RangeFinder: Accelerating ball-interference queries against steady lattices

Submitted June 30th, 2018

Kelsey Kurzeja, Jarek Rossignac
kkurzeja3@gatech.edu, jarek@cc.gatech.edu

School of Interactive Computing,
Georgia Institute of Technology, Atlanta, USA

Abstract

Advances in additive manufacturing techniques are enabling the fabrication of new microstructures and materials. These may often be defined in terms of a set of balls and of beams that each connects two balls. To support application needs, we must support lattices with billions of such elements. To address this problem, we focus on architected and periodic structures in which the connectivity pattern repeats in three directions, and in which the positions and radii of the balls evolve through the structure in a prescribed and steady way that is defined by three similarity transforms. We propose here an algorithm that accelerates the Ball-Interference Query (BIQ), which establishes which elements of the lattice interfere with a query ball *Q*. Our RangeFinder (RF) solution reduces the asymptotic complexity of BIQs, which, in our tests, reduced the query time by a factor of between 45 and 5500. RF does not use any spatial occupancy data structure and can be trivially parallelized. We demonstrate the effectiveness of RangeFinder through the generation of multi-level lattices that we call Lattice-in-Lattice (LiL).

Keywords: lattice, pattern, query, point-membership classification, procedural modeling, steady affine motions

1 Introduction

We consider three-dimensional, truss-like lattices that are each the union of a set of **balls** and **beams**. The balls are organized into a three-dimensional array of groups of balls. Each **group**, *G*[i,j,k], is identified by a triplet of indices, (i,j,k). All the groups have the same number of balls. Each ball *B*[i,j,k,b] in *G*[i,j,k] is defined by the four-tuple (i,j,k,b) and has center, C[i,j,k,b], and radius r[i,j,k,b]. The **periodic connectivity** of the lattice is defined in terms of a set of **edge-patterns**. An edge-pattern is defined by the five-tuple (b_1,i',j',k',b_2) and defines all edges from ball (i,j,k,b_1) to ball (i+i',j+j',k+k',b_2) for valid triplets (i,j,k) for which the two balls exist and are not explicitly marked as omitted by the designer. To each such edge of each edge-pattern

corresponds a beam that is a solid of revolution around the axis passing through the centers of the two balls and that smoothly blends between the two balls. In its simplest form, discussed in this paper, the beam is a truncated cone, tangent to the two balls. Though, beams with non-linear silhouettes may also be used [1].

We say that the lattice is **regular** if three vectors, U, V, and W, exist so that, for each ball-index, b, C[i,j,k,b] = C[0,0,0,b] + iU + jV + kW and all values r[i,j,k,b] are identical. Regular lattices have many advantages. The most important ones are that (1) they do not require storing the explicit location of all the balls and (2) geometric queries, such as **point-membership classification** (**PMC**) or area/volume calculations may be computed lazily, in constant time (assuming a relatively small number of balls per group and of edge patterns), regardless of the complexity (number of groups) of the lattice. An example is shown in Figure 1.

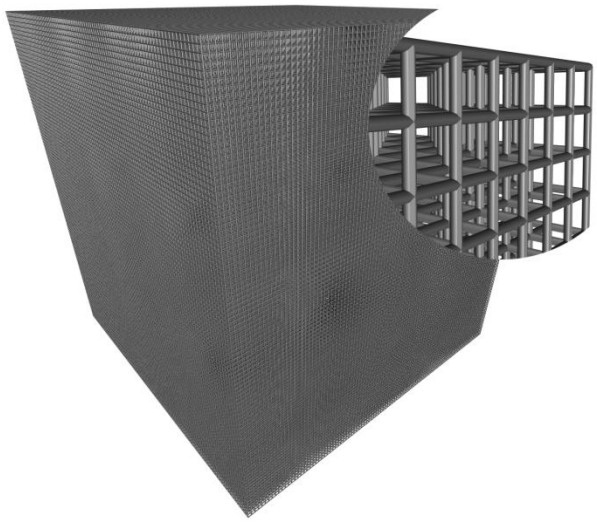

Figure 1: Regular lattice with 100^3 groups of a single ball and 3 edge-patterns. The top-right is magnified to show detail.

Our objective is to preserve, as much as possible, these advantages, while relaxing the regularity constraint. Hence, we focus on **semi-regular** lattices, in which the centers, C[i,j,k,b], and radii r[i,j,k,b] of the balls are defined by mathematical formulae or efficient algorithms, that can be evaluated quickly (hence supporting the lazy evaluation advantage mentioned above) and yet allow the designer to deform (bend, twist, taper) the lattice to some extent.

Such semi-regular lattices may be approximated by a regular lattice and a **Free-Form Deformation** (**FFD**) of it [2, 3]. The FFD may be defined via a tri-viariate polynomial (for example, Bezier or B-Spline) map. Such an approach offers design flexibility (for example, 64 control points for a cubic Bezier tri-variate) and mathematical continuity. However, it suffers from several drawbacks:

1. Direct manipulation of the control points may often result in unacceptable geometries that locally destroy the periodic structure of the pattern and may produce unwanted interferences

between beams. Hence, computer-assistance may be required to detect such problems and help the designer avoid or rectify them.

2. The non-monotonic variations of beam length, of angles between beams, or of other geometric measures along one of the three principal directions of the lattice may result in abrupt spatial variations of homogenized mechanical properties.
3. The large number of control parameters (for example, 192 coordinates of the control points) increases optimization cost.
4. Simple queries, such as PMC or the computation of integral properties (surface area, volume, center of mass, inertia) may be prohibitively expensive, even when efficient inversion solvers are available for computing the triplets (u,v,w) of parameters (corresponding location in the regular lattice) that the FFD maps to for a given query point Q.
5. The FFD bends beams and maps balls into complex shapes. Correcting for these deformations so as to produce a valid lattice with straight-axis beams and spherical nodes exacerbates the previous drawbacks. For example, a natural formula for inferring the radius of each ball from the local derivative of the map may produce a lattice in which pairs of instances of two different balls overlap, but only in a small portion of the lattice. Furthermore, straightening the beams may result in pairs of beams interfering in some portions of the lattice. Consequently, optimization steps may need to be interleaved with computationally expensive interference or pathology detection and remediation passes. Finally, the (u,v,w) parameters of a query point Q cannot be used against the regular (undeformed version of the) lattice because a point with coordinates (u,v,w) would have to be classified against the (now bent) preimages of the corrected balls and beams.

Steady lattices offer an appealing compromise between the regular lattices and their FFDs. They support a limited, but useful, range of deformations (see an example in Figure 2) while avoiding or reducing significantly the drawbacks of FFDs of regular lattices. This compromise is particularly appealing for designing, optimizing, and querying highly complex lattices with billions of beams.

We say that a lattice is steady if three orientation-preserving, similarity transforms, **U**, **V**, and **W**, exist so that, for each ball-index, b, $B[i,j,k,b] = \mathbf{G} \circ \mathbf{W}^k \circ \mathbf{V}^j \circ \mathbf{U}^i \circ B[0,0,0,b]$, where **G** controls the position, orientation and size of the **base-group** $G[0,0,0]$. These **incremental similarity** transformations may be computed from their cumulative counterparts, $\mathbf{U}^u$, $\mathbf{V}^v$, $\mathbf{W}^w$ and the corresponding **repetition counts** u, v, w [4]. Hence, the deformation of a regular lattice into the standard form of a steady lattice may be defined using four similarity transforms, **G**, $\mathbf{U}^u$, $\mathbf{V}^v$, and $\mathbf{W}^w$, or equivalently four local coordinate systems obtained by transforming the global coordinates by **G**, $\mathbf{G} \circ \mathbf{U}^u$, $\mathbf{G} \circ \mathbf{V}^v$, and $\mathbf{G} \circ \mathbf{W}^w$. Each coordinate system is defined by 7 parameters (for example, three coordinates of its origin, 3 angles, and one uniform scaling). So, the standard steady lattice layout has only 28 degrees of freedom, about 7 times less than its FFD counterpart. More advanced variants of steady lattices, not discussed here, expose additional degrees of freedom, at the expense of some of the benefits of purely steady lattices. They include a variant which allows the designer to control 8 frames, one at each corner of the lattice, resulting in 56 degrees of freedom. A

significant fraction of structures in architecture, CAE, urban planning, mechanical CAD, material engineering, art, and nature contain structures that can be modelled as (or closely approximated by) steady lattices.

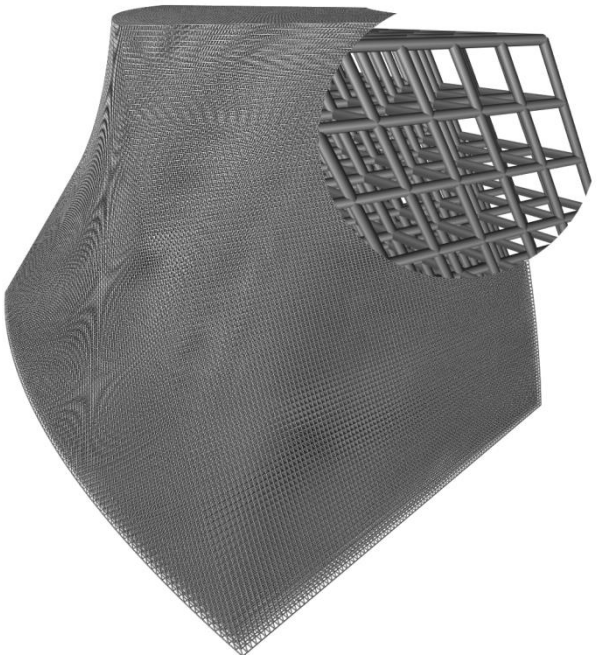

Figure 2: Steadily bent, twisted, and tapered semi-regular lattice with 100^3 groups of a single ball and 3 edge-patterns. The top-right is magnified to show detail.

The key feature of all steady lattices is that their balls can be organized into **steady rows** (see Figure 3) which have the form $R_k = \mathbf{S}^k \circ R_0$, where R_0 is the **template shape** and **S** is an incremental similarity, proven in [4]. The set of balls B[i,j,k,b] for a fixed pair (i,j) and all valid values of k forms a steady row of balls, i.e. B[i,j,k,b] = $\mathbf{W}^k \circ B$[i,j,0,b]. The beams of a steady lattice may also be organized into steady rows of beams, such that, if T[i,j,k,b,t] is the beam (tube), t, that connects to ball b in group (i,j,k), then T[i,j,k,b,t] = $\mathbf{W}^k \circ T$[i,j,0,b,t], which is also proven in [4].

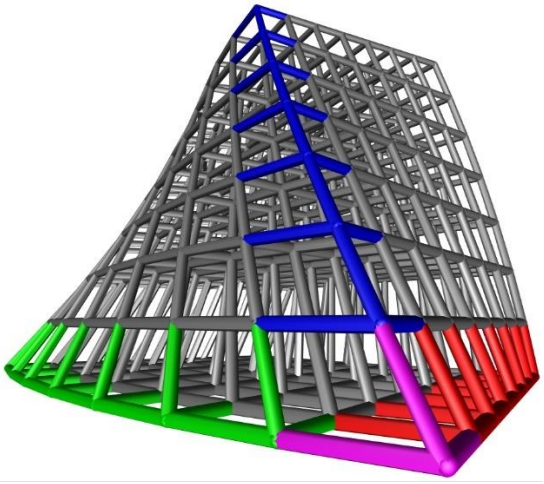

Figure 3: The base-group *G*[0,0,0] and beams originating from it are shown in magenta, and some rows along the U (red), V (green), and W (blue) directions are also shown. Corresponding balls in each of the three rows forms a steady row of balls, but corresponding beams are only guaranteed to form a steady row of beams along the W direction.

An approach that takes advantage of these properties of steady lattices to accelerate the computation of surface areas, volumes, and centroids of steady rows of balls or of beams is addressed elsewhere [4]. In this paper, we focus on exploiting steadiness to accelerate Point-Membership Classification (PMC) and **Ball-Interference Queries** (**BIQs**). Specifically, given a candidate point Q and a steady lattice *L*, PMC establishes whether Q lies inside any of the elements (ball or beam) of *L*. Similarly, given a candidate ball *Q* and a steady lattice *L*, BIQ establishes whether *Q* interferes (i.e. has non-empty intersection) with any of the elements of *L*, and can be used to retrieve a list of the elements of *L* that interfere with *Q*. BIQ is a generalized form of PMC, because PMC is equivalent to a BIQ with a ball of radius zero.

A naive implementation of these queries requires testing *Q* against each element of *L*. Exploiting the steadiness of *L*, we reduce the expected asymptotic complexity from O(uvw) tests to O(uv) tests, at least, for queries where the radius of *Q* is small enough and the elements of *L* are well separated enough so that *Q* may only ever interfere with a number of elements of *L* that is small compared to u, v, and w. The proposed improvement may result in a 100-to-1 performance improvement for a lattice with 100x100x100 elements (see Figure 1 and Figure 2), which is a lattice of modest complexity in the scale of micro- and nano-structures. For special configurations of steady lattices, we further reduce the expected complexity to O(u) and O(1).

In some applications, it may be possible and advantageous to precompute flat or hierarchical (bounding volume hierarchy or space partition) data structures and to use them to accelerate BIQ queries. We focus here on applications where such an approach is not possible or not desirable, for example, because the time or space cost of building such a structure would be prohibitive or because the memory access costs to such a structure would be prohibitive on massively parallel architectures. Furthermore, in an optimization loop that adjusts the lattice parameters, the data structure would have to be reconstructed after each operation.

Hence, we advocate distributing the implicit definition of the lattice (the balls in the base-group, its connectivity, the three similarities, and the repetition counts) to the various processors and, within each thread, using it to evaluate the parameters that define a specific steady row (i,j) of balls or beams, and then performing the query or query sequence on that steady row.

So, the problem addressed here may be reduced to the following. A steady row R is defined by a template shape R_0 (ball or beam), an incremental similarity **S**, and a repetition count n. R is the union of all n+1 copies $R_k = \mathbf{S}^k \circ R_0$. Given such a row R and a query ball Q, we want to establish whether Q interferes with R.

We decompose this problem into two subtasks: (1) Identify the range $[k_{min}, k_{max}]$ of values for which we can guarantee, when k is outside of $[k_{min}, k_{max}]$, that Q does not interfere with R_k and (2) for all values k in $[k_{min}, k_{max}]$, use precise geometric tests to establish whether Q interferes with R_k. The second task consists of a hopefully small and constant number of standard geometric tests. Hence, we focus on the first one and refer to its solution as **RangeFinder**.

We wish to compute $[k_{min}, k_{max}]$ quickly using closed form expressions, without iterations or recursions. At the same time, we wish to reduce the number of false positives, i.e., we want to make $[k_{min}, k_{max}]$ as tight as possible.

One important application of RangeFinder is accelerating higher level queries that build off of BIQs. For example, a grid of query points or balls can be used to voxelize a lattice for analysis or additive manufacturing. Also, a BIQ with a query ball of center Q and radius r can be used to implement a local distance query by retrieving a list of lattice elements within distance r from Q and computing the minimum distance to the lattice interior by testing and storing the minimum distance from Q to each element in the list. Higher level queries may even build off of each other, for example, the local distance query may be used to implement raycasting via sphere tracing [5].

Another particularly important application of RangeFinder is its use for displaying and processing a multi-level lattice directly from its implicit definition. For example, Figure 4 shows a two-level lattice modeled using a coarse steady lattice C and a fine steady lattice F. Specifically, we use a variant of BIQ to identify the good balls of F, which we define as those interfering with C. We then discard the rest of the balls of F and the beams of F that are not connecting two good balls. In other words, C is used to provide a procedural definition of a trimming volume for F. This **Lattice-in-Lattice** (**LiL**) idea may be cascaded to provide a structure with more than two levels, as shown in Figure 5.

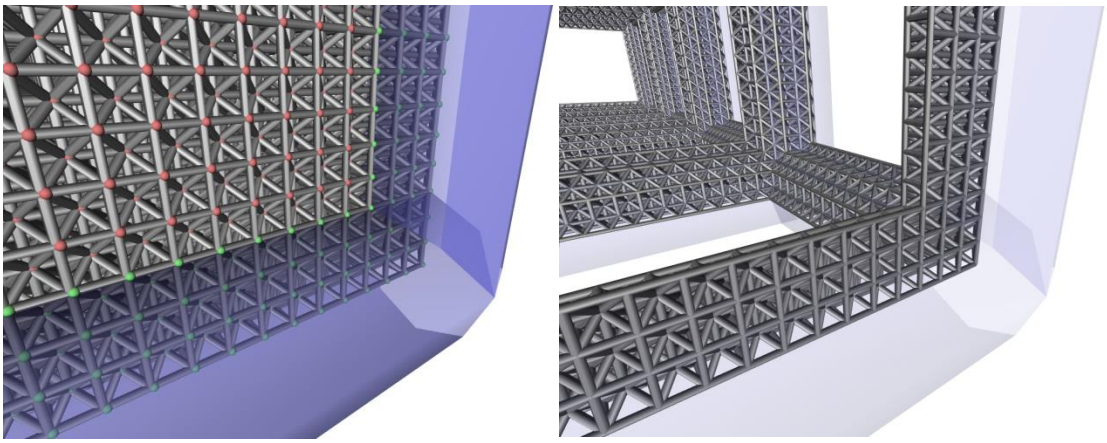

Figure 4: (Left) Coarse lattice *C* shown in transparent blue and a fine lattice *F* of which its balls are colored green if they interfere with *C* and red if they do not. (Right) The LiL resulting from removing from *F* all beams with at least one red ball.

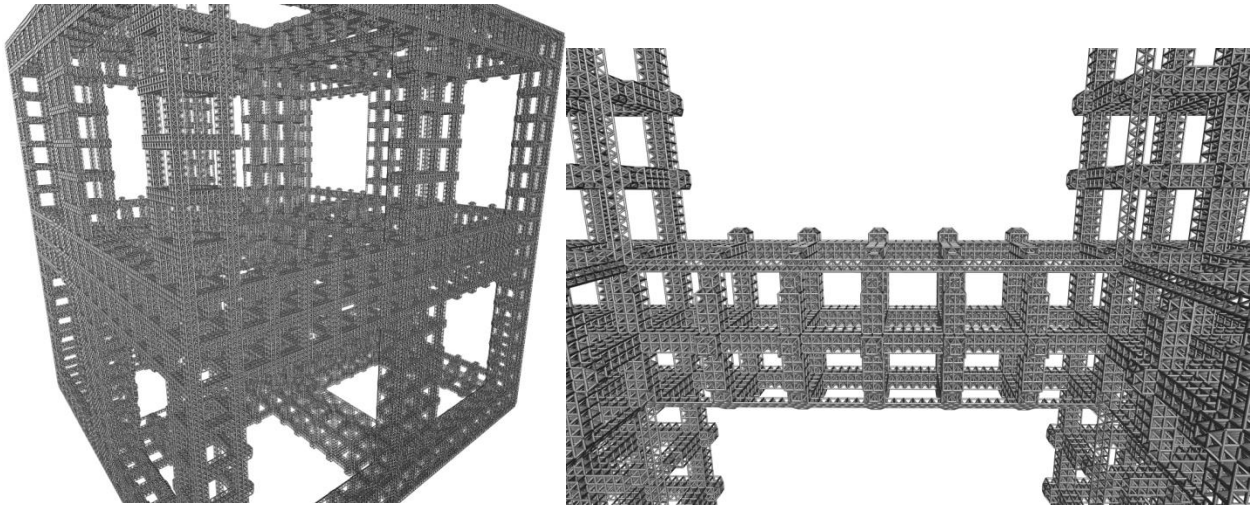

Figure 5: (Left) Cascaded LiL produced from two coarse lattices and a fine lattice with 138^3 groups. The resulting structure has 1,494,074 beams. (Right) Magnification on a multi-level beam.

The remainder of the paper is organized as follows:

- In section 2, we list our notations.
- In section 3, we review related prior art.
- In section 4, we first present the overview of the RangeFinder approach and prove the key theorem upon which it is based, and then we explain the details of RangeFinder, with a focus on steady rows of balls.
- In section 5, we extend the details, from section 4, to handle steady rows of beams.
- In section 6, we discuss how RangeFinder may be applied to accelerate BIQs on steady lattices, and we offer modifications to the BIQ algorithm for performance improvements on special cases of steady lattices.
- In section 7, we provide results of using RangeFinder to accelerate the generation of LiLs.

- In section 8, we conclude by restating our contributions, highlighting the limitations of our results, and discussing directions for future research.

2 Notations

First, we give brief descriptions and examples of the notations used:

k, f	Integers, reals
Q, V	Points, vectors
AB	Vector from A to B
A, R_k	Point-set object
S, **M**(X)	Transformation, Function
C, ***Q***	Integer or real interval
$X_{[A]}$	Closest projection of X onto A
A·B	Dot product of A and B
A×B	Cross product of A and B
|V|	Magnitude of V
$\underline{V}$	Normalization of V
S∘X	X transformed by **S**
V°(w,D)	V rotated by w radians around D
U^V	Angle from U to V
⌊a⌋, ⌈b⌉	Floor of a, ceiling of b

3 Prior art

A significant number of man-made and natural objects contain patterns that may be modeled as steady rows, or steady rows-of-rows, of which the template shape may or may not be a ball or beam. M. Pauly et al. introduce a computational approach for discovering steady, similarity-based patterns in 3D geometric models [6]. The method is applied to architectural, artistic, and natural structures that are good candidates for RangeFinder acceleration, if BIQs against the models are desired during some analysis.

N. Stolte proposed a method, known as infinite implicit replication [7], of implicitly modeling infinitely repeating patterns, from a template surface, with a single closed-form equation. The paper describes, at a high level, a mapping from points in space to integers such that the integer corresponding to a region of space specifies how many times an incremental transformation is to be applied to the template surface to generate the instance contained by that region. The approach described is mostly abstract, and the only concrete replication given is a steady translation along an axis. One limitation of this approach is that only a single instance of the template surface may lie in each region of space. Patterns with overlapping instances must be modeled as the union of multiple infinite implicit replications where none of the individual replications have overlapping instances. The idea of mapping points in space to integers is similar to RangeFinder, however, RangeFinder maps a ball to a range of indices representing multiple instances that it may interfere with.

A. Pasko et al. discuss function-based models of periodic, volumetric microstructures [8]. They formulate the idea of infinite implicit replication in terms of a unit cell of geometry that is implicitly repeated using a replication function that maps points of space into the unit cell. Like Stolte, Pasko et al. only mention orthogonal-translation based patterns. However, they create curved and graded structures by warping axis aligned structures. This “post-processing warp” approach creates distorted instances of the unit cell. Cylindrical mapping, tapering, twisting, and a few other warps are demonstrated. O. Fryazinov et al. further discuss applying infinite implicit replication to microstructures, with a focus on modeling multi-scale structures [9]. The applications addressed in these papers are similar to ours, with a focus on microstructures and lattices. However, these approaches use the concept of a unit cell, which does not translate well into our method of modeling lattices. Furthermore, their structures are axis aligned, unless warped.

G. Elber uses trivariate B-Splines to map microstructures into a volume [3]. The Free-Form Deformation (FFD) approach [2] used by Elber allows more general warps than the ones demonstrated by Pasko et al., but this increase in flexibility comes with an increased computational cost for inverting the warp. B-Spline subdivision based solvers are commonly used to do the inverse mapping [10], from points in space to the unwarped geometry.

The use of multi-level lattices to architect useful, novel materials is a recent and developing area of research. Multi-level lattices have been manufactured that have a higher strength and stiffness, relative to density, when compared to existing single-level lattices [11]. These multi-level lattices also demonstrated high recoverability and resistance to failure from compression. Multi-level lattices have also been used to design scaffolds on which tissue such as bone can be grown, where a multi-level structure increases porosity to enhance nutrient transport [12].

Existing research on designing multi-level lattices, such as our Lattice-in-Lattice (LiL), is scarce. As mentioned, [9] and [3] model multi-level structures, but neither is appropriate for our definition of lattices since both rely on a regular, translational pattern of unit-cells. [11] designs self-similar, multi-level lattices recursively by repeating instances of a unit-cell along the beams of another instance of that unit-cell.

4 RangeFinder solution

In this section, we present and justify our **RangeFinder** solution to the problem of Ball-Interference Queries (BIQs) on a steady row *R* of balls.

Later, we extend this solution to handle BIQs on a steady row of beams.

4.1 Overall strategy for RangeFinder

We start here by restating the problem, outlining the general nature of our strategy, and revealing the mathematical formulation that it is based on.

We assume that the similarities discussed here are orientation-preserving (i.e., they have a positive dilation factor).

PROBLEM: Given a query ball *Q* and a steady row *R* with incremental similarity **S**, repetition count n, and template shape R_0, find a conservative, but hopefully tight, **candidate range *C*** that contains the indices of all shapes R_k that interfere with *Q*.

OVERALL STRATEGY: We divide the problem into two simpler RangeFinder sub-problems and return the intersection of the candidate ranges produced as solutions to these.

This approach requires computing a particular **canonical decomposition** of **S**, which is justified by the following property:

CANONICAL DECOMPOSITION: Similarity **S** may always be decomposed into a **commutative product** of two **primitive similarities**, one of which is a rotation and the other is either a translation or dilation.

Depending on the **dilation factor** d of **S**, one of the following two decompositions always exists:

1. If d = 1, **S** = **R**∘**T**, the product of a rotation **R** around an axis *A* with a translation **T** parallel to *A*
2. If d ≠ 1, **S** = **R**∘**D**, the product of a rotation **R** around an axis *A* with a dilation **D** about a fixed point F on *A*

If not directly available, the dilation factor d, of **S**, may be computed as the cube-root of the determinant of the 3x3 matrix that represents the linear part of the similarity [13].

The subtype of similarity associated with the d ≠ 1 case is a **swirl**. The subtype for the case when d = 1 is a **screw**, and although a screw may be considered as a special case of the general family of swirls, we use a different derivation to compute the decomposition and the candidate range.

The canonical decomposition of a screw is discussed in numerous papers and books (see for example [14]). The canonical decomposition for the swirl is presented in [4]. Note that these decompositions include special cases in which one or both primitive similarities degenerate to identities. These special cases may be easily detected and may require special treatments that are simple and hence are not discussed in this paper.

In the next section, we assume that **S** has been decomposed into its two primitive similarities as either **R**∘**T** or **R**∘**D**, and we explain how RangeFinder identifies the candidate range ***C*** for each one of the three possible primitive similarity types: **T**, **D**, **R**.

4.2 The essence of RangeFinder

First, we present the overall essence of RangeFinder.

Consider a point X_0 of the template shape R_0 and its images $X_k = \mathbf{S}^k \circ X_0$ in the consecutive instances R_k. Consider a **normalized mapping**, **M**, from space to the real line for which $\mathbf{M}(X_k) = \mathbf{M}(X_0) + k$. Our goal is to define such a mapping and to use it to identify indices k of the instances R_k that are guaranteed not to contain a given point X.

DEFINITION: M is a normalized mapping if and only if, for every point X in the valid domain, $\mathbf{M}(\mathbf{S}^{k} \circ X) = \mathbf{M}(X) + k$.

Assume that we have precomputed the **extents** $s_0 = \min(\mathbf{M}(X))$ and $e_0 = \max(\mathbf{M}(X))$ for all points X in R_0 under **M**. The **template-extents**, $[s_0, e_0]$, bounds the image of R_0 on the real line. Likewise, let the **query-extent**s $[s_q, e_q]$ bound the image of ball *Q* on the real line, where $s_q = \min(\mathbf{M}(Y))$ and $e_q = \max(\mathbf{M}(Y))$ for all points Y of *Q*.

THEOREM: *Q* may interfere with R_k only if $k \in \boldsymbol{C} = [\max(0, [s_q - e_0]), \min(n, [e_q - s_0])]$.

Proof: Shape R_k is mapped to the interval $[s_0+k, e_0+k]$. So, shape *Q* may interfere with R_k only if the intervals $[s_0+k, e_0+k]$ and $[s_q, e_q]$ overlap, yielding the conditions $s_q <= k+e_0$ and $k+s_0 <= e_q$, i.e., if $k \in [s_q-e_0, e_q-s_0]$. The integer indices that lie in this interval are the members of integer interval $[\lceil s_q-e_0 \rceil, \lfloor e_q-s_0 \rfloor]$. Finally, the indices need to be clamped to [0, n].

If ***C*** is empty, then *Q* interferes with no instances.

Thus, for each one of the RangeFinders for our three primitive similarities, we need to define the appropriate mapping **M** and closed-form expressions for computing the extent of the image of a ball under **M**. To determine their image bounds on the real line, the extent computation is applied to both the template shape and the query ball. The ball template shape is particularly useful because it is simple and a single ball or a union of such balls may be used as a bounding container for more complex shapes.

In the subsequent subsections, we consider primitive similarities **T**, **D**, and **R**, one at a time. For each such similarity, **X**, we explain how to define **M** (see definition marked by **X**-MAP) and how to compute the extent [s,e] of a ball *B*, with center C and radius r, under **M** (see definition marked by **X**-RANGE).

4.3 RangeFinder for translation T

Consider that **T** is a translation by vector V. Let O be an arbitrarily chosen origin.

T-MAP: The **normalized translation mapping**, **M**, for a translation by vector V with chosen reference point O is $\mathbf{M}(X) = OX \bullet V / V^2$.

JUSTIFICATION: **M**(X) is composed of a projection and a normalization. The expression $OX \bullet V / |V|$ computes the projection (measured from O) of X onto a line passing through O and parallel to V. The $1 / |V|$ normalization ensures that the measure reported by the mapping is expressed in the proper unit, so that, $\mathbf{M}(O+V) = 1$ (see Figure 6).

T-RANGE: The extents [s, e] of *B*, as a ball of center C and radius r, for a translation by vector V is $\mathbf{E}(B) = [\mathbf{M}(C) - r / |V|, \mathbf{M}(C) + r / |V|]$.

JUSTIFICATION: Interval [s, e] defines the smallest slice of space orthogonal to V that contains *B*. Let $c = \mathbf{M}(C)$ be the image of the center C. We extend the interval in both directions around c to ensure that it covers the projection of *B* onto a line parallel to V. However, we need to

normalize this extension to be expressed in the proper unit, which here is 1 / |V|, because the distance between a point X_0 of *B* and the corresponding point X_1 of $\mathbf{T}\circ B$ is |V|. (see Figure 6).

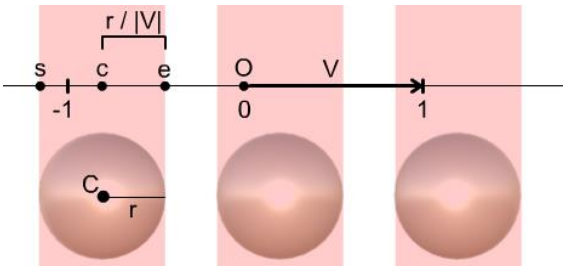

Figure 6: Normalized translation mapping and extent computation.

4.4 RangeFinder for dilation D

Consider that **D** is a dilation by factor d about fixed point F.

D-MAP: The **normalized dilation mapping**, **M**, for a dilation about fixed point F by dilation factor d > 0 is $\mathbf{M}(X) = \log_d(|FX|)$.

JUSTIFICATION: Let *sphere*(C,r) denote a sphere of center C and radius r. **M** maps all points of *sphere*(F,1) to 0 and all points of *sphere*(F,d) to 1. The mapping is logarithmic. For example, *sphere*(F,$d^{1/2}$) maps to ½. Hence, the same transformation, a dilation by $d^{1/2}$ maps *sphere*(F,1) to *sphere*(F,$d^{1/2}$) and *sphere*(F,$d^{1/2}$) to *sphere*(F,d), thus producing a steady pattern.

At point F, the image **M**(F) is undefined, because the logarithm of 0 is undefined. So, for simplicity and elegance, we assume that *B* does not contain F. (When it does, RangeFinder returns the full range [0,n]. However, a RangeFinder for dilation when *B* contains F can be constructed by assuming **M**(F) = -∞ when d>1 and **M**(F) = ∞ when 0<d<1.)

D-RANGE: The extents [s, e] of *B*, as a ball of center C and radius r, for a dilation about fixed point F by dilation factor d is $\mathbf{E}(B) = [\min(\log_d(|FC| \pm r)), \max(\log_d(|FC| \pm r))]$.

JUSTIFICATION: Interval [s, e] defines a spherical shell, i.e. the solid bounded by the union of *sphere*(F, |FC|-r) and *sphere*(F, |FC|+r), such that the larger sphere contains *B* and the smaller sphere does not. When d > 1, **M**(X) increases as |FX| increases, and s must be smaller than e, so points on *sphere*(F, |FC|-r) map to s and points on *sphere*(F, |FC|+r) map to e. However, when 0 < d < 1, **M**(X) decreases as |FX| increases, so points on the larger sphere map to s and points on the smaller sphere map to e. Consequently, $s = \min(\log_d(|FC| \pm r))$ and $e = \max(\log_d(|FC| \pm r))$. (See Figure 7).

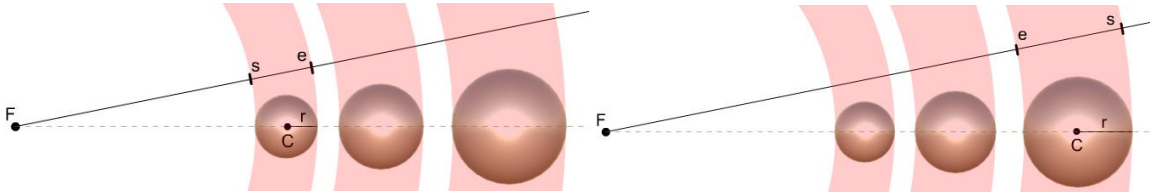

Figure 7: Normalized dilation mapping and extent computation with d > 1 on the left and 0 < d < 1 on the right.

4.5 RangeFinder for rotation R

For clarity and elegance, we present here the RangeFinder solution in which we assume that the total angle sustained by the entire row is less than 2π. In other words, the row does not even complete a single **wrap** around its axis. Such "tame" steady rows may be adequate for some applications, but not for others.

Our extension to more general steady rows that fully wrap around the axis at least once applies RangeFinder to each wrap, where each application may yield a different candidate range. A previously computed candidate range for the other primitive similarity (**T** or **D**) can be used to reduce the number of wraps tested.

Consider that **R** is a rotation by angle θ around axis *A*.

We assume below that *A* does not intersect ball *B*. When it does, RangeFinder returns the full range [0,n].

Let **angle**(N,V,W) be the angle that rotates N to V around W, assuming that N and V are vectors perpendicular to W and that |N|=|W|=1. It may be computed as **atan2**((W×N)•V, N•V).

R-MAP: The **normalized rotation mapping**, **M**, by rotation angle θ about axis *A* of direction W and given an arbitrary reference vector N orthogonal to *A* returns **M**(X,b) = **angle**(N, $\underline{X_{[A]}X}$, W)/θ + 2πb/|θ|, where $\underline{X_{[A]}X}$ is the normalized vector from $X_{[A]}$ to X, and where b, the **branch ID**, represents the number of full rotations around *A* to be added to the angle measurement. If no b is given, as in **M**(X), then **M** represents a mapping to the infinite values as the union of **M**(X,b) for all integers b.

JUSTIFICATION: **M** is composed of the sum of a normalized angle measurement **angle**(N, $\underline{X_{[A]}X}$, W)/θ and the normalized angle 2πb/|θ| of b rotations around *A*. The normalization ensures that the measure reported by the mapping is expressed in the proper unit, so that, there exists a value b such that **M**($X_{[A]}$+N°(θ,W), b) = 1, where $X_{[A]}$ may any point on *A*. (See Figure 8). The b rotations around *A* are normalized by the absolute value of θ, because increasing b should increase the result, regardless of the direction of rotation.

We define a **branch** of ID b to be the interval [(2πb - π)/|θ|, (2πb + π)/|θ|) on the real line. A single value of image **M**(X), of any point X, lies in each branch b, for any integer b.

R-RANGE: The extents [s, e] of *B*, as a ball of center C and radius r, for a rotation by angle θ around axis *A* is **E**(*B*,b) = [**M**(C,b) - h/|θ|, **M**(C,b) + h/|θ|] where h = $\mathbf{sin}^{-1}$(r / |$CC_{[A]}$|) and b is the desired branch for C to map to. If no b is given, then the extent **E**(*B*) is the union of the infinite intervals corresponding to every possible b.

JUSTIFICATION: Interval [s, e] defines the smallest wedge of space extending infinitely radially from *A* that contains *B*. This wedge is the intersection of two linear half-spaces, each containing *A* in its bounding plane. Hence, we represent that wedge by the angles of their oriented

bounding planes around *A*, with respect to reference vector N. Let c = M(C,b) be the image of center C in branch b. We want to extend the interval in both directions around c to ensure that it covers the image of all points in *B*. Doing so requires extending in both directions by h, half of the angle (from $C_{[A]}$) subtending *B*, where the extension is normalized to the proper unit, $1/|\theta|$, because the angle between a point X_0 of *B* and the corresponding point X_1 of $\mathbf{R}\circ\mathbf{B}$ around *A* is θ.

Although c is in branch b, ***either*** s might be in branch b-1 ***or*** e might be in branch b+1, because we extended the interval around c, and c may be anywhere in branch b, including its boundary. If s is in branch b-1, then we can add $2\pi/|\theta|$ to both s and e, to ensure each lies in either branch b or b+1.

The image **E**(*Q*) of query ball *Q* must map to infinite branches, because it may overlap with the image of *R* in any branch. However, the extents of the entire row *R* can be restricted to lie in only branches 0 and 1, because the total angle sustained by *R* is less than 2π, so the only portions of **E**(*Q*) that may overlap the image of *R* are Q_0 = **E**(*Q*,0) and Q_1 = **E**(*Q*,1). Evaluating RangeFinder with both query-extents, $[s_q,e_q] = Q_0$ and $[s_q,e_q] = Q_1$, each along with template-extents $[s_0,e_0]$ = **E**(R_0,0) will yield two ranges, only one of which may be non-empty, because a continuous mapping of the steady row covers less than $2\pi/|\theta|$ on the real line. If one range is non-empty, then it is returned by RangeFinder.

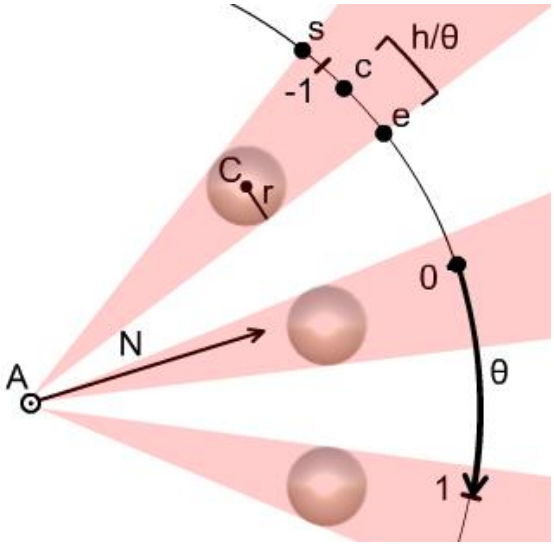

Figure 8: Normalized rotation mapping and extent computation.

4.6 Combining ranges from the two primitive similarities

Assume the candidate ranges [a,b] returned by RangeFinder for translation or dilation and [c,d] returned by RangeFinder for rotation have already been computed. The final candidate range is the intersection of the two ranges and can be computed as [max(a,c), min(b,d)].

5 Computing the extents of a beam

Here we describe how to compute the extents of beams defined as the convex hull of two balls, *X* and *Y*.

Computing the extents of a beam directly, rather than approximately with a bounding ball, reduces the number of false-positive candidate indices returned by RangeFinder, sometimes significantly, because a bounding ball is a poor approximation for a long, thin beam.

Let $[s_X,e_X]$ and $[s_Y,e_Y]$ be the extents of balls *X* and *Y* respectively, under a primitive similarity.

Translation extents: $[s,e] = [\min(s_X, s_Y), \max(e_X, e_Y)]$.

Justification: The most extreme points of the beam along the direction of translation must be in *X* or *Y*, so the extent is constructed from only the min and max values of the ball extents. See Figure 9.

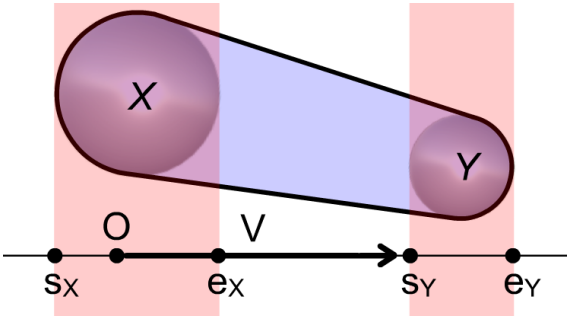

Figure 9: Computation of beam extents for translation primitive.

Dilation extents: $[s,e] = [\min(s_X, s_Y, \log_d f), \max(e_X, e_Y, \log_d f)]$, where f is the distance from the fixed point F to the interior of the beam.

Justification: The point of the beam farthest from F must be in *X* or *Y*. However, the point of the beam closest to F may or may not be in *X* or *Y*. Including $\log_d f$ accounts for this possibility, and it is used in both min and max to account for dilation factors less than or greater than 1. See Figure 10.

[15] describes an efficient computation of the distance f from a point to a beam, called “cone-sphere”.

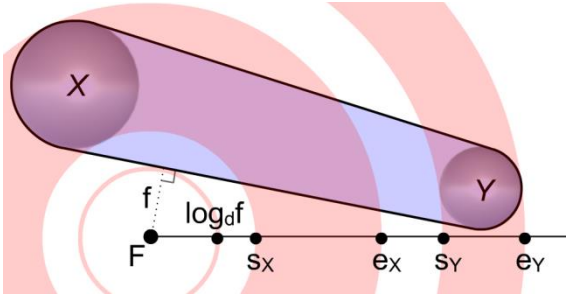

Figure 10: Computation of beam extents for dilation primitive with dilation factor d>1. F is closer to the conical part of the beam than to the balls.

Rotation extents: [min(s_X, y - h_Y/|θ|), max(e_X, y + h_Y/|θ|)], where θ is the rotation angle around axis *A*, y = (s_X+e_X)/2 + **angle**($C_{X[A]}C_X$, $C_{Y[A]}C_Y$, W)/θ is the **proper mapping** of C_Y relative to the mapping of C_X (described in justification), W is the direction of *A*, and h_Y = **sin**$^{-1}$(r_Y / |$C_YC_{Y[A]}$|) is half of the angle subtending *Y* around *A*. $C_{X[A]}$ denotes the closest projection of the center of *X* onto *A*. We assume *A* does not intersect the beam.

Justification: Consider starting with the extents [s_X,e_X] of *X*. The extents [s_Y,e_Y] of *Y* must be computed relative to s_X and e_X, since computing s_Y and e_Y from *Y*, in isolation, will not guarantee a mapping to the correct branch relative to the mapping of *X*. The relative offset from a mapping of C_X to the proper mapping of C_Y is **angle**($C_{X[A]}C_X$, $C_{Y[A]}C_Y$, W)/θ, which is composed of the angle from C_X to C_Y around *A* normalized by the primitive's rotation angle. From the correct mapping of C_Y, the extents of *Y* can be computed by subtracting (for s_Y) and adding (for e_Y) half of the angle subtending *Y* from $C_{Y[A]}$, normalized by the primitive's rotation angle. Given the extent of *X* and correct relative extent of Y, the extent of the beam is computed by combining the two in the same manner as the translation case. See Figure 11.

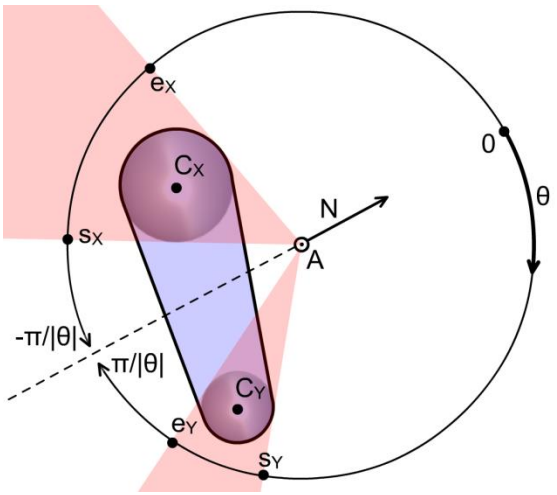

Figure 11: Computation of beam extents for rotation primitive. Given a mapping of C_X into branch 0, the correct mapping of C_Y here is into branch -1.

6 RangeFinder in steady lattices

Without loss of generality, assume we are performing a BIQ on a lattice of which the base-group has a single ball with a single beam *B* originating from it.

We want to compute the set of group index triplets (i,j,k) of all groups whose instance of *B* interferes with the query ball *Q*.

The naive solution, iterating through all valid index triplets and testing each instance of *B* for interference with *Q*, takes O(uvw) time.

We have earlier introduced RangeFinder for performing BIQs on steady rows in expected constant time.

The lattice can be organized into u by v steady rows [4], allowing for BIQs in expected O(uv) time by using RangeFinder on each row. These steady rows are each composed of w repetitions, an incremental similarity **W**, and a template shape as an instance of *B* in the **base-slab**, where the base-slab is the union of all groups *G*[*,*,0], where asterisks denote any valid index. Each use of RangeFinder returns a range of candidate values ***C*** so that for each value k in ***C***, a ball-vs-beam interference test needs to be performed between the query ball and the k^{th} instance of the steady row.

6.1 Special case improvements

For special configurations of a steady lattice, the expected time complexity of a BIQ can be further reduced, from O(uv), to either O(u) or O(1), using simple modifications to the algorithm above.

6.1.1 O(u) BIQ case

Consider the case in which the BIQ algorithm can be modified to reduce the time complexity to an expected O(u). We want to perform u constant-time procedures, one for each group of the **base-row** (the union of all groups *G*[*,0,0]), where each computes the set of (j,k) pairs identifying instances of *B* that must be tested for ball-vs-beam interference.

The set of (j,k) pairs is computed using two RangeFinders:

1. to compute a range of k-values, using a steady row of w repetitions, incremental similarity **W**, and a template shape as an instance of *B* in the base-row;
2. to compute a range of j-values, using a steady row of v repetitions, incremental similarity **V**, and the same template shape as the RangeFinder for k-values.

The (j,k) pairs to be tested are formed as all possible combinations of values from both ranges.

The O(u) BIQ modification is valid if the following three conditions are met:

1. The row of balls in the **V** direction, of which the template shape is the ball in *G*[0,0,1], has an incremental similarity of **V**.
2. Applying **V** to the beam originating at *G*[*,0,0] does not change its extents with respect to the primitives of **W**.
3. Applying **W** to the beam originating at *G*[*,0,0] does not change its extents with respect to the primitives of **V**.

The first condition implies that the lattice's rows of beams in the **V** direction are steady. Rows of beams in the **V** direction are not guaranteed to be steady, unlike in the **W** direction, as proven in [4], but they must be steady in order for RangeFinder to work. This condition implies steadiness for the rows of beams in the **V** direction because any row of balls of which the template shape is a ball in the base-slab is steady in the **V** direction with an incremental similarity of **V**, and it is only when all rows of balls along the **V** direction that are further along the **W** direction than the

base-slab also have an incremental similarity of **V** that all beams must be connecting two balls that each belong to a steady row in the **V** direction with an incremental similarity of **V**, and if the two end balls of a beam are transformed by the same similarity **V**, then the whole beam is transformed by the similarity **V**.

The second condition implies that the range of k-values, returned by the RangeFinder on the row in the **W** direction, is the same range that would be returned by a RangeFinder on any of the other v rows in the **W** direction, for a fixed index i, and so, it is valid to use the single returned range as the set of indices into each of the rows. The condition implies this because the only parameter into RangeFinder that varies between the rows is the template shape, which is transformed by **V** between consecutive rows, and if applying **V** to the template shape does not change its extents with respect to the primitives of **W**, then the output of RangeFinder will not change.

The third condition has a symmetric justification to the second.

6.1.2 O(1) BIQ case

Now consider the case in which the BIQ algorithm can be modified to reduce the time complexity to an expected O(1). It is much like the O(u) BIQ case, except more restrictive. Three RangeFinders are performed, one for a row in each direction, **U**, **V**, and **W**, where each returns a range of indices of for which all possible combinations form the (i,j,k) index triplets identifying instances of *B* to test for ball-vs-beam interference.

The O(1) BIQ modification is valid if the following conditions are met:

1. The row of balls in the **V** direction, of which the template shape is the ball at *G*[0,0,1], has an incremental similarity of **V**.
2. The row of balls in the **U** direction, of which the template shape is the ball at *G*[0,0,1], has an incremental similarity of **U**.
3. Applying **U** or **V** to the beam originating at *G*[0,0,0] does not change its extents with respect to the primitives of **W**.
4. Applying **U** or **W** to the beam originating at *G*[0,0,0] does not change its extents with respect to the primitives of **V**.
5. Applying **V** or **W** to the beam originating at *G*[0,0,0] does not change its extents with respect to the primitives of **U**.

These conditions can be justified similarly to the conditions from the O(u) BIQ case.

7 Results

Here we demonstrate performance improvements, due to RangeFinder, in accelerating BIQs against steady rows. We do this first by demonstrating improvements in two-level LiL generation, and then we show results from a purer test of 100 random BIQs against a steady lattice.

Table 1 shows the time taken to compute the sets of beams, to be displayed, for the LiLs shown in Figure 12. Each test was performed on 8 threads, where the parameters that implicitly describe the lattices were available to each thread. Each thread was assigned an approximately equal number of groups from the fine lattice, divided along the **U** direction. For each valid beam *B* originating from the assigned groups, its assigned thread analyzed *B* for membership in the final LiL by performing two BIQs, one for both balls of *B*, against the coarse lattice.

The results clearly show a benefit of RangeFinder accelerated BIQs over naïve BIQs. However, the benefit of O(uv) RangeFinder BIQs over naïve O(uvw) BIQs may not be as high as expected. The reason for this is that the BIQs were performed against coarse lattices, which contained no more than 9 groups in the **W** direction, which is very small. Therefore, we also performed pure BIQ tests on more complex steady lattices, without the LiL concept.

Consider the curved, steady lattice in Figure 2, with 100^3 groups. For two cases, one with naïve BIQs and the other with O(uv) RangeFinder BIQs, we performed, on a single thread, 100 BIQs with query balls of which the centers were randomly placed in the axis-aligned bounding box of the lattice and of which the radii were randomly chosen to be between 90%-110% of the average radius of the lattice's balls. For the naïve case, it took 4.04 minutes to perform all 100 BIQs. For the O(uv) RangeFinder case, it took **only 0.0894 minutes** to perform the 100 BIQs, yielding a **45x improvement** over the naïve case. We performed the same tests of 100 random BIQs on the fine lattices from the cylindrical LiL and the regular LiL in Figure 12. For the cylindrical lattice, the test of 100 naïve BIQs took 6.38 minutes while the test with **100 O(u) BIQs took 0.00173 minutes**, yielding a **3700x improvement**. For the regular lattice, the test of 100 naïve BIQs took 1.37 minutes while the test with **100 O(1) BIQs took 0.000252 minutes**, yielding a **5500x improvement**.

These tests were performed on a on a machine with an I7-6820HQ@2.70GHz and 32GB DDR4 RAM.

	Regular lattice	**Cylindrical lattice**	**Warped lattice**
O(uvw) BIQ, Naïve	10.3 minutes	37.2 minutes	19.10 minutes
O(uv) BIQ, Standard	2.20 minutes	5.98 minutes	5.87 minutes
O(u) BIQ	0.31 minutes	0.541 minutes	-
O(1) BIQ	0.067 minutes	-	-

Table 1: Time taken to compute the sets of beams displayed in the LiLs shown in Figure 12. Results are displayed when either naïve, O(uv), O(u), or O(1) BIQs are used.

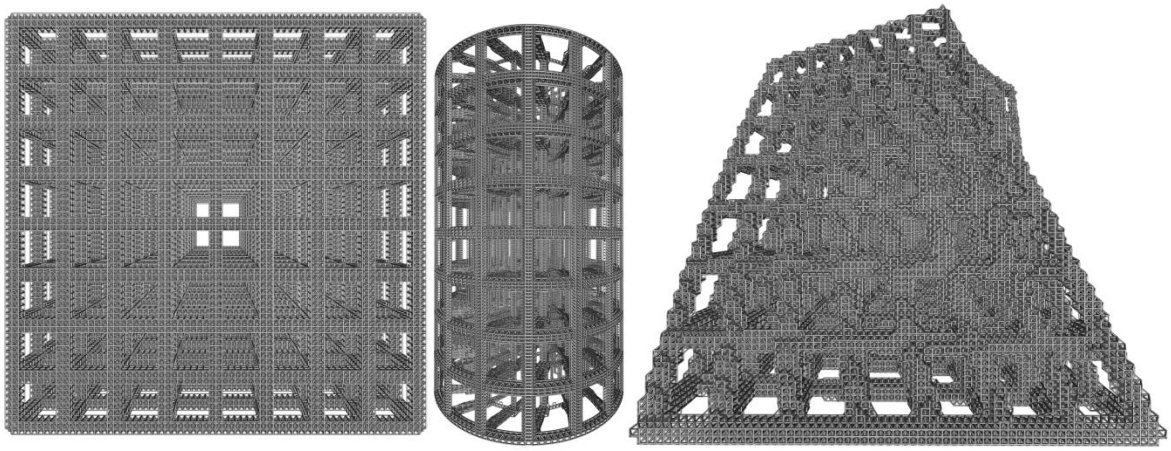

Figure 12: (Left) LiL created from a regular fine lattice of 75^3 groups and a regular coarse lattice of 9^3 groups. (Middle) LiL created from a cylindrical, steady fine lattice of 32x128x128 groups and a cylindrical, steady coarse lattice of 5x17x9 groups. (Right) LiL created from a regular fine lattice of 64^3 groups and a semi-regular coarse lattice of 8^3 groups. All three examples have a fine lattice of 2 balls per group and 14 edge-patterns and have a coarse lattice of 1 ball per group and 3 edge-patterns.

8 Conclusion

Here we review our contributions, discuss limitations, and propose questions to be explored in future work.

We presented a simple solution for identifying, in expected constant time, a conservative but tight range of instances, from a steady row of balls that is guaranteed to contain all instances that interfere with a query ball *Q*. We extended that solution for steady rows of beams, and we applied the extension to compute, in expected O(uv) time, the list of beams of a steady lattice that interfere with query ball *Q*. We described conditions and algorithm modifications to improve the ball-interference query (BIQ) complexity to O(u) or O(1) for special cases of steady lattices. We applied BIQs to steady lattices to generate multi-level lattices, called Lattice-in-Lattice (LiL), and we reported performance improvements from the application of RangeFinder to LiL generation.

There are a few limitations that were not addressed in this paper.

One limitation is that we do not yet have a general solution to improve the BIQ complexity to the theoretical optimum in all cases. Consider the cylindrical lattice from Figure 12. A human could design a O(1) BIQ for this case, but our solution is O(u) time. We wish to further improve BIQ costs for a wider variety of lattices.

Another, more minor, limitation is that RangeFinder sometimes returns false-positive candidates when the query ball *Q* clearly cannot interfere with any instance. For example, given a translational steady row, if *Q* moves any distance orthogonally to the direction of translation, the candidate range will not change. Early rejection tests may be developed to cull non-interfering queries before RangeFinder is ever performed.

Finally, we introduced the concept of Lattice-in-Lattice (LiL), to construct a multi-level lattice from the beams of a fine lattice that have both balls interfering with a coarse lattice, but we barely discussed it. To our knowledge, LiL has not been introduced previously. Further work should be done to explore the design and properties of LiLs. One direction of research might be to constrain the resulting lattice to be fully contained by the coarse lattice, which our algorithm does not do.

Steady lattices are an appealing option for generative design of materials and other structures. They may be used alone or as efficient primitives in more complex structures. Several queries may be built from RangeFinder accelerated BIQs, and other queries on steady lattices have been explored elsewhere, such as efficient computation of integral properties [4]. We suspect that these benefits will enable generative design of new lattice-based materials that could not be previously realized, because analysis of such complex structures has traditionally been too costly to perform in an optimization loop.

9 Acknowledgements

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

10 References

[1] Ashish Gupta, George Allen, and Jarek Rossignac. Quador: Quadric-of-revolution beams for lattices. *Computer-Aided Design*, 102:160–170, 2018.

[2] Thomas W Sederberg and Scott R Parry. Free-form deformation of solid geometric models. *ACM SIGGRAPH computer graphics*, 20(4):151–160, 1986.

[3] Gershon Elber. Precise construction of micro-structures and porous geometry via functional composition. In *International Conference on Mathematical Methods for Curves and Surfaces*, pages 108–125. Springer, 2016.

[4] Ashish Gupta, Kelsey Kurzeja, Jarek Rossignac, George Allen, Pranav Kumar, and Suraj Musuvathy. Designing and processing parametric models of steady lattices. in review.

[5] John C Hart. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12(10):527–545, 1996.

[6] Mark Pauly, Niloy J Mitra, Johannes Wallner, Helmut Pottmann, and Leonidas J Guibas. Discovering structural regularity in 3d geometry. *ACM transactions on graphics (TOG)*, 27(3):43, 2008.

[7] Nilo Stolte. Infinite implicit replication: Case study for voxelizing and representing cyclical parametric surfaces implicitly. In *Shape Modeling International, 2002. Proceedings*, pages 105–274. IEEE, 2002.

[8] Alexander Pasko, Oleg Fryazinov, Turlif Vilbrandt, Pierre-Alain Fayolle, and Valery Adzhiev. Procedural function-based modelling of volumetric microstructures. *Graphical Models*, 73(5):165–181, 2011.

[9] Oleg Fryazinov, Turlif Vilbrandt, and Alexander Pasko. Multi-scale space-variant frep cellular structures. *Computer-Aided Design*, 45(1):26–34, 2013.

[10] Nicholas M Patrikalakis and Takashi Maekawa. *Shape interrogation for computer aided design and manufacturing*. Springer Science & Business Media, 2009.

[11] Lucas R Meza, Alex J Zelhofer, Nigel Clarke, Arturo J Mateos, Dennis M Kochmann, and Julia R Greer. Resilient 3d hierarchical architected metamaterials. *Proceedings of the National Academy of Sciences*, 112(37):11502–11507, 2015.

[12] Paul F Egan, Stephen J Ferguson, and Kristina Shea. Design of hierarchical three-dimensional printed scaffolds considering mechanical and biological factors for bone tissue engineering. *Journal of Mechanical Design*, 139(6):061401, 2017.

[13] Peter Shirley and Steve Marschner. *Fundamentals of Computer Graphics*. AK Peters, Ltd., 2009.

[14] Ignacio Llamas, Byungmoon Kim, Joshua Gargus, Jarek Rossignac, and Chris D Shaw. Twister: a space-warp operator for the two-handed editing of 3d shapes. *ACM transactions on graphics (TOG)*, 22(3):663–668, 2003.

[15] Aurelien Barbier and Eric Galin. Fast distance computation between a point and cylinders, cones, line-swept spheres and cone-spheres. *Journal of Graphics tools*, 9(2):11–19, 2004.