 1, pp. 149–185, 2000.

[18] R. H. Byrd, M. E. Hribar, and J. Nocedal, "An interior point algorithm for large-scale nonlinear programming," *SIAM Journal on Optimization*, vol. 9, no. 4, pp. 877–900, 1999.

[19] R. A. Waltz, J. L. Morales, J. Nocedal, and D. Orban, "An interior algorithm for nonlinear optimization that combines line search and trust region steps," *Mathematical Programming*, vol. 107, no. 3, pp. 391–408, 2006.

[20] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed., ser. Springer Series in Operations Research. Springer Verlag, 2006.

[21] P. Spellucci, "A new technique for inconsistent qp problems in the sqp method," *Mathematical Methods of Operations Research*, vol. 47, no. 3, p. 355–400, 1998.

[22] K. Tone, "Revisions of constraint approximations in the successive QP method for nonlinear programming problems," *Mathematical Programming*, vol. 26, no. 2, p. 144–152, 1983.

[23] R. Fletcher and M. J. D. Powell, "A rapidly convergent descent method for minimization," *Computer Journal*, vol. 6, pp. 163–168, 1963.

[24] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*. London: Academic Press, 1981.

[25] D. Goldfarb, "A family of variable metric updates derived by variational means," *Mathematics of Computing*, vol. 24, pp. 23–26, 1970.

[26] S. P. Han, "A globally convergent method for nonlinear programming," *Journal of Optimization Theory and Applications*, vol. 22, p. 297, 1977.

[27] M. J. D. Powell, "A fast algorithm for nonlinearly constrained optimization calculations," in *Numerical Analysis*, ser. Lecture Notes in Mathematics, G. A. Watson, Ed. Philadelphia: Springer-Verlag, 1978, vol. 630, pp. 245–262.

[28] ——, "The convergence of variable metric methods for nonlinearly constrained optimization calculations," in *Nonlinear Programming 3*, O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, Eds. Academic Press, 1978.

[29] D. Kraft, "A software package for sequential quadratic programming," DLR German Aerospace Center – Institute for Flight Mechanics, Koln, Germany, Tech. Rep. DFVLR-FB 88-28, 1988.

[30] M. Lalee, J. Nocedal, and T. Plantega, "On the implementation of an algorithm for large-scale equality constrained optimization," *SIAM Journal on Optimization*, vol. 8, no. 3, pp. 682–706, 1999.

[31] L. Zhang, Y. J. Kim, G. Varadhan, and D. Manocha, "Generalized penetration depth computation," in *SPM '06: Proceedings of the 2006 ACM symposium on solid and physical modeling*. New York: ACM Press, 2006, pp. 173–184.

[32] C. L. Bajaj and T. K. Dey, "Convex decomposition of polyhedra and robustness," *SIAM Journal of Computing*, vol. 21, no. 2, pp. 339–364, 1992.

[33] B. M. Chazelle, "Convex decompositions of polyhedra," in *STOC '81: Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*, May 1981, pp. 70–79.

Mark A. Abramson is an Associate Professor of Mathematics at Utah Valley University. His degrees include a BS from Brigham Young University, 2 MS degrees from the Air Force Institute of Technology, an MSAA from University of Washington, and MA and PhD from Rice University. He is a retired US Air Force officer, and also worked for the Boeing Company for 7 years. His research interests are in optimization and numerical analysis.

Griffin D. Kent is an graduate student in the Department of Industrial and Systems Engineering at Lehigh University. He received his BS degree in Statistics from Utah Valley University in 2020.

Gavin W. Smith received his PhD in Mathematics from Washington State University in 2013. He has taught at Gonzaga University and now works for the Intel Corporation in their Advanced Design department.