Fast Approximate Union Volume in High Dimensions with Line Samples

Scott A. Mitchell, Muhammad A. Awad, Mohamed S. Ebeida, Laura P. Swiler

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550


Approved for public release; further dissemination unlimited.
Abstract

The classical problem of calculating the volume of the union of $d$-dimensional balls is known as “UnionVolume.” We present line-sampling approximation algorithms for UnionVolume. Our methods may be extended to other Boolean operations, such as setminus; or to other shapes, such as hyper-rectangles. The deterministic, exact approaches for UnionVolume do not scale well to high dimensions. However, we adapt several of these exact approaches to approximation algorithms based on sampling. We perform local sampling within each ball using lines. We have several variations, depending on how the overlapping volume is partitioned, and depending on whether radial, axis-aligned, or other line patterns are used.

Our variations fall within the family of Monte Carlo sampling, and hence have about the same theoretical convergence rate, $1/\sqrt{M}$, where $M$ is the number of samples. In our limited experiments, line-sampling proved more accurate per unit work than point samples, because a line sample provides more information, and the analytic equation for a sphere makes the calculation almost as fast. We performed a limited empirical study of the efficiency of these variations. We suggest a more extensive study for future work. We speculate that different ball arrangements, differentiated by the distribution of overlaps in terms of volume and degree, will benefit the most from patterns of line samples that preferentially capture those overlaps.
Acknowledgement

We thank Karl Bringman for explaining his BF-ApproxUnion (ApproxUnion) algorithm [3] to us. We thank Josiah Manson for pointing out that spoke darts oversample the center and we might get a better answer by uniform sampling. We thank Vijay Natarajan for suggesting random chord sampling. The authors are grateful to Brian Adams, Keith Dalbey, and Vicente Romero for useful technical discussions.

This work was sponsored by the Laboratory Directed Research and Development (LDRD) Program at Sandia National Laboratories. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research (ASCR), Applied Mathematics Program. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-NA0003525.
Contents

1 Introduction ................................................................. 7
  1.1 Our contribution ......................................................... 9
2 Background ................................................................. 10
  2.1 Deterministic methods for UnionVolume ................................ 10
  2.2 UnionVolume sampling approaches .................................... 11
3 Volume estimation by line samples ...................................... 14
  3.1 Neighbor finding ......................................................... 14
  3.2 Number of samples per ball ............................................ 14
  3.3 Cell volume estimation ................................................ 15
  3.4 Cell types .............................................................. 19
  3.5 Sampling primitives .................................................. 21
4 Complexity analysis ....................................................... 23
5 Extensions ................................................................. 24
6 Experimental efficiency .................................................. 25
  6.1 Relative error, $\eta$, proxy by $\sigma$ .................................... 25
  6.2 Ball distribution types $F$ ................................................ 25
  6.3 Algorithms, $A$ .......................................................... 26
  6.4 Experimental results .................................................. 26
7 Proposed experiments ..................................................... 30
  7.1 Relationship of $M, \sigma, \text{and } T$ ...................................... 30
  7.2 Scaling by $N$ ............................................................ 31
  7.3 Scaling by dimension .................................................. 32
  7.4 $F$ other distribution types ........................................... 32
8 Conclusion ................................................................. 34
References ........................................................................ 35

Figures

1 Various line sampling patterns useful for volume estimation. Both types of slats provide uniform volume coverage. Chords undersample the center, and spokes oversample the center, so samples must be weighted. .................. 8
2 Three types of cell partitions of the union. ................................ 9
3 A line sample is trimmed by the disks or hyperplanes defining the cell. We retain only the segments inside the cell. Zero, one, or more segments may be retained depending on the cell and line sample type. Here for exposition we only show one ray of each spoke line sample. ........................................ 16
4 How to weight chord samples for volume integration and estimation. ........ 19
5 Cone sizing function with 100 (left) and 1000 (right) balls. The additional balls are typically very small and concentrated near the $F = 0$ level set, the dark blue circle. ..... 26
6 Convergence rate by error. ................................................ 27
Scaling by dimension. The different linear slopes come from different constant factors in the $O(d)$ scaling.

Scaling by number of balls, $n$. The different linear slopes come from different constant factors in the $O(n)$ scaling.

Scaling of time / $n$ by number of balls, $n$. All curves are expected to be flat. Lower curves have smaller constants in their $O(n)$ scaling.

Tables

1. BF-ApproxUnion relative error ($\epsilon$) and required # samples per ball.
1 Introduction

The problem, and why it is important. We consider the problem of computing the volume of a union of balls, the problem known as UnionVolume. This is a classic geometry problem. Modeling with balls and UnionVolume have important applications in uncertainty quantification. In particular, in concurrent work, Probability-Of-Failure Darts (POF-Darts) [12], we construct a set of balls approximating the failure region, and the relative UnionVolume approximates the probability of failure. In other UQ contexts, balls, points and spherical neighborhoods, are useful devices in high dimensional spaces to determine the fraction of space adequately explored by sampling, or within trust regions of surrogates, etc. High dimensions, say 10 or 100 dimensional parameter spaces, are common.

In geometric modeling, complex shapes and shape functions may be approximated or bounded by a set of overlapping balls [24, 5], which individually are very simple and analytic. Some overlapping ball models arise from physical space. One 3-d application is the volume of an arrangement of atoms in a molecule, where the Van der Waals radii overlap. Often these radii are expanded, e.g. to explore solvent accessible regions, as in $\alpha$-shapes, with applications in computational biology. A 2-d application example, from design engineering, is computing the area of coverage for cell phone towers. The medial axis transform represents a solid by an infinite union of balls, with centers on the medial axis. A finite subset can approximate the shape, as in the power crust [1]. Selecting the best set of balls to approximate a Graphics or CAD model is an active research topic [26]. Inner, outer, and sandwiching approximations are considered. Quality is often measured by the difference in volume between the approximation and the original. Other applications of modeling with balls include image matching and interpolation, shape model extraction from image data, two and three-dimensional shape simplification, polygonal surface reconstruction [27], and collision detection [15].

A generalization of UnionVolume is integration of some quantity over the volume of the shape, such as mass or heat. There has been a surge in high dimensional volume and integration (e.g. quadrature) research. In our context, integrating a probability density function would allow POF calculations over non-uniform distributions. High dimensional models also arise in informatics problems such as statistical marginal densities, pattern matching, and approximate database queries.

Why it is hard in high dimensions. The sum of the volumes of the individual balls is larger than UnionVolume, because it overcounts the volume contained in multiple balls. The standard deterministic algorithm is to partition the union into disjoint cells, one for each ball, and add the cell volumes. This works well in low dimensions, say 2 or 3. Because the combinatorial complexity of the faces of the boundary grows factorially with dimension, it is intractable to calculate the cell boundary and perform the partition for 100 dimensions, or even 10 dimensions [7]. So instead we seek to estimate UnionVolume without computing cell or union boundaries, by sampling. After all, approximating a complicated and noisy geometric boundary is often one of the motivations for constructing the balls in the first place.

Traditional point sampling, and new line sampling. Point sampling is traditional. That is,
perform the standard partition conceptually, then generate sample points inside each ball, and test whether each sample point is inside the ball’s cell. The estimate of the cell’s volume is the fraction of sample points inside the cell times the volume of the ball.

![Figure 1. Various line sampling patterns useful for volume estimation. Both types of slats provide uniform volume coverage. Chords undersample the center, and spokes oversample the center, so samples must be weighted.](image)

We propose line sampling as a more efficient alternative. For each line, we calculate the subsegments actually in the ball’s cell, and weight them as in integration to estimate cell volume. Each cell volume estimate is the average weighted fraction of the lines’ segments, times the ball volume. See Figure 1 for line sample types. Our initial experiments included only radial lines through the ball center, spokes. For spokes, each segment is weighted by its distance to the center, as in its swept volume for integration in polar coordinates. A second type of lines are uniform random, where the weighting is a constant and independent of distance.

**Types of partitions.** Three types of partitions are traditional in three different fields; see Figure 2. From computational geometry, CoveredCells partition, a cell is the intersection of the ball and its power cell. Power cells are a weighted variant of Voronoi cells. Graphics assigns an order-
ing to the balls for the OccludedCells partition. A cell is the subset of the ball that is outside all balls prior to it in the ordering. In topology, DepthCells is a weighted partition. The number of balls containing a point is its depth. Partition each ball into regions of constant depth. Weight each region by its depth. That is, the cell of a ball is the entire ball, but the subregion covered by two balls is weighted by 1/2, covered by three balls is weighted by 1/3, etc.

![Diagram](image_url)

(a) Covered (power) cells. (b) Occluded cells. (c) Depth cells.

**Figure 2.** Three types of cell partitions of the union.

### 1.1 Our contribution

We consider the practical efficiency of approximate union volume algorithms in high dimensions. There are several standard deterministic approaches for union volume based on partitioning. We adapt these to point and line sampling algorithms. The line sampling algorithms are new, and we call these generically BalloonDarts. In addition, we suggest comparison to other algorithms which scale well with dimension. These include the union volume approximation algorithm BF-ApproxUnion [4], and random walk methods for convex polytopes [14].

We implemented the BalloonDarts UnionVolume variants described in this paper. According to the BF-ApproxUnion [4] authors, we are also the first to implement their algorithm for balls. BF-ApproxUnion has been previously implemented for boxes, BF-HypervolumeApproximator [18, 29]. BF-ApproxUnion appears to be fundamentally point based and we were not able to discover a line-sampling variation of it. Random walks are inherently line based.

**Practical performance.** In our limited tests, line sampling outperformed point sampling, and OccludedCell is the preferred partition. CoveredCells is known to be the preferred 2-d deterministic method because it is easy to calculate the volume of each partition analytically. However, this benefit is lost when sampling, and OccludedCells is both faster and easier to implement. OccludedCells is even more efficient if we can order the balls so that many large ones are unoccluded, so their cell volume is just the analytic ball volume. We suggest additional experiments to compare point sampling, line sampling, and the alternatives. Considerations include scalability by dimension, desired accuracy, number of balls, and ball distribution.
2 Background

2.1 Deterministic methods for UnionVolume

The deterministic methods for UnionVolume all work by partitioning the volume into cells, calculating the volume of each cell analytically, then summing the volume of all the cells. Constructing partitions and analytically calculating the volumes can only be done in low dimensions. See Figure 2 for the partition types. Let the covered volume denote space in the union of the balls.

- **Covered cells—Computational Geometry.** Partition the covered volume into the power cells of each ball [13, 6], the “covered cells.” See Figure 2(a).

- **Occluded cells—Graphics.** Partition the covered volume into the part of each ball “above” all other balls, defined by any ordering. See Figure 2(b).

- **Depth cells—Set Topology.** Partition into regions covered by exactly $\delta$ balls. Sum the weighted volume of each ball, where the volume of an $\delta$-covered subregion is weighted by $1/\delta$. See Figure 2(c).

Recall that the power cell for a ball is a generalization of the Voronoi cell for a vertex. The power cell of a ball is the domain region “closer” to its center than another other ball center. Using standard Euclidian distance $e$ to define “closer” yields the Voronoi cell. Defining distance (a.k.a. power) by $e^2 - r^2$ for a ball of radius $r$ yields the power cells. The separating hyperplane for the power cells of two overlapping $d$-balls passes through their $(d - 2)$-sphere of intersection, known as the radical axis. I.e. in 2-d, the separator is the line through the two intersection points of the two circles. Two major advantages of this approach are that cells are convex, and in low dimensions the planes allow the decomposition of each cell into regions with analytic volumes. This method for 3-d was described two decades ago [13], and robust software with certified accuracy was published more recently, in 2011 [6].

In graphics, occlusion culling [8] is often done over triangles to determine the subset of object space that is visible to the camera. The main goal is to eliminate triangles that are completely occluded. This can be done in hardware. The visible sub-polygons of each remaining triangle can be sampled from by light rays on the fly. One may also deterministically construct the unoccluded polygons from the triangle overlaps. For UnionVolume of balls, this deterministic approach is unappealing even in low dimensions because the segments are curved.

For weighted depth sum, since each subregion is counted $\delta$ times, once for each ball covering it, the sum of weighted volumes gives the covered volume. For the deterministic algorithm, the complexity depends strongly on the depths of the overlaps. BF-ApproxUnion can be viewed as a fast randomized version of weighted depth sum.

Another set topology variation is the alternating depth sum. Consider the regions covered by at least $n$ balls. The alternating-sign sum over $n$ of the volume of these regions gives the
covered volume. The complexity of the deterministic version of weighted depth sum is the same as alternating depth sum. However, getting accurate answers with sampling appears to be inefficient, because one would be finding the difference between some large approximate values. So we did not experiment with this approach.

2.2 UnionVolume sampling approaches

Here we describe some alternative sampling methods for UnionVolume from the literature. We performed limited experiments against the first, BF-ApproxUnion.

BF-ApproxUnion

Bringmann and Friedrich’s [3] BF-ApproxUnion algorithm is a different style of point sampling algorithm. It selects a test ball and a test point in it, then samples balls until one covers the test point. The UnionVolume is the fraction of test balls to sample balls, times the sum of the volumes of all the balls. It can be understood as a re-organized estimate of weighted depth sum. It appears to achieve its theoretical efficiency by estimating both the depth and volume of overlap regions simultaneously.

BF-ApproxUnion provides an estimate of the volume $V_{est}$ that is within an $\epsilon$ factor of the true volume, $V_{est} \in [1 - \epsilon, 1 + \epsilon]V_{true}$ with probability $3/4$. It runs in $O(nd/\epsilon^2)$. This is ideal scaling in $n$ and $d$, since just representing the input is $O(nd)$. The convergence rate in $\epsilon$ is the same as standard Monte Carlo sampling, although the minimum achievable $\epsilon$ depends on the accuracy of three oracles. The algorithm requires oracles to compute the volume of an individual ball, perform Ball Sampling (Section 3.5), and test if a point is inside a ball. All these are $O(d)$ time. It also picks a random ball with probability proportional to its volume; with $O(n)$ preprocessing this can be done in $O(1)$ time [30, 4].

BF-ApproxUnion works by computing the sum of the volumes of the individual spheres, $V'$, then scaling this by an estimate of the amount of overlap. The algorithm has two loops. In the outer loop it samples a point $x$, by picking a ball (by volume) then a random point from it. In the inner loop it samples balls (by index) until one contains $x$. (The inner loop is a depth estimate.) The ratio of the total number of inner loop (ball) iterations $S$ to outer loop (point) iterations $X$ estimates the amount of overlap between the spheres: $C = S/(nX)$ and $V_{est} = CV'$. 

The algorithm determines a priori the number of inner iterations $S$ based on $\epsilon$: $S$ is $O(n/\epsilon^2)$. For small $\epsilon$, the number of samples per ball $S/n$ may exceed the number of balls, i.e. the runtime exceeds $n^2$; see Table 1. In this case it is more efficient to pre-compute the neighbor lists.

We implemented and tested two variants of BF-ApproxUnion. One is as published [3] without neighbor lists. The other is BF-ApproxUnion-Nbr with neighbor lists precomputed; this greatly improves its efficiency in practice for certain distributions. In the inner loop, we draw the ball from the neighbors instead of the entire population, and increase the iteration count ($S$) by $n/g$ instead
Table 1. BF-ApproxUnion relative error ($\varepsilon$) and required # samples per ball.

<table>
<thead>
<tr>
<th>$\varepsilon$</th>
<th>$S/n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9</td>
<td>40</td>
</tr>
<tr>
<td>.75</td>
<td>50</td>
</tr>
<tr>
<td>0.5</td>
<td>100</td>
</tr>
<tr>
<td>0.1</td>
<td>$1.8 \times 10^3$</td>
</tr>
<tr>
<td>0.01</td>
<td>$1.7 \times 10^5$</td>
</tr>
<tr>
<td>0.001</td>
<td>$1.7 \times 10^7$</td>
</tr>
</tbody>
</table>

The probability that the error is less than $\varepsilon$ is $3/4$. Here the oracles are assumed to be perfectly accurate; otherwise more samples are needed.

of 1, where $g$ the number of neighbors.

To our knowledge, and the knowledge of BF-ApproxUnion’s authors, we are the first to implement BF-ApproxUnion for balls. The algorithm is fairly simple, and our implementation is careful and efficient. Much of the speed depends on the random number generator, so for consistency we used the same generators and data structures in the implementations of all the BF-ApproxUnion and BalloonDarts variants. The only other implementation of BF-ApproxUnion we (and BF-ApproxUnion’s authors) are aware of is for boxes [18, 29], BF-HypervolumeApproximator. That implementation contains an inefficiency that spoils its performance: selecting a box weighted by volume is done by brute-force. In contrast, in our implementation we precompute the volumes of individual balls (objects) and select from them efficiently.

Other Estimation Methods

We describe some other methods for approximating UnionVolume that we did not compare against. One category of algorithms are based on nesting the object of interest between approximating shapes [14, 16]. Similar to standard point sampling, the fraction of random points between the nested shapes that are inside the object provides a multiplier to estimate its volume. Since the shapes are complex and may have high aspect ratio, uniform random points from a bounding box of sphere is often an inefficient distribution. Instead approximately-uniform random points are produced inside complex shapes by random walks: walking until hitting the boundary, then bouncing off in arbitrary direction. With enough bounces, the endpoint of the walk approaches uniformity. These methods are typically designed to estimate the volume of convex polytopes in high dimensions, say 10 to 100. These types of methods have long been attractive in theory, and practical implementations are just now appearing [14, 16].

McVol [28] and Voronoia [25] estimate the UnionVolume of 3-d balls for molecular applica-
tions. They use background grids (boxes) for the point sample locations, but the concepts are similar to Monte Carlo point sampling. Low discrepancy sequences are another useful way of generating sample points. They sample the domain more evenly than uniform-random points which exhibit clumping. Low discrepancy sequences can efficiently estimate volume properties [10].

“k-d Darts: Sampling by k-Dimensional Flat Searches” [11] reviews some history of higher dimensional sampling, and provides a general recipe for converting a point sampling algorithm to k-d dart sampling. k-d Darts includes experimental convergence rates for estimating the volume of a single ball (or ellipsoid) for uncertainty quantification. k-d darts are axis-aligned lines (hyperplanes) chosen uniformly from a cube. The paper also considers randomly oriented lines. These darts for boxes are analogous to our slats for spheres.
3 Volume estimation by line samples

The general outline of all our line sampling algorithms is to conceptually partitioning the Union-Volume into cells, estimate the volume of each cell, and add the estimates. Each cell volume is estimated using line samples; lines are trimmed to the subset of segments actually inside the cell (see Figure 3), and weighted appropriately. We first describe some top level parameters and choices; then cell estimation algorithms; the partition types; and finally note some implementation details of the common primitives.

3.1 Neighbor finding

In all of our algorithms, we trim each sample line by the balls defining the cell. These balls are called the neighbors of the ball. The neighbors are a subset of all of the balls. If the neighbor set is smaller than the entire set, then if we determine these neighbors once, before any sampling, this will speed up the trimming. In some cases, such as occlusion cells when a ball is completely covered by another single disk, we can immediately tell that its cell volume is zero. The brute force approach to neighbor finding takes $O(n^2)$ time, $O(n)$ for each ball. Without any guarantees about the position and size of the balls, all pairs of balls may overlap, so this is worst-case tight.

However, for certain types of configurations there are good proximity search speedups. But these tend to be more effective in low dimensions.

Our goal is to investigate the efficiency of the UnionVolume algorithms, independent of neighbor finding. We treat neighbor finding as a black box. For the algorithm variants that require or benefit from neighbor lists, we precompute them using brute-force and report that time separately.

3.2 Number of samples per ball

In our algorithms we estimate the volume of some subset $C$ of a ball. The general trend for Monte Carlo point sampling is for the relative error (variance) to decrease as the square root of the number of samples. The error in the estimate of the union is the sum of the absolute errors, not the relative errors. When deciding where to place the next sample, we should add it where the marginal error is the largest. In an optimal sampling strategy, the marginal errors will all be equal. This leads to

$$m_i = \frac{MV_i^{2/3}}{\sum_j V_j^{2/3}}$$

where $m_i$ is the number of samples in the $i$th ball, $M$ is the total number of samples, and $V_i$ is the cell’s volume. See Section 3.2 for details.

For certain line samplings over the cells arising from certain ball distributions, we expect that the error rate will not go down exactly as $\sqrt{m}$. The reason is that the different line sampling algorithms use lines distributed differently in the ball, and the location of the cell within the ball is different depending on the type of cells used and the ball distribution. For example, line spokes are a more accurate estimate of the center of the balls, and often power cells are concentrated near the ball center. We may either use $\sqrt{m}$ as a less-than-perfect proxy, or use observed variance trends to
inform our marginals.

**Importance sampling derivation**

The absolute error of the Union Volume is $E_a = \sum_i \sigma_i V_i$ where $\sigma_i$ is the relative error of the estimate of the $i$th volume, with the expectation $\sigma_i = a N_i^{-1/2}$ for some constant $a$. The marginal error rate for placing the next sample in the $i$th cell is $\partial E_a / \partial N_i = -a V_i N_i^{-3/2} / 2$. In an optimal sampling strategy we would consistently place the next sample where the marginal is largest; at optimality the marginals are equal (to $M$) so $-a V_i N_i^{-3/2} / 2 = M$ or $N_i \propto V_i^{2/3}$. Since the sum of the $N_i$ is the total number of samples $N$ we have $N_i / N = V_i^{2/3} / \sum_j V_j^{2/3}$.

Our initial estimate of $V_i$ is the ball’s volume. (A more sophisticated approach would be to modify this by the number of neighbors, or their pairwise overlap, etc.) For sampling each ball in parallel, using this fixed value is reasonable. For adaptive serial sampling, we may update our estimate of $V_i$ using the current estimate. (We begin the updates only after some number of initial samples, so we are not misguided by the large variance for small sample sizes).

### 3.3 Cell volume estimation

We now describe how to estimate the volume of a cell using line samples. In all cases a line sample is generated in the ball, then trimmed to retain just its subsegments actually inside the cell. Figure 3 illustrates some trimming examples. The trimmed lines estimate the volume of the cell, by some constant (an area) times the average weighted trimmed segment lengths. The untrimmed lines estimate the volume of the ball, using the same constant but untrimmed lengths. We also know the ball volume analytically. The intuitive argument for our best estimate is that sampled lines are random and will not be perfectly distributed in any given instance, and this will be measured by comparing the estimated and analytic ball volumes. We use this ratio to adjust the cell volume estimate. k-d Darts [11] provides a formal argument which we summarize here. The metaphor of integration provides the following volume estimates,

$$\text{vol}_{\text{samp}}(C) = \text{area ave}(\text{weighted}(l_{\text{trimmed}})),$$

$$\text{vol}_{\text{samp}}(B) = \text{area ave}(\text{weighted}(l_{\text{full}})).$$

Our best cell estimate, in the absence of any additional information, is given by

$$\frac{\text{vol}_{\text{est}}(C)}{\text{vol}_{\text{analytic}}(B)} = \frac{\text{ave}(\text{weighted}(l_{\text{trimmed}}))}{\text{ave}(\text{weighted}(l_{\text{full}}))}.$$

The weighted contribution of a segment depends on the distribution of the line samples, that is it depends on how we generated the lines. The characteristics of each distribution follows; see Figure 1. Slats sample the ball uniformly by volume. Spokes strongly oversample the center of
the ball, exponential in $d$. Chords slightly oversample the crust of the ball. Chords are an unbiased way of sampling the surface, but not the volume. In each case we compensate for the oversampling by weighting the line segments.

Figure 3. A line sample is trimmed by the disks or hyperplanes defining the cell. We retain only the segments inside the cell. Zero, one, or more segments may be retained depending on the cell and line sample type. Here for exposition we only show one ray of each spoke line sample.

Point sampling

As a common foundation, we first describe standard Monte Carlo point sampling over a cell; see Algorithm 1.

Algorithm 1 PointDarts(cell type $C$, ball $B$, neighbors $G$)

$m = 0 \{\text{number of samples}\}$
$h = 0 \{\text{number of samples hitting } C(B,G)\}$

repeat
  $m = m + 1$
  Pick a random point $p$ from $B$
  if $p \in C(B,G)$ then
    $f = 1$
    if $C$ is weighted depth then
      $f = f / \text{depth}(p \in B \cup G)$
    end if
    $h = h + f$
  end if
until desired accuracy or desired $m$

return $\text{vol}_{\text{est}}[C] = \text{vol}(B)h/m$
Slat sampling

See Algorithm 2. The simplest extension of point sampling to line sampling is when the lines form a uniform-random distribution over the volume. This is the case for vertical slats and random slats. Because of the uniformity, the weighted contribution of a segment is simply its length. A relative volume estimate is provided by the relative lengths of the trimmed to untrimmed segments, times the volume of the ball.

Algorithm 2 SlatDarts(cell type $C$, ball $B$, neighbors $G$)

\[
\begin{align*}
m &= 0 \{ \text{number of samples} \} \\
t &= 0 \{ \text{contribution to } \text{vol}(B), \text{untrimmed length} \} \\
h &= 0 \{ \text{contribution to } \text{vol}(C), \text{trimmed length} \} \\
\text{repeat} & \\
S &= \text{random slat} \{ \text{axis-aligned or random} \} \\
m &= m + 1 \\
t &= t + \text{length}(S \cap B) \\
\{s\} &= \text{trim}(S) \text{ into segments by } \mathcal{C}(C, B, G) \\
\text{for all } s \equiv [a, b] & \text{ do} \\
f &= b - a \{ \text{length of segment} \} \\
\text{if } C \text{ is weighted depth} & \text{ then} \\
f &= f / \text{depth}(s \in B \cup G) \\
\text{end if} \\
h &= h + f \\
\text{end for} \\
\text{until desired accuracy or desired } m \\
\text{return } \text{vol}_{\text{est}}[C] = \text{vol}(B)h/t
\]

To understand that slats are uniform sampling, let us construct the height function of the ball over the diameter. Vertical slats uniformly sample this height function, and thus the mean slat height tends to the average height. This is also true for random slats; they are just taken from a uniform distribution of diameters rather than a fixed one. Our intuition is that random slat estimates may have lower variance than vertical slats, because the sampling is not structured by the choice of diameter. Slats are much like line darts from a box in $k$-d Darts [11], but by selecting points from a diameter, we can generate just the lines that will hit the enclosed ball, and skip the factorial-in-$d$ lines in the box that miss the ball.

Spoke sampling

See Algorithm 3. In SpokeDarts sampling, we sample lines through the ball center uniformly by orientation. The contribution weight of a trimmed segment depends both on its length, and its distance from the center, as is well known in integration over polar coordinates. Every untrimmed segment is the same, a diameter, and so contributes the same amount to the ball estimate, and we do not need to track these individually.
**Algorithm 3 SpokeDarts** (cell type $C$, ball $B$, neighbors $G$)

$m = 0$ \{number of samples\}

$h = 0$ \{contribution to vol$(\mathcal{C})$, radius-weighted trimmed length\}

place origin at ball center, $c = \text{center}(B) = 0$ for segments.

repeat

\begin{itemize}
  \item $u =$ random direction
  \item $S =$ line $cu$ \{spoke\}
  \item $m = m + 1$
  \item $\{s\} =$ trim$(S)$ into segments by $c$ and $\mathcal{C}(C, B, G)$
\end{itemize}

for all $s \equiv [a, b]$ do

\begin{itemize}
  \item $f = b^d - a^d$ \{relative volume integral of segment\}
  \item if $C$ is weighted depth then
    \item $f = f / \text{depth}(s \in B \cup G)$
  \end{itemize}

end if

\begin{itemize}
  \item $h = h + f / 2$ \{because $cu$ is two unit rays from $c$\}
\end{itemize}

end for

until desired accuracy or desired $m$

constant $V_1 =$ vol(unit $d$-ball)

return $\text{vol_{est}}[\mathcal{C}] = V_1 h / m$

---

**Chord sampling**

See Algorithm 4 and Figure 4. In ChordDarts sampling, we sample random chords through the ball, uniformly by sphere area, by selecting the two endpoints of the chord uniformly at random from the surface of the ball. To derive the proper weighting, consider determining the volume of a ball by integrating chords. Fix $x$, and allow $y$ to vary uniformly over the surface area. Consider discrete integration by pyramids, before taking the limit as the size of the surface elements $\Delta A$ goes to zero. For a pyramid, the base has area $\Delta A$ and height equal to the chord length $l = |\vec{x} \vec{y}|$. The base is not normal to the height, but is tilted by angle $\theta$. The effective area of the base is $\Delta A \sin \theta$. The volume of the pyramid is $\Delta A \sin(\theta)l / d$ (exact in the limit). Using dot products, $\vec{x} \vec{y} \cdot \vec{c} \vec{x} = |\vec{x} \vec{y}||\vec{c} \vec{x}| \cos (\pi / 2 - \theta) = lr \sin \theta$. Thus an unbiased estimate of the ball volume is

$$V_{\text{samp}}(B) = \frac{\text{area}(B)}{rd} \text{ave}(\vec{x} \vec{y} \cdot \vec{c} \vec{x}).$$

The weighted contribution of the untrimmed chord to the volume is simply $g = \vec{x} \vec{y} \cdot \vec{c} \vec{x}$, where the $d$, $r$, and sphere area are constants that are the same for the trimmed chord and hence will cancel themselves in our best estimate.

We modify this for trimmed chords as follows. Suppose instead of extending to $l$, the chord and its pyramidal volume element just extends length $w$ from $x$. Then the volume of this smaller pyramid is $(w/l)^d$ times the volume of the larger pyramid, where one $(w/l)$ factor arises from its shorter height, and a $(w/l)^{d-1}$ factor arises from its smaller base area. In general, a pyramidal section $[v, w]$ has volume ratio $(w/l)^d - (v/l)^d$. But, $x$ and $y$ are interchangeable, since they were independently and identically chosen. Interchanging $x$ and $y$ does not affect the untrimmed pyramid.
volume, but does effect the trimmed pyramid. The most accurate estimate is to use the average of the two orderings of \( x \) and \( y \). That is, the weighted volume contribution is

\[
\frac{((w/l)^d - (v/l)^d)/2 + ((1-v/l)^d - (1-w/l)^d)/2}{(w^d - v^d + (1-v)^d - (1-w)^d)/(2l^d)}.
\]

(a) Volume integration by uniform-by-surface-area chords. (b) Weighting the surface area element by angle. (c) Trimmed pyramid volume.

**Figure 4.** How to weight chord samples for volume integration and estimation.

### 3.4 Cell types

Here we describe some efficiency, accuracy, and implementation consequences of the choice of cell type.

**CoveredCell and trimming**

Recall a covered cell is the intersection of the ball with its power cell; see Figure 3. Both are convex so a covered cell is convex. Thus after trimming, there will always be zero or one segment. Algorithmically, it is easy to successively trim one spoke by the separating hyperplane between \( B \) and a neighboring ball. This streamlines the serial implementation and makes for easier GPU parallelism, because no dynamic memory allocation or large static storage is needed. Note the center of the ball is not necessarily in the covered cell, however. For the other cells types, there may be more than one retained segment. For occluded cells, ordering the balls so the biggest ones are unoccluded tends to reduce the number of segments generated. Depth cells always produce the most segments. However, each of these segments does inform the estimate.

**OccludedCell ordering and neighbors**

Occlusion is defined by some ordering of the disks \( 1, 2, \ldots n \). Any arbitrary ordering works, but we have the freedom to select an ordering that improves efficiency. In our implementation, we just
Algorithm 4 ChordDarts(cell type $C$, ball $B$, neighbors $G$)

$m = 0$ \{number of samples\}
$t = 0$ \{contribution to vol($B$), weighted untrimmed length\}
$h = 0$ \{contribution to vol($C$), weighted trimmed length\}

repeat
  $x = \text{random point from surface}(B)$
  $y = \text{random point from surface}(B)$
  chord $S = xy$
  $m = m + 1$
  $g = S \cdot \hat{e}_y$ \{dot product for sin angle\}
  $t = t + g$
  $l = \|S\|$ \{untrimmed chord length\}
  \{$s\} = \text{trim}(S)$ into segments by $\mathcal{C}(C,B,G)$
  place segment origin at $x$: segments are positive distances from $x$
  for all $s \equiv [v,w]$ do
    \{treat $x$ and $y$ symmetrically\}
    $f = (w^d - v^d + (l - v)^d - (l - w)^d)/(2l^d)$
    if $C$ is weighted depth then
      $f = f/\text{depth}(s \in B \cup G)$
    end if
    $h = h + fg$
  end for
until desired accuracy or desired $m$
return $\text{vol}_{\text{est}}[C] = \text{vol}(B)h/t$
ordered balls by decreasing radius. The accuracy of the estimated union volume benefits the most from knowing the largest balls analytically and exactly.

Ordering them so that a maximum independent set comes first would be most efficient, since many of the disks would not be occluded at all and we could compute their volume analytically. A greedy heuristic that combines both goals follows. It computes a maximal independent set. Consider disks in order of decreasing radius (volume). In the first pass, accept a ball into the ordering, as not-occluded, if it does not overlap any previously accepted ball. In a second pass, add all the remaining balls in order, each of which will be at least partially occluded.

The number of neighbor disks is greatest for DepthCells, middle for CoveredCells, and least for OccludedCells. This is because for OccludedCells a disk does not need to consider any disk after it in the ordering.

3.5 Sampling primitives

Here we give details on how each type of line sample is generated, to show what primitives we require. Spokes are generated by picking one random point \( p \) on the sphere uniformly by surface area. Random chords are generated by picking two random points \( p \) and \( q \) on the sphere. Vertical slats are generated by picking a random point \( p \) on the horizontal \((d-1)\)-dimensional diameter ball \((x_1 = 0)\) uniformly by diameter area, then taking the vertical line \((x_1 = \text{free})\) through it. Random slats are the same as verticals slats, but for each sample we use an independent diameter with uniform-random orientation. This diameter is selected by picking a normal vector with uniform random orientation, by picking a point on the sphere. Thus, the necessary sampling primitives are picking a point in a \( d \) or \( d-1 \) dimensional ball, or from a \( d \) dimensional sphere.

Uniform point-in-ball and point-on-sphere

There are several standard algorithms for generating a uniform point in a ball. For efficiency in high dimensions, the best method is also the oldest [22]: generate each coordinate independently from a normal (Gaussian) distribution; then scale the vector of coordinates. If we normalize to the sphere radius \( r \), we get a uniform-random radial direction. To get a uniform random point, scale to radius \( R = ru^{1/d} \), where \( u \) is a uniform random variable from \([0, 1]\) and \( d \) is the dimension of the ball. This and several other techniques use a random normal variable. There are other approaches that avoid normal variables [17, 23].

Random normal variables

Algorithms and libraries for random normal variables in \( d \)-dimensions are well-known, but we describe some implementation options for further experimentation. The standard option is Box-Muller [2], which transforms a pair of uniform variables to a pair of normal variables. For our
setting, where the tails are unimportant and we seek computational efficiency, the poor-man’s method (Equation (26.48) [19]) of getting an approximate normal variable is appealing: add 12 uniform \([0, 1]\) variables and subtract 6. Since in our context we only care about the level sets being circular and not about a particular standard deviation, we may add \(m\) variables and subtract \(m/2\). One may also use the normal_distribution template from C++’s (random) library [9].
4 Complexity analysis

The amount of work for neighbor finding is $O(dn^2)$, which we report separately. The amount of work for the actual volume estimate is $O(dnmK(g))$. Here $d$ is the dimension; $n$ is the number of balls; and $m$ is the number of samples per ball. The leading $d$ arises simply because every point and line has $d$ coordinates so all primitives take at least $O(d)$. This dependence on $d$ is quite mild.

Here $K(g)$ is the complexity of constructing the sample segments in a cell, where $g$ is the number of neighbors per ball. Note $g = O(n)$. For CoveredCells, $K(g) = O(g)$ because there is just one segment; for the other cells, $K(g) = O(g \log g)$. In practice the differences in the number of segments and the constants involved in the different primitives are more significant than this $\log g$ factor.
5 Extensions

Parallel sampling. Each cell $C$ and each sample may be generated independently, for easy parallelism. This is in contrast to BF-ApproxUnion, where the published algorithm is inherently serial because a discrete decision to switch to a new ball and point is dependent on the result of a ball sample. It is possible to add some coarse-grain parallelism by running several instances of $BF - ApproxUnion$ in independent threads and combining their independent estimates.

Beyond line sampling, we explored hyperplane sampling in theory and experimented with plane-sampling. Plane-sampling did not improve the estimation efficiency in practice because of the complexity of the calculations required.

Other shapes, Booleans, and bounded domains. Ball UnionVolume is closely related to calculating the union of the volume of hyper rectangles, ellipses, or convex shapes. It is straightforward to perform spoke sampling from some point inside these other shapes. Enclosing these shapes inside a large ball and performing the other sampling types is also straightforward, but estimation efficiency will suffer because it depends on how closely the ball matches the shape, which tends to degrade as the dimension increases.

Trimming lines by these other shapes is easy. It is also easy to trim lines by domain boundaries, and by other Booleans between balls or other shapes. In particular we estimate the volume of one class of balls that are strictly outside two other classes of balls in POF-Darts [12]; this is trivial for OccludedCells. Another useful variation is to estimate the volume of high-dimensional Voronoi and power cells [21]. These might arise from Centroidal Voronoi Tesselations (CVT).

BF-ApproxUnion [3] is designed for general shapes. We are unaware of how to extend it to other Booleans. For bounded domains, it requires an oracle for the volume of each individual clipped sphere; we are unaware of how to do this if the sphere is clipped by multiple arbitrary hyperplanes.
6 Experimental efficiency

We performed a limited set of experiments. For these, we report average times $t$ over combinations of five parameters: number of balls $n$, dimension $d$, relative error $\eta$, ball distribution type $F$, and algorithm $A$.

**Time, $t$.** We measure the time taken to achieve a given level of accuracy. The time for generating the neighbor lists (NeighborLists) is broken out.

**Replicates.** We average the results of each experiment over 20 replicates using different random number seeds for the sampling.

**Sizes, $n$.** We use sets of $n$ balls, with $n \approx \{10^1, 10^2, 10^3, 10^4\}$.

**Dimensions, $d$.** Balls have dimension $d \in \{2, 3, 6, 10, 20\}$.

6.1 Relative error, $\eta$, proxy by $\sigma$

One would like to measure accuracy by the relative error $\eta$. However, except in very simple cases, we do not know the true solution. Instead we measure the standard deviation of the estimate. This is reasonable because we assert that in all cases the estimates are unbiased. This means that the estimates converge to the true union volume, and further the standard deviation of the estimates converges to the standard deviation of the error.

6.2 Ball distribution types $F$

Our test data sets are balls centered in a unit box. (Balls may extend outside this box and are not truncated.) We have two types of ball configurations. The first is generated by Relaxed MPS [11] following a Cone sizing function; see Figure 5. The second is Chain, overlapping balls generated by a random walk.

For Cone, the radius is $r(x) = \|x - c\|_2 - a$, where $c$ is the domain center and $a$ is some scalar constant. Note $a$ separates the balls into two non-overlapping sets: $\|x - c\|_2 > a$ and $\|x - c\|_2 < a$. We consider only $a = 0.01$.

We also consider Chain, a chain of balls chosen so that consecutive balls overlap, plus possibly others, to mimic a molecular application. We determine the set iteratively. The volume (not radius) of the next ball is chosen uniformly between 0 and 1. Its direction from the prior ball is chosen uniformly at random. The distance between centers is chosen uniformly between $|r_{prior} - r_{next}|$ and $|r_{prior} + r_{next}|$. That is, somewhere between inner and outer tangency. Centers may be outside the unit box.
Figure 5. Cone sizing function with 100 (left) and 1000 (right) balls. The additional balls are typically very small and concentrated near the $F = 0$ level set, the dark blue circle.

6.3 Algorithms, A

We tested the following algorithm variants using points and spokes.

- Weighted depth sum: DepthDarts with 0-spoke (points) and 1-spoke (line-spokes); BF-ApproxUnion, BF-ApproxUnion-Nbr.
- Covered cell: CoveredDarts with 0-spoke, 1-spoke, and 2-spoke.
- Occluded cell: OccludedDarts with 0-spoke, 1-spoke.

6.4 Experimental results

Scaling by accuracy

The first set of experiments is meant to address how the methods’ timings scale by the accuracy of the estimate. See Figure 6.4 and Figure 6.4. We see that all methods scale by $1/\sigma^2$, as expected with Monte Carlo based sampling, regardless of whether points or spokes are used or the dimension of the space. We used the Cone and 100 balls. The time for finding neighbors is negligible. 0-OccludedDarts in 2-d appears to be an outlier.
The next set of experiments is meant to demonstrate the linear time scaling by dimension of all of the methods. See Figure 7. We varied $d \in \{2, 3, 6, 10, 20\}$. We note that theory and experimental results from Figure 6.4 suggest that for all methods time scales as $1/\sigma^2$; i.e. $t\sigma^2$ is roughly invariant. All data points were chosen to have approximately the same sigma, $\sigma \approx 0.001$, but to gain greater accuracy, we plot $t\sigma^2 10^6$ rather than just $t$. We used the Chain model, with 10 balls. This has relatively few neighbors, and may have even fewer neighbors as the dimension increases. We expected time to be linear in $d$, but it appears that the random Chain data set for
\(d = 6\) was particularly easy. 0-OccludedDarts in \(d = 20\) appears to be an outlier. A static chain of centers might reduce the noise in the plots.

**Scaling by number of balls** \(n\)

**(a)** Scaling by \(n\), linear scale.

**(b)** Closeup of scaling by number of balls.

**Figure 8.** Scaling by number of balls, \(n\). The different linear slopes come from different constant factors in the \(O(n)\) scaling.

**(a)** Scaled time / \(n\).

**(b)** Closeup of scaled time / \(n\).

**Figure 9.** Scaling of time / \(n\) by number of balls, \(n\). All curves are expected to be flat. Lower curves have smaller constants in their \(O(n)\) scaling.
This set of experiments is meant to demonstrate the linear scaling in \( n \), the number of balls, for all the methods. We have \( n = \{10^2, 10^3, 10^4\} \). The sizing function is the Cone, in two dimensions. Since our brute-force implementation of neighbor finding is \( O(n^2) \), we remove the neighbor-find time. In the \( n = 10^4 \) case, the neighbor find time was roughly the same as the volume estimation time (for all Balloon Darts methods). We choose \( \sigma \) and scale time as in Section 6.4. In Figure 8 we plot scaled times on a linear scale, and we expect to find a constant slope. In Figure 9 we plot the scaled times divided by \( n \), and expect to find a zero slope. The slight downward slope is expected from the algorithmic steps that do not depend on \( n \). While the data are noisy and there are a few outliers, the expected trends roughly hold.
7 Proposed experiments

We propose additional experiments to better capture the trends and provide more general conclusions. We have 5 sample types: points, vertical slats (⊥-slats), random slats (o-slats), chords and spokes. We have 3 cell types: covered, occluded, and depth. And we have 2 variations: neighbor-aware and unknown-neighbors. In addition we seek to compare to BF-ApproxUnion [3].

For each of these algorithms we have the total number of samples $M$ as input and the variance $\sigma$ and time $T$ as outputs. We wish to test these (30+) algorithm variations over different distribution types. In particular, we are interested in scaling by $n$ number of balls, $N$ number of neighbors, how much the neighbors overlap, whether ball radii $r$ are uniform or varying, and by dimension $d$.

The number of combinations, and the trends we wish to demonstrate, at first appear staggering. The following experiments, in sequence, are designed to simplify this plethora of combinations. The early experiments should establish relationships, which will reduce the number of experiments needed in later steps.

An analytic answer is rarely known, but we can calculate it for balls with depth at most 2. We propose using such examples to demonstrate that our methods provide an unbiased estimate. Once established, then, for other examples, the mean estimate tends to the true mean, so we may use variance as a proxy for error. However, we must still perform many replicates of an experiment in order to get a good estimate of that variance.

7.1 Relationship of $M$, $\sigma$, and $T$

We propose verifying $\sigma = O(1/M^2)$, as expected for Monte Carlo sampling, where $M$ is the total number of samples. We suggest a test distribution of ten balls on a line that pairwise overlap, with radius 1 and center-distance 1. For any dimension, is possible to calculate the analytic answer. To verify the $\sigma$ trend, we suggest that it is sufficient to perform experiments in dimension 7. To demonstrate linear scaling by $d$ for the primitives, we also suggest it is sufficient to use the 10 balls example, but vary $d = 1 \ldots 100$. We suggest that other (implicit) effects of dimension are best captured by considering different distribution types.

Volume of 10 overlapping balls in a line

The analytic UnionVolume of 10 balls with center distance 1 and radius 1 is calculable as follows. The analytic expression for the volume of a hyperspherical cap is [20]

$$V_d^\text{cap}(r, \phi) = \frac{1}{2} V_d(r) I_{\sin^2 \phi} \left( \frac{d + 1}{2} , \frac{1}{2} \right);$$

where $I_x(a, b)$ is the regularized incomplete Beta function; $\phi \in [0, \pi/2]$ is the cone angle of the cap; and $V_d(r) = \frac{\pi^{d/2}}{\Gamma(d/2 + 1)} r^d$ is the volume of the complete $d$-sphere of radius $r$ in terms of the gamma
function $\Gamma$.

For 10 spheres, there are 9 overlaps pairs, so the volumes of 18 caps must be subtracted from the volume of 10 spheres. Each cap has angle $\phi = \pi/3 = 60^\circ$, so $\sin^2 \phi = 3/4$, independent of dimension; and $r = 1$. Thus

$$V_{d \text{ cap}}(1, \pi/3) = \frac{1}{2} V_d(1) I_{3/4} \left( \frac{d+1}{2}, \frac{1}{2} \right);$$

and

$$V_{\text{union}} = V_d(1) \left( 10 - 9 I_{3/4} \left( \frac{d+1}{2}, \frac{1}{2} \right) \right);$$

where

$$V_d(1) = \frac{\pi^{d/2}}{\Gamma(d/2 + 1)}.$$

For clarity, when we report the volume for the 10 balls, we should report it as a multiple of the volume of a single ball. That way the numbers do not vanish when the dimension goes up, and are always between 6 and 10.

### 7.2 Scaling by $N$

For a fixed dimension, we propose reporting how $\sigma$ varies by the number of samples. For each dimension, record the number of samples $N$ vs. $\sigma^2$, using the known exact volume for computing $\sigma$, not the variance, for the 10 balls example. We propose to plot $N$ vs. $\sigma^2$. Because we are using lines, for the same number of samples we speculate that we will have smaller relative error in low dimensions than in high dimensions for a fixed number of samples.

For each fixed dimension, $d \in \{2, 3, 6, 10, 20\}$, we propose to plot $\sqrt{N}$ vs. $\sigma^2$ as follows. First, scale so that we expect horizontal lines in the plots: that is, plot $N$ vs. $\sigma^2 \sqrt{N}$. All the lines may be placed in a single graphic for conciseness, with offsets for the lines so they are not on top of each other. Once the trends are observed, we can reduce each dimension to a single number, the slope of its line. That is,

$$\sigma^2 \sqrt{N} = a_2 \text{ in } 2\text{-d}$$

$$\sigma^2 \sqrt{N} = a_3 \text{ in } 3\text{-d}$$

... We may then plot or make a table of $a_d$ vs. $d$ to report its trend. We propose to determine how many samples are needed in each dimension to get $\sigma = 0.01$, or some other value that has feasible runtime, for use in the next study.
7.3 Scaling by dimension

We propose experiments for a fixed number of samples, and fixed relative error. For a fixed relative error, we may use the $a_i$ from Section 7.2. For both, plot $d$ vs. $t$.

(0b) To measure the effect that spoke sampling concentrates lines at ball centers, and therefore estimates the cell volume near the center more accurately, we propose rerunning the 10-balls experiment where the distance between centers is larger, 1.9, so the balls just overlap a little bit at their extremes. For all methods, we expect a smaller relative error for small sample size, because the volume of the union is closer to the sum of volumes. However, when comparing to chords or slats, and we expect the relative error for spokes to decrease less quickly than for the center-distance-1 balls.

7.4 Other distribution types

We suggest experiments to determine scaling by $d$ and $N$ for other distributions. Which distributions are relevant depends on the application. The following distributions may be of independent interest as well.

Box

The box distribution has one ball at each corner of a unit box. Balls have dimension $d \in \{2, 3, 6, 10, 20\}$ and radius $r = \{\sqrt{d}, \sqrt{d-1}, \sqrt{d-2}, \ldots 1\}/1.5$. For the largest radius all balls pairwise overlap. For each decrement in the series, the dimension of the largest hypercubes whose balls at its far corners overlap decrements. We do not know how to easily obtain the analytic UnionVolume. An analytic variant that might be of some use is to add one more ball that contains all the others, but to structure the experiments so that the algorithms cannot exploit it to get a perfect answer.

White noise

We propose studying scaling by $n$ (samples per ball), and the effect of varying $r$ per ball. We propose an experiment to test the strategies of using a different number of samples for each ball according to its volume.

The following distribution may be effective. Pick balls with random centers uniformly inside a sphere of radius 1. Pick a random radius between 0 and 1/2, such that the volume is uniformly distributed. Let balls have dimension $d \in \{2, 3, 6, 10, 20\}$. Consider sets of $n$ balls, with $n \in \{10^1, 10^2, 10^3, 10^4\}$. For each increase in $n$, just enrich the prior set.

As with the box distributions, an analytic variant that might be of some use is to add one more
ball that contains all the others, and structure the algorithms so this is not exploited.

Spheres around a point

For this experiment, we propose to pick ball centers uniformly by volume from a radius 1 sphere around the origin. For each ball center, we set the radius to be 1.1 times its distance from the origin, so that the origin is heavily overlapped. This experiment may illuminate scaling by $N$ with $n$ fixed in a different way than the box. Analytic variant includes a sphere of radius 2.2 containing everything.
8 Conclusion

Line sampling (1-d darts) improved the efficiency of all the UnionVolume algorithms over point sampling (0-d darts) in our limited tests. BF-ApproxUnion and BF-ApproxUnion-Nbr have good theoretical scaling, and their competitiveness in practice may depend on the setting. The performance scaling of CoveredDarts, OccludedDarts, and DepthDarts were roughly the same. Which one is the most efficient probably depends on the configuration of balls being probed. We speculate that different ball arrangements, differentiated by the distribution of overlaps in terms of volume and degree, will benefit the most from patterns of line samples that preferentially capture those overlaps. We suggest more extensive empirical studies for future work. Some aspects of our implementation are quadratic in the number of neighbors, and we could improve that using known techniques.

The implementation of the 1-d dart methods were all fairly easy, although OccludedDarts was the easiest. BF-ApproxUnion is also easy to implement. Line sampling is more versatile than BF-ApproxUnion in terms of supporting Boolean operations besides union. BF-ApproxUnion is more versatile in terms of more easily supporting more complex shapes than balls.
References

[1] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust, unions of balls, and
Curves and Surfaces.


45th ACM Symposium on Theory of Computing (STOC-13), page accepted, Palo Alto, USA,
2013. ACM.

[5] Frédéric Cazals and Tom Dreyfus. Multi-scale geometric modeling of ambiguous shapes
with tolerated balls and compoundly weighted $\alpha$-shapes. Computer Graphics Forum,

2011.

Geometry (SoCG’08).

New York, NY, USA, 1997. ACM.

normal_distribution/, 2013.


and John D. Owens. $k$-d darts: Sampling by $k$-dimensional flat searches. ACM Trans. Graph.,

[12] Mohamed Salah Ebeida, Scott A Mitchell, Laura Painton Swiler, Vicente Jose Romero, and
Ahmad A Rushdi. POF-Darts: Geometric adaptive sampling for probability of failure. Reli-

annual Symposium on Computational Geometry, SCG ’93, pages 218–231, New York, NY,
USA, 1993. ACM.


DISTRIBUTION:

1  MS 0359  D. Chavez, LDRD Office, 1911
1  MS 0899  Technical Library, 9536 (electronic copy)