k-d Darts: Sampling by k-Dimensional Flat Searches

MOHAMED S. EBEIDA
Sandia National Laboratories
ANJUL PATNEY
University of California, Davis
SCOTT A. MITCHELL and KEITH R. DALBEY
Sandia National Laboratories
and
ANDREW A. DAVIDSON and JOHN D. OWENS
University of California, Davis

We formalize sampling a function using k-d darts. A k-d dart is a set of independent, mutually orthogonal, k-dimensional hyperplanes called k-d flats. A dart has d choose k flats, aligned with the coordinate axes for efficiency. We show k-d darts are useful for exploring a function's properties, such as estimating its integral, or finding an exemplar above a threshold. We describe a recipe for converting some algorithms from point sampling to k-d dart sampling, if the function can be evaluated along a k-d flat.

We demonstrate that k-d darts are more efficient than point-wise samples in high dimensions, depending on the characteristics of the domain: for example, the subregion of interest has small volume and evaluating the function along a flat is not too expensive. We present three concrete applications using line darts (1-d darts): relaxed maximal Poisson-disk sampling, high-quality rasterization of depth-of-field blur, and estimation of the probability of failure from a response surface for uncertainty quantification. Line darts achieve the same output fidelity as point sampling in less time. For Poisson-disk sampling, we use less memory, enabling the generation of larger point distributions in higher dimensions. Higher-dimensional darts provide greater accuracy for a particular volume estimation problem.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Sampling, dimension, line search, thin regions, rendering, depth-of-field, Poisson-disk sampling, Monte Carlo integration, Latin hypercube sampling, uncertainty quantification

ACM Reference Format:

Mohamed S. Ebeida, Anjul Patney, Scott A. Mitchell, Keith R. Dalbey, Andrew A. Davidson, and John D. Owens. 2014. k-d darts: sampling by k-dimensional flat searches. ACM Trans. Graph. 33, 1, Article 3 (January 2014), 16 pages.
DOI: http://dx.doi.org/10.1145/2522528

1. INTRODUCTION

In many applications we are interested in estimating some global property of a function because it is difficult to calculate that property exactly. *Sampling* is the process of randomly selecting *samples*, subsets of a domain. The function is evaluated at these subsets, and the global property is estimated based on those values.

In typical sampling processes, the samples are points. However, a recurring challenge is to deal efficiently with the case that the interesting part of the domain is very small compared to the entire domain. For example, suppose we have a function over a domain, and we are interested in estimating the volume of the subdomain where the function is negative. If this subdomain has a very small volume, only a correspondingly very small fraction of uniform sample points will land in it; see Figure 1. Consequently, point sampling will require a large number of samples to get any estimate, and will be inefficient since most samples will not contribute to the estimate.

We propose the k-d dart to address this problem. One key idea is that rather than evaluate the function at one single point at a time, we evaluate it with a higher-dimensional sample. Specifically, we

M. S. Ebeida, S. A. Mitchell, and K. R. Dalbey thank Sandia's Validation and Verification program, and Computer Science Research Institute for supporting this work. A. Patney, A. A. Davidson, and J. D. Owens thank the National Science Foundation (grant no. CCF-1017399), Sandia LDRD award no. 13-0144, UC Lab Fees Research Program award no. 12-LR-238449, NVIDIA and Intel Graduate Fellowships, and the Intel Science and Technology Center for Visual Computing for supporting this work. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Authors' addresses: M. S. Ebeida, Sandia National Laboratories, 1515 Eubank, Albuquerque, NM 87123; A. Patney, University of California, Davis, 1 Shields Ave, Davis, CA 95616; S. A. Mitchell (corresponding author) and K. R. Dalbey, Sandia National Laboratories, 1515 Eubank, Albuquerque, NM 87123; email: samitch@sandia.gov; A. A. Davidson and J. D. Owens, University of California, Davis, 1 Shields Ave, Davis, CA 95616.

© 2014 ACM 0730-0301/2014/01-ART3 $15.00
DOI: http://dx.doi.org/10.1145/2522528

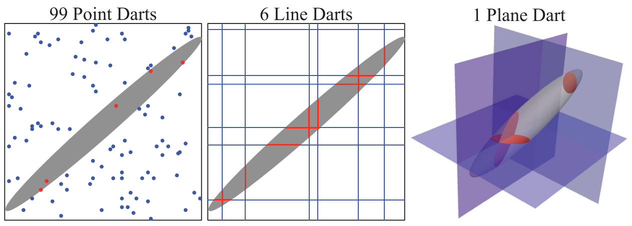


Fig. 1. Sampling long and thin subregions (gray) using points (left), lines (center), and planes (right). Point samples may be cheap to generate and evaluate, but they contribute nothing to the final result if they miss the region of interest. Misses (blue) are frequent for regions with a small volume. Samples of higher dimensions, or k-d darts, often intersect (red) the region of interest, especially if the region is long and thin. A k-d dart's greater expense is offset by it providing more information.

evaluate the function along a set of higher-dimensional flats (i.e., lines, planes ... hyperplanes). The second key idea is to use a set of mutually orthogonal flats, aligned with the coordinate axes; a k-d dart denotes this set of flats. Randomly oriented flats have been considered before, but orthogonal flats are more efficient in many settings. Moreover, if our subregion of interest is long and thin, there is a better chance that a good fraction of our flats will hit it compared to randomly oriented flats. Each fixed-coordinate value of each flat is chosen independently. This helps ensure that darts are *unbiased,* meaning that the expected mean estimate is equal to the true value. An important case of flats are one-dimensional lines. Using our previous example, we may find the points along the line where the function value is zero, then partition the line into segments where the function value f is strictly positive or negative, and finally estimate the volume where $f < 0$ from the negative-interval lengths. While these samples are more expensive to compute, they are more powerful; depending on the function they can generate better results for the same amount of effort.

A simple example that demonstrates this concept is estimating the volume of a unit ball by sampling from its bounding box: $f = -1$ inside the ball and 0 outside it, and we seek an estimate of $\int_{f<0} 1$. Figure 2 shows error versus time as the sample size increases. We sampled using k-d flats of dimension $k = 0, 1, 2$. For a point sample, we checked if the point was inside the ball. For higher dimensions, we calculated the fraction of the flat inside the ball; see Figure 3. We performed both Monte Carlo (MC) and Latin Hypercube Sampling (LHS). For each sample size we ran 100 experiments and calculated the error in the volume estimate. Plane samples consumed less CPU time than point samples for the same RMS error. For MC sampling the payoff was about a factor of 5, and for LHS sampling the payoff was 3 to 8 orders of magnitude! The reasons behind these gains are the following.

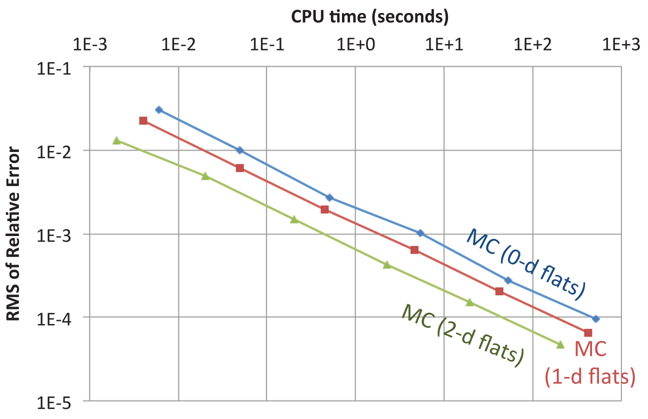
—Evaluating f along k-d flats is cheap; in this case we exploited the analytic function of the ball.

—A k-d flat gives more information as k increases.

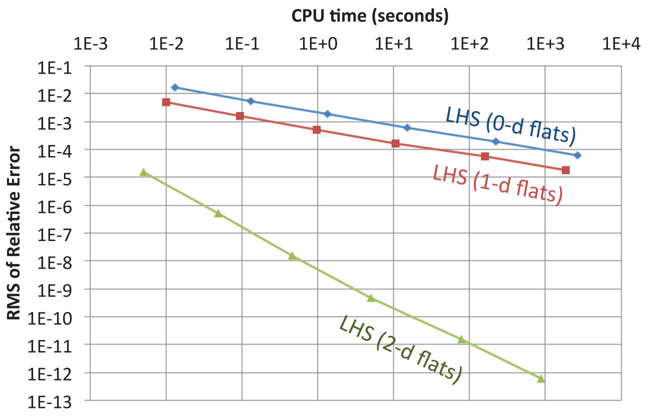
—A flat is cheap to generate. Each k-d flat requires $d - k$ random numbers; here $d = 3$.

—$(d - 1)$-dimensional flats distributed in LHS fashion boosted the convergence rate from $O(\frac{1}{\sqrt{n}})$ to $O(\frac{1}{n})$.

In general, evaluating the integration function along a k-d flat costs more than at a single point. However, for many problems, this extra cost is offset by the superior capability of a k-d flat to capture narrow regions. For instance, consider Figure 4(a), where a line flat perpendicular to that narrow region of interest will capture

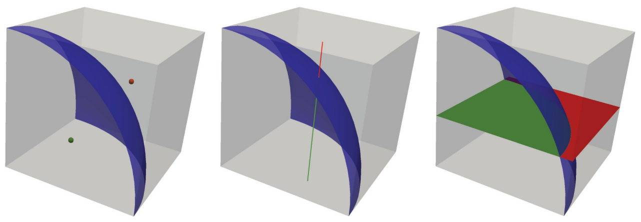


(a) Monte Carlo sampling (MC)



(b) Latin Hypercube Sampling (LHS)

Fig. 2. Estimating the volume of a ball using random sampling via k-d flats, $k = 0, 1, 2$. For each sample size, we performed 100 experiments and calculated the RMS error. The reported CPU time is the total time consumed by these experiments. For MC sampling (a) plane samples consumed an order of magnitude less time to achieve the same error as point samples. The savings were even more for LHS (b).



(a) 0-d flats (b) 1-d flat (c) 2-d flat

Fig. 3. k-d flats used to estimate the volume of a ball. The fraction of the subflats inside the ball estimates the function average.

it regardless of its thickness. On the other hand, the probability of a point sample landing in the region approaches zero as the thickness decreases. Suppose the region $f < 0$ is thin in k directions and fat in the others, and we use k-dimensional flats. Because the flats deterministically cycle through all coordinate directions, some fraction of them will be roughly orthogonal to the subregion's fat directions even if these fat directions are not axis aligned. These flats are likely to intersect the subregion.

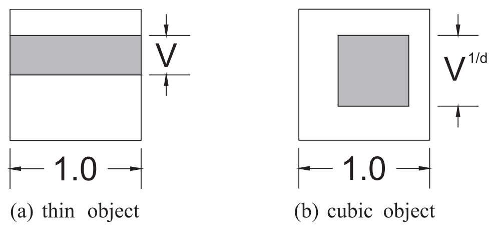


Fig. 4. Extreme subregion shapes. In general, a k-d flat has a better chance to intersect a region of interest as k increases. For a given region volume, the advantage is higher for stretched regions than for square ones.

The purpose of this article is to formalize and demonstrate the k-d dart approach. In Section 4, we show three practical applications: completing a relaxed maximal Poisson-disk sampling, in dimensions 4–30; rendering depth-of-field blur in four dimensions; and estimating the probability of failure from a response surface for uncertainty quantification, where the probability is small, for example, 1e-5, and the dimension is large, for instance, 15. In each of these three applications the space is of moderate dimension, and line darts are particularly effective. The experiments in Section 5 verify the accuracy of higher-dimensional darts, using a volume estimation application.

2. PREVIOUS WORK

Our work generalizes sampling. There are many patterns for generating samples. For graphics, maximal Poisson-disk sampling (Section 2.1) is common. The traditional focus is dimensions 2 and 3, but our interests extend beyond. Rendering applications use sampling in many forms (Section 2.3), often with high dimension. The field of uncertainty quantification (nongraphics, Section 2.4) attempts to quantify the range of a computation, typically by performing the computation many times with a well-distributed selection of input parameters. This is important in computational science for predictions and reliability.

Our k-d dart is a particular type of high-dimensional sampling. Line sampling has been explored in a variety of graphics and nongraphics contexts. Unfortunately much of this work is isolated and considers fixed-dimensional domains. We hope to help provide a unified view of these approaches.

Lines appeared early in the history of Monte Carlo sampling. “Buffon’s needle problem” was published in 1777. It is a problem in the field of geometric probability. It considers the number of intersections of randomly oriented short line segments (needles) with axis-aligned, regularly spaced infinite lines. The solution gives an estimate of the value of π. There are approaches using integral geometry [Ramaley 1969] and MC experiments [Hall 1873]. There are some striking similarities between the needle problem and estimating volume by line darts. Both use finite objects, varying orientations, and infinite probes. But which of these are known, and which are measured or estimated, are different. Also both use uniform-random and uniform-deterministic distributions, but for different objects. For example, darts are sets of orthogonal flats whose fixed-dimension indices are deterministic and evenly spaced, but their geometric position is uniform random. In Buffon’s problem the geometric position of the rule lines is deterministic, and the position and angle of the needles are uniform random.

In neutron transport physics simulations, a class of Monte Carlo algorithms known as “track length estimators” essentially performs Monte Carlo estimation using line segments [Spanier 1966]. This is in contrast to the “collision estimators” class that estimates using point samples. In graphics, these collision estimators correspond to standard volumetric photon mapping estimation using photon scattering locations [Jensen and Christensen 1998]. Track length estimators using line samples correspond to photon beams, where the estimation uses random-walk path segments [Jarosz et al. 2011]. In surface reconstruction and CAD modeling, we can count the intersections of unoriented line samples with the surface, then make use of the integral geometry Cauchy-Crofton formula to estimate integration quantities such as surface area or enclosed volume [Liu et al. 2006]. To get the right estimate with low variance, it is crucial to select the sample lines using the right probability model. For example, models for bundles of uniformly spaced parallel lines, reminiscent of both k-d darts and the regularly spaced lines in Buffon’s needle problem; models for chordal lines; and pseudo-random and other numerical sequences have all been studied in the context of sampling surfaces [Rovira et al. 2005].

2.1 Relaxed Maximal Poisson-Disk Sampling

Maximal Poisson-disk Sampling (MPS) is a popular graphics technique to distribute a set of points in a domain. The region of interest is dynamic: the remaining uncovered areas are called “voids.” A candidate sample that lands in a void is a “hit” and is accepted, otherwise it is a “miss.” The points are random and have a blue-noise spectrum, which is well suited to the human visual system and helps avoid visual artifacts. The points have a minimum distance between them, r_f, which helps to use the point budget efficiently. We denote the maximum distance from a domain point to its nearest sample by r_c. In a maximal sample, the r_f disks around the points overlap to cover the whole domain, leaving no room to add another point, and $r_c \le r_f$. Otherwise, maximality is *relaxed*, and we measure how far the geometry is from maximality by the distribution aspect ratio $\beta = r_c/r_f \ge 1$. Point sets with a meaningful upper bound on β are separated yet dense, also known as “well spaced.”

MPS algorithms abound, and often achieving maximality is the most challenging part. In some applications maximality is not required [Lagae and Dutré 2008]. The acceptable relaxation of maximality depends on the application. For example, in Voronoi mesh generation [Ebeida and Mitchell 2011], the cells have an aspect ratio bound that varies smoothly with the relaxation, 2β. Some methods sacrifice maximality to terminate more quickly, but explicit statements about the achieved β are rare.

Many methods use some form of a background grid for tracking the voids, and point location and proximity queries [Ebeida et al. 2011; Jones and Karger 2011; Wei 2008]. The grid may be refined, as in a quadtree [Gamito and Maddock 2009; White et al. 2007]. This can be made efficient in dimensions up to about 5 [Ebeida et al. 2012]. However, even in dimensions below 5, refinement methods can run out of memory as the sample size increases. Memory problems are exacerbated on a GPU.

Uncertainty quantification motivates MPS sampling in higher dimensions, for example, 10–30. No MPS methods in the literature scale to these dimensions due to the so-called “curse of dimensionality.” Classical dart throwing [Cook 1986; Dippé and Wold 1985] is not strongly dependent on dimension, but as the number of accepted samples increases, the runtime for the next sample becomes prohibitive and the algorithm must terminate well before maximality.

Consider sampling a unit box with disk radius r in dimension d. Some issues are fundamental to the problem, independent of any specific algorithm or application. The size of a maximal sampling n is indeterminate, but its lower and upper bounds grow exponentially as $O(1/r)^{O(d)}$. The kissing number, the number of disks that can touch another disk, grows exponentially in d. (The geometry literature discusses these issues extensively. The densest packings and largest kissing numbers by dimension are summarized in Nebe and Sloane [2012].)

The curses of dimensionality for grid-based methods go beyond those unavoidable issues.

(1) The base grid size grows exponentially and faster than the output point set.
(2) A grid cell is refined into 2^d subcells.
(3) The number of nearby cells that might contain a conflicting sample grows faster than the kissing number.
(4) The ratio of misses/hits grows exponentially with the dimension [Ebeida et al. 2012].

Item 1 arises because the size of the base grid is usually chosen such that each cell can accommodate at most one point: base squares have diagonal length r and edge length $r/\sqrt{d}$. Worse, at maximality, the number of empty base cells grows exponentially with the dimension. These empty cells increase the time, and especially the memory requirements, to prohibitive levels. One possible solution is to choose a base grid level with *edge* length $2r$, so that every square must contain one point. However, due to Item 2, this approach just defers the problem until cells are required to be refined a couple of times to represent voids. Because of the kissing number, representing voids through geometric constructions does not appear to be a viable solution in high dimensions.

Some approximate methods [Wei 2008] put an upper bound on the number of misses per cell, which partly addresses Item 4. The drawback is that the sampling is not maximal. How far it is from maximality has not been analyzed, but volume arguments suggest that for a fixed box size, the number of allowed misses must grow exponentially in d to bound the linear distance between an uncovered point and a sample's disk.

While some of these issues affect runtime, the real curse is the memory requirements for quadtrees. Simple MPS [Ebeida et al. 2012] is the quadtree method with the best memory scaling by dimension. It seems unlikely to extend to even $d = 10$ in the near future. Simple MPS uses a flat quadtree of same-sized squares, periodically refining all remaining squares uniformly. We compare our results to Simple MPS in Section 4.1.4.

2.2 Motivation for High-Dimensional Point Sets

In the design of computer experiments, we generate points in parameter space, then evaluate a function at the sample points. We often build a surrogate model based on those points' values, for example, Kriging models. The dimension is equal to the number of parameters, so can be very high. The time it takes to generate the points is often very small compared to evaluating the function, so it is worthwhile to spend the time to find a set of points that span the space efficiently. Well-spaced points use the point budget efficiently, and provide bounds on the condition number and interpolation error.

For some applications, if the function is not too expensive, it may make sense to sample the function directly using k-d darts. For example, if the integral of a function over a flat is available analytically, then sampling it directly using k-d darts would be very efficient. Otherwise, k-d darts can provide well-spaced points (see Section 4.1); we can build a surrogate model over those points; then k-d darts can integrate the *surrogate model* analytically.

Another application where well-spaced point sets are required is finite element simulation, where we need a computational mesh of the points.

2.3 Rendering

High-quality rendering is an important application of multidimensional sampling. Photorealistic effects like motion blur, depth-of-field (defocus), and soft shadows can be expressed as integrals over multiple dimensions. Classical techniques often employ stochastic point sampling to estimate these integrals. Noise-free rendering using point sampling can require a large number of samples, which can be extremely expensive for complicated scenes.

Thus a long history of research has targeted choosing samples wisely. The goal of Mitchell's classic antialiasing paper [1987] is to reduce sample density while producing a high-quality image. Metropolis light transport [Veach and Guibas 1997] "performs especially well on problems that are usually considered difficult" by using mutations to preferentially (but in an unbiased way) sample light paths in interesting regions of the path space.

Besides choosing samples carefully, another approach toward the same goal is to reduce the number of required samples through techniques such as sample reuse and/or caching. For example, a notable recent advance in this area, by Lehtinen et al. [2011], specifically notes "a clear need for methods that maximize the image quality obtainable from a given set of samples" and exploits anisotropy in the temporal light field to reuse samples between pixels. Our work has a similar goal—making the best use of a limited number of samples—but instead reduces the number of necessary samples by increasing their dimensionality.

k-d darts formalize a general way of multidimensional sampling. Several early graphics applications use multidimensional samples in limited and specific scenarios. The OpenGL accumulation buffer [Haeberli and Akeley 1990] uses a form of k-d darts for motion blur: each output pixel is an aggregate of several input pixels, each of which may span a 2-d region (pixel area) for a constant shutter time. Essentially, the accumulation buffer samples 2-d x-y images in a 3-d x-y-time space. Max [1990] uses a scan-line visible surface algorithm that generates line samples, describing how to use the information in these samples to create antialiased images.

Recent research has also shown promise in rendering high-quality motion blur using multidimensional samples. Gribel et al. [2010, 2011] present the use of line samples (our "1-d flats"). In their 3-d domain (x, y, t) they fix x, y and perform a 1-d flat in the t domain[1]. They also extend their implementation to render motion-correct ambient occlusion.

Recently, line samples have proven useful in the representation of light. Sun et al. [2010] represent lighting and viewing rays directly in a 6-d Plücker space, which allows an efficient formulation of finding nearby lighting rays. This, in turn, allows accurate, fast rendering of large scenes with single scattering even in the presence of occlusions and specular bounces. The previously mentioned work of Jarosz et al. [2011] concentrates on better representations for light paths in photon tracing for the purposes of rendering participating media in light interaction; previous methods had used photon particles. Instead, they represent and store full light paths (samples), resulting in a more compact and expressive lighting representation with corresponding performance benefits.

Jones and Perry [2000] experimented with using analytical line sampling for antialiased polygon rendering. They shoot single-dimensional darts across a pixel's surface, analytically compute triangle coverage for each, and then average to obtain pixel colors.

[1]In their 2011 paper, Gribel et al. use 2-d darts but call them line samples because they are lines in x-y space.

Our article generalizes the idea of multidimensional darts for sampling, and it is this generalization that helps us design renderers that use k-d darts in different configurations. For rendering depth-of-field effects with added antialiasing, we present a generalized configuration using a full 1-d dart in Section 4.2. That is, we use a set of orthogonal flats rather than just a single flat as in prior work. We compute high-quality depth-of-field images efficiently.

While we demonstrate just one configuration of k-d darts, several different strategies are possible for depth-of-field and other effects. In general, k-d darts offer a sampling process that converges faster and has lower noise than point sampling; the use of k-d darts offers the potential benefit of faster convergence but must be weighed against the higher complexity per k-d dart.

2.4 Monte Carlo Sampling

Random sampling is one of the oldest [Hall 1873] and most robust methods for uncertainty quantification. The principal use of Monte Carlo (MC) sampling is to approximate a high-dimensional integral with a sample mean. The primary drawback of MC is the slow rate at which the sample mean converges to its true value. The standard error in the *computed* mean for n samples is

$$\sigma_{\text{err}} = \frac{\sigma}{\sqrt{n}}. \tag{1}$$

Although this rate of convergence in n is very slow, the number of dimensions, d, does not appear and MC is not subject to the curse of dimensionality. Significant effort has been invested in developing variants of MC with faster rates of convergence; a full review is out of scope. Latin Hypercube Sampling (LHS) [McKay et al. 1979; Owen 1992] is known as "N-rooks sampling" in the graphics community. Importance sampling [Glynn and Iglehart 1989] involves preferentially sampling rare cases and compensating by reducing their weight.

3. DART FRAMEWORK

Given the ideas of the method (Section 1) and where it might be useful (Section 2), we now define it formally and give a general recipe for converting a point sampling algorithm to a k-d dart sampling algorithm. We consider two application scenarios. The first is sampling to estimate the average of a function, as in numerical integration; see Section 3.1. The second is sampling to find locations in the domain where the function has a particular value; see Section 3.2.

We begin with terminology. Function f is defined over domain $\mathcal{X} \subset \mathbb{R}^d$. Often $\mathcal{X} = [0, 1]^d$ or a hyperrectangle box. A *flat* F is a k-dimensional hyperplane, the vector space defined by leaving k coordinates free and fixing the remaining $d - k$ coordinates, clipped at the domain boundary. A k-d dart s^k is a sample defined by a set of $I = \binom{d}{k}$ k-dimensional flats, s_i^k, one for each combination of fixed coordinates. So in $\mathbb{R}^2$, a 1-d dart might consist of the two 1-d flats at $x = 0.5$ and $y = 0.3$.

3.1 Averaging a Function over a Domain

Consider an algorithm that depends on the average value of f, and receives a series of single values, $y = f(x)$, from evaluating f at uniform-random points x.

3.1.1 *Algorithm for Box Domains.* We can instead return a series of y' values generated over uniform-random k-d darts. We generate a k-d dart's flats one at a time. For each flat F we choose its fixed coordinates uniformly at random, then:

(1) clip the hyperplane at the domain boundary and calculate its relative volume $V(F) = \int_F 1$. For boxes this is trivial;
(2) integrate the function along the clipped hyperplane: $G = \int_F f$;
(3) return y', the average of f over the flat, $y' = H = G/V(F)$.

Each flat of any dart is unbiased, in the sense that the expected value of y' is the average value of $f(x)$, the same as the expected value of each y from point sampling. To get an estimate with lower variance, average H over all flats of a k-d dart, several darts, or over some other uniform-random collection: $y' = \sum_{i=1}^{I} H_i/I$.

3.1.2 *Nonbox Domains.* This algorithm can be extended to general domains. Extend the domain $\mathcal{X}$ to its bounding box $B(\mathcal{X})$, and extend f to f_B over $B(\mathcal{X})$ by defining $f_B = 0$ in $B(\mathcal{X}) \setminus \mathcal{X}$.

Now estimate the average of f_B over $B(\mathcal{X})$ as before using H for box domains. To get an estimate of f over $\mathcal{X}$, use $y' = H \cdot V(B(\mathcal{X}))/V(\mathcal{X})$. This is unbiased if $V(\mathcal{X})$ and $V(B(\mathcal{X}))$ are known exactly. If either volume must be estimated, then y' is only guaranteed to be unbiased in the limit as the sample size approaches infinity. This is because the product of unbiased estimates is not usually an unbiased estimate of their product. The best way to use these values is to average H over many samples, average the volume estimates over many samples, then return the product.

3.1.3 *Efficiency.* In many cases we can obtain a fast expression for f over a flat by substituting fixed-coordinate constants into the expression for f. If the result is analytic, integrating over the flat is easy and efficient. If not, the substitution may have at least reduced the expense of evaluating f at the remaining free coordinates. One option is numerical integration over a discretization of the flat. Represent a flat with a uniform grid (mesh). (We assume it is too expensive to mesh the entire domain.) In (1), clip the grid elements at the domain boundary, generating simplices. In (2), evaluate f at the grid points and perform standard numerical integration over the grid simplices.

3.1.4 *Applications.* Our depth-of-field, probability of failure, and volume estimation applications follow this recipe, because they use Monte Carlo integration, but the integration along the line flats can be made faster and more accurate given application-specific knowledge about the function. For probability of failure, the function we integrate is the 0–1 indicator function of failure, rather than the continuous value of the response function. We improve accuracy by finding the roots of a single-variable equation to find the boundary points where the indicator function switches its value. For depth-of-field, f is the contribution of one ray (photon) to a pixel, and we seek to estimate the contribution of all photons over all focal depths for each pixel. For efficiency we use discrete algorithms to find the occlusion boundaries of triangles, then integrate a continuous function over each nonoccluded segment of a triangle.

3.2 Finding a Point in the Domain with a Particular Function Value

MPS represents a different category of application. Instead of integrating $f(x)$, we are looking for a random x that satisfies $f(x) > 0$, that is, finding a point outside all prior disks. Point-sampling techniques would sample the entire domain until such a point was found, which is expensive if the acceptable region is small. We can speed up the search using k-d darts.

(1) Clip F at the domain boundary, retaining g. (For MPS, we clip a line by the cube.)

(2) Retain the portions of g where the function value is acceptable, $f > 0$. (For MPS, these are the subsegments outside all prior disks.)

(3) Return a random point from g.

The likelihood of generating a particular g after several iterations of this process is usually different than its likelihood from uniform point sampling; Section 4.1.2 discusses this in more detail.

3.3 Darts are Unbiased for Averaging a Function

We prove that the algorithm in Section 3.1.1 is "unbiased," meaning the expected mean estimate using darts is equal to the true mean. Actually we show something stronger: the expected value of a single, uniformly sampled, axis-aligned flat is equal to the true mean. The expected value of sampling by darts is the mean of the expected value of all the flats; so, by the stronger result, this is the mean of a set of expected values that are themselves the true mean.

The stronger result follows immediately from point sampling being unbiased and the order independence of multidimensional integration. Let $\mathcal{D}$ be the set of domain coordinates, and partition it into the set of fixed and free coordinates for a flat: $\mathcal{D} = \mathcal{D}\text{free} \cup \mathcal{D}\text{fixed}$. Convert f from a function over a d-dimensional domain to $f_{\mathcal{D}\text{fixed}}$ over the fixed coordinates by integrating over the free coordinates: $f_{\mathcal{D}\text{fixed}} = \int_{\mathcal{D}\text{free}} f$.

By definition of the true mean $\overline{u}$ of f, we have

$$\overline{u} \int_{\mathcal{D}} 1 = \int_{\mathcal{D}} f = \int_{\mathcal{D}\text{fixed}} \int_{\mathcal{D}\text{free}} f = \int_{\mathcal{D}\text{fixed}} f_{\mathcal{D}\text{fixed}}.$$

Since a uniform point sample s is unbiased, $\mathbb{E}(f(s)) = \overline{u}$. Let $\overline{u_{\mathcal{D}\text{fixed}}}$ be the true mean of $f_{\mathcal{D}\text{fixed}}$. Since picking a particular flat is done by uniform point sampling over $\mathcal{D}$fixed, we have

$$\mathbb{E}(f_{\mathcal{D}\text{fixed}}(s_{\mathcal{D}\text{fixed}})) = \overline{u_{\mathcal{D}\text{fixed}}} = \frac{\int_{\mathcal{D}\text{fixed}} f_{\mathcal{D}\text{fixed}}}{\int_{\mathcal{D}\text{fixed}} 1}.$$

Combining these two we have

$$\mathbb{E}(f_{\mathcal{D}\text{fixed}}(s_{\mathcal{D}\text{fixed}})) \int_{\mathcal{D}\text{fixed}} 1 = \overline{u} \int_{\mathcal{D}} 1.$$

Since the domain is a box,

$$\begin{aligned} \int_{\mathcal{D}} 1 &= \int_{\mathcal{D}\text{fixed}} \int_{\mathcal{D}\text{free}} 1 = \left(\int_{\mathcal{D}\text{fixed}} 1\right)\left(\int_{\mathcal{D}\text{free}} 1\right) \\ &= \left(\int_{\mathcal{D}\text{fixed}} 1\right) V(\text{flat}). \end{aligned}$$

That is, simply divide the integral of f over a flat by the flat's relative volume to get an unbiased estimate:

$$\frac{\mathbb{E}(f_{\mathcal{D}\text{fixed}}(s_{\mathcal{D}\text{fixed}}))}{V(\text{flat})} = \overline{u}.$$

We reemphasize that this shows that a single axis-aligned flat is an unbiased estimator when chosen uniformly from a box. We did not rely on the dimensionality of the flat, cycling deterministically over the set of fixed dimensions, nor flats being orthogonal; these may affect the variance but not the mean estimate.

Except for the uniform sampling over the fixed coordinates and the volume of a flat being constant, we did not rely on the domain being a box nor the flats being axis aligned. For comparison we perform unbiased sampling with unaligned flats in Section 5.

4. APPLYING THE FRAMEWORK

We now describe how to apply the k-d dart framework to our representative applications in more detail.

4.1 Relaxed MPS

Our relaxed maximal Poisson-disk sampling algorithm is a variant of traditional dart throwing, throwing point darts and keeping those that hit an uncovered region (void). Algorithm 1 specifies our implementation in detail; in brief, we cast a line dart into the domain and intersect it with the disks of previously accepted samples to generate a set of uncovered segments, then uniformly sample from those segments with a point dart. The user specifies an acceptable void volume V, the fraction of the domain uncovered by sample disks. We have a conservative stopping criteria based on the number of successive misses that usually achieves a smaller void volume. Figure 5 can be used as a guide for selecting V in four dimensions.

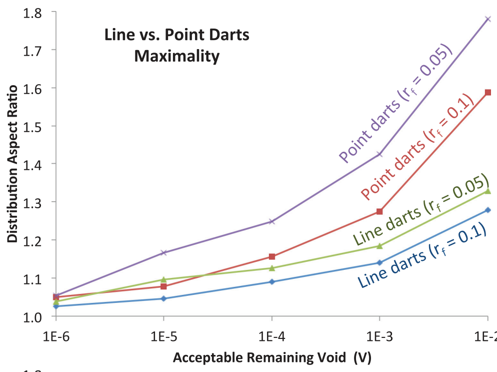
ALGORITHM 1: A classical dart throwing algorithm using line darts.

```
while maximality estimates are inadequate do
  generate a line dart $s^1$
  for all $i$ = permutation($1..d$) do
    generate line segments $g = s_i^1 \cap$ domain
    for all samples $p$ do
      subdivide $g = g \setminus D(p)$
    end for
    if $g \neq \emptyset$ then
      count $s^1$ as a hit
      select a sample point uniformly from $g$
      skip to next line dart
    end if
  end for
  count $s^1$ as a miss
end while
```

4.1.1 *Complexity.* The memory requirements are only $O(nd)$, which is what is required to represent the output point set because each sample has d coordinates. Only $2n + d$ floats are needed for scratch space for line segments g. We may generate and store only one flat of one dart at a time. The runtime is $O(dn \log n + nd^2)$ per dart throw. The most significant feature is that the complexity does not suffer from a curse of dimensionality: there are no exponents containing d. The number of throws is a function of V and the miss rate; we are not aware of any statistical techniques or theorems that would allow us analytically bound the miss rate, but we show that for line darts it can be made reasonable, or at least more reasonable than the alternatives.

Our approach is efficient because a line dart is more likely to intersect an uncovered region than a point dart. Only extremely simple and one-dimensional data structures are needed. The cost of throwing a line dart is nearly the same as a point dart.

4.1.2 *Output and Process.* The main drawback is that the output is not maximal, but its deviation can be estimated. A second potential drawback is that the process is not identical to MPS. There is no proof that the expected outputs are the same. Indeed, for a nonmaximal sample, the probability of inserting the next sample point in a given disk-free subregion depends only on the subregion's area in MPS; but for our variant the probability depends also



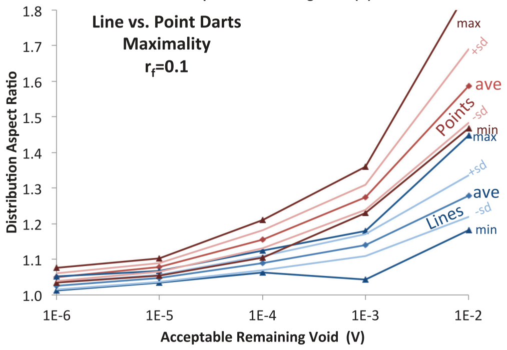
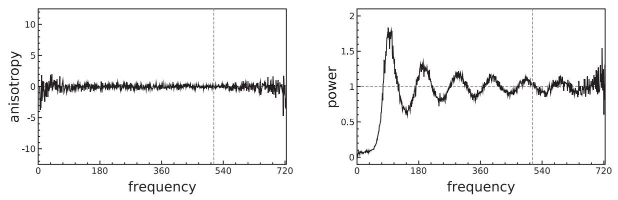
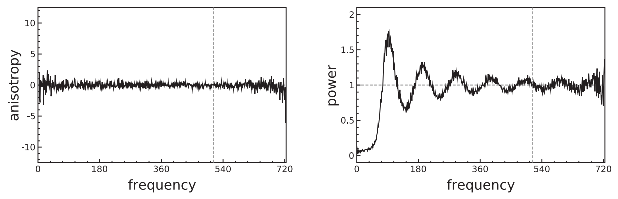


Fig. 5. Achieved distribution aspect ratio for different void volume thresholds V in $d = 4$. Top shows the means across different r_f, and bottom shows more statistics for $r_f = 0.01$, including standard deviation, sd. This is over 25 experiments.



(a) Line darts



(b) Point darts

Fig. 6. Radial anisotropy and mean power estimates for line darts (top) and point darts (bottom), averaged over ten samplings.

on how much prior disks cover the axis-aligned lines through the subregion. The main effect appears to be the order in which points are introduced, rather than their spatial distribution near maximality. Choosing the position of the next sample is dependent on many prior random decisions, so there are few noticeable patterns between one run and another with a different random number seed.

MPS and Algorithm 1 follow a different *process*. The quality of the *outcome*, the point positions, does not appear to be sensitive to this, and we see little distinction between the outputs using the standard measures of FFT spectrum, power, and anisotropy. Additionally, we are unaware of any formal definition of an ideal point set nor any proof that MPS produces it, so exactly matching the MPS process and its output are not strict requirements. The conventional goal in the graphics literature is "blue noise," meaning no discernible axis- or boundary-aligned patterns in the FFT as in Figure 6, and smooth anisotropy as in Figure 6 left; and something resembling the FFT of a Heaviside step function for the mean radial power, as in Figure 6 right.

4.1.3 *Implementation Details.*

4.1.3.1 *k-d Tree.* To speed up the iteration over prior samples when generating segments g we use a k-d tree. This allows us to prune samples whose disks are too far away to intersect the line of s_i^1. A k-d tree requires little memory regardless of the number of dimensions. A k-d tree uses axis-aligned partitions, which is a perfect match for checking the proximity of disks to our axis-aligned flats.

4.1.3.2 *Line Segments g.* We compute and store g, an array of segments, subsets of the line sample s_i^1. (This is our only data structure besides the k-d tree and a list of the accepted samples.) Each segment is inside the domain but outside the sample disks processed so far. Each coordinate is the start (a) or end (b) of an uncovered segment. Each segment is distinguished by a subscript, that is, $g = [a_0 b_0 a_1 b_1 \dots a_q b_q]$.

To update g for a new disk D we must remove from g the parts of its segments covered by the disk. We first compute the covered segment $g' = D \cap l = [b', a']$. We find the position in g where b' and a' should appear. If they are beyond a_0 or b_q, the disk does not intersect g. If b' and a' both lie between a_j and b_j, the latter segment is split into two: $a_j b' a' b_j$. Otherwise, if only b' lies between a_j and b_j, then a segment is trimmed by replacing b_j by b'; a similar step applies for a'. Any endpoints between b' and a' are covered by the disk and discarded.

To choose a point from g uniformly, we sum the lengths of the segments L, choose a random number between 0 and L, and find the corresponding point on g.

In practice, implementing g using a fixed-length array is efficient enough. (The number of segments is less than n, assuming the domain is convex.) An efficient implementation of g is a balanced tree. We can find the positions of b', a', and a random point in $O(\log n)$ time. We can update the tree in amortized $O(1)$ time.

4.1.3.3 *Maximality Estimates.* The user sets the remaining void volume V that is acceptable. Here we show how to translate that into a conservative stopping criteria. Denote the probability of hitting the void with a k-d dart by P_k. Given a lower bound on P_k, we set $m \geq \lceil 1/P_k \rceil$. We stop after m consecutive misses.

We seek lower bounds on P_k in terms of V in order to find a sufficiently large value of m. For point darts, $k = 0$ and $P_0 = V$ regardless of void shape. For higher-dimensional darts, the worst shape for a void is a hypercube with edge length $b = V^{\frac{1}{d}}$. By worst shape, we mean the shape that has the smallest P_k for a fixed V, assuming the entire domain is a hypercube. For a hypercube void, the probability of hitting that void with a k-d flat is given by $p_k = V^{\frac{d-k}{d}}$. A k-d dart contains $l_k = \binom{d}{k}$ k-d flats and hence $P_k = 1 - (1 - p_k)^{l_k}$. This gives a lower bound on P_k, and a sufficient value of m. We may consider this as an MC estimation of the remaining void using a sample size of the last $m + 1$ k-d darts. As in any MC process, there is some variance, which decreases as k increases. Moreover, the remaining void is typically scattered throughout the domain,

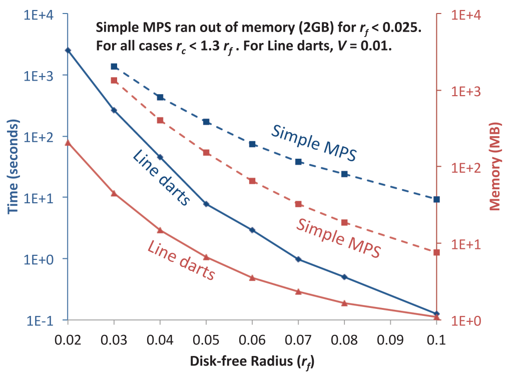


Fig. 7. Time (blues) and memory (reds) for line darts compared to Simple MPS for the same acceptable void volume, $V = $ 1e-2, in $d = 4$.

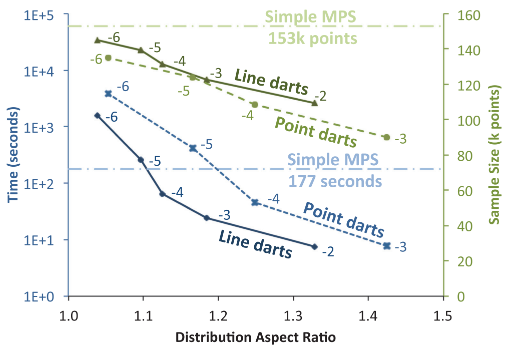


Fig. 8. V threshold effects on time (blues) and sample size (greens) for line and point darts in $d = 4$. Data points are labeled with $\log_{10} V$. As V decreases, the sample approaches maximality and distribution aspect ratio 1. Simple MPS [Ebeida et al. 2012] dashed lines are for a maximal distribution.

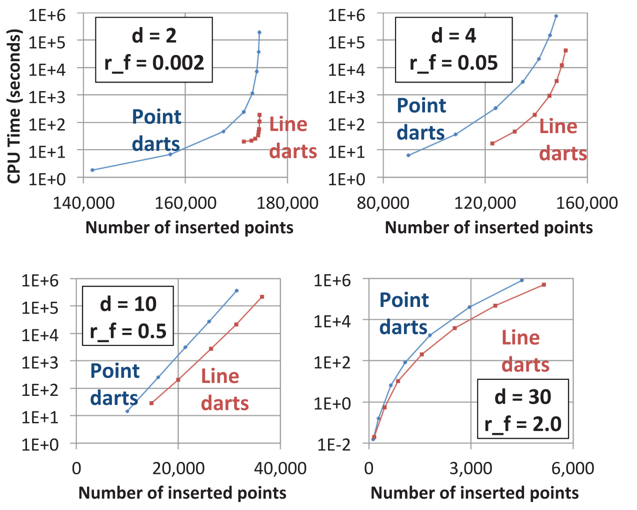


Fig. 9. Line darts approach maximality faster than point darts, as measured by the number of inserted points in a given runtime.

resulting in a smaller r_c than the worst case where the void is a single hypercube.

4.1.4 *Experimental Results.*

4.1.4.1 *Distribution Aspect Ratio.* The free radius, r_f, is the disk radius, the minimum distance between any two samples. The coverage radius, r_c, is the maximum distance between a domain point and its nearest sample. We define the *distribution aspect ratio* as $\beta = r_c/r_f \geq 1$. This is a measure of maximality.

To compute this for a point set, we used Qhull [Barber et al. 1996] to generate a Voronoi diagram. For each Voronoi vertex interior to the domain we retrieved the distance to its closest sample point: r_c is the maximum of these distances.

Figure 5 shows the relation between the distribution aspect ratio and the acceptable void volume in $d = 4$ for various disk-free radii. Figure 7 shows time and memory across different radii for a fixed void volume. Figure 8 shows runtimes as the point sets approach maximality. Line darts consistently produced better results than point sampling.

4.1.4.2 *Speed of Approaching Maximality.* We tested our code over 2-, 4-, 10-, and 30-dimensional domains using point darts and line darts. Figure 9 shows the number of points inserted over time. The expected void volume V is related to the number of points; in practice the number of inserted points is a better indicator of maximality than our loose estimates of V based on successive misses. Line darts were able to generate larger samples for all V and d.

4.1.4.3 *Efficiency by Method.* Figure 7 compares the performance of traditional point darts and line darts (this work) to Simple MPS [Ebeida et al. 2012]. Recall Simple MPS is based on a flat quadtree, and is currently the fastest and most memory efficient of the provably correct MPS methods. Our method is attractive at large values of r_f for its speed, and at low values of r_f for its memory consumption. For example, for $r_f < 0.025$, we generated 4M points in half an hour using 107 MB of memory, while Simple MPS ran out of memory at 2 GB.

4.1.4.4 *Output Quality.* We measure the quality of the distribution of 2-d output points using the PSA spectrum analysis tool [Schlömer 2011]. Using point darts, our process is the same as classic dart throwing, so we use it as our standard of correct output. Figures 6 and 10 compare the outputs' blue-noise properties; the difference between point and line darts was insignificant, at least for these three metrics.

4.2 Depth-of-Field with Antialiasing

k-d darts can be used for fast and high-quality rendering of Depth-Of-Field (DOF) effects in computer-synthesized images. Mathematically, computing a pixel's color in the presence of DOF can be expressed as a four-dimensional integral over the pixel's spatial (x, y) and lens aperture (u, v) dimensions. In most high-quality renderers, this is calculated using Monte Carlo integration over many point samples. This method suffers from a low rate of convergence; reducing noise for a good-quality image usually requires a very large number of samples per pixel. k-d darts offer the promise of faster convergence with low noise. Instead of using point samples for reconstruction, we use 1-d (line) darts, thrown in the 4-d (x, y, u, v) space.

We use Latin Hypercube Sampling (LHS) or jittered sampling for each dimension [Cook 1986]. Given that our sample space is four-dimensional, each line dart consists of four line flats. We select n points, that is, $4n$ flats. Each line requires a fixed location in 3-d and a variable fourth dimension.

We compute coverage for these darts using a method inspired by Gribel et al.'s work [2010, 2011] on rendering motion blur. Gribel

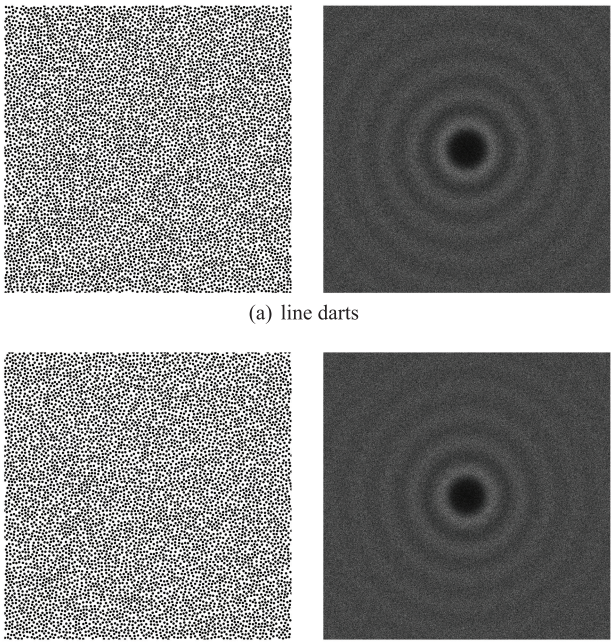


Fig. 10. FFT spectra (right) for the relaxed MPS point sets (left) generated by line darts (top) and point darts (bottom).

et al. fix x and y and shoot line samples in the time domain; instead, we use line darts that shoot line flats in all spatial dimensions to compute depth-of-field blur. We also sample a higher-dimensional problem.

4.2.1 *Contrast to Tzeng et al.* Tzeng et al. [2012] considered line sampling for DOF. They only consider the two (u, v) dimensions, not the four (u, v, x, y) dimensions as we do. Further, they take advantage of the structure that the (u, v) subspace of interest is uniformly circular. This reduces 2-d (u, v) sampling to 1-d sampling of lines through the origin, pinwheel sampling by angle. In their implementation, for each pixel, they fix x and y and use a pinwheel of line samples to vary u and v. The resulting DOF had high performance when compared to point sampling strategies, with low noise. However, due to both the pinwheel configuration and the fixing of x and y for each pixel's sample, strobing artifacts tended to occur in regions with high-frequency changes. The formulation we present in this article differs in the following ways.

- —Both implementations address DOF, but we also address antialiasing.
- —Tzeng et al.'s line darts are all radial and specified completely by their angle; they live in 1-d (θ) space. In this article, we use axis-aligned darts in 4-d (x, y, u, v) space. The positive and negative consequences follow:
  - —Tzeng et al. exhibits screen-space aliasing because x and y are fixed.
  - —Tzeng et al. achieves higher-quality DOF for the same number of samples, because 1-d spaces require fewer samples to cover than 4-d spaces.
- —Tzeng et al. use nonrandom dart locations, but we randomly position darts.
- —Tzeng et al.'s work specifically targets the GPU pipeline; we do not discuss (or consider) implementation details.

4.2.2 *Triangle Edge Equations in 4-d.* We now describe the triangle equations and how we create a line sample. For a given triangle, we start by computing a signed radius of Circle of Confusion (CoC) for each vertex, obtained using the following expression [Hammon 2008]

$$\text{CoC} = A\frac{f(z - z_f)}{z(z_f - f)},$$

where A and f are the camera aperture and focal length, respectively, and z_f and z indicate the respective depths of the focal plane and the given vertex. Note z is simply the w coordinate of the vertex in clip space. Now that we have the circle of confusion for each triangle's vertex, we can begin formulating a sampling strategy for each pixel.

Given a set of coordinates on the lens u and v, we assume a linear apparent motion of the vertex screen coordinates. For a given screen-space vertex $i \in \{0, 1, 2\}$, with coordinates (x_i, y_i), circle of confusion c_i, and $u, v \in [-0.5, 0.5]$, we have

$$x_i^u = x_i + c_i u \qquad (2)$$
$$y_i^v = y_i + c_i v. \qquad (3)$$

For each pixel we have a four-dimensional space (x, y, u, v), where x and y are the subpixel regions, and u and v are coordinates on the lens. To get a realistic and noise-free image, we seek to sample this space uniformly in an effective manner. Since we have four varying dimensions, we choose to use four different hypercubes for our Latin Hypercube Sampling (LHS). Each hypercube chooses three dimensions within which to sample, and one that will vary in our next stage.

For each triangle, we can consider the edge equations in this four-dimensional space to be the following: we substitute Eqs. (2) and (3) into the equations for testing whether a point (x, y) lies within one of the triangle edges i. Let ES_i represent the edge-sum at point (x, y) for the ith edge, between vertices i and j,

$$ES_i(x, y, u, v) = \left(y - y_i^v\right)\left(x_j^u - x_i^u\right) - \left(x - x_i^u\right)\left(y_j^v - y_i^v\right).$$

Expanding, we get equivalent right-hand sides

$$\begin{array}{cr} \Leftrightarrow & (y - y_i - c_i v)(x_j - x_i + u(c_j - c_i)) \\ - & (x - x_i - c_i u)(y_j - y_i + v(c_j - c_i)) \\ \Leftrightarrow & ES_i(x, y, 0, 0) \\ + & u(c_j(y - y_i) - c_i(y - y_j)) \\ - & v(c_j(x - x_i) - c_i(x - x_j)). \end{array}$$

This simplifies to

$$ES_i(x, y, u, v) = C_i(x, y) + uA_i(y) - vB_i(x) \geq 0.$$

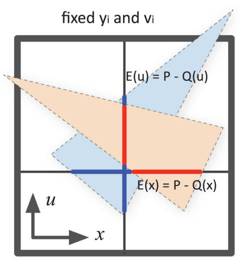
Here we have four dimensions of variability (x, y, u, v). In conventional point sampling, one would randomly (or pseudo-randomly) select a set of points in this space. We instead decide to employ line sampling using our Latin hypercubes. To illustrate this concept, consider fixing three of these dimensions with points on our hypercube from LHS: select x_1, y_1 and u_1 for x, y and u. Now we have a line equation as follows:

$$ES_i(v) = C_i(x_1, y_1) + u_1 A_i(y_1) - vB_i(x_1),$$

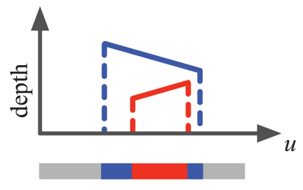
which simplifies to

$$ES_i(v) = P - vQ.$$

This line equation is easy to analytically solve for each of our edge equations and determines where line coverage exists. The same



(a) two line darts in (x, u)



(b) resolving occlusion depth

Fig. 11. Our technique for computing analytical coverage using line darts. In this example, consider two possible line flats in the x and u directions. The domain can be transformed and we can test the triangles for occlusion along the line flat as shown in (a). Depth resolution per flat is shown in (b).

concept can be extended to our other hypercube setups, fixing three of the dimensions and varying the last.

4.2.3 *Analytical Coverage in the Hypercube Domain.* Knowing the edge equations in our 4-d domain, we can now compute their coverage along a line sample. Figure 11 summarizes our technique for determining coverage.

We instantiate a set of line flats along each of our four hypercube configurations. A line dart is the combination of four different line flats (one in the direction of each of the four dimensions of the domain) with initial points chosen using our LHS.

Rendering consists of testing each incoming triangle against potentially covered line flats. For each pixel sample in the triangle's bounding box, we use equations from Section 4.2.2 to transform triangle edges to test for the correct hypercube domain.

To finish the calculation we follow Gribel et al.'s approach [2010] to construct and resolve line darts.

For each line flat, we analytically compute its segment covered by the triangle. A per-line-sample queue stores the color and depth of covered segments. Once all triangles have been processed, we resolve the final color for each sample. We sweep across the flat while aggregating triangles closest in depth, and then use all pixel samples to compute the final color for the pixel.

4.2.4 *Implementation and Results.* To test our formulation, we built a simple CPU-based renderer. It is capable of rendering scenes using traditional triangle rasterization. We integrated two additional capabilities:

—a stochastic sampler based on point darts;

—a sampler based on line darts.

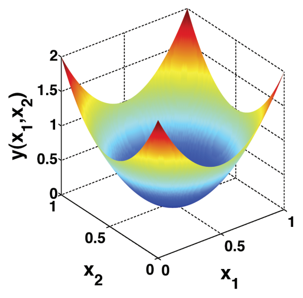
The two noise artifacts that typically occur using a DOF scheme are noise in blurry regions and noisy aliasing artifacts near the point in focus. For scenes far away from the focal plane, line flats in the x and y direction become particularly *narrow* when compared to the sampling space, and this causes additional noise. In these regions the *length* of flats in the x and y dimensions are significantly smaller than flats in either the u or v direction.

If we consider all such flats to have equal contributions, this results in the x and y samples adding a noticeable amount of noise into our system in blurry regions. To address this, rather than considering all line flats to have equal contribution, we select a weight constant ($\alpha = 0.2$ in our case) such that x and y line flats are scaled by α and contribute less to the scene.

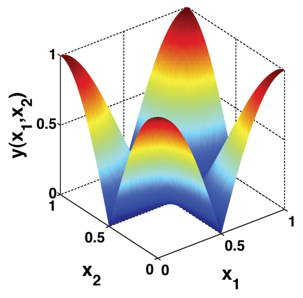
This weight can be modified based on the aperture of the lens for the scene. In cases where the aperture is small, contributions from x and y are deemed more important (for antialiasing), and the weight is adjusted accordingly. A more accurate dart sampling method might also consider the ratios of flat lengths per tile (worst case) and decide on an α weight accordingly. Research is needed to determine the optimal amount for each flat to contribute. Still, our simple heuristic seems effective.

Table I. Performance of Our Point vs. Line Darts

Sample Type	Sample Count	Rendering Time (s) Cessna	Rendering Time (s) Teapot
Points	64	29.6	52.1
	256	116.7	198.6
	1024	453.0	792.1
Line Darts	4	14.9	24.5
	16	56.8	91.9
	30	105.1	169.4



(a) circular parabola



(b) planar cross

Fig. 12. Plots of the test functions in 2-d.

Figure 13 compares two scenes rendered with our point dart and line dart techniques. Renderings based on k-d darts are virtually free of noise and aliasing artifacts. Point darts, however, retain noticeable aliasing artifacts even with 1024 darts. Although 16 line darts produce a bit more noise in unfocused regions than 256 point darts, 30 line darts has better quality than 256 point darts and around the same quality as 1024 point darts in unfocused regions, and no noticeable aliasing artifacts in focused areas.

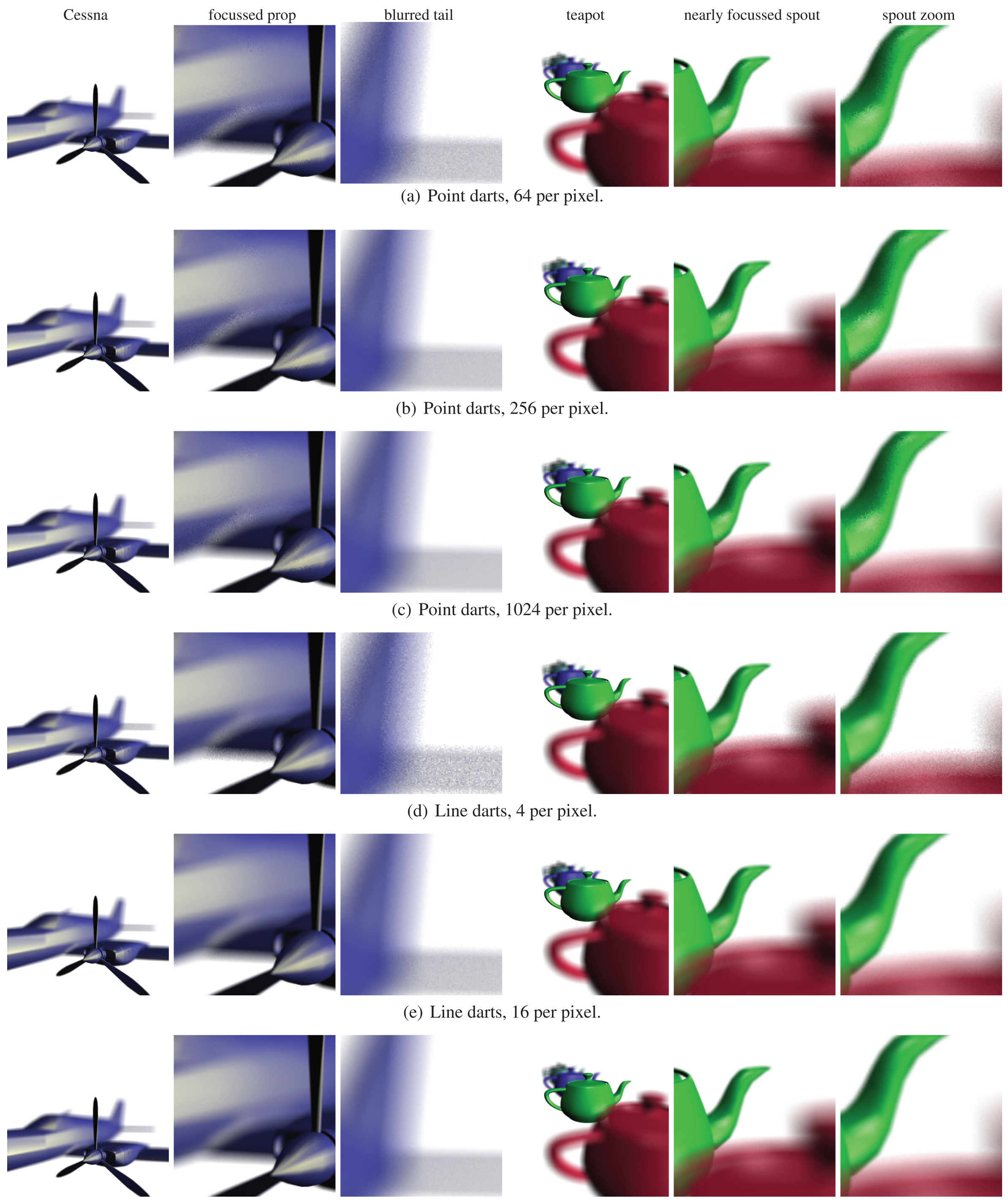
Table I shows the performance of our samplers. Clearly, throwing one line dart is more expensive than throwing one point dart. However, fewer are needed, and correctly weighted line darts converge more quickly to a less noisy image without aliasing artifacts.

4.3 Probability of Failure

Uncertainty quantification usually explores a vast high-dimensional space with a limited budget of sample points. Efficiency is crucial because typically the function evaluation is expensive and we want more sample points than we can afford. Sometimes we can afford more data by evaluating a cheaper surrogate model instead. But, if the failure region is small enough, then even that is not enough for Monte Carlo (MC) sampling to accurately estimate the probability of failure. Here we show that k-d darts can improve MC efficiency.

We test the "circular parabola" (Eq. (4)) and "planar cross" (Eq. (5)) surrogate models; see Figure 12.

$$y(x) = \sum_{i=1}^{d} (2x_i - 1)^2, \quad 0 < x_i < 1. \tag{4}$$



(a) Point darts, 64 per pixel.

(b) Point darts, 256 per pixel.

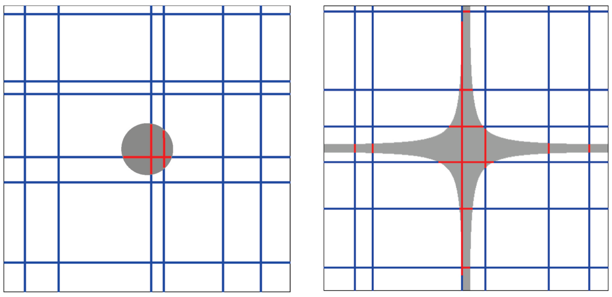
(c) Point darts, 1024 per pixel.

(d) Line darts, 4 per pixel.

(e) Line darts, 16 per pixel.

(f) Line darts, 30 per pixel.

Fig. 13. Depth-of-field (DOF) images using conventional point sampling (rows a–c) versus our *k*-d darts (rows d–f). *k*-d darts produce high-quality, antialiased images. The "blurred tail" column shows a close-up of an extremely blurry region, the Cessna's tail. In regions close to the focal plane we see some aliasing artifacts for point darts but not for line darts: for example, at the body–wing junction in the "focussed prop" column; and the transition shades on the green teapot spout in the rightmost two columns. Furthermore, line darts tend to be faster for the same quality blur; see Table I.



(a) circular parabola (b) planar cross

Fig. 14. Shape of the failure regions in 2-d.

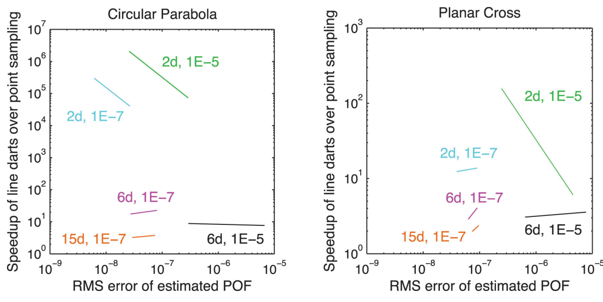


Fig. 15. Speedup of line darts over point sampling for estimating the Probability Of Failure (POF) of the circular parabola (left) and planar cross (right) analytic test functions. Each trend line indicates the Root-Mean-Square (RMS) error of the estimated POF over 64 trials, for a test function in a given dimension (labeled #d) with known analytic POF y_t (labeled 1E-#). Test functions with the same dimension and POF are colored the same in the left and right. Note that the speedups are greater than 1 for these problems, indicating that line darts are faster than point sampling. Since line darts in essence reduce the dimensionality of the problem by one, as the dimension increases the speedup decreases; maintaining speedup may require higher-dimensional darts.

$$y(x) = \left[\prod_{i=1}^{d} \frac{1+\cos(2\pi x_i)}{2}\right]^{1/d}, \quad 0 < x_i < 1. \tag{5}$$

Failure is defined as the function value below some constant threshold: $y(x) < y_t$. The shape of the failure region is different for the two test functions: a d-dimensional ball for the parabola and a fattened plus-sign for the planar cross; see Figure 14. For uniform distributions, the probability of failure is the fraction of the domain volume where $y(x) < y_t$. We choose y_t so the probability of failure is exactly 10^{-5} or 10^{-7}. We tested dimensions 2, 6, and 15. We estimate the failure volume using line darts. For a line flat, we find the roots of the single-variable equation $y(x_i) = y_t$. The length of the line segment between the two roots (if real roots exist) is used to estimate the volume of the failure region. Figure 15 demonstrates the benefit of line darts over conventional point sampling in reducing the time required to achieve a given accuracy level. Root-finding for line darts is expensive, but the information is worth it: for all tests line darts were more efficient than point darts. Our root-finding implementation is more expensive for the planar cross than the circular parabola, so the speedups for line darts on the planar cross are less even though the failure region is longer and thinner.

Table II. k-d Dart Parameter Study

d	k	s	r	n
2	0–1	$\frac{1}{10}, \frac{1}{2}, 1, 2, 10$	10	10^2–10^6
2	0–1	0.5	0, 1, 5, 10, 20	10^2–10^6
3	0–2	$\frac{1}{10}, \frac{1}{2}, 1, \sqrt{2}, \sqrt{10}$	10	10^2–10^6
3	0–2	0.5	0, 1, 5, 10, 20	10^2–10^6
10	0–9	$\frac{1}{10}, \frac{1}{2}, 1, \sqrt[9]{2}, \sqrt[9]{10}$	10	10^2–10^6
10	0–9	0.5	0, 1, 5, 10, 20	10^2–10^6

The darts are axis aligned except for some $d = 2$ experiments. We repeat each parameter combination 1000 times, $N = 1000$. Squish parameters s are symmetric around 1 with respect to ellipse volume: for example, for $d = 2$, $s = 1/2$ and $s = 2$ define ellipses with the same volume.

5. ACCURACY EXPERIMENTS

5.1 Problem Motivation

We provide some experimental results on the accuracy of darts for the canonical Monte Carlo problem of estimating the volume of an object in high dimensions. (Volume estimation is in the same category as the probability-of-failure problem in Section 4.3.) In particular, we seek to show that the method produces good estimates, regardless of the size, shape, dimension, and orientation of the object, and regardless of the dimension and orientation of the darts. The average estimate should be close to the true estimate, and the higher moments of the estimates should be low. We design our experiments to show the effects (if any) of the following factors:

—d, the dimension of the object;
—k, the dimension of the dart. Of particular interest is comparing our results to standard MC point sampling, $k = 0$;
—s, the squish factor of the object, which controls its aspect ratio;
—r, the number of rotations of the object. This allows us to compare axis-aligned objects to unaligned ones;
—axis-aligned darts versus unaligned darts.

We perform N experiments of n flats over the prior parameters, as described in Table II. Note that we keep the number of flats constant, rather than the number of darts, because the computational expense is more closely tied to the number of flats for objects with an analytic expression, and because middle-dimensional darts have many more flats than high- and low-dimensional ones.

5.2 Object Generation

Instead of a spherical object, we estimate the volume of an ellipse (a.k.a. ellipsoid), randomly oriented and squished. An ellipse provides enough generality to test the factors in Section 5.1, but enough simplicity to isolate numerical from methodology issues. In particular, we choose an ellipse because it is possible to analytically calculate the volume of an ellipse's intersection with a k-d dart.

Our object is a d-dimensional ellipse centered at the origin. We construct it as follows. We start with a d-ball centered at the origin with radius 1. This fits in an origin-centered cube with side length 2, the two-cube. We ensure the final ellipse also lies in the two-cube.

—s: squish factor. We scale the ellipse along the x-axis by multiplying its x-extent by s. The ball has $s = 1$. Note $s < 1$ gives thin, coin-shaped objects. For $s > 1$, we then shrink the ellipse so it fits in the two-cube: multiply all coordinates by a factor of $1/s$. The net effect is keeping the x-coordinate fixed and scaling the other axes by $1/s$. This gives needle-shaped objects.
—r: number of rotations. The more rotations we perform, the less the object is aligned with the coordinate axes. We perform r

rotations in sequence. Each is a Givens rotation with a random pair of coordinate indices i and j, and a random angle $\theta \in [0, \pi]$. To define a Givens rotation, take the identity matrix and replace the 2×2, $\{i, j\}$ submatrix [1 0; 0 1] by [$\cos\theta$ -$\sin\theta$; $\sin\theta$ $\cos\theta$]. Then multiply the coordinate matrix by the Givens matrix.

5.3 Dart Generation

Most implementers will choose axis-aligned darts for three reasons. First, it is easy to distribute aligned darts uniformly, which ensures that the expected mean of the function estimates is accurate. Second, it is easiest to implement aligned darts, since it involves simply fixing coordinate values. Third, in many cases it is most efficient because we may obtain an expression for the underlying function along a dart by substituting in the fixed-coordinate values. However, for completeness we provide some experimental results on the accuracy of unaligned darts. We shoot k-d darts into the two-cube as follows.

—A point dart ($k = 0$) is generated by selecting a random point. Each of the d coordinates is chosen independently and uniformly in [0, 1].

—Aligned darts are generated by their flats. Each flat has a unique combination of $d - k$ fixed-coordinate indices; the remaining k coordinates are allowed to vary. The coordinates for the fixed indices are chosen independently and uniformly as for point darts.

—Unaligned flats are generated so that the orientation of the flats is uniformly random. The only experimental setting where we generate unaligned flats is for $k = 1$ and $d = 2$, line darts in the plane. We choose angle $\theta \in [0, \pi]$, which determines the orientation of the flat. Any line that intersects the square crosses one of its main diagonals. (It is guaranteed to cross the diagonal to which it is more perpendicular, which depends only on θ.) We pick a point p uniformly at random along the appropriate diagonal. We now have a point and an angle, which together define a line flat. For random darts, the second flat is a line perpendicular to the first line, passing through some random point q of the other diagonal.

Aligned 1-d darts are labeled "k = 1a," random flats are labeled "k = 1r," and random darts, pairs of orthogonal flats, are labeled "k = 1o" in the top two rows of Figures 16 and 17.

5.4 Object-Dart Intersection

For point darts, the volume estimation is the fraction of darts that landed inside the ellipse, multiplied by the volume of the two-cube sampling domain, 2^d. For $k > 0$ darts, instead of this discrete ratio, we average the geometric fraction of each dart inside the ellipse object. The details of these calculations follow.

For point darts, we simply back-project the points to the domain of the ball: apply the inverse Givens rotations to the dart's point in reverse order; then scale the x-coordinate by $1/s$ (or, for $s > 1$, all the other coordinates by s). If the distance from the transformed dart to the origin is less than 1, it is inside the ball, and the original dart is inside the ellipse.

For k-d darts, we back-project their hyperplanes into the ball domain, where we can calculate the volume of intersection analytically, and then forward-weight it by the scaling.

To back-project a flat, we back-project $k + 1$ points spanning the flat. Each dart has $d - k$ fixed coordinates and k free coordinates. We pick spanning point p_0 with 0 for all of its free coordinates, and spanning point $p_{i>0}$ with 1 for its ith free coordinate and 0 for its other free coordinates. Each p_i is back-projected to q_i using the same procedure as for a point dart. The k vectors from q_0 to $\{q_{i>0}\}$ span the transformed flat, but are no longer orthonormal because of the final scaling step, so we must reconstruct an orthonormal basis. Now we are ready to calculate volumes, using forward transformations. We calculate the distance from the flat to the origin. This tells us the radius of the k-dimensional subball that is the intersection of the flat with the d-ball. We compute the volume of this subball. We multiply this volume by the sum of the x-components of the orthonormal basis, which gives the volume of the (unrotated) ellipse of intersection. The (forward) rotations do not affect the volume so are skipped.

For unaligned line darts in the plane, the distance from the origin is easy to measure. We use a process similar to the prior paragraph but it is a little easier because the 1-ball is simply a line segment.

5.5 Results

We plot the mean of the absolute value of the relative error, $|\text{mean} - \text{true}|/\text{true}$, versus the number of flats n in Figure 16. We plot the histograms of the ratios of estimated/true volume for $n = 10^6$ in Figure 17. Each subfigure shows results for all k for some combination of the other parameters.

In Figure 16 the experimental slopes for all k and d are about $-1/2$, in agreement with theory (the n to the power $-1/2$ in Eq. (1)). The accuracy is insensitive to the orientation of the object.

In Figure 17 the histograms are all sharply peaked at the true value. This shows that the variations, the higher-order moments of the estimates, are reasonable.

The major trend of these figures is that the accuracy of the estimates improves with k. Moreover, the larger the k, the smaller the variation of the estimate and the sharper the peak near the true value.

For small-volume objects, aligned darts are more accurate than unaligned darts. This is illustrated by the red curves in the top right and top left subfigures in Figures 16 and 17. We were initially surprised by this, but the explanation is that unaligned flats are shorter on average than aligned ones, because they might clip a corner of the square rather than having a length equal to the side of the square, so they miss the object more often. (They will be even shorter on average as the dimension of the space increases.) For moderate-volume objects, the accuracy is about the same regardless of object orientation. For unaligned flats, it appears that our n was large enough that using pairs of orthogonal flats or independently random individual flats does not make much difference. Our conclusion is that aligned darts are universally better when the sample domain is a square.

The accuracy is primarily sensitive to the volume of the object and, secondarily, to the squish value. Higher-dimensional darts are better than lower-dimensional ones, and the advantage is more pronounced for small-volume objects. This can be seen by considering the second-from-bottom row in Figures 16. To see the volume dependence, note that the lines are closer together for $s = 1$, and farther apart for larger and smaller squish factors. In low dimensions, 2 and 3, the smaller the volume of the object, the less accurate are all estimates, for all dimensions k. Moderate-dimensional darts in high dimensions, for example, $d = 10$ and $k = 4$, also exhibit this trend. However, 9-d darts in 10-d space have the same accuracy regardless of the volume or squish, because they are close to the dimension of the underlying space and they more fully span it. For example, a dart with $k = d$ always evaluates to the true value. That is, the advantage of higher-dimensional darts over lower-dimensional darts is greater for small objects in high dimensions, which is where we are advocating their use.

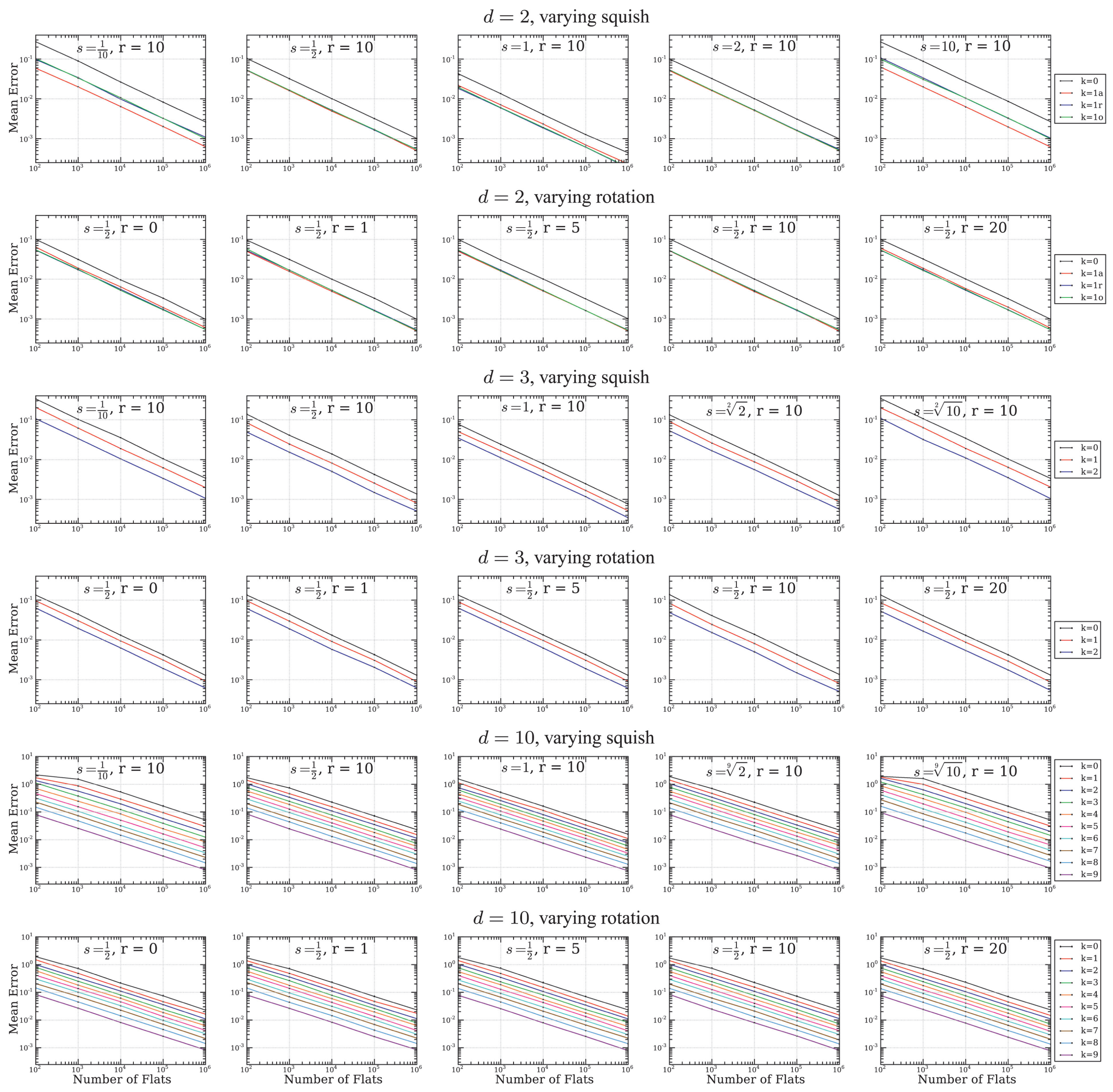


Fig. 16. Mean error of volume estimation, |mean − true|/true by n. See also Table II and Figure 17.

A secondary phenomenon is that the estimate is slightly more accurate for squish $s = 1/10$ than for $s = \sqrt[9]{10}$, despite the volume being the same. This can be seen by careful examination of the height of the lines in the $d = 10$, varying squish row in Figure 16. For instance, the mean error is 10% lower in the first column than the fifth for $n = 10^4$ and $k = 4$. This supports our intuition that darts are effective at hitting thin, coin-shaped regions. Since $\sqrt[9]{10} \approx 1.3$, a 10-d object with this squish factor is actually roundish and not very needle-shaped. Preliminary experiments show that the accuracy gained by increasing k is a complicated function of s and the volume. High-dimensional darts have more advantage over low-dimensional ones for very small and sharp needle-shaped objects, compared to their advantage for coin-shaped objects.

6. CONCLUSIONS

In this work, we introduced k-d darts as a particular type of higher-dimensional sampling. We described a k-d dart framework for hyperplanes of general dimension k, and then demonstrated efficiency and accuracy over three applications using $k = 1$, and accuracy for one application using $k \geq 1$. In particular, darts produce accurate estimates of the volume of an object regardless of the dimension, orientation, and aspect ratio of the object. Axis-aligned darts are universally preferable to unaligned ones for sampling square domains, and we expect this to extend to hyperrectangles, such as bounding boxes. Darts also produce accurate mean estimates for function integration.

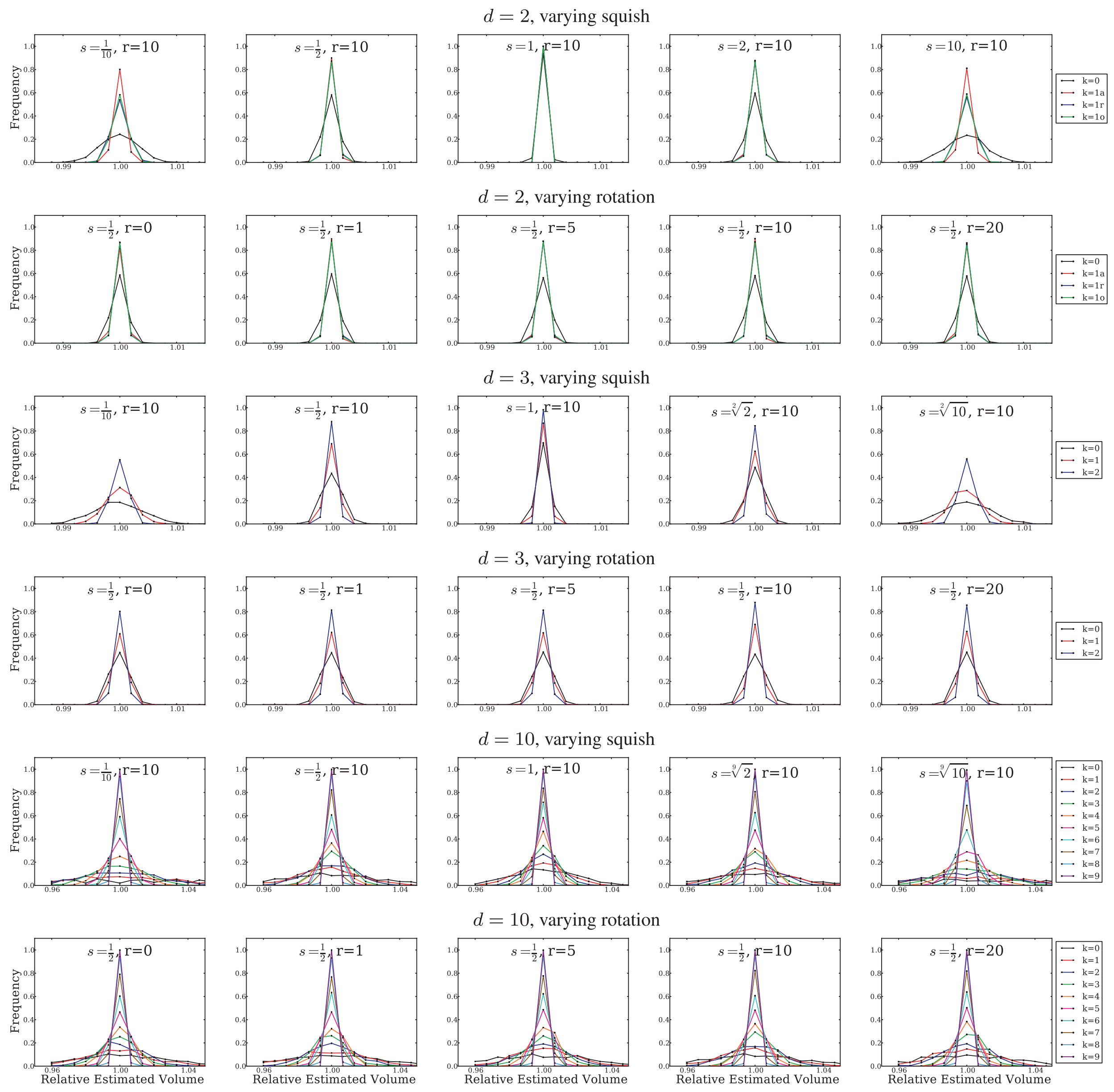


Fig. 17. Volume estimation histograms, estimate/true by frequency, for $n = 10^6$. "1a" is axis-aligned darts; "1r" is random-angle, unaligned lines; and "1o" is random-orientation darts, pairs of orthogonal flats. See also Table II and Figure 16.

Our implementation of darts samples each hyperplane orientation in a deterministically uniform way rather than randomly uniform. This yields a deterministic guarantee, rather than a probabilistic one, that at least some of the darts will be perpendicular to the thin directions of the object being probed. We hoped this would lower the variance. In our experiments, we had low variance regardless, perhaps because we had so many darts. For proving that the expected mean estimate is the true mean, deterministic uniform was not necessary.

In principle, thin objects are more efficiently sampled by higher-*k* darts. Demonstrating that efficiency for applications requires either analytical expressions, as in the toy ball-volume problem in the Introduction; or efficient numerical techniques for evaluating the underlying function along higher-dimensional flats. In future work we plan to explore a numerical technique using recursive sampling designs by dimension.

For maximal Poisson-disk sampling, line darts are helpful in getting close to maximality in high dimensions. Below a given acceptable void ratio, they are more efficient than point darts. In terms of bounding the distance from a domain point to the nearest sample point, we are actually closer to maximality as the dimension increases. Any difference in the distribution of points produced by classical maximal Poisson-disk sampling and our line darts is small by the standard measures. Our line dart algorithm is efficient with respect to memory usage, which enables the production of larger samples in higher dimensions.

For depth-of-field, generalized k-d darts give us a high-quality noise-free image without aliasing. Although each 1-d dart requires more processing than a point sample, we only need a few of them to render a high-quality image. Thus our k-d dart method outperforms point sampling. We suggest sampling over a higher-dimensional space to render both depth-of-field and motion blur in animations. We suggest exploring weighted darts, where scene information determines which flats contribute more to a scene.

For uncertainty quantification, k-d dart Monte Carlo sampling can be more efficient than point sampling. The key for all these applications is exploiting the problem structure to take advantage of what k-d darts provide.

ACKNOWLEDGMENTS

We thank the reviewers for their helpful feedback and suggestions that made the article better. Thanks to Laura Swiler for discussing Section 3.3 with us.

REFERENCES

C. B. Barber, D. P. Dobkin, and H. T. Huhdanpaa. 1996. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.* 22, 4, 469–483.

R. L. Cook. 1986. Stochastic sampling in computer graphics. *ACM Trans. Graph.* 5, 1, 51–72.

M. A. Z. Dippé and E. H. Wold. 1985. Antialiasing through stochastic sampling. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'85).* 69–78.

M. S. Ebeida and S. A. Mitchell. 2011. Uniform random voronoi meshes. In *Proceedings of the 20th International Meshing Roundtable*. 258–275.

M. S. Ebeida, S. A. Mitchell, A. Patney, A. A. Davidson, and J. D. Owens. 2012. A simple algorithm for maximal Poisson-disk sampling in high dimensions. *Comput. Graph. Forum* 31, 2, 785–794.

M. S. Ebeida, A. Patney, S. A. Mitchell, A. Davidson, P. M. Knupp, and J. D. Owens. 2011. Efficient maximal Poisson-disk sampling. *ACM Trans. Graph.* 30, 4, 49:1–49:12.

M. N. Gamito and S. C. Maddock. 2009. Accurate multidimensional Poisson-disk sampling. *ACM Trans. Graph.* 29, 1, 8:1–8:19.

P. W. Glynn and D. L. Iglehart. 1989. Importance sampling for stochastic simulations. *Manag. Sci.* 35, 11, 1367–1392.

C. J. Gribel, R. Barringer, and T. Akenine-Möller. 2011. High quality spatio-temporal rendering using semi-analytical visibility. *ACM Trans. Graph.* 30, 54:1–54:11.

C. J. Gribel, M. Doggett, and T. Akenine-Möller. 2010. Analytical motion blur rasterization with compression. In *Proceedings of Conference on High Performance Graphics (HPG'10).* 163–172.

P. E. Haeberli and K. Akeley. 1990. The accumulation buffer: Hardware support for high-quality rendering. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'90).* 309–318.

A. Hall. 1873. On an experimental determination of π. *Messenger Math.* 2, 113–114.

E. Hammon Jr. 2008. Practical post-process depth of field. In *GPU Gems 3*, H. Nguyen, Ed., Addison-Wesley, 583–605.

W. Jarosz, D. Nowrouzezahrai, I. Sadeghi, and H. W. Jensen. 2011. A comprehensive theory of volumetric radiance estimation using photon points and beams. *ACM Trans. Graph.* 30, 1, 5:1–5:19.

H. W. Jensen and P. H. Christensen. 1998. Efficient simulation of light transport in scenes with participating media using photon maps. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'98).* 311–320.

T. R. Jones and D. R. Karger. 2011. Linear-time Poisson-disk patterns. *J. Graph. GPU Game Tools* 15, 3, 177–182.

T. R. Jones and R. N. Perry. 2000. Antialiasing with line samples. In *Proceedings of the Eurographics Workshop on Rendering Techniques*. Springer, 197–206.

A. Lagae and P. Dutré. 2008. A comparison of methods for generating Poisson disk distributions. *Comput. Graph. Forum* 27, 1, 114–129.

J. Lehtinen, T. Aila, J. Chen, S. Laine, and F. Durand. 2011. Temporal light field reconstruction for rendering distribution effects. *ACM Trans.Graph.* 30, 4, 55:1–55:12.

Y.-S. Liu, J.-H. Yong, H. Zhang, D.-M. Yan, and J.-G. Sun. 2006. A quasi-Monte Carlo method for computing areas of point-sampled surfaces. *Comput.-Aided Des.* 38, 1, 55–68.

N. L. Max. 1990. Antialiasing scan-line data. *IEEE Comput. Graph. Appl.* 10, 1, 18–30.

M. D. McKay, R. J. Beckman, and W. J. Conover. 1979. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* 21, 2, 239–245.

D. P. Mitchell. 1987. Generating antialiased images at low sampling densities. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'87).* 65–72.

G. Nebe and N. Sloane. 2012. A catalogue of lattices. http://www.math.rwth-achen.de/$\sim$Gabriele.Nebe/LATTICES/index.html.

A. B. Owen. 1992. A central limit theorem for Latin hypercube sampling. *J. Royal Statist. Soc. Series B (Methodological)* 54, 2, 541–551.

J. F. Ramaley. 1969. Buffon's noodle problem. *Amer. Math. Monthly* 76, 8, 916–918.

J. Rovira, P. Wonka, F. Castro, and M. Sbert. 2005. Point sampling with uniformly distributed lines. In *Proceedings of the 2nd Eurographics / IEEE VGTC Conference on Point-Based Graphics (SPBG'05).* Eurographics Association, 109–118.

T. Schlömer. 2011. PSA point set analysis. http://code.google.com/p/psa/.

J. Spanier. 1966. Two pairs of families of estimators for transport problems. *J. SIAM Appl. Math.* 14, 702–713.

X. Sun, K. Zhou, S. Lin, and B. Guo. 2010. Line space gathering for single scattering in large scenes. *ACM Trans. Graph.* 29, 4, 54:1–54:8.

S. Tzeng, A. Patney, A. Davidson, M. S. Ebeida, S. A. Mitchell, and J. D. Owens. 2012. High-quality parallel depth-of-field using line samples. In *Proceedings of the Conference on High Performance Graphics (HPG'12).* 23–31.

E. Veach and L. J. Guibas. 1997. Metropolis light transport. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'97).* 65–76.

L.-Y. Wei. 2008. Parallel Poisson disk sampling. *ACM Trans. Graph.* 27, 3, 20:1–20:9.

K. B. White, D. Cline, and P. K. Egbert. 2007. Poisson disk point sets by hierarchical dart throwing. In *Proceedings of the IEEE Symposium on Interactive Ray Tracing (RT'07).* 129–132.

Received January 2013; revised June 2013; accepted August 2013