Numerically Accurate Hyperbolic Embeddings Using Tiling-Based Models

Tao Yu
Department of Computer Science
Cornell University
Ithaca, NY, USA
`tyu@cs.cornell.edu`

Christopher De Sa
Department of Computer Science
Cornell University
Ithaca, NY, USA
`cdesa@cs.cornell.edu`

Abstract

Hyperbolic embeddings achieve excellent performance when embedding hierarchical data structures like synonym or type hierarchies, but they can be limited by numerical error when ordinary floating-point numbers are used to represent points in hyperbolic space. Standard models such as the Poincaré disk and the Lorentz model have unbounded numerical error as points get far from the origin. To address this, we propose a new model which uses an integer-based tiling to represent *any* point in hyperbolic space with provably bounded numerical error. This allows us to learn high-precision embeddings without using BigFloats, and enables us to store the resulting embeddings with fewer bits. We evaluate our tiling-based model empirically, and show that it can both compress hyperbolic embeddings (down to 2% of a Poincaré embedding on WordNet Nouns) and learn more accurate embeddings on real-world datasets.

1 Introduction

In the real world, valuable knowledge is encoded in datasets with hierarchical structure, such as the IBM Information Management System to describe the structure of documents, the large lexical database WordNet [14], various networks [8] and natural language sentences [24, 5]. It is challenging but necessary to embed these structured data for the use of modern machine learning methods. Recent work [11, 26, 27, 7] proposed using *hyperbolic spaces* to embed these structures and has achieved exciting results. A hyperbolic space is a manifold with constant negative curvature and endowed with various geometric properties, in particular, Bowditch [4] shows that any finite subset of an hyperbolic space looks like a finite tree according to the definition in [18]. Therefore, the hyperbolic space is well suited to model hierarchical structures.

A major difficulty that arises when learning with hyperbolic embeddings is the numerical instability, sometimes informally called "the NaN problem". Models of hyperbolic space commonly used to learn embeddings, such as the Poincaré ball model [26] and the Lorentz hyperboloid model [27], suffer from significant numerical error caused by floating-point computation and amplified by the ill-conditioned Riemannian metrics involved in their construction. To address this when embedding a graph, one technical solution exploited by Sarkar [32] is to carefully scale down all the edge lengths by a factor before embedding, then recover the original distances afterwards by dividing by the factor. However, this scaling increases the distortion of the embedding, and the distortion gets worse as the scale factor increases [30]. Sala et al. [30] suggested that, to produce a good embedding in hyperbolic space, one can either increase the number of bits used for the floating-point numbers or increase the dimension of the space.

While these methods can greatly improve the accuracy of an embedding empirically, they come with a computational cost, and the floating-point error is still unbounded everywhere. Despite these previous

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

adopted methods, as points move far away from the origin, the error caused by using floating-point numbers to represent them will be unbounded. Even if we try to compensate for this effect by using BigFloats (non-standard floating-point numbers that use a large quantity of bits), no matter how many bits we use, there will always be numerical issues for points sufficiently far away from the origin. No amount of BigFloat precision is sufficient to accurately represent points *everywhere* in hyperbolic space.

To address this problem, it is desirable to have a way of representing points in hyperbolic space that: (1) can represent any point in the space with small fixed bounded error; (2) supports standard geometric computations, such as hyperbolic distances, with small numerical error; and (3) avoids potentially expensive BigFloat arithmetic.

One solution is to avoid floating-point arithmetic and do as much computation as possible with integer arithmetic, which introduces no error. To gain intuition, imagine solving the same problem in the more familiar setting of the Euclidean plane. A simple way to construct a constant-error representation is by using the integer-lattice square tiling (or tessellation) [9] of the Euclidean plane. With this, we can represent any point in the plane by (1) storing the coordinates of the square where the point is located as integers and (2) storing the coordinates of the point within that square as floating point numbers. In this way, the worst-case representation error (Definition 1) will only be proportional to the machine epsilon of the floating-point format—but not the distance of the point from the origin.

We propose to do the same thing in the hyperbolic space: we call this a *tiling-based model*. Given some tiling of hyperbolic space, we can represent a point in hyperbolic space as a pair of (1) the tile it is on and (2) its position within the tile represented with floating point coordinates. In this paper, we show how we can do this, and we make the following contributions:

- We identify tiling-based models for both the hyperbolic plane and for higher-dimensional hyperbolic space in various dimensions. We prove that the representation error (Definition 1) is bounded by a fixed value, further, the error of computing distances and gradients are independent of how far the points are from the origin.
- We show how to compute efficiently over tiling-based models, and we offer algorithms to compress and learn embeddings for real-world datasets.

The reminder of this paper is organized as follows. In Section 2, we discuss related work regarding hyperbolic embeddings on various models. In Section 3, we detail the standard models of hyperbolic space which we use in our theory and experiments. In Section 4, we introduce the L-tiling model and show how it can be used to accurately represent any point in the hyperbolic plane (2-dimensional hyperbolic space). In Section 5, we show how to use the L-tiling model to learn embeddings with traditional manifold optimization algorithms. In Section 6, we develop the H-tiling model, which generalizes our methods to higher dimensional spaces. Finally, in Section 7, evaluate our methods on two different tasks: (1) compressing a learned embedding and (2) learning embeddings on multiple real-world datasets.

2 Related Work

Hyperbolic space [1] is a simply connected Riemannian manifold with constant negative (sectional) curvature, which is analogous to a high dimensional sphere with constant positive curvature. The negative-curvature metric of the hyperbolic space results in very different geometric properties, which makes it widely employed in many settings. One noticeable property is the volume of the ball in hyperbolic space: it increases exponentially with respect to the radius (for large radius), rather than polynomially as in the Euclidean case [6]. For comparison to hierarchical data, consider a tree with branching factor b, where the number of leaf nodes increases exponentially as the tree depth increases [27], this property makes hyperbolic space particularly well suited to represent hierarchies.

Nickel and Kiela [26] introduced the Poincaré embedding for learning hierarchical representations of symbolic data, which captured the attention of the machine learning community. The Poincaré ball model of hyperbolic space was used to embed taxonomies and graphs with state-of-the-art results in link prediction and lexical entailment. Similarly, it was also proposed in [7] to learn neural embeddings of graphs in hyperbolic space, where the performances on downstream tasks were improved significantly. The Poincaré ball model was used in several subsequent works, including unsupervised learning of word and sentence embeddings [35, 13], directed acyclic graph embeddings and hierarchical relations learning using a family of nested geodesically convex cones [16]. Further, Ganea et al. [15] proposed hyperbolic neural networks to embed sequential data and perform classification based on the Poincaré ball model.

In a later work [27], the Poincaré model and the Lorentz model of hyperbolic space were compared to learn the same embeddings, and the Lorentz model was observed to be substantially more efficient than the Poincaré model to learn high-quality embeddings of large taxonomies, especially in low dimensions. Similarly, Gulcehre et al. [20] built the new hyperbolic attention networks on top of the Lorentz model rather than the Poincaré model. Further along this direction, Gu et al. [19] explored a product manifold combining multiple copies of different model spaces to get better performance on a range of datasets and reconstruction tasks. This suggests that the numerical model used for learning embeddings can have significant impact on its performance. Sala et al. [30] analyzed the tradeoffs between precision and dimensionality of hyperbolic embeddings to show this is a fundamental problem when using float arithmetic. More broadly, different models have been used in different tasks like hierarchies embedding [26], text embedding [35, 13] and question answering system [34]. However, all these models can be limited by numerical precision issues.

3 Models of Hyperbolic Space

Typically, people work with hyperbolic space by using a *model*, a representation of hyperbolic space within Euclidean space. There exists multiple important models for hyperbolic space, most notably the Poincaré ball model, the Lorentz hyperboloid model, and the Poincaré upper-half space model [1], which will be described in this section. These all model the same geometry in the sense that any two of them can be related by a transformation that preserves all the geometrical properties of the space, including distances and gradient [6]. Generally, one can choose whichever model is best suited for a given task [27].

Poincaré ball model. The Poincaré ball model is the Riemannian manifold $(\mathcal{B}^n, g_p)$, where $\mathcal{B}^n = \{\boldsymbol{x} \in \mathbb{R}^n : \|\boldsymbol{x}\| < 1\}$ is the open unit ball. The metric and distance on $\mathcal{B}^n$ are defined as

$$g_p(\boldsymbol{x}) = \left(\frac{2}{1-\|\boldsymbol{x}\|^2}\right)^2 g_e, \quad d_p(\boldsymbol{x}, \boldsymbol{y}) = \operatorname{arcosh}\left(1 + 2\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{(1-\|\boldsymbol{x}\|^2)(1-\|\boldsymbol{y}\|^2)}\right),$$

where g_e is the Euclidean metric, due to its conformality (angles measured at a point are the same as they are in the actual hyperbolic space), its convenient parameterization, and clear visualization results, the Poincaré ball model is widely used in many applications. However, it can be seen from this equation that the distance within the Poincaré ball model changes rapidly when the points are close to the boundary (i.e. $\|x\| \approx 1$), and hence it is very poorly conditioned.

Lorentz hyperboloid model. The Lorentz model is arguably the most natural model algebraically for hyperbolic space. It is defined in terms of a nonstandard scalar product called the *Lorentzian scalar product*. For two-dimensional hyperbolic space, it is defined as

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle_L = \boldsymbol{x}^T g_l \boldsymbol{y}, \quad \text{where} \quad g_l = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The Lorentz model of 2-dimensional hyperbolic space is then defined as the Riemannian manifold $(\mathcal{L}^2, g_l)$, where $\mathcal{L}^2$ and associated distance function are given as

$$\mathcal{L}^2 = \{\boldsymbol{x} \in \mathbb{R}^3 : \langle \boldsymbol{x}, \boldsymbol{x} \rangle_L = -1, x_0 > 0\}, \quad d_l(\boldsymbol{x}, \boldsymbol{y}) = \operatorname{arcosh}(-\langle \boldsymbol{x}, \boldsymbol{y} \rangle_L).$$

This model generalizes easily to higher dimensional spaces by increasing the number of 1s on the diagonal of the matrix g_l. Points in the Lorentz model lie on the upper sheet of a two-sheeted n-dimensional hyperbola. Unlike the Poincaré disk model, which is confined in the Euclidean unit ball, the Lorentz model is unbounded. However, like other models, it can experience severe numerical error for points far away in hyperbolic distance from the origin as shown in Theorem 1.

Definition 1. *[Representation error] We are concerned with representing points in hyperbolic space* $\mathbb{H}^n$ *using floating-points* fl*. Define the* representation error *of a particular point* $x \in \mathbb{H}^n$ *as* $\delta_{\text{fl}}(x) = d_{\mathbb{H}^n}(x, \text{fl}(x))$*, and the* worst case representation error *of floating-points representation as a function of the distance-to-origin* d*, which is the maximum representation error of any point with a distance-to-origin at most* d*,*

$$\delta_{\text{fl}}^d = \max_{x \in \mathbb{H}^n,\ d_{\mathbb{H}^n}(x, O) \le d} \delta_{\text{fl}}(x).$$

Theorem 1. *The worst-case representation error (Definition 1) in the Lorentz model using floating-point arithmetic (with machine epsilon* ϵ_m*) is* $\delta_l^d = \operatorname{arcosh}(1 + \epsilon_m(2\cosh^2(d) - 1))$*, where* d *is the hyperbolic distance to origin. This becomes* $\delta_l^d = 2d + \log(\epsilon_m) + o(\epsilon_m^{-1}\exp(-2d))$ *if* $d = O(-\log\epsilon_m)$*.*

Poincaré half-space model. The Poincaré upper half-space model of the hyperbolic space is the manifold $(\mathcal{U}^n, g_u)$, where $\mathcal{U}^n = \{\boldsymbol{x} \in \mathbb{R}^n : x_n > 0\}$ is the upper half space of the n-dimensional Euclidean space. The metric and corresponding distance function is

$$g_u(\boldsymbol{x}) = \frac{g_e}{x_n^2}, \quad d_u(\boldsymbol{x}, \boldsymbol{y}) = \operatorname{arcosh}\left(1 + \frac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{2x_n y_n}\right)$$

Here g_e is the Euclidean metric. The half-space model is also unbounded and conformal, and has a particularly nice interpretation in two dimensions as a mapping on the complex plane. Note that although it is unbounded, this model still has an "edge" where $x_n = 0$ and it can exhibit numerical issues similar to the Poincaré ball as x_n approaches 0.

4 A Tiling-Based Model for Hyperbolic Plane

As we saw in the previous section, the standard models of hyperbolic space exhibit unbounded numerical error as the hyperbolic distance from the origin increases. In this section, we will describe a tiling-based model that avoids this problem. Our model is constructed on top of the Lorentz model for the two-dimensional hyperbolic plane $\mathbb{H}^2$.

In hyperbolic geometry, a uniform tiling [9, 12, 33] is an edge-to-edge filling of the hyperbolic plane which has regular congruent polygons as faces and is vertex-transitive (there is an isometry mapping any vertex onto any other) [28]. Any tiling is associated with a discrete group G of orientation-preserving isometries of $\mathbb{H}^2$ that preserve the tiling [38, 22]; discrete subgroups of isometries of $\mathbb{H}^2$ (like G) are called *Fuchsian groups* [21, 2, 37]. Importantly, not only does the tiling determine G, but G also determines the shape of the tiling. One way to see this is to consider the images of a single point in $\mathbb{H}^2$ under the group action G (called an *orbit* of the action). Then the Voronoi diagram associated with the orbit (which partitions each point in $\mathbb{H}^2$ into tiles based on which point in the orbit it is closest to) will be a regular tiling of $\mathbb{H}^2$. This equivalence between tilings and groups means that we can reason about tilings by reasoning about Fuchsian groups.

In the 2-dimensional Lorentz model, isometries can be represented as matrices operating on $\mathbb{R}^3$ that preserve the Lorentzian scalar product. That is, a matrix $A \in \mathbb{R}^{3\times 3}$ is an isometry if $A^T g_l A = g_l$. If we have some discrete group of isometries G, and we choose the tile which contains the origin to be the *fundamental domain* [37, 36] F, then we can start to define a tiling-based model on top of the Lorentz model of the hyperbolic plane.

L-tiling model. Our first insight is to represent points in the hyperbolic plane as a pair consisting of an element of the group and an element of the fundamental domain. The point represented by this pair is the result of the group element applied to the fundamental domain element. For example, the ordered pair $(\boldsymbol{g}, \boldsymbol{x}) \in G \times F$ would represent the point $\boldsymbol{gx}$. The L-tiling model of the hyperbolic plane is defined as the Riemannian manifold $(\mathcal{T}_l^n, g_{lt})$, where $g_{lt} = g_l$ and

$$\mathcal{T}_l^n = \{(\boldsymbol{g}, \boldsymbol{x}) \in G \times F : \langle \boldsymbol{x}, \boldsymbol{x}\rangle_L = -1\}, \quad d_{lt}((\boldsymbol{g}_x, \boldsymbol{x}), (\boldsymbol{g}_y, \boldsymbol{y})) = \operatorname{arcosh}\left(-\boldsymbol{x}^T \boldsymbol{g}_x^T g_{lt} \boldsymbol{g}_y \boldsymbol{y}\right).$$

Of course, this is useless unless we have a group G that we can store and compute with easily. Our second insight is to construct a Fuchsian group that can be represented with *integers* so that group operations can be computed exactly and efficiently. The naive way to do this is to try the subgroup of orientation-preserving isometries in $\mathbb{R}^{3\times 3}$ that have all-integer coordinates: unfortunately, this group (called the modular group) results in a tiling with unbounded fundamental domain, which makes it impossible to bound the representation error, so it is not suitable for our purpose. Instead, we constructed a special Fuchsian group to get a particularly useful L-tiling model of hyperbolic plane.

Definition 2. *Let* g_a *and* $g_b \in \mathbb{Z}^{3\times 3}$ *and* $L \in \mathbb{R}^{3\times 3}$ *be defined as*

$$g_a = \begin{bmatrix} 2 & 1 & 0 \\ 0 & 0 & -1 \\ 3 & 2 & 0 \end{bmatrix}, \; g_b = \begin{bmatrix} 2 & -1 & 0 \\ 0 & 0 & -1 \\ -3 & 2 & 0 \end{bmatrix}, \; \textit{and } L = \begin{bmatrix} \sqrt{3} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Define G *to be the fuchsian group generated by* $L \cdot g_a \cdot L^{-1}$ *and* $L \cdot g_b \cdot L^{-1}$*. It is straightforward to verify that* $(L \cdot g_a \cdot L^{-1})^T g_l (L \cdot g_a \cdot L^{-1}) = (L \cdot g_b \cdot L^{-1})^T g_l (L \cdot g_b \cdot L^{-1}) = g_l$*. Note that*

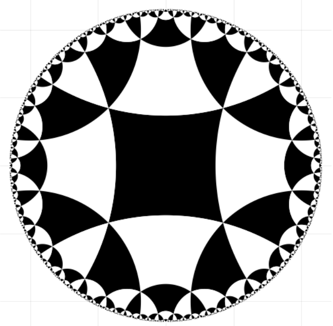


Figure 1: The regular quadrilateral tiling of hyperbolic space produced by the group G on the Poincaré disk.

Algorithm 1 Map Lorentz model to L-tiling model

```
Require: $\boldsymbol{x} \in \mathcal{L}^2$
  initialize $R \leftarrow I$
  while $x \notin F$ do
    if $x_2 \leq -|x_3|$ then $S \leftarrow g_a^{-1}$
    else if $x_2 \geq |x_3|$ then $S \leftarrow g_b^{-1}$
    else if $x_3 < -|x_2||$ then $S \leftarrow g_b$
    else if $x_3 > |x_2|$ then $S \leftarrow g_a$
    $(R, \boldsymbol{x}) \leftarrow (R \cdot S, L \cdot S^{-1} \cdot L^{-1} \cdot \boldsymbol{x})$
    $x_1 = \sqrt{x_2^2 + x_3^2 + 1}$    ▷ renormalize $\boldsymbol{x}$
  end while
output $(R, \boldsymbol{x})$
```

$g_a^6 = g_b^6 = (g_a g_b)^3 = I$, and so this group has presentation

$$G = L \cdot \langle g_a, g_b | g_a^6 = g_b^6 = (g_a g_b)^3 = 1 \rangle \cdot L^{-1}.$$

Importantly, *any* element of G can be represented in the form $g = LZL^{-1}$ where $Z \in \mathbb{Z}^{3\times 3}$ is an all-integers matrix. For this reason, we can store elements of G and take group products and inverses using *only integer arithmetic*. This property makes G of particular interest for use with an L-tiling model. But before we can construct an L-tiling model for this group, we need to choose an appropriate fundamental domain.

Theorem 2. *$F = \{(x_1, x_2, x_3) \in \mathcal{L}^2 | \max(2x_2^2 - x_3^2, 2x_3^2 - x_2^2) < 1\}$ is a fundamental domain of G. Any point in $\mathcal{L}^2$ can be mapped by G to one unique point in F or to a point on its boundary.*

Figure 1 illustrates the tiling generated by group G and F centered at the origin in the Poincaré disk model. Now that we have a group and a fundamental domain, we can start computing with our new L-tiling model. The first step is to build a relationship between standard hyperbolic models and the L-tiling model, i.e., convert points into the L-tiling model from other models: to this end, we offer a "normalization" procedure (Algorithm 1), which transforms the Lorentz model to the L-tiling model. The convergence and complexity of this algorithm are characterized in Theorem 3.

Theorem 3. *For any point in the Lorentz model, Algorithm 1 converges and stops within $1 + 7d$ steps, where $d = d(\boldsymbol{x}, \boldsymbol{O})$ denotes the distance from $\boldsymbol{x}$ to the origin.*

Representing points. For a point $(g, \boldsymbol{x})$ in the L-tiling model, where $g \in G$, $\boldsymbol{x} \in F$, we represent this point with $(g, \mathrm{fl}(\boldsymbol{x}))$. Here g is exact because it is represented by the related integer matrix, while fl denotes float arithmetic with error bounded by some machine epsilon ϵ_m. This floating point arithmetic introduces some representation error, which we can bound as follows:

$$d_{lt}((g, \boldsymbol{x}), (g, \mathrm{fl}(\boldsymbol{x}))) = \operatorname{arcosh}(-\boldsymbol{x}^T g^T g_{lt} g \mathrm{fl}(\boldsymbol{x})) = \operatorname{arcosh}(-\boldsymbol{x}^T g_{lt} \mathrm{fl}(\boldsymbol{x}))$$

Since $\boldsymbol{x} \in F$, which is bounded as shown in Theorem 2, this approximation error can also be bounded (Theorem 4). In comparison, for the Lorentz model, the worst case error (Theorem 1) is unbounded.

Theorem 4. *The representation error (Definition 1) in L-tiling model is bounded as $\delta_{lt}^d \leq \sqrt{5\epsilon_m} + 15\epsilon_m/4 + o(\epsilon_m)$, where ϵ_m is the machine error.*

By convention, for $(g, \boldsymbol{x})$ in the L-tiling model, where $g \in G$, $\boldsymbol{x} \in F$, firstly we will usually denote g using its related integer matrix $\hat{g} = L^{-1} g L$; Secondly for the point $x \in F$, even though x is part of the Lorentz model and lies in 3-dimensional space, in fact only two coordinates suffice to determine its position. For simplicity, we define a biejective function $h(x_2, x_3) = (\sqrt{1 + x_2^2 + x_3^2}, x_2, x_3)$ which maps $\mathbb{R}^2$ to the hyperboloid model (this is sometimes called the *Gans model* [17]). In this way, we can represent $(g, \boldsymbol{x}) \in \mathcal{T}_{lt}^2$ as $(\hat{g}, h^{-1}(\boldsymbol{x}))$. We can then store the integer matrix and floating-point coordinates $h^{-1}(\boldsymbol{x}) \in \mathbb{R}^2$. In future sections, we assume we will use this integer matrix and two-coordinate representation rather than $(g, \boldsymbol{x})$ unless otherwise specified.

5 Learning in the L-tiling Model

In this section, we provide an efficient and precise way to compute distances and gradients accordingly in the L-tiling model, with which we can construct learning algorithms to train and derive embeddings. We also present error bounds for these computations, which avoid the "Nan" problem.

Algorithm 2 RSGD in the L-tiling model

```
Require: Objective function $f$, fuchsian group $G$ with fundamental domain $F$, exponential map $\exp_{\beta_t}(v) = \cosh(\|v\|_L)\beta_t + \sinh(\|v\|_L)\frac{v}{\|v\|_L}$, where $\|v\|_L = \sqrt{\langle v, v\rangle_L}$.
Require: $(\beta_t, U_t) \in F \times G$, Epochs $T$, and learning rate $\eta$
  for $t = 0$ to $T - 1$ do
    $l_t \Leftarrow g_{\beta_t}^{-1}\nabla_{\beta_t} f(LU_tL^{-1}\beta_t)$   ▷ Riemannian
    grad $f \Leftarrow l_t + \langle\beta_t, l_t\rangle_L\beta_t$   ▷ Projection
    $\beta_{t+1} \Leftarrow \exp_{\beta_t}(-\eta \text{ grad} f)$   ▷ Update
    if $\beta_{t+1} \notin F$ then
      $W \Leftarrow \arg\min_{W \in G} d(LW^{-1}L^{-1}\beta_{t+1}, O)$
      $U_{t+1} \Leftarrow U_t \cdot W$   ▷ Normalize if $\beta_{t+1} \notin F$
      $\beta_{t+1} \Leftarrow LW^{-1}L^{-1}\beta_{t+1}$
    else
      $U_{t+1} \Leftarrow U_t$
    end if
  end for
output $(\beta_{t+1}, U_{t+1})$
```

Distance and Gradient. For two points $(U, u), (V, v)$ in the L-tiling model, the formula to compute distance is

$$d((U, u),(V, v)) = \operatorname{arcosh}(h(u)^T L^{-T} Q L^{-1} h(v))$$

where $Q = -U^T L^T g_{lt} LV$ can be computed exactly with integer arithmetic. A potential difficulty here is that the entries in Q can be very large (possibly even larger than can be represented in floating-point). To solve this, observe that Q_{11} has the largest absolute value in the matrix (Lemma 2). So we define and compute $\hat{Q} = Q/Q_{11}$, which is guaranteed to not overflow the floating-point format, since all the entries of $\hat{Q}$ are in $[-1, 1]$. Let $d_c = h(u)^T L^{-T} \hat{Q} L^{-1} h(v)$, this reduces our distance to

$$d((U, u), (V, v)) = \operatorname{arcosh}(Q_{11} \cdot d_c) = \log(Q_{11}) + \log\left(d_c + \sqrt{d_c^2 - Q_{11}^{-2}}\right)$$

Note that (assuming that we can compute $\log(Q_{11})$ without overflow) this expression can be computed in floating-point without any overflow, since all the numbers involved are well within range. The corresponding formula for the gradient can also be derived as

$$\nabla_u d((U, u), (V, v)) = \frac{\nabla h(u)^T L^{-T} \hat{Q} L^{-1} h(v)}{\sqrt{d_c^2 - Q_{11}^{-2}}}, \quad \text{where } \nabla h(u) = \left[\frac{u}{\sqrt{1 + \|u\|^2}},\ I\right].$$

Again, this avoids any possibility of overflow. We provide the error of computing distance (Theorem 6) and gradient (Theorem 7) in L-tiling model together with that in Lorentz model in Appendix. By computing with integer arithmetic, the error will be independent of how far the points are from the origin, which guarantees that it avoids the "NaN" problem. Since we can compute distances and derivatives, we can use all the standard gradient-based optimization algorithms. In Algorithm 2 we present the most powerful one, RSGD, adapted for use with the L-tiling model.

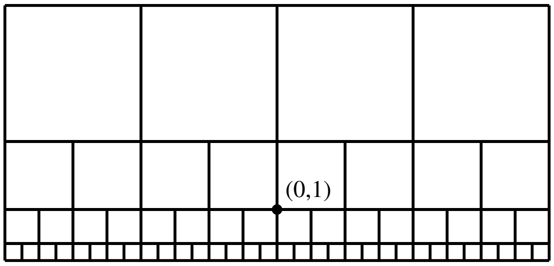
6 Extension to Higher Dimensional Space

Extending the L-tiling model to higher dimension seems simple: just find a cocompact (to ensure a bounded fundamental domain) discrete subgroup of the higher-dimensional space's isometry group. Such a group would induce a *honeycomb*, a higher-dimensional analog of a regular tiling of the hyperbolic plane. Unfortunately, a classic result by Coxeter [10] says this is impossible in general: there are no such regular honeycombs in six or more dimensions.

In order to derive a high dimensional tiling-based model which may be necessary for complicated datasets, we consider two possibilities.

- Take the Cartesian product of multiple copies of the L-tiling model in the hyperbolic plane. The use of multiple copies of models in the hyperbolic plane was previously proposed in Gu et al. [19].
- Construct honeycombs and tilings from a set of isometries that is not a group.

Practically we can embed data into products of $\mathbb{H}^2$s as we do in Section 7, however, the first possibility (tilings over $\mathbb{H}^2 \times \mathbb{H}^2 \times \cdots \mathbb{H}^2$) is something fundamentally different from tiling a single



Datasets	Nodes	Edges
Bio-yeast[29]	1458	1948
WordNet[14]	74374	75834
↳ Nouns	82115	769130
↳ Verbs	13542	35079
↳ Mammals	1181	6541
Gr-QC[23]	4158	13422

Figure 2: (Left) The infinite square tiling of hyperbolic space on the half-plane model; (Right) Datasets.

high dimensional hyperbolic space (tilings over $\mathbb{H}^n$), which we aims to do in this section. Fortunately for the second possibility, in half-space model, we find that horizontal translation and homotheties are hyperbolic isometries, which can produce the (infinite) square tiling illustrated in Figure 2 [1, 6]. It consists of the image of the unit square S, with vertical and horizontal sides and whose lower left corner is at $(0, ..., 0, 1)$, under the maps

$$p \to 2^j(p+k), (j,k) \in \mathbb{Z} \times (\mathbb{Z}^{n-1} \times \{0\}).$$

Here each square is isometric to every other square, and the unit square S takes on the role of the fundamental domain in Theorem 2. With these maps, we can define a tiling-based model on top of the half-space model as follows.

H-tiling model. The H-tiling model of the hyperbolic space is defined as the Riemannian manifold $(\mathcal{T}_h^n, g_{ht})$, where

$$\mathcal{T}_h^n = \{(j, \boldsymbol{k}, \boldsymbol{x}) \in \mathbb{Z} \times (\mathbb{Z}^{n-1} \times \{0\}) \times S\}, \qquad g_{ht}(j, \boldsymbol{k}, \boldsymbol{x}) = \frac{g_e}{(2^j x_n)^2}$$

The associated distance function on $\mathcal{T}_h^n$ is then given as

$$d((j_1, \boldsymbol{k}_1, \boldsymbol{x}), (j_2, \boldsymbol{k}_2, \boldsymbol{y})) = \operatorname{arcosh}\left(1 + \frac{\|2^{j_1}z_1 - 2^{j_2}z_2 + 2^{j_1}k_1 - 2^{j_2}k_2)\|^2}{2^{j_1+j_2+1}z_{1n}z_{2n}}\right).$$

Similarly, we derive the representation error for this model, which is bounded by a constant depending on the machine epsilon as shown in Theorem 5.

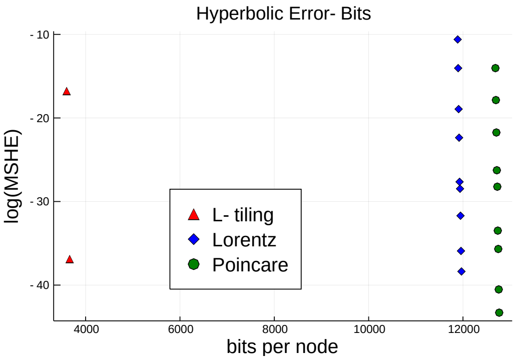
Theorem 5. *The representation error (Definition 1) in H-tiling model is bounded as $\delta_{ht}^d = \sqrt{(n+3)\epsilon_m}/2 + (n+3)\epsilon_m/4 + o(\epsilon_m)$, where ϵ_m is the machine error.*

We can compute distances and gradients in a numerically accurate way, and run RSGD algorithm on this model for optimization, just as we could in the L-tiling model. For lack of space, we defer that discussion and more learning details of Sections 5 and 6 to Appendix A. Also note that we are not tied to the half-space model here: while the half-space model gives a convenient way to describe the set of transformations we are using, we could use the same transformations with any underlying model we choose by adding an appropriate conversion.

7 Experiments

Compressing embeddings. We consider storing 2-dimensional embeddings using the L-tiling model for compression: storage using few bits. While storing the integer matrices exactly is convenient for computation, it does tend to take up a lot of extra memory (especially when BigInts are needed to store the integer values in the matrix). This motivates us to look for alternative storage methods. To store the matrix g, we prorpose and evaluate the following methods:

- Matrix: store all 9 integers in the matrix g as Int or BigInt.
- Entries: store just g_{21}, g_{31} as Int or BigInt, which we can show is sufficient to reconstruct the whole matrix (Lemma 1 in Appendix D).
- Order: store the generator order with respect to g_a, g_b as a string.
- VBW: store the generator order with respect to g_a, g_b using a variable bit-width encoding. We use binary code `10` to represent g_a^1 and g_b^1, `001` to represent g_a^2 and g_b^2, `010` to represent g_a^3 and g_b^3, `011` to represent g_a^4 and g_b^4, `11` to represent g_a^5 and g_b^5, and `000` to represent the end of the string. This encoding disambiguates the generators by taking advantage of the fact that powers of g_a and g_b must alternate to appear.



Models	size (MB)	bzip (MB)
Poincaré(16512B)	372	119
Poincaré(12688B)	287	81
Lorentz(11898B)	396	171
Matrix(VLQ)	600(286)	260(251)
Entries(VLQ)	132(63)	57(55)
Order	111	8.52
VBW	33.1	6.07
fpt-f32	6.2	1.96
fpt-f16	4.25	1.07

Figure 3: (Left) Hyperbolic error for WordNet Nouns; (Right) Compression statistics for WordNet under the same MSHE, first block contains the size of original poincare embedding, second block contains the size of compressed baseline models, third block contains the size of matrix part in the L-tiling model (size of compressed integers using VLQ is also reported), the last block contains size of float points (fpt, f32 or f16) in the fundamental domain of L-tiling model.

The generator order and corresponding VBW encoding of a given matrix can be derived using Algorithm 1 as shown in Lemma 1. Additionally, for Int or BigInt, we can use variable length quantity (VLQ) to compress [31]. To test our compression methods, we use combinatorial construction [30] to derive 2-dimensional Poincaré disk embeddings for WordNet (Tree-like) and Bio-yeast datasets (Figure 2), then we transform embeddings and compress them. We calculate the mean squared hyperbolic error (MSHE) with respect to the original embedding to show the error of compression.

For Bio-yeast, we evaluate different compressions using MSHE and mean average precision (MAP). As shown in Table 1, representation and compression in the L-tiling model (with different floating number for points in the fundamental domain) does not hurt MAP performance, while the compression of the Poincaré embedding to the same size hurts MAP severely. For WordNet, we plot the scatter of the relationship between $\log(\text{MSHE})$ and bits to store per node in Figure 3. Under the same MSHE, the L-tiling model requires approximately $2/3$ less bits per node compared to that of Lorentz and Poincaré models. We measure the size of different models under the same MSHE in Figure 3. The L-tiling model can represent the hyperbolic embedding with only (6.07+1.07) MB, which is 2% of the original 372 MB, while it will cost at least 81 MB for any reasonably accurate baseline model.

Learning embeddings. As we have shown, our tiling-based models represent hyperbolic space accurately, and so they can be used for learning embeddings with generic objective functions. However, since we analyzed hyperbolic distance and gradient computation error in this paper, we evaluate our learning methods empirically on objective functions that depend on distances. As proposed by Nickel and Kiela [26], to consider the ability to embed data that exhibits a clear latent hierarchical structure, we conduct reconstruction experiments on the transitive closure of the Gr-QC, WordNet Nouns, Verbs and Mammals hierarchy as summarized in Table 2. We firstly embed the data and then reconstruct it from the embedding to evaluate the representation capacity of the embedding. Let $\mathcal{D} = \{(u, v)\}$ be the set of observed relations between objects. We aim to learn embeddings of $\mathcal{D}$ such that related objects are close in the embedding space. To do this, we minimize the loss [26]

$$\mathcal{L}(\Theta) = \sum_{(u,v)\in\mathcal{D}} \log \frac{e^{-d(\boldsymbol{u},\boldsymbol{v})}}{\sum_{\boldsymbol{v}'\in\mathcal{N}(u)} e^{-d(\boldsymbol{u},\boldsymbol{v}')}}, \tag{1}$$

where $\mathcal{N}(u) = \{v \mid (u, v) \notin \mathcal{D}\} \cup \{u\}$ is the set of negative examples for u (including u). We randomly sample $|\mathcal{N}(u)| = 50$ negative examples per positive example during training.

Table 1: Compression of Bio-yeast

Models	MSHE	MAP
Poincaré(8128B)	0.00	0.873
Poincaré(6360B)	4.84e-17	0.873
Poincaré(1832B)	1.01e+03	0.310
L-tiling-f64(1832B)	9.76e-17	0.873
L-tiling-f32(1768B)	5.12e-08	0.873
L-tiling-f16(1736B)	4.30e-05	0.873
L-tiling-f0(1704B)	5.90e-01	0.873

Table 2: Learning Mammals

Models	MAP	MR
Poincaré	0.805±0.011	2.22±0.10
Lorentz	0.855±0.013	1.89±0.13
H-tiling-SGD	0.892±0.031	2.14±0.70
L-tiling-RSGD	**0.930**±0.005	**1.49**±0.09
H-tiling-RSGD	0.923±0.016	1.56±0.20

Dimension	Models	WordNet Nouns		WordNet Verbs		Gr-QC	
		MAP	MR	MAP	MR	MAP	MR
2	Poincaré	0.124±0.001	68.75±0.26	0.537±0.005	4.74±0.17	0.561± 0.004	67.91±1.14
	Lorentz	0.382±0.004	17.80±0.55	**0.750**±0.004	2.11±0.06	0.563±0.003	68.40±1.20
	H-tiling-rsgd	0.390±0.002	17.18±0.52	0.747±0.003	2.10±0.05	0.560±0.004	66.17±1.05
	L-tiling-sgd	0.341±0.001	20.27±0.39	0.696±0.003	2.33±0.07	**0.574**±0.005	**63.04**±1.97
	L-tiling-rsgd	**0.413**±0.007	**15.26**±0.57	0.746±0.004	**2.07**±0.03	0.564± 0.002	63.88±1.47
4	2×Lorentz	0.460±0.001	10.12±0.03	**0.873**±0.001	1.31±0.01	**0.718**±0.003	11.59±0.32
	2×L-tiling-rsgd	**0.464**±0.002	**9.99**±0.09	0.871±0.004	1.33±0.01	0.716±0.005	**10.88**±0.42
5	Poincaré	0.848±0.001	4.16±0.04	0.948±0.001	1.19±0.01	0.714±0.000	34.60±0.52
	Lorentz	0.865±0.005	**3.70**±0.12	0.947±0.001	**1.16**±0.00	**0.715**±0.003	33.51± 1.04
	H-tiling-rsgd	**0.869**±0.001	**3.70**±0.06	**0.949**±0.001	**1.16**±0.01	0.714±0.002	**33.46**±0.66
10	Poincaré	0.876±0.001	3.47±0.02	0.953±0.002	1.16±0.01	0.729±0.000	29.51±0.21
	Lorentz	0.865±0.004	3.36±0.04	0.948±0.001	1.15±0.00	0.724±0.001	29.34±0.23
	H-tiling-rsgd	**0.888**±0.004	**3.22**±0.02	0.954±0.002	1.15±0.00	0.729±0.001	27.75±0.39
	5×Lorentz	0.672±0.000	4.42±0.00	0.958±0.003	1.07±0.01	0.944±0.007	3.06±0.03
	5×L-tiling-rsgd	0.674±0.000	4.41±0.00	**0.961**±0.002	**1.06**±0.00	**0.953**±0.002	**3.03**±0.01

Table 3: Learning experiments on different datasets. Results are averaged over 5 runs and reported in mean+std style.

We consider the L-tiling models trained with RSGD and SGD, H-tiling models trained with RSGD and the Cartesian product of multiple copies of 2-dimensional L-tiling models (proposed in Gu et al. [19]). The Poincaré ball model [26] and Lorentz model [27] were included as baselines. All models were trained in float64 for 1000 epochs with the same hyper-parameters. To evaluate the quality of the embeddings, we make use of the standard graph embedding metrics in [3, 25]. For an observed relationship (u, v), we rank the distance $d(\boldsymbol{u}, \boldsymbol{v})$ among the set $\{d(u, v')|(u, v') \in \mathcal{D})\}$, then we evaluate the ranking on all objects in the dataset and record the mean rank (MR) as well as the mean average precision (MAP) of the ranking.

We start by evaluating all 2-dimensional embeddings on the Mammals dataset. As shown in Table 2, all tiling-based models outperform baseline models: the performances of L-tiling model and H-tiling model with RSGD are nearly the same. In particular, the L-tiling model achieves a 8.8% MAP improvement on Mammals compared to Lorentz model.

Embedding experiments on other three large datasets are presented in Table 3. These results show that tiling-based models generally perform better than baseline models in various dimensions. We found three observations particularly interesting here. First, the group-based tiling model (L-tiling) performs better than the non-group tiling model (H-tiling) in two dimensions. Second, tiling-based models perform particularly better than baseline models for the largest WordNet Nouns dataset, which further validates that numerical issue happens when the embeddings are far from the origin and affects the embedding performances. Third, the Cartesian product of multiple copies of 2-dimensional L-tiling models performs even better than high dimensional models when the datasets are not too large and complex such as WordNet Verbs and Gr-QC, especially for the dense graph Gr-QC.

More experiment details are provided in Appendix B. We release our compression code[*] in Julia and learning code[†] in PyTorch publicly for reproducibility.

8 Discussions and Conclusions

In this paper, we introduced tiling-based models of hyperbolic space, which use a tiling backed by integer arithmetic to represent any point in hyperbolic space with fixed and provably bounded error. We showed that L-tiling model using one particular group G can achieve substantial compression of an embedding with minimal loss, and can perform well on embedding tasks compared with other methods. A notable observation that could motivate future work is that our group based tiling model (L-tiling) performs better than the non-group tiling model (H-tiling) in two dimensions: it is interesting to ask if this reflects some advantages of the group, and if we can use this to find better non-regular tilings in high dimensions. Overall, it is our hope that this work can help make hyperbolic embeddings more numerically robust and thereby make them easier for practitioners to use.

---

[*] `https://github.com/ydtydr/HyperbolicTiling_Compression`

[†] `https://github.com/ydtydr/HyperbolicTiling_Learning`

References

[1] J. Anderson. *Hyperbolic Geometry*. Springer Undergraduate Mathematics Series. Springer London, 2005. ISBN 9781852339340. 2, 3, 7

[2] Alan F Beardon. *The geometry of discrete groups*, volume 91. Springer Science & Business Media, 2012. 4, 16

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013. 9

[4] Brian H Bowditch. A course on geometric group theory. *Mathematical Society of Japan*, 16 of MSJ Memoirs, 2006. 1

[5] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015. 1

[6] James W. Cannon, William J. Floyd, Richard Kenyon, Walter, and R. Parry. Hyperbolic geometry. In *In Flavors of geometry*, pages 59–115. University Press, 1997. 2, 3, 7

[7] Benjamin Paul Chamberlain, James Clough, and Marc Peter Deisenroth. Neural embeddings of graphs in hyperbolic space. *Proceedings of the 13th international workshop on mining and learning from graphs held in conjunction with KDD*, 2017. 1, 2

[8] Aaron Clauset, Cristopher Moore, and Mark EJ Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98, 2008. 1

[9] J.H. Conway, H. Burgiel, and C. Goodman-Strauss. *The Symmetries of Things*. Ak Peters Series. Taylor & Francis, 2008. ISBN 9781568812205. URL `https://books.google.com/books?id=EtQCk0TNafsC`. 2, 4, 15

[10] H. S. M. Coxeter. Regular honeycombs in hyperbolic space. In *III, Noordhoff, Groningen, and North-Holland*, page 155, 1956. 6

[11] Andrej Cvetkovski and Mark Crovella. Multidimensional scaling in the poincaré disk. *Applied Mathematics & Information Sciences*, abs/1105.5332, 05 2011. doi: 10.18576/amis/100112. 1

[12] Basudeb Datta and Subhojoy Gupta. Uniform tilings of the hyperbolic plane. *arXiv e-prints*, art. arXiv:1806.11393, Jun 2018. 4, 15

[13] Bhuwan Dhingra, Christopher Shallue, Mohammad Norouzi, Andrew Dai, and George Dahl. Embedding text in hyperbolic spaces. In *Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-12)*, pages 59–69. Association for Computational Linguistics, 2018. 3

[14] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998. 1, 7

[15] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. In *Advances in neural information processing systems*, pages 5345–5355, 2018. 3

[16] Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic entailment cones for learning hierarchical embeddings. In *Proceedings of the 35th International Conference on Machine Learning*, 2018. 3

[17] David Gans. A new model of the hyperbolic plane. *The American Mathematical Monthly*, 73(3):291–295, 1966. ISSN 00029890, 19300972. URL `http://www.jstor.org/stable/2315350`. 5

[18] Mikhael Gromov. Hyperbolic groups. *Essays in group theory*, page 75–263, 1987. 1

[19] Albert Gu, Frederic Sala, Beliz Gunel, and Christopher Ré. Learning mixed-curvature representations in product spaces. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HJxeWnCcF7`. 3, 6, 9

[20] Caglar Gulcehre, Misha Denil, Mateusz Malinowski, Ali Razavi, Razvan Pascanu, Karl Moritz Hermann, Peter Battaglia, Victor Bapst, David Raposo, Adam Santoro, et al. Hyperbolic attention networks. *International Conference on Learning Representations*, 2018. 3

[21] Svetlana Katok. *Fuchsian groups*. University of Chicago press, 1992. 4, 16

[22] Benedikt Kolbe and Vanessa Robins. Tiling the euclidean and hyperbolic planes with ribbons. *arXiv preprint arXiv:1904.03788*, 2019. 4

[23] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2, 2007. 7

[24] Shigeru Miyagawa, Robert C Berwick, and Kazuo Okanoya. The emergence of hierarchical structure in human language. *Frontiers in psychology*, 4:71, 2013. 1

[25] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. Holographic embeddings of knowledge graphs. In *Thirtieth Aaai conference on artificial intelligence*, 2016. 9

[26] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6338–6347. Curran Associates, Inc., 2017. 1, 2, 3, 8, 9, 14

[27] Maximillian Nickel and Douwe Kiela. Learning continuous hierarchies in the Lorentz model of hyperbolic geometry. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 3779–3788. PMLR, 2018. 1, 2, 3, 9

[28] Melissa Potter and Jason M. Ribando. Isometries, tessellations and escher, oh my. *American Journal of Undergraduate Research*, 3, 03 2005. doi: 10.33697/ajur.2005.005. 4, 15

[29] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015. URL `http://networkrepository.com`. 7

[30] Frederic Sala, Chris De Sa, Albert Gu, and Christopher Re. Representation tradeoffs for hyperbolic embeddings. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 4460–4469, Stockholmsmässan, Stockholm Sweden, 2018. PMLR. 1, 3, 8

[31] David Salomon. *Variable-length Codes for Data Compression*. Springer-Verlag, Berlin, Heidelberg, 2007. ISBN 1846289580. 8

[32] Rik Sarkar. Low distortion delaunay embedding of trees in hyperbolic plane. In *International Symposium on Graph Drawing*, pages 355–366. Springer, 2011. 1

[33] Teruhisa Sugimoto. Convex pentagons for edge-to-edge tiling, ii. *Graphs and Combinatorics*, 31(1):281–298, 2015. 4, 15

[34] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Hyperbolic representation learning for fast and efficient neural question answering. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 583–591. ACM, 2018. 3

[35] Alexandru Tifrea, Gary Becigneul, and Octavian-Eugen Ganea. Poincare glove: Hyperbolic word embeddings. In *International Conference on Learning Representations*, 2019. 3

[36] John Voight. Computing fundamental domains for fuchsian groups. *Journal de théorie des nombres de Bordeaux*, 21(2):467–489, 2009. doi: 10.5802/jtnb.683. URL `http://www.numdam.org/item/JTNB_2009__21_2_467_0`. 4, 16

[37] John Voight. The arithmetic of quaternion algebras. *preprint*, 2014. 4, 16, 17

[38] Robert Yuncken. Regular tessellations of the hyperbolic plane by fundamental domains of a fuchsian group. *Moscow Mathematical Journal*, 3, 03 2011. doi: 10.17323/1609-4514-2003-3-1-249-252. 4

Supplementary Material: Numerically Accurate Hyperbolic Embeddings Using Tiling-Based Models

A Learning Details

Efficient Computation in L-tiling Model For two points $(U, u), (V, v)$ in L-tiling model, the formula to compute distance is

$$
\begin{aligned}
d((U,u),(V,v)) &= \operatorname{arcosh}\left(h(u)^T L^{-T} Q L^{-1} h(v)\right) \\
&= \operatorname{arcosh}(Q_{11} \cdot d_c) \\
&= \log(Q_{11}) + \log\left(d_c + \sqrt{d_c^2 - Q_{11}^{-2}}\right)
\end{aligned}
$$

where $Q = -U^T L^T g_{lt} L V, \hat{Q} = \frac{Q}{Q_{11}}, d_c = h(u)^T L^{-T} \hat{Q} L^{-1} h(v)$. Since Q_{11} can be super large, then we extract Q_{11} out here to avoid potential overflow, also there is no underflow problem since Q_{11} is a positive integer. Then the corresponding formula for the gradient of this distance is

$$
\begin{cases}
\nabla_u d((U,u),(V,v)) = \frac{\nabla h(u)^T L^{-T} \hat{Q} L^{-1} h(v)}{\sqrt{(h(u)^T L^{-T} \hat{Q} L^{-1} h(v))^2 - Q_{11}^{-2}}} = \frac{\nabla h(u)^T L^{-T} \hat{Q} L^{-1} h(v)}{\sqrt{d_c^2 - Q_{11}^{-2}}} \\
\nabla_v d((U,u),(V,v)) = \frac{\nabla h(v)^T L^{-T} \hat{Q}^T L^{-1} h(u)}{\sqrt{(h(u)^T L^{-T} \hat{Q} L^{-1} h(v))^2 - Q_{11}^{-2}}} = \frac{\nabla h(v)^T L^{-T} \hat{Q}^T L^{-1} h(u)}{\sqrt{d_c^2 - Q_{11}^{-2}}}
\end{cases}
$$

where

$$
\nabla h(u) = \left[\frac{u}{\sqrt{1 + ||u||^2}},\ I\right], \nabla h(v) = \left[\frac{v}{\sqrt{1 + ||v||^2}},\ I\right]
$$

We provide the error bound for distance and gradient computation in L-tiling model using float arithmetic in Theorem 6, Theorem 7, where errors are independent of how far points are from the origin and solves the "Nan" problem.

SGD in the L-tiling model We offer SGD algorithm below, with the addition of a normalization when the parameter goes out of the L-tiling model, which is performed using Algorithm 1, whose convergence and complexity were shown in Theorem 3.

Algorithm 3 SGD using group representation

```
Require: Objective function $f$, fuchsian group $G$ with fundamental domain $F \subset \mathbb{R}^2$
Require: Tuple $(\theta_t, U_t) \in F \times G$
Require: Number of epochs $T$, and learning rate $\alpha$
  for $t =$ to $T - 1$ do
    $l_t \Leftarrow \nabla_{\theta_t} f(L U_t L^{-1} h(\theta_t))$          ▷ Euclidean gradient w.r.t. $\theta_t$
    $\theta_{t+1} \Leftarrow \theta_t - \alpha l_t$                          ▷ Update $\theta_t \in F$
    if $\theta_{t+1} \notin F$ then
      $W \Leftarrow \arg\min_{W \in G} d(L W^{-1} L^{-1} h(\theta_{t+1}), 0)$
      $U_{t+1} \Leftarrow U_t \cdot W$                                        ▷ Normalize if $\theta_{t+1} \notin F$
      $\theta_{t+1} \Leftarrow L W^{-1} L^{-1} h(\theta_{t+1})$
    else
      $U_{t+1} \Leftarrow U_t$
    end if
  end for
output $(\theta_{t+1}, U_{t+1})$
```

RSGD in the L-tiling model We show RSGD in the L-tiling model here, which is in correspondence to Algorithm 1. Equivalence of this algorithm to that in Lorentz model is shown in Theorem D For (U_t, u_t) in the L-tiling model, let f be the objective function, denote $\nabla_{u_t} f$ to be the Euclidean gradient of f w.r.t. u_t. To do RSGD in the L-tiling model, firstly transform the Euclidean gradient to Riemannian gradient using the pull-back hyperbolic metric:

$$h_t = g_{u_t}^{-1} \nabla_{u_t} f,$$

then project it into the tangent space at u_t,

$$\text{grad}_{u_t} f = h_t + \langle u_t, h_t \rangle_L u_t.$$

then the RSGD algorithm in the L-tiling model updates u^t as follows:

$$u_{t+1} = \exp_{u_t}(v) = \cosh(||v||_L) u_t + \sinh(||v||_L) \frac{v}{||v||_L},$$

where $v = -\eta \cdot \text{grad}_{u_t} f$ and $\|v\|_L = \sqrt{\langle v, v \rangle_L}$.

Efficient Computation in H-tiling Model For points (j_1, k_1, z_1) and (j_2, k_2, z_2) in H-tiling model, directly computing distance as follows works in most cases,

$$d_h((j_1, \boldsymbol{k}_1, \boldsymbol{z}_1), (j_2, \boldsymbol{k}_2, \boldsymbol{z}_2)) = \text{arcosh}(1 + \frac{||2^{j_1}(z_1 + k_1) - 2^{j_2}(z_2 + k_2)||^2}{2^{j_1+j_2+1} z_{1n} z_{2n}})$$

However, we do consider situations where overflows may happen and provide an alternative way to compute distance. Suppose without loss of generality that $j_2 \geq j_2$, then we can write distance as

$$d_h((j_1, \boldsymbol{k}_1, \boldsymbol{z}_1), (j_2, \boldsymbol{k}_2, \boldsymbol{z}_2)) = \text{arcosh}(1 + 2^{j_1-j_2} \frac{||z_1 - 2^{j_2-j_1} z_2 + k_1 - 2^{j_2-j_1} k_2)||^2}{2 z_{1n} z_{2n}})$$

let $k_1 - 2^{j_2-j_1} k_2 = 2^s I$ where s is some natural number scale factor such that $\|I\| < 1$. This is easy to compute exactly using integer arithmetic. (Note that $k_1 - 2^{j_2-j_1} k_2$ is an integer vector, choose $s = \lceil \log_2(\|k_1 - 2^{j_2-j_1} k_2\|^2)/2 \rceil$, where the values inside the $\log_2$ are all integers). Then we get

$$\begin{aligned}
d_h((j_1, \boldsymbol{k}_1, \boldsymbol{z}_1), (j_2, \boldsymbol{k}_2, \boldsymbol{z}_2)) &= \text{arcosh}(1 + 2^{j_1-j_2} \frac{||2^s I + z_1 - 2^{j_2-j_1} z_2||^2}{2 z_{1n} z_{2n}}) \\
&= \text{arcosh}(1 + 2^{2s+j_1-j_2} \frac{||I + 2^{-s} z_1 - 2^{j_2-j_1-s} z_2||^2}{2 z_{1n} z_{2n}}) \\
&= \text{arcosh}(1 + 2^{2s+j_1-j_2} X) \\
&= \log(1 + 2^{2s+j_1-j_2} X + \sqrt{(1 + 2^{2s+j_1-j_2} X)^2 - 1}) \\
&= (2s + j_1 - j_2) \log(2) + \log(2^{-2s-j_1+j_2} + X + \sqrt{X^2 + 2^{1-2s-j_1+j_2} X})
\end{aligned}$$

where

$$X = \frac{||I + 2^{-s} z_1 - 2^{j_2-j_1-s} z_2||^2}{2 z_{1n} z_{2n}}$$

then we can get following gradients,

$$\begin{cases}
\dfrac{\partial d}{\partial X} = \dfrac{1 + \frac{X + 2^{-2s-j_1+j_2}}{\sqrt{X^2 + 2^{1-2s-j_1+j_2} X}}}{2^{-2s-j_1+j_2} + X + \sqrt{X^2 + 2^{1-2s-j_1+j_2} X}} \\
\nabla_{z_{1i}} X = \dfrac{I_i + 2^{-s} z_{1i} - 2^{j_2-j_1-s} z_{2i}}{2^s z_{1n} z_{2n}} \\
\nabla_{z_{2i}} X = -\dfrac{I_i + 2^{-s} z_{1i} - 2^{j_2-j_1-s} z_{2i}}{2^{s+j_1-j_2} z_{1n} z_{2n}} \\
\nabla_{z_{1n}} X = \dfrac{I_n + 2^{-s} z_{1n} - 2^{j_2-j_1-s} z_{2n}}{2^s z_{1n} z_{2n}} - \dfrac{||I + 2^{-s} z_1 - 2^{j_2-j_1-s} z_2||^2}{2 z_{1n}^2 z_{2n}} \\
\nabla_{z_{2n}} X = -\dfrac{I_n + 2^{-s} z_{1n} - 2^{j_2-j_1-s} z_{2n}}{2^{s+j_1-j_2} z_{1n} z_{2n}} - \dfrac{||I + 2^{-s} z_1 - 2^{j_2-j_1-s} z_2||^2}{2 z_{1n} z_{2n}^2}
\end{cases}$$

Using chain rule, Euclidean gradients of d_h w.r.t. z_1, z_2 can be derived. We provide the error bound for this distance computation in H-tiling model using float arithmetic in Theorem 8. we neglect the error bound for gradient computation here which can also be derived similarly. In H-tiling model, these computation errors are independent of how far points are from the origin and solves the “Nan" problem.

RSGD in the H-tiling model For (j_t, k_t, z^t) in the H-tiling model, let f be the objective function, denote $\nabla_{z^t} f$ to be the Euclidean gradient of f w.r.t. z^t. To do RSGD in the H-tiling model, firstly transform the Euclidean gradient to Riemannian gradient using the pull-back metric:

$$\text{grad}_{z^t}\ f = z_n^t \nabla_{z^t} f$$

take the learning rate η into consideration, denote $v = -\eta \cdot \text{grad}_{z^t} f$, firstly compute its norm as $s = \sqrt{v^T v}$, then the RSGD algorithm in the H-tiling model updates z^t as follows:

$$\begin{cases} z_i^{t+1} &= z_i^t + \dfrac{z_n^t}{\frac{s}{\tanh s} - v_n} \cdot v_i \\ z_n^{t+1} &= \dfrac{z_n^t}{\cosh s - \frac{\sinh s}{s} v_n} \end{cases}$$

B Experiment Details

Compression Compression experiments were implemented in Julia. We compress L-tiling model using storage methods mentioned in Section 7, round Poincaré ball model towards zero since it is bounded in the Euclidean unit ball, round Lorentz model to the nearest to compress baseline models.

Learning We implemented learning experiments in PyTorch using float64. Notably, for tiling-based models, we also use float64 to store the integers, in order to avoid potential numerical imprecision when the integers overflow and are out of the expressible range of float64, we developed a secure method to express a integer matrix U and do accurate integer arithmetic using two float64 type matrices. Specifically, we express it as $U = U_1 + U_2$, where $2^t|U_1, |U_2| < 2^t$, and U_1, U_2 are float64 type. Alternatively, we can similarly use n float64 type matrices to express the integer matrix and pick suitable t to prevent overflow. In our experiments, we found that two float matrices and $t = 20$ are sufficient to prevent overflow and get exact computation of integer arithmetic using float64.

We initialize embeddings randomly from the uniform distribution $U(-0.0001, 0.0001)$, except from embeddings in the H-tiling model, whose last elements z_n were initialized from the uniform distribution $U(1, 1.0001)$ in order to make the division to z_n stable. Matrices $g \in G$ in L-tiling model were initialized to be identity matrices, integer vectors and the exponential integer in H-tiling model were initialized to be zeros. Then we project those embedding to the manifold accordingly before training.

Second, similar as [26], to get a good initial angular layout which is helpful to find good embeddings, we train during an initial "burn-in" phase (20 epochs) with a reduced learning rate $\eta/100$. We train the embedding using multi-threads N to speed up convergence. Those hyperparameters together with batch size b for different datasets were summarized in Table 4.

Hyperparameters	η	b	N
Gr-QC	0.3	10	2
WordNet Mammals	0.3	10	2
WordNet Verbs	0.5	10	5
WordNet Nouns	0.5	50	5

Table 4: Hyperparameters

C More Experiments

Embedding in hyperbolic space reaches better performance compared to Euclidean space, as a simple experiment, consider embedding of a simple tree in Figure 4, where the lengths of edges are in different scale. When embedded into Euclidean space, the global distortion is 0.1395, the worst-case distortion is 2.15. However, when embedded into Poincaré ball model of hyperbolic space, the

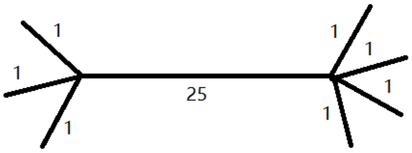


Figure 4: A simple tree.

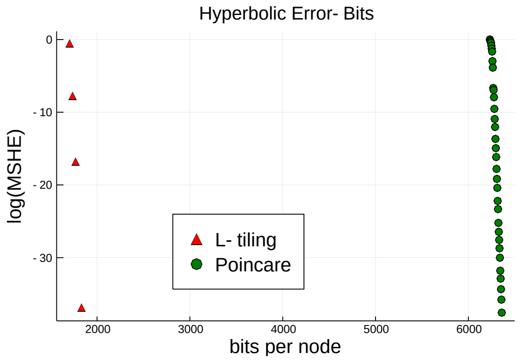


Figure 5: hyperbolic error for bio-yeast dataset.

global distortion is just 0.0007 and the worst-case distortion is 1.025, which is far better than that in Euclidean space.

Compression Experiments For compression of Bio-yeast dataset mentioned in Section 7, we plot the scatter of the relationship between log(MSHE) and bits to store per node in Figure 5, under the same MSHE, L-tiling model store per node with approximately $3/4$ less bits compared to that of Poincaré model.

We also consider compressing embeddings that were trained using optimization algorithms like SGD and RSGD, to learn 2-dimensional Poincaré disk embedding with great performance, but this method fails to learn a great embedding for larger dataset like WordNet Nouns using 2 dimension model, so we use this method to derive 2-dimensional embeddings for Mammals (Tree-like) and Gr-QC (Dense) dataset. We show the MSHE of different compressions in Table 5, and it leads to the same conclusion that those compression will not hurt the performance of the embedding such as MAP and MR while largely shrinking the size of embeddings.

Table 5: Compression Performance

Models	Mammals			grqc		
	MSHE	MAP	MR	MSHE	MAP	MR
Poincaré(128b)	0	0.7936	2.36	0	0.5382	73.88
Poincaré(64b)	1.59e-2	0.7935	2.36	1.65e-1	0.5382	73.88
Lorentz(128b)	1.51e-11	0.7935	2.36	1.39e-10	0.5382	73.88
Lorentz(64b)	9.44e-3	0.7935	2.36	1.77e-1	0.5382	73.88
L-tiling-f32	5.17e-08	0.7935	2.36	5.29e-8	0.5382	73.88
L-tiling-f16	4.16e-04	0.7935	2.36	4.19e-4	0.5382	73.88

Learning Experiments We also include the previous results where all models were trained using float32 in PyTorch as shown in Table 6.

D Mathematical Background and Proofs

Definition 1. *[Representation error] We are concerned with representing points in hyperbolic space $\mathbb{H}^n$ using floating-points* fl. *Define the* representation error *of a particular point $x \in \mathbb{H}^n$ as $\delta_{\text{fl}}(x) = d_{\mathbb{H}^n}(x, \text{fl}(x))$, and the* worst case representation error *of floating-points representation as a function of the distance-to-origin d, which is the maximum representation error of any point with a distance-to-origin at most d,*

$$\delta_{\text{fl}}^{d} = \max_{x \in \mathbb{H}^n,\, d_{\mathbb{H}^n}(x,O) \le d} \delta_{\text{fl}}(x).$$

Definition 3 ([9, 12, 33, 28])**.** *A tiling of the plane is a collection of sets ("tiles") whose union is the entire plane, but the interiors of different tiles are disjoint. A uniform tiling is an edge-to-edge filling*

Table 6: Previous Learning Experiments

Dimension	Models	WordNet Nouns		WordNet Verbs		Gr-QC	
		MAP	MR	MAP	MR	MAP	MR
2	Poincaré	0.092	95.01	0.478	6.24	0.566	69.11
	Lorentz	0.371	19.07	0.701	2.35	0.556	**63.62**
	L-tiling-RSGD	**0.390**	**17.52**	0.721	2.36	0.564	71.36
	L-tiling-SGD	0.341	21.53	0.726	**2.10**	**0.582**	65.19
	H-tiling	0.385	17.70	**0.741**	2.28	0.568	65.84
4	2*L-tiling-RSGD	/	/	0.858	1.37	0.717	12.32
5	Poincaré	0.850	4.76	**0.953**	1.23	0.712	34.77
	Lorentz	0.851	3.78	0.935	1.19	0.712	33.95
	H-tiling	**0.869**	**3.62**	**0.953**	**1.17**	**0.716**	**32.57**
6	3*L-tiling-RSGD	/	/	0.935	1.13	**0.852**	**4.45**
10	Poincaré	0.875	3.88	**0.954**	1.23	**0.730**	29.86
	Lorentz	0.872	3.45	0.951	**1.14**	0.724	29.50
	H-tiling	**0.894**	**3.25**	**0.954**	1.15	0.726	**29.45**

of the hyperbolic plane, which has regular congruent polygons as faces and is vertex-transitive (there is an isometry mapping any vertex onto any other).

Definition 4 ([21, 2, 37]). *A Fuchsian group G is a discrete subgroup of the 2×2 projective special linear group over $\mathbb{R}$, $PSL(2, \mathbb{R})$.*

Definition 5 ([37]). *The Dirichlet domain for G centered at $z_0 \in \mathbb{H}^2$ is*

$$\square(G; z_0) = \{z \in \mathbb{H}^2 : d(z, z_0) \leq d(gz, z_0), \forall g \in G\}$$

Definition 6 ([37, 36]). *A fundamental domain for G is a close set $F \subset \mathbb{H}^2$ such that*

- $\{gx | \forall g \in G, x \in F\} = \mathbb{H}^2$
- $\{gx | \forall x \in F^{\circ}\} \cap F^{\circ} = \emptyset, \forall g \in G/\{1\}$, *where* $^{\circ}$ *denotes the interior.*

Theorem 1. *The worst-case representation error (Definition 1) in the Lorentz model using floating-point arithmetic (with machine epsilon ϵ_m) is $\delta_l^d = \operatorname{arcosh}(1 + \epsilon_m(2\cosh^2(d) - 1))$, where d is the hyperbolic distance to origin. This becomes $\delta_l^d = 2d + \log(\epsilon_m) + o(\epsilon_m^{-1}\exp(-2d))$ if $d = O(-\log \epsilon_m)$.*

Theorem 4. *The representation error (Definition 1) in L-tiling model is bounded as $\delta_{lt}^d \leq \sqrt{5\epsilon_m} + 15\epsilon_m/4 + o(\epsilon_m)$, where ϵ_m is the machine error.*

Proof of Theorem 1 and Theorem 4. For a real point $(g, \boldsymbol{x})$ in L-tiling model, where $g \in G$, $\boldsymbol{x} \in F$, we represent it as $(g, \mathrm{fl}(\boldsymbol{x}))$, then we get the representation error as follows:

$$\delta_{lt}^d = d_{lt}((g, \boldsymbol{x}), (g, \mathrm{fl}(\boldsymbol{x}))) = \operatorname{arcosh}(-\boldsymbol{x}^T g^T g_{lt} g \mathrm{fl}(\boldsymbol{x})) = \operatorname{arcosh}(-\boldsymbol{x}^T g_{lt} \mathrm{fl}(\boldsymbol{x}))$$

Note that $|x_i - \mathrm{fl}(x_i)| \leq \epsilon_m x_i$, so we have

$$\begin{aligned} \delta_{lt}^d &= \operatorname{arcosh}\left(-(x_1, x_2, x_3)\, g_{lt} \begin{pmatrix} (1+\epsilon_1)x_1 \\ (1+\epsilon_2)x_2 \\ (1+\epsilon_3)x_3 \end{pmatrix} \right) \\ &= \operatorname{arcosh}((1+\epsilon_1)x_1^2 - (1+\epsilon_2)x_2^2 - (1+\epsilon_3)x_3^2) \\ &= \operatorname{arcosh}(1 + \epsilon_1 x_1^2 - \epsilon_2 x_2^2 - \epsilon_3 x_3^2) \end{aligned}$$

since $x_1^2 + x_2^2 + x_3^2 \leq B_f$, then we derive that $\delta_{lt}^d \leq \operatorname{arcosh}(1 + \epsilon_m B_f) = \sqrt{\epsilon_m B_f} + 3\epsilon_m B_f/4 + o(\epsilon_m)$, simple calculation will lead to $B_f = 5$, then $\delta_{lt}^d \leq \sqrt{5\epsilon_m} + 15\epsilon_m/4 + o(\epsilon_m)$. If we consider the representation error in Lorentz model, the only difference is that x is not bounded in the fundamental domain any more. Then we can get that $\delta_l^d = \operatorname{arcosh}(1+\epsilon_m||x||^2)$, noticed that $\cosh d = -\boldsymbol{x}^T g_{lt}\boldsymbol{O} = x_1$, where d is the distance to origin, then

$$\delta_l^d = \operatorname{arcosh}(1+\epsilon_m(x_1^2+x_2^2+x_3^2)) = \operatorname{arcosh}(1+\epsilon_m(2x_1^2-1)) = \operatorname{arcosh}(1+\epsilon_m(2\cosh^2(d)-1)),$$

which becomes $\delta_l^d = 2d+\log(\epsilon_m)+o(\epsilon_m^{-1}\exp(-2d))$ if $d = O(-\log\epsilon_m)$, this error also generalize similarly to high dimensional Lorentz model. $\square$

Theorem 2. $F = \{(x_1,x_2,x_3)\in\mathcal{L}^2 | \max(2x_2^2-x_3^2, 2x_3^2-x_2^2) < 1\}$ *is a fundamental domain of* G*. Any point in* $\mathcal{L}^2$ *can be mapped by* G *to one unique point in* F *or to a point on its boundary.*

Proof. To begin with, we prove that F is the Dirichlet domain for G centered at $O\in\mathbb{H}^2$ denoted as $\square(G)$. Firstly, we show that $F\subset\square(G)$, that is, for any $z\in F$, we have $d(z,O)\le d(Uz,O)$ for all $U\in G$. It suffices to show

$$z_1 \le z_1u_{11}+z_2u_{12}+z_3u_{13},$$

where $z_1^2 = 1+z_2^2+z_3^2$. Also note that $U\in G$, then $U^Tg_lU = g_l$, from which we can derive that $u_{11}^2 = 1+u_{12}^2+u_{13}^2$. Further, from the construction of G, we can write $u_{11} = t_{11}, u_{12} = \sqrt{3}t_{12}, u_{13} = \sqrt{3}t_{13}$, where t_{1i} is an integer, then $t_{11}^2 = 1+3t_{12}^2+3t_{13}^2$. Consider following:

$$\begin{aligned}
& z_1 \le z_1u_{11}+z_2u_{12}+z_3u_{13} \\
\Longleftrightarrow\ & z_1 \le z_1t_{11}+\sqrt{3}z_2t_{12}+\sqrt{3}z_3t_{13} \\
\Longleftrightarrow\ & -\sqrt{3}(z_2t_{12}+z_3t_{13}) \le z_1(t_{11}-1) \\
\Longleftarrow\ & 3(z_2t_{12}+z_3t_{13})^2 \le z_1^2(t_{11}-1)^2 && \triangleright\, z_1, t_{11}\ge 1 \\
\Longleftrightarrow\ & 3(z_2^2t_{12}^2+z_3^2t_{13}^2+2t_{12}t_{13}z_2z_3) \le (1+z_2^2+z_3^2)(1+3t_{12}^2+3t_{13}^2-2t_{11}+1) \\
\Longleftrightarrow\ & 6t_{12}t_{13}z_2z_3 \le (3t_{12}^2+3t_{13}^2-2t_{11}+2)+z_2^2(3t_{13}^2-2t_{11}+2)+z_3^2(3t_{12}^2-2t_{11}+2) \\
\Longleftarrow\ & 2t_{11}z_1^2 \le (3t_{12}^2+3t_{13}^2)+2z_1^2 && \triangleright\, z_2^2t_{13}^2+z_3^2t_{12}^2 \ge 2z_2z_3t_{13}t_{12} \\
\Longleftrightarrow\ & 2t_{11}z_1^2 \le (t_{11}^2-1)+2z_1^2 \\
\Longleftrightarrow\ & 2z_1^2(t_{11}-1) \le t_{11}^2-1 \\
\Longleftrightarrow\ & 2z_1^2 \le t_{11}+1 && \triangleright\, t_{11}-1\ge 0 \\
\Longleftarrow\ & 5 \le t_{11} && \triangleright\, z_1\le\sqrt{3} \Leftarrow z\in F
\end{aligned}$$

Hence, if $t_{11}\ge 5$, then the inequality is proved. If $t_{11} < 5$, since t_{11} is an integer, then $t_{11} = 1,2,3,4$.

- If $t_{11} = 1$, then U is identity matrix, the inequality is satisfied.
- If $t_{11} = 2$, then the integer solutions to $t_{11}^2 = 1+3t_{12}^2+3t_{13}^2$ is $\{g_a, g_b, g_a^{-1}, g_b^{-1}\}$, the inequality is satisfied by simply checking one by one.
- If $t_{11} = 3$, then there is no integer solutions to $t_{11}^2 = 1+3t_{12}^2+3t_{13}^2$.
- If $t_{11} = 4$, then the solutions to $t_{11}^2 = 1+3t_{12}^2+3t_{13}^2$ is $(t_{11}, t_{13}, t_{13}) = (4,2,1), (4,1,2)$. Manually check will find that the inequality is satisfied in both cases.

Then $F\subset\square(G)$, also note that with Algorithm 1, for any $z\in\square(G)$, it will be mapped by some $V\in G$ such that $Vz\in F$ and $d(Vz,O)\le d(z,O)$. From the definition of $\square(G)$, $d(Vz,O)\ge d(z,O)$, then $(Vz)^Tg_lO = d(Vz,O) = d(z,O) = z^Tg_lO$, which leads to $V_{11} = 1$, then $V = I$ considering $V^Tg_lV = g_l$, thus, $z = Vz\in F$ and $\square(G)\subset F$, which shows that $F = \square(G)$.

For the second part of the proof, we show that F is a fundamental domain for G. According to Theorem 37.1.10 in [37], it suffices to show that $Stab_G(O) = \{1\}$. Consider $TO_H = O_H$, where $T\in G$. From $(T-I)O_H = 0$, we can get that

$$T = \begin{bmatrix} 1 & 0 \\ 0 & B \end{bmatrix}.$$

where $B^T = B$. Also note $T\in G$, then $B^2 = I$, these two conditions lead to that $B = I$. Hence, $Stab_G(O) = \{1\}$, then F is a fundamental domain for G. Since fundamental domain F only contains one element in the orbit, then for any point in the space, it can only be mapped to one unique point in F. $\square$

Theorem 3. *For any point in the Lorentz model, Algorithm 1 converges and stops within* $1 + 7d$ *steps, where* $d = d(\boldsymbol{x}, \boldsymbol{O})$ *denotes the distance from* $\boldsymbol{x}$ *to the origin.*

Proof. We just consider $x_2 \leq -|x_3|$ case in the Algorithm, other cases can be proved in the same way, then we have

$$\frac{\cosh d(L \cdot g_a \cdot L^{-1} \cdot \boldsymbol{x}, O)}{\cosh d(\boldsymbol{x}, \boldsymbol{O})} = 2 + \frac{\sqrt{3}x_2}{\sqrt{1 + x_2^2 + x_3^2}} < 1$$

then the distance to the origin in the space is monotonically decreasing as Algorithm 1 goes, note that this distance is bounded by 0, then it will converge.

To see the steps required for the algorithm to finish, we may assume that $\max\{|x_2|, |x_3|\} \geq C_0 > 1$, due to the symmetry of this algorithm, also consider the case $x_2 \leq -|x_3|$, then $|x_2| \geq C_0$, we have

$$\frac{\cosh d(L \cdot g_a \cdot L^{-1} \cdot \boldsymbol{x}, \boldsymbol{O})}{\cosh d(\boldsymbol{x}, \boldsymbol{O})} \leq 2 - \sqrt{\frac{3C_0}{2C_0 + 1}} \leq 1$$

Hence,

$$\begin{aligned} & d(\boldsymbol{x}, \boldsymbol{O}) - d(L \cdot g_a \cdot L^{-1} \cdot \boldsymbol{x}, \boldsymbol{O}) \\ \geq & \operatorname{arcosh}(\sqrt{1 + x_2^2 + x_3^2}) - \operatorname{arcosh}\left((2 - \sqrt{\frac{3C_0}{2C_0 + 1}})\sqrt{1 + x_2^2 + x_3^2}\right) \\ \geq & -\log(2 - \sqrt{\frac{3C_0}{2C_0 + 1}}) \end{aligned}$$

so $d(\boldsymbol{x}, \boldsymbol{O})$ will decrease monotonically for at most $s_0(C_0)$ steps, where

$$s_0(C_0) = \frac{d(\boldsymbol{x}, \boldsymbol{O})}{-\log(2 - \sqrt{\frac{3C_0}{2C_0+1}})}$$

Consider $\max\{|x_2|, |x_3|\}$ at the boundary between F and its neighborhood tiles:

$$\min_{x \in L\{g_a, g_b, g_a^{-1}, g_b^{-1}\}L^{-1}F} \max\{|x_2|, |x_3|\} = \frac{5}{2}\sqrt{2}$$

Hence, we can choose $C_0 = \frac{5}{2}\sqrt{2}$, then $d(\boldsymbol{x}, \boldsymbol{O})$ will decrease monotonically until $\max\{|x_2|, |x_3|\} < C_0$ within $s_0(5\sqrt{2}/2)$ steps, which means x lies either in F or its 4 neighborhood tiles, so totally it will cost at most $s_0(5\sqrt{2}/2) + 1 \leq 1 + 7d$ steps. ☐

Lemma 1. *If the integer matrix* U *is given, then corresponding VBW encoding can be derived using Algorithm 1. Further, if only* U_{21}, U_{31} *are given, then* U *can be reconstructed using Algorithm 1.*

Proof. If U is given, consider the point (U, O) in the L-tiling model, which is in correspondence to $x = LUL^{-1}O$ in the Lorentz model, then we can map x to (U', u') with Algorithm 1, where we choose a generator at each step to get a generator order string, with which we can reconstruct U'. Since each point in the Lorentz model will be mapped to a unique point in F as Theorem 2 states, also x can be mapped to (U, O) and (U', u'), then $u' = O$. The question is whether $U = U'$, consider $LUL^{-1}O = LU'L^{-1}O$, which leads to $(LU'^{-1}UL^{-1} - I)O = 0$, since $Stab_G(O) = \{1\}$ as the second part in the proof of Theorem 2 proved, then $LU'^{-1}UL^{-1} = I$ to get $U = U'$. Hence, given U, we can get its generator order string, which can be then used to get the VBW encoding accordingly. Further, note that $U^T M_3 U = M_3$, then we have

$$U_{11}^2 = 1 + \frac{U_{21}^2 + U_{31}^2}{3}.$$

Therefore, we can compute U_{11} if only U_{21}, U_{31} were given to get the first column of U. Since $x = LUL^{-1}O$ and $O = (1, 0, 0)$, then the first column of U suffices to get x, then we can reconstruct U out using Algorithm 1. ☐

Lemma 2. Q_{11} *has the largest absolute value in* $Q = U^T M_3 V$.

Proof. Note that $Q = U^T M_3 V = M_3 U^{-1} V = M_3 T$, where T is an integer matrix generated by g_a and g_b, so we have $T^T M_3 T = M_3$, using this relation, we can get following equations:

$$\begin{aligned} t_{11}^2 &= 1 + \frac{t_{21}^2 + t_{31}^2}{3} \\ t_{11}^2 &= 1 + 3(t_{12}^2 + t_{13}^2) \\ 3t_{12}^2 &= (t_{22}^2 + t_{32}^2) - 1 \\ 3t_{13}^2 &= (t_{23}^2 + t_{33}^2) - 1 \end{aligned}$$

since $Q_{11} = 3t_{11}$, from first formula, we get that $Q_{11}^2 = 9t_{11}^2 \geq 3t_{11}^2 > t_{21}^2 + t_{31}^2 = Q_{21}^2 + Q_{31}^2$, so Q_{11} has the largest absolute value in the first column of Q. From the second formula, we get that $Q_{11}^2 \geq t_{11}^2 > t_{12}^2 + t_{13}^2$, so Q_{11} has the largest absolute value in the first row of Q. Then combine formulas 2,3,4 and we get that

$$1 + t_{11}^2 = t_{22}^2 + t_{32}^2 + t_{23}^2 + t_{33}^2$$

From first formula, we know that $t_{11} \geq 1$, then we have $Q_{11}^2 \geq 2t_{11}^2 \geq 1 + t_{11}^2 = t_{22}^2 + t_{32}^2 + t_{23}^2 + t_{33}^2$ Combine above results, clearly that Q_{11} has the largest absolute value in Q, which finishes our proof ☐

Theorem 5. *The representation error (Definition 1) in H-tiling model is bounded as $\delta_{ht}^d = \sqrt{(n+3)\epsilon_m}/2 + (n+3)\epsilon_m/4 + o(\epsilon_m)$, where ϵ_m is the machine error.*

Proof. For a real point $(j, \boldsymbol{k}, \boldsymbol{x}) \in \mathbb{Z} \times (\mathbb{Z}^{n-1} \times \{0\}) \times S$ in H-tiling model, we represent it as $(j, \boldsymbol{k}, \mathrm{fl}(\boldsymbol{x}))$, then we get the representation error as follows:

$$\delta_{ht}^d = d_{ht}((j, \boldsymbol{k}, \boldsymbol{x}), (j, \boldsymbol{k}, \mathrm{fl}(\boldsymbol{x}))) = \operatorname{arcosh}(1 + \frac{||\boldsymbol{x} - \mathrm{fl}(\boldsymbol{x}))||^2}{2x_n \mathrm{fl}(x_n)})$$

Note that $|x_i - \mathrm{fl}(x_i)| \leq \epsilon_m x_i$, so we have

$$\delta_{ht}^d = \operatorname{arcosh}(1 + \frac{\sum_{i=1}^{n} \epsilon_i x_i^2}{2(1+\epsilon_n)x_n^2})$$

since $0 \leq x_1, \cdots, x_{n-1} < 1 \leq x_n < 2$, then we derive that $\delta_{ht}^d \leq \operatorname{arcosh}(1 + \frac{(n+3)\epsilon_m}{2(1-\epsilon_m)}) = \sqrt{(n+3)\epsilon_m}/2 + (n+3)\epsilon_m/4 + o(\epsilon_m)$. ☐

Proof of RSGD algorithm. Here we show the equivalence between the RSGD algorithm of L-tiling model described in appendix A and that in Lorentz model. To begin with, consider the RSGD algorithm in Lorentz model. Let $x_t, y_t \in \mathcal{H}^2$, then we have $d(x_t, y_t) = \operatorname{arcosh}(-x_t^T g_l y_t)$, the Euclidean gradient of x_t can be computed as

$$\nabla_{x_t} d(x_t, y_t) = \frac{-g_l y_t}{\sqrt{(x_t^T g_l y_t)^2 - 1}},$$

to get the Riemannian gradient in the model, we make use of the pull-back metric as follows,

$$h_t = g_{l,x_t}^{-1} \nabla_{x_t} d(x_t, y_t) = \frac{-y_t}{\sqrt{(x_t^T g_l y_t)^2 - 1}},$$

further we project this Riemannian gradient into the tangent space at x_t,

$$\operatorname{grad}_{x_t} d = h_t + \langle x_t, h_t \rangle_L x_t = -\frac{y_t + (x_t^T g_l y_t) x_t}{\sqrt{(x_t^T g_l y_t)^2 - 1}},$$

then we make use of the exponential map in Lorentz model to update,

$$x_{t+1} = \exp_{x_t}(v) = \cosh(||v||_L) x_t + \sinh(||v||_L) \frac{v}{||v||_L},$$

where

$$v = -\eta \cdot \text{grad}_{x_t} d = \eta \frac{y_t + (x_t^T g_l y_t) x_t}{\sqrt{(x_t^T g_l y_t)^2 - 1}}.$$

Now consider the norm of v under hyperbolic metric,

$$\begin{aligned}
||v||_L^2 =& v^T g_l v \\
=& \frac{\eta^2}{(x_t^T g_l y_t)^2 - 1} (y_t^T g_l y_t + (x_t^T g_l y_t)^2 + (x_t^T g_l y_t)^2 + (x_t^T g_l y_t)^2 x_t^T g_l x_t) \\
=& \frac{\eta^2}{(x_t^T g_l y_t)^2 - 1} (-1 + (x_t^T g_l y_t)^2 + (x_t^T g_l y_t)^2 - (x_t^T g_l y_t)^2) \\
=& \eta^2
\end{aligned}$$

hence we derived the RSGD algorithm in the Lorentz model as

$$x_{t+1} = \exp_{x_t}(v) = \cosh(\eta) x_t + \sinh(\eta) \frac{y_t + (x_t^T g_l y_t) x_t}{\sqrt{(x_t^T g_l y_t)^2 - 1}} \tag{2}$$

For the second part, we turn to L-tiling model, let $x_t = LUL^{-1}u_t, y_t = LVL^{-1}v_t$, the distance is

$$d(x_t, y_t) = \text{arcosh}(u_t^T L^{-T} Q L^{-1} v_t),$$

then the Euclidean gradient of u_t can be computed as

$$\nabla_{u_t} d(x, y) = \frac{L^{-T} Q L^{-1} v_t}{\sqrt{(u_t^T L^{-T} Q L^{-1} v_t)^2 - 1}}.$$

In the same way, make use of the pull-back by the metric matrix, we derived the Riemannian gradient

$$h_t = g_{lt,u_t}^{-1} \nabla_{u_t} d(x_t, y_t) = \frac{g_{lt} L^{-T} Q L^{-1} v_t}{\sqrt{(u_t^T L^{-T} Q L^{-1} v_t)^2 - 1}},$$

also project it into the tangent space at u_t,

$$\text{grad}_{u_t} d = h_t + \langle u_t, h_t \rangle_L u_t = \frac{g_{lt} L^{-T} Q L^{-1} v_t + (v_t^T L^{-T} Q^T L^{-1} u_t) u_t}{\sqrt{(u_t^T L^{-T} Q L^{-1} v_t)^2 - 1}}.$$

then the update rule in the tiling-based model is

$$u_{t+1} = \exp_{u_t}(v),$$

where

$$v = -\eta \cdot \text{grad}_{u_t} d = -\eta \frac{g_{lt} L^{-T} Q L^{-1} v_t + (v_t^T L^{-T} Q^T L^{-1} u_t) u_t}{\sqrt{(u_t^T L^{-T} Q L^{-1} v_t)^2 - 1}},$$

then consider the norm of v under hyperbolic metric,

$$\begin{aligned}
||v||_L^2 =& v^T g_{lt} v \\
=& \frac{\eta^2}{(u_t^T L^{-T} Q L^{-1} v_t)^2 - 1} [g_{lt} L^{-T} Q L^{-1} v_t + (v_t^T L^{-T} Q^T L^{-1} u_t) u_t]^T \cdot \\
& \qquad g_{lt} [g_{lt} L^{-T} Q L^{-1} v_t + (v_t^T L^{-T} Q^T L^{-1} u_t) u_t] \\
=& \frac{\eta^2}{(u_t^T L^{-T} Q L^{-1} v_t)^2 - 1} [v_t^T L^{-T} Q^T L^{-1} g_{lt} L^{-T} Q L^{-1} v_t \\
& \qquad + 2(v_t^T L^{-T} Q^T L^{-1} u_t)^2 + (v_t^T L^{-T} Q^T L^{-1} u_t) u_t^T g_{lt} u_t] \\
=& \frac{\eta^2}{(u_t^T L^{-T} Q L^{-1} v_t)^2 - 1} [v_t^T L^{-T} Q^T L^{-1} g_{lt} L^{-T} Q L^{-1} v_t \\
& \qquad + 2(v_t^T L^{-T} Q^T L^{-1} u_t)^2 - (v_t^T L^{-T} Q^T L^{-1} u_t)] \\
=& \frac{\eta^2}{(u_t^T L^{-T} Q L^{-1} v_t)^2 - 1} [(L^{-T} Q L^{-1} v_t)^T g_{lt} (L^{-T} Q L^{-1} v_t) + (v_t^T L^{-T} Q^T L^{-1} u_t)^2]
\end{aligned}$$

Since $U \in G_0$, then it follows

$$Q = U^T M_3 V = M_3 U^{-1} V = M_3 W, \quad W \in G.$$

So we get

$$L^{-T} Q L^{-1} v_t = L^{-T} M_3 W L^{-1} v_t,$$

hence

$$\begin{aligned}(L^{-T} Q L^{-1} v_t)^T g_{lt} (L^{-T} Q L^{-1} v_t) =& v_t^T L^{-T} W^T M_3^T L^{-1} g_{lt} L^{-T} M_3 W L^{-1} v_t \\ =& - v_t^T L^{-T} W^T M_3 W L^{-1} v_t \\ =& - v_t^T L^{-T} M_3 L^{-1} v_t \\ =& v_t^T g_{lt} v_t \\ =& - 1\end{aligned}$$

so $\|v\|_L^2 = \eta^2$, then the RSGD algorithm in L-tiling model is

$$u_{t+1} = \exp_{u_t}(v) = \cosh(\eta) u_t - \sinh(\eta) \frac{g_{lt} L^{-T} Q L^{-1} v_t + (v_t^T L^{-T} Q^T L^{-1} u_t) u_t}{\sqrt{(u_t^T L^{-T} Q L^{-1} v_t)^2 - 1}} \tag{3}$$

For the third part, again consider the RSGD algorithm in Lorentz model, from Equation 2, we have that

$$L U L^{-1} u_{t+1} = \cosh(\eta) L U L^{-1} u_t + \sinh(\eta) \frac{L V L^{-1} v_t - (v_t^T L^{-T} Q^T L^{-1} u_t) L U L^{-1} u_t}{\sqrt{(v_t^T L^{-T} Q^T L^{-1} u_t)^2 - 1}},$$

so RSGD algorithm in Lorentz model is equivalent to:

$$u_{t+1} = \cosh(\eta) u_t + \sinh(\eta) \frac{L U^{-1} V L^{-1} v_t - (v_t^T L^{-T} Q^T L^{-1} u_t) u_t}{\sqrt{(v_t^T L^{-T} Q^T L^{-1} u_t)^2 - 1}}, \tag{4}$$

note that $U^T M_3 U = M_3$, then $U^{-1} = M_3^{-1} U^T M_3$, with simple computation, we get that

$$L U^{-1} V L^{-1} = L M_3^{-1} U^T M_3 V L^{-1} = L M_3^{-1} Q L^{-1} = -g L^{-T} Q L^{-1},$$

hence, RSGD algorithm in L-tiling model (Equation 3) becomes the same as RSGD algorithm in Lorentz model (Equation 4), which finishes our proof. □

Error for Computing in L-tiling Model We approximate $(U, u), (V, v)$ with $(U, \mathrm{fl}(u))$ and $(V, \mathrm{fl}(v))$, here we provide the error of computing in L-tiling model together with that in Lorentz model.

Theorem 6. *The worst case error of computing distance in Lorentz model using float is*

$$|d_l^{fl}(fl(x), fl(y)) - d_l(x, y)| = \frac{2\cosh(d(x, O))\cosh(d(y, O))}{\sinh(d(x, y))} \epsilon_m + \epsilon_m d(x, y) + o(\epsilon_m),$$

the error of computing distance in L-tiling model using float is

$$|d_{lt}^{fl}((U, fl(u)), (V, fl(v))) - d_{lt}((U, u), (V, v))| = \begin{cases} d\epsilon_m + A_1(C_0)\epsilon_m + o(C_0^{-2} + \epsilon_m), d \geq C_0 \\ d\epsilon_m + [A_2(C_0) + \dfrac{A_3(C_0)}{\tanh(d)}]\epsilon_m + o(\epsilon_m), d < C_0 \end{cases}$$

where d is the real distance between two points, $A_i(C_0)$ are constants only depends on C_0, d^{fl} means that inside computation like multiplication are performed with machine error ϵ_m.

Remark: The worst case error of distance computation in Lorentz model using float is dominated by $d(x, O), d(y, O)$, this will cause the "NaN" problem when two points are far away from the origin. However, in L-tiling model, the error only depends on d, i.e., how far two points are to each other, also $\tanh$ term is bounded, which controls the distance error and solves the "NaN" problem.

Proof. We consider the Lorentz model at first, let $z = x^T M y$, $\hat{z} = \mathrm{fl}(x)^T M \mathrm{fl}(y)$, then

$$\begin{aligned}|\hat{z} - z| &\le |(1+\epsilon_m)^6 z - z| = 6\epsilon_m |x|^T |y| + o(\epsilon_m) \le 2\epsilon_m x_0 y_0 + o(\epsilon_m)\\ &= 2\epsilon_m \cosh(d_x)\cosh(d_y) + o(\epsilon_m) = \delta_z\end{aligned}$$

thus,

$$\begin{aligned}|d_l^{\mathrm{fl}}(\mathrm{fl}(x), \mathrm{fl}(y)) - d_l(x,y)| &= |(1+\epsilon)\operatorname{arcosh}(x^T M y + \delta_z) - \operatorname{arcosh}(x^T M y)|\\ &= \frac{\delta_z}{\sqrt{(x^T M y)^2 - 1}} + \epsilon d_l + o(\delta_z) = \frac{2\cosh(d_x)\cosh(d_y)}{\sinh(d_l)}\epsilon_m + \epsilon_m d_l + o(\epsilon_m)\end{aligned}$$

As for the distance error in L-tiling model, here in the same way denote $z = h(u)^T L^{-T} \hat{Q} L^{-1} h(v)$, since $h(u), h(v)$ are in the fundamental domain, which is bounded by B_f, so $\|h(u)\|, \|h(v)\| \le \sqrt{B_f} = \sqrt{5}$, also $\hat{Q}$ is bounded, then using Cauchy inequality, we have

$$|z| \le \|h(u)\|\left(\frac{\sqrt{7}}{3}|h(v)_1| + \sqrt{\frac{7}{3}}|h(v)_2| + \sqrt{\frac{7}{3}}|h(v)_3|\right) \le \frac{7}{3}\|h(u)\|\|h(v)\| \le \frac{7}{3}B_f^2 = 35/3$$

so the distance is

$$d_{lt}((U,u),(V,v)) = \log(Q_{11}) + \log\left(z + \sqrt{z^2 - Q_{11}^{-2}}\right)$$

then $\log(Q_{11}) \le d$, further note that

$$\begin{aligned}\operatorname{arcosh}(Q_{11}/3) =& d(L^{-T}UL^{-1}O, L^{-T}VL^{-1}O)\\ \le& d(L^{-T}UL^{-1}O, L^{-T}UL^{-1}h(u)) + d(L^{-T}UL^{-1}h(u), L^{-T}VL^{-1}h(v))\\ &+ d(L^{-T}VL^{-1}O, L^{-T}VL^{-1}h(v))\\ =& d(O, h(u)) + d(L^{-T}UL^{-1}h(u), L^{-T}VL^{-1}h(v)) + d(O, h(v))\\ \le& d + 2\operatorname{arcosh}(\sqrt{3}) = d + 2\log(\sqrt{3} + \sqrt{2})\end{aligned}$$

in this way, we get

$$\begin{aligned}z^2 = z^2 - Q_{11}^{-2} + Q_{11}^{-2} =& Q_{11}^{-2}(\cosh^2(d) - 1) + Q_{11}^{-2} \ge Q_{11}^{-2} + \frac{\sinh^2(d)}{9\cosh^2(d + 2B_f)}\\ z =& Q_{11}^{-1}\cosh(d) \ge \frac{\cosh(d)}{3\cosh(d + 2\log(\sqrt{3} + \sqrt{2}))}\end{aligned}$$

Now, we consider the first term of calculating distance $\log(Q_{11})$, in order to avoid overflow, we computed with following formula.

$$\log(Q_{11}) = \frac{\log(3)}{2} + \log(U^{11}) + \log(V_{11}) + \log\left(1 + \frac{U^{12}}{U^{11}}\frac{V_{12}}{V_{11}} + \frac{U^{13}}{U^{11}}\frac{V_{13}}{V_{11}}\right)$$

then

$$
\begin{aligned}
\mathrm{flc}(\log(Q_{11})) =& \left(\frac{\log(3)}{2} + \log(U^{11}) + \log(V_{11})\right)(1+\epsilon) \\
&+ \mathrm{fl}\left(\log(1 + \frac{U^{12}}{U^{11}}\frac{V_{12}}{V_{11}}(1+\epsilon)^2 + \frac{U^{13}}{U^{11}}\frac{V_{13}}{V_{11}}(1+\epsilon)^2)\right) \\
=& \left(\frac{\log(3)}{2} + \log(U^{11}) + \log(V_{11})\right)(1+\epsilon) \\
&+ \left(\log(1 + \frac{U^{12}}{U^{11}}\frac{V_{12}}{V_{11}}(1+\epsilon)^2 + \frac{U^{13}}{U^{11}}\frac{V_{13}}{V_{11}}(1+\epsilon)^2)\right)(1+\epsilon) \\
=& \left(\frac{\log(3)}{2} + \log(U^{11}) + \log(V_{11})\right)(1+\epsilon) \\
&+ (1+\epsilon)\log(1 + \frac{U^{12}}{U^{11}}\frac{V_{12}}{V_{11}}(1+2\epsilon) + \frac{U^{13}}{U^{11}}\frac{V_{13}}{V_{11}}(1+2\epsilon) + o(\epsilon)) \\
=& \left(\frac{\log(3)}{2} + \log(U^{11}) + \log(V_{11})\right)(1+\epsilon) + (1+\epsilon)\log(1 + \frac{U^{12}}{U^{11}}\frac{V_{12}}{V_{11}} + \frac{U^{13}}{U^{11}}\frac{V_{13}}{V_{11}}) \\
&+ (1+\epsilon)\frac{2\epsilon(\frac{U^{12}}{U^{11}}\frac{V_{12}}{V_{11}} + \frac{U^{13}}{U^{11}}\frac{V_{13}}{V_{11}}) + o(\epsilon)}{1 + \frac{U^{12}}{U^{11}}\frac{V_{12}}{V_{11}} + \frac{U^{13}}{U^{11}}\frac{V_{13}}{V_{11}}} \\
=& \log(Q_{11})(1+\epsilon) + (1+\epsilon)\left(\frac{2\epsilon(\frac{U^{12}}{U^{11}}\frac{V_{12}}{V_{11}} + \frac{U^{13}}{U^{11}}\frac{V_{13}}{V_{11}}) + o(\epsilon)}{1 + \frac{U^{12}}{U^{11}}\frac{V_{12}}{V_{11}} + \frac{U^{13}}{U^{11}}\frac{V_{13}}{V_{11}}}\right) \\
=& \log(Q_{11})(1+\epsilon) + 2\epsilon\frac{U^{12}V_{12} + U^{13}V_{13}}{U^{11}V_{11} + U^{12}V_{12} + U^{13}V_{13}} + o(\epsilon)
\end{aligned}
$$

Here flc means calculating with float arithmetic, hence, the error of computing the first term is

$$
\delta_Q = |\mathrm{flc}(\log(Q_{11})) - \log(Q_{11})| \le \log(Q_{11})\epsilon_m + \frac{1}{2}\epsilon_m + o(\epsilon_m)
$$

Then we consider the error of computing $z = h(u)^T L^{-T}\hat{Q}L^{-1}h(v)$, given by following formula:

$$
\begin{aligned}
|\mathbf{flc}(z) - z| &\le |(1+\epsilon)^7 h(u)^T L^{-T}\hat{Q}L^{-1}h(v) - z| \le 7\epsilon_m |h(u)|^T L^{-T}|\hat{Q}|L^{-1}|h(v)| + o(\epsilon_m) \\
&\le \frac{245}{3}\epsilon_m + o(\epsilon_m) = \delta_z
\end{aligned}
$$

based on this error, we consider the error for the second term of distance

$$
\begin{aligned}
\delta_2 =& \mathrm{flc}\left(\log(z + \sqrt{z^2 - Q_{11}^{-2}})\right) - \log(z + \sqrt{z^2 - Q_{11}^{-2}}) \\
=& (1+\epsilon_1)(\log((1+\epsilon_2)(z + \delta_z + (1+\epsilon_3)\sqrt{(1+\epsilon_4)((1+\epsilon_5)(z+\delta_z)^2 - (1+\epsilon_6)Q_{11}^{-2})}))) \\
&- \log(z + \sqrt{z^2 - Q_{11}^{-2}}) \\
=& (1+\epsilon_1)(\log(z + \delta_z + (1+\epsilon_3)\sqrt{(1+\epsilon_4)((1+\epsilon_5)(z+\delta_z)^2 - (1+\epsilon_6)Q_{11}^{-2})})) \\
&- \log(z + \sqrt{z^2 - Q_{11}^{-2}}) + \epsilon_2 + o(\epsilon_m) \\
=& (1+\epsilon_1)(\log(z + \delta_z + (1+\epsilon_3)\sqrt{(1+\epsilon_4)} \cdot \sqrt{(z+\delta_z)^2 - Q_{11}^{-2} + \epsilon_7((z+\delta_z)^2 + Q_{11}^{-2})})) \\
&- \log(z + \sqrt{z^2 - Q_{11}^{-2}}) + \epsilon_2 + o(\epsilon_m) \\
=& (1+\epsilon_1)(\log(z + \frac{245}{3}\epsilon_m + (1 + \frac{2}{3}\epsilon_8)\sqrt{(z + \frac{245}{3}\epsilon_m)^2 - Q_{11}^{-2} + \epsilon_7((z+\delta_z)^2 + Q_{11}^{-2})})) \\
&- \log(z + \sqrt{z^2 - Q_{11}^{-2}}) + \epsilon_2 + o(\epsilon_m)
\end{aligned}
$$

We divide it into two cases, firstly, consider Taylor expansion here when $Q_{11}z \geq C_0$, where $C_0 \geq 1$ is a large constant, then

$$\begin{aligned}
&\log(z + \frac{245}{3}\epsilon_m + (1+\frac{2}{3}\epsilon_8)\sqrt{(z+\frac{245}{3}\epsilon_m)^2 - Q_{11}^{-2} + \epsilon_7((z+\frac{245}{3}\epsilon_m)^2 + Q_{11}^{-2})})\\
=&\log(z + (1+\frac{2}{3}\epsilon_8)\sqrt{z^2 - Q_{11}^{-2} + \epsilon_7(z^2+Q_{11}^{-2})}) + \frac{245\epsilon_m}{3\sqrt{z^2 - Q_{11}^{-2} + \epsilon_7(z^2+Q_{11}^{-2})}} + o(\epsilon_m)\\
=&\log(z + \sqrt{z^2 - Q_{11}^{-2} + \epsilon_7(z^2+Q_{11}^{-2})}) + \frac{2}{3}\epsilon_8 - \frac{2\epsilon_8 z}{3[z+\sqrt{z^2 - Q_{11}^{-2} + \epsilon_7(z^2+Q_{11}^{-2})}]}\\
&+ \frac{245\epsilon_m}{3\sqrt{z^2 - Q_{11}^{-2} + \epsilon_7(z^2+Q_{11}^{-2})}} + o(\epsilon_m)\\
=&\log(z+\sqrt{z^2-Q_{11}^{-2}}) + \frac{(z^2+Q_{11}^{-2})\epsilon_7}{2\sqrt{z^2-Q_{11}^{-2}}(z+\sqrt{z^2-Q_{11}^{-2}})} + \frac{2}{3}\epsilon_8 - \frac{2\epsilon_8 z}{3[z+\sqrt{z^2-Q_{11}^{-2}}]}\\
&+ \frac{245\epsilon_m}{3\sqrt{z^2-Q_{11}^{-2}}} + o(\epsilon_m)\\
=&\log(z+\sqrt{z^2-Q_{11}^{-2}}) + \frac{\epsilon_7}{4} + \frac{7\epsilon_7}{16(Q_{11}z)^2} + \frac{2}{3}\epsilon_8 - \frac{\epsilon_8}{3} - \frac{\epsilon_8}{12(Q_{11}z)^2} + \frac{245\epsilon_m}{3z}\\
&+ O(\epsilon_m Q_{11}^{-2} z^{-3}) + o((Q_{11}z)^{-3}) + o(\epsilon_m)\\
=&\log(z+\sqrt{z^2-Q_{11}^{-2}}) + \frac{\epsilon_7}{4} + \frac{7\epsilon_7}{16(Q_{11}z)^2} + \frac{1}{3}\epsilon_8 - \frac{\epsilon_8}{12(Q_{11}z)^2} + \frac{245\epsilon_m}{3z}\\
&+ O(\epsilon_m Q_{11}^{-2} z^{-3}) + o((Q_{11}z)^{-3}) + o(\epsilon_m)\\
=&\log(z+\sqrt{z^2-Q_{11}^{-2}}) + \frac{\epsilon_7}{4} + \frac{7\epsilon_7}{16(Q_{11}z)^2} + \frac{1}{3}\epsilon_8 - \frac{\epsilon_8}{12(Q_{11}z)^2} + \frac{245\epsilon_m}{3z} + o(C_0^{-2}) + o(\epsilon_m)\\
=&\log(z+\sqrt{z^2-Q_{11}^{-2}}) + (\frac{7}{12} + \frac{25}{48(Q_{11}z)^2})\epsilon_9 + \frac{245\epsilon_m}{3z} + o(C_0^{-2}) + o(\epsilon_m)
\end{aligned}$$

where $|\epsilon_i| \leq \epsilon_m$, the machine error, then the error is

$$\begin{aligned}
&|(1+\epsilon_1)(\log(z+\sqrt{z^2-Q_{11}^{-2}}) + (\frac{7}{12} + \frac{25}{48(Q_{11}z)^2})\epsilon_9 + \frac{245\epsilon_m}{3z} + o(C_0^{-2}) + o(\epsilon_m))\\
&- \log(z+\sqrt{z^2-Q_{11}^{-2}}) + \epsilon_2 + o(\epsilon_m)|\\
=&|(1+\epsilon_1)\left((\frac{7}{12} + \frac{25}{48(Q_{11}z)^2})\epsilon_9 + \frac{245\epsilon_m}{3z} + o(C_0^{-2})\right) + \epsilon_1\log(z+\sqrt{z^2-Q_{11}^{-2}}) + \epsilon_2 + o(\epsilon_m)|\\
=&|(\frac{7}{12} + \frac{25}{48(Q_{11}z)^2})\epsilon_9 + \frac{245\epsilon_m}{3z} + \epsilon_1\log(z+\sqrt{z^2-Q_{11}^{-2}}) + \epsilon_2 + o(C_0^{-2}) + o(\epsilon_m) + \frac{7}{12}\epsilon_1|\\
\leq&\frac{277}{432}\epsilon_m + 63\epsilon_m C_0 Q_{11}^{-1} + \epsilon_m \log 70/3 + \frac{19}{12}\epsilon_m + o(C_0^{-2}) + o(\epsilon_m)\\
\leq&(\frac{961}{432} + \frac{245C_0}{3} + \log 70/3)\epsilon_m + o(C_0^{-2}) + o(\epsilon_m) = \delta_2
\end{aligned}$$

On the other hand, if $Q_{11}z \leq C_0$, then notice that

$$\begin{aligned}
\operatorname{arcosh}(Q_{11}/3) =& d(L^{-T}UL^{-1}O, L^{-T}VL^{-1}O)\\
\leq& d(L^{-T}UL^{-1}O, L^{-T}UL^{-1}h(u)) + d(L^{-T}UL^{-1}h(u), L^{-T}VL^{-1}h(v))\\
&+ d(L^{-T}VL^{-1}O, L^{-T}VL^{-1}h(v))\\
=& d(O, h(u)) + d(L^{-T}UL^{-1}h(u), L^{-T}VL^{-1}h(v)) + d(O, h(v))\\
\leq& \operatorname{arcosh}(Q_{11}z) + 2\log(\sqrt{3}+\sqrt{2})\\
\leq& \operatorname{arcosh}(C_0) + 2\log(\sqrt{3}+\sqrt{2})
\end{aligned}$$

so we can get $Q_{11} \leq E(C_0)$, where $E(C_0)$ is a constant depending on C_0, then we further have

$$\begin{aligned}
&\log(z+\frac{245}{3}\epsilon_m+(1+\frac{2}{3}\epsilon_8)\cdot\sqrt{(z+\frac{245}{3}\epsilon_m)^2-Q_{11}^{-2}+\epsilon_7((z+\frac{245}{3}\epsilon_m)^2+Q_{11}^{-2})})\\
=&\log(Q_{11}z+\frac{245}{3}\epsilon_m Q_{11}-\log(Q_{11})\\
&+(1+\frac{2}{3}\epsilon_8)\cdot\sqrt{(Q_{11}z+\frac{245}{3}\epsilon_m Q_{11})^2-1+\epsilon_7((Q_{11}z+\frac{245}{3}\epsilon_m Q_{11})^2+1))}\\
=&\log(Q_{11}z+(1+\frac{2}{3}\epsilon_8)\sqrt{(Q_{11}z)^2-1+\epsilon_7((Q_{11}z)^2+1)})-\log(Q_{11})+o(\epsilon_m)\\
&+\frac{245}{3}\epsilon_m Q_{11}[1+\frac{Q_{11}z}{\sqrt{(Q_{11}z)^2-1}}]\frac{1}{Q_{11}z+\sqrt{(Q_{11}z)^2-1}}\\
=&\log(Q_{11}z+\sqrt{(Q_{11}z)^2-1+\epsilon_7((Q_{11}z)^2+1)})-\log(Q_{11})+o(\epsilon_m)\\
&+(\frac{245}{3}\epsilon_m Q_{11}+\frac{2}{3}\epsilon_8)[1+\frac{Q_{11}z}{\sqrt{(Q_{11}z)^2-1}}]\frac{1}{Q_{11}z+\sqrt{(Q_{11}z)^2-1}}\\
=&\log(Q_{11}z+\sqrt{(Q_{11}z)^2-1})-\log(Q_{11})+o(\epsilon_m)\\
&+(\frac{245}{3}\epsilon_m Q_{11}+\frac{2}{3}\epsilon_8+\epsilon_7)\cdot[1+\frac{Q_{11}z}{\sqrt{(Q_{11}z)^2-1}}]\frac{1}{Q_{11}z+\sqrt{(Q_{11}z)^2-1}}\\
=&\log(z+\sqrt{z^2-Q_{11}^{-2}})+o(\epsilon_m)\\
&+(\frac{245}{3}\epsilon_m Q_{11}+\frac{2}{3}\epsilon_8+\epsilon_7)\cdot[1+\frac{Q_{11}z}{\sqrt{(Q_{11}z)^2-1}}]\frac{1}{Q_{11}z+\sqrt{(Q_{11}z)^2-1}}\\
=&(\frac{245}{3}C_0E(C_0)+\frac{5}{3})\epsilon_m\cdot[1+\frac{Q_{11}z}{\sqrt{(Q_{11}z)^2-1}}]\frac{1}{Q_{11}z+\sqrt{(Q_{11}z)^2-1}}+o(\epsilon_m)\\
&+\log(z+\sqrt{z^2-Q_{11}^{-2}})
\end{aligned}$$

Hence, we get the error to be

$$\begin{aligned}
\delta_2 =&|\epsilon_1\log(z+\sqrt{z^2-Q_{11}^{-2}})+\epsilon_2+o(\epsilon_m)\\
&+(\frac{245}{3}C_0E(C_0)+\frac{5}{3})\epsilon_m\cdot[1+\frac{Q_{11}z}{\sqrt{(Q_{11}z)^2-1}}]\frac{1}{Q_{11}z+\sqrt{(Q_{11}z)^2-1}}|\\
\leq&(\frac{245}{3}C_0E(C_0)+\frac{5}{3})\cdot[1+\frac{Q_{11}z}{\sqrt{(Q_{11}z)^2-1}}]+1)\epsilon_m+(\log(C_0+\sqrt{C_0^2-1})\\
&-\log(Q_{11})+o(\epsilon_m)\\
\leq&[\log(C_0+\sqrt{C_0^2-1})-\log(Q_{11})+(\frac{245}{3}C_0E(C_0)+\frac{8}{3})\cdot(1+\frac{45}{8\tanh(d)})]\epsilon_m\\
&+o(\epsilon_m)\\
\leq&[\log(2C_0)+(\frac{245}{3}C_0E(C_0)+\frac{8}{3})\cdot(1+\frac{45}{8\tanh(d)})]\epsilon_m+o(\epsilon_m)
\end{aligned}$$

All in all, we can get the total error of computing distance

$$\delta=\delta_Q+\delta_2=\log(Q_{11})\epsilon_m+\frac{1}{2}\epsilon_m+o(\epsilon_m)+\delta_2$$

Because $\log(Q_{11}) \leq d$, then we get that if $Q_{11}z \geq C_0$,

$$\delta \leq d\epsilon_m+(\frac{1825}{432}+\frac{245C_0}{3}+\log 70/3)\epsilon_m+o(C_0^{-2})+o(\epsilon_m)$$

if $Q_{11}z \leq C_0$,

$$\delta \leq d\epsilon_m+[\frac{1}{2}+\log(2C_0)+(\frac{245}{3}C_0E(C_0)+\frac{8}{3})\cdot(1+\frac{45}{8\tanh(d)})]\epsilon_m+o(\epsilon_m)$$

□

Also, in the same way, we give the error for computing gradient:

Theorem 7. *The worst case error for computing gradient of distance in Lorentz model using float is*

$$
\begin{aligned}
|\nabla_x d_l^{fl}(fl(x), fl(y)) - \nabla_x d_l(x,y)| =& \frac{2\cosh(d(x,O))\cosh(d(y,O))}{\sinh(d(x,y))\tanh(d(x,y))}\epsilon_m \nabla_x d \\
&+ \frac{3}{4\tanh^2(d(x,y))}\epsilon_m \nabla_x d + \frac{1}{2}\epsilon_m \nabla_x d + o(\epsilon_m)
\end{aligned}
$$

the error of computing gradient of distance in L*-tiling model using float is*

$$
|\nabla_u d_{lt}^{fl} - \nabla_u d_{lt}|_1 = \begin{cases} [(B_1(C_0) + B_2(C_0)\exp(d))|\nabla_x d|_1 + B_3(C_0)\exp(d)]\epsilon_m + o(\epsilon_m C_0^{-1}), \\ \hfill d \geq C_0 \\ [\frac{B_4(C_0)}{\tanh(d)} + (\frac{B_5(C_0)}{\tanh^2(d)} + \frac{B_6(C_0)}{\tanh(d)} + B_7(C_0))|\nabla_x d|_1]\epsilon_m + o(\epsilon_m), \\ \hfill d \leq C_0 \end{cases}
$$

where d *is the real distance between two points,* $B_i(C_0)$ *are constants only depends on* C_0*,* B_f *is a fixed constant,* d^{fl} *means that inside computation like multiplication are performed with machine error* ϵ_m*.*

Remark: Similar to the worst case error of computing distance, the worst case error of computing gradient in Lorentz model using float is dominated by $d(x,O), d(y,O)$, this will also cause the "NaN" problem when two points are far away from the origin. In L-tiling model, the gradient error only depends on the gradient itself, i.e., also $\tanh$ term is bounded, which controls the error and solves the "NaN" problem.

Proof. We consider the Lorentz model at first, the gradient is

$$
\nabla_x d = \frac{My}{\sqrt{(x^T My)^2 - 1}},
$$

then we have the error to be

$$
\begin{aligned}
&|\text{flc}(\nabla_x d) - \nabla_x d| \\
&=|(1+\epsilon_1)\frac{(1+\epsilon_2)My}{(1+\epsilon_3)\sqrt{(1+\epsilon_4)[(1+\epsilon_5)(x^T My + \delta_z)^2 - 1]}} - \frac{My}{\sqrt{(x^T My)^2 - 1}}| \\
&= \frac{\epsilon_m My}{2\sqrt{(x^T My)^2 - 1}} + \frac{\delta_z (x^T My) My}{((x^T My)^2 - 1)^{3/2}} + \frac{3\epsilon_m (x^T My)^2 My}{4((x^T My)^2 - 1)^{3/2}} \\
&= \frac{1}{2}\epsilon_m \nabla_x d + \frac{3}{4\tanh^2(d(x,y))}\epsilon_m \nabla_x d + \frac{2\cosh(d(x,O))\cosh(d(y,O))}{\sinh(d(x,y))\tanh(d(x,y))}\epsilon_m \nabla_x d + o(\epsilon_m)
\end{aligned}
$$

Here flc means calculating with float arithmetic. As for the gradient error in L-tiling model, note that the gradient is

$$
\nabla_u d_{lt}((U,u),(V,v)) = \frac{\nabla h(u)^T L^{-T} \hat{Q} L^{-1} h(v)}{\sqrt{z^2 - Q_{11}^{-2}}}
$$

where $\nabla h(u) = \left[\frac{u}{\sqrt{1+||u||^2}},\ \ I\right]$. First, consider

$$
\begin{aligned}
&\mathrm{flc}(\sqrt{(z+\delta_1)^2 - Q_{11}^{-2}})\\
=&(1+\epsilon_1)\sqrt{(1+\epsilon_2)((1+\epsilon_3)(z+\frac{245}{3}\epsilon_m)^2 - (1+\epsilon_4)Q_{11}^{-2})}\\
=&(1+\epsilon_1)(1+\epsilon_2/2)\sqrt{((1+\epsilon_3)(z+\frac{245}{3}\epsilon_m)^2 - (1+\epsilon_4)Q_{11}^{-2})} + o(\epsilon_m)\\
=&(1+\frac{2}{3}\epsilon_5)\sqrt{(z+\frac{245}{3}\epsilon_m)^2 - Q_{11}^{-2} + \epsilon_6((z+\frac{245}{3}\epsilon_m)^2 + Q_{11}^{-2}))} + o(\epsilon_m)\\
=&(1+\frac{2}{3}\epsilon_5)[\sqrt{z^2 - Q_{11}^{-2} + \epsilon_6(z^2 + Q_{11}^{-2})} + \frac{245\epsilon_m z}{3\sqrt{z^2 - Q_{11}^{-2} + \epsilon_6(z^2 + Q_{11}^{-2})}}] + o(\epsilon_m)\\
=&\sqrt{z^2 - Q_{11}^{-2} + \epsilon_6(z^2 + Q_{11}^{-2})} + \frac{245\epsilon_m z}{3\sqrt{z^2 - Q_{11}^{-2} + \epsilon_6(z^2 + Q_{11}^{-2})}}\\
&+ \frac{2}{3}\epsilon_5\sqrt{z^2 - Q_{11}^{-2} + \epsilon_6(z^2 + Q_{11}^{-2})} + o(\epsilon_m)\\
=&\sqrt{z^2 - Q_{11}^{-2} + \epsilon_6(z^2 + Q_{11}^{-2})} + \frac{245\epsilon_m z}{3\sqrt{z^2 - Q_{11}^{-2}}} + \frac{2}{3}\epsilon_5\sqrt{z^2 - Q_{11}^{-2}} + o(\epsilon_m)\\
=&\sqrt{z^2 - Q_{11}^{-2}} + \frac{\epsilon_6(z^2 + Q_{11}^{-2})}{2\sqrt{z^2 - Q_{11}^{-2}}} + \frac{245\epsilon_m z}{3\sqrt{z^2 - Q_{11}^{-2}}} + \frac{2}{3}\epsilon_5\sqrt{z^2 - Q_{11}^{-2}} + o(\epsilon_m)
\end{aligned}
$$

In the same way, we divide this into two cases, if $Q_{11}z \geq C_0$, then the error of this term is

$$
\begin{aligned}
\delta_4 =&|\mathbf{fl}(\sqrt{(z+\delta_z)^2 - Q_{11}^{-2}}) - \sqrt{z^2 - Q_{11}^{-2}}|\\
=&|\frac{\epsilon_6(z^2 + Q_{11}^{-2})}{2\sqrt{z^2 - Q_{11}^{-2}}} + \frac{245\epsilon_m z}{3\sqrt{z^2 - Q_{11}^{-2}}} + \frac{2}{3}\epsilon_5\sqrt{z^2 - Q_{11}^{-2}} + o(\epsilon_m)|\\
=&|\frac{\epsilon_6 z}{2}(1 + \frac{3}{2(Q_{11}z)^2}) + \frac{245}{3}\epsilon_m(1 + \frac{1}{2(Q_{11}z)^2}) + \frac{2}{3}\epsilon_5\sqrt{z^2 - Q_{11}^{-2}}\\
&+ O(\epsilon_m(Q_{11}z)^{-4}) + o(\epsilon_m)| \leq E_1\epsilon_m + O(\epsilon_m C_0^{-2}) + o(\epsilon_m)
\end{aligned}
$$

Also, if $Q_{11}z \leq C_0$, then the error of this term is

$$
\begin{aligned}
\delta_4 =&|\mathbf{fl}(\sqrt{(z+\delta_1)^2 - Q_{11}^{-2}}) - \sqrt{z^2 - Q_{11}^{-2}}|\\
=&|\frac{\epsilon_6(z^2 + Q_{11}^{-2})}{2\sqrt{z^2 - Q_{11}^{-2}}} + \frac{245\epsilon_m z}{3\sqrt{z^2 - Q_{11}^{-2}}} + \frac{2}{3}\epsilon_5\sqrt{z^2 - Q_{11}^{-2}} + o(\epsilon_m)|\\
\leq&\frac{E_2}{\tanh(d)}\epsilon_m + E_3\epsilon_m + o(\epsilon_m)
\end{aligned}
$$

For the numerator term $z_p = \nabla h(u)^T L^{-T}\hat{Q}L^{-1}h(v)$, where $\nabla h(u) = \left[\frac{u}{\sqrt{1+||u||^2}},\ \ I\right]$, also because $|z_p|$ is bounded, then we can easily get that $\|\mathrm{fl}(z_{pi}) - z_{pi}\| \leq E_4\epsilon_m + o(\epsilon_m)$. Hence, we have

$$\begin{aligned}\nabla_u d_{lt}((U,u),(V,v))_i &= (1+\epsilon_1)\frac{z_{pi}+E_4\epsilon_m}{\sqrt{z^2-Q_{11}^{-2}}+\delta_4}\\ &= (1+\epsilon_1)[\frac{z_{pi}+E_4\epsilon_m}{\sqrt{z^2-Q_{11}^{-2}}}-\frac{z_{pi}\delta_4}{z^2-Q_{11}^{-2}}]\\ &= \frac{z_{pi}+E_4\epsilon_m}{\sqrt{z^2-Q_{11}^{-2}}}+\frac{z_{pi}\delta_4}{z^2-Q_{11}^{-2}}+\frac{z_{pi}\epsilon_1}{\sqrt{z^2-Q_{11}^{-2}}}\end{aligned}$$

Then, the error of gradient is

$$\begin{aligned}\delta_{gi} =&\frac{E_4\epsilon_m}{\sqrt{z^2-Q_{11}^{-2}}}+\frac{z_{pi}\delta_4}{z^2-Q_{11}^{-2}}+\frac{z_{pi}\epsilon_1}{\sqrt{z^2-Q_{11}^{-2}}}\\ =&\frac{E_4\epsilon_m}{\sqrt{z^2-Q_{11}^{-2}}}+\frac{\delta_4\nabla_u d_{lti}}{\sqrt{z^2-Q_{11}^{-2}}}+\epsilon_1\nabla_u d_{lti}\end{aligned}$$

So using this formula, we can get that if $Q_{11}z \geq C_0$, then the error of this term is

$$\begin{aligned}\delta_{gi} \leq& E_4 Q_{11}\epsilon_m C_0^{-1}+[\delta_4 Q_{11}C_0^{-1}+\epsilon_m]\nabla_u d_{lti}+o(\epsilon_m C_0^{-1})\\ \leq& E_4 Q_{11}\epsilon_m C_0^{-1}+[E_1 Q_{11}C_0^{-1}+1]\epsilon_m\nabla_u d_{lti}+o(\epsilon_m C_0^{-1})\\ \leq& E_4\epsilon_m C_0^{-1}\exp(d)+[E_1 C_0^{-1}\exp(d)+1]\epsilon_m\nabla_u d_{lti}+o(\epsilon_m C_0^{-1})\end{aligned}$$

If $Q_{11}z \leq C_0$, then the error of this term is

$$\begin{aligned}\delta_{gi} \leq&\frac{E_4}{\tanh(d)}\epsilon_m+[\frac{\delta_4}{\tanh(d)}+\epsilon_m]\nabla_u d_{lti}\\ \leq&\frac{E_4}{\tanh(d)}\epsilon_m+[\frac{E_2}{\tanh^2(d)}+\frac{E_3}{\tanh(d)}+1]\epsilon_m\nabla_u d_{lti}+o(\epsilon_m)\end{aligned}$$

All in all, if $Q_{11}z \geq C_0$, then the error of gradient is

$$|\delta_g|_1 \leq 3E_4\epsilon_m C_0^{-1}\exp(d)+[E_1 C_0^{-1}\exp(d)+1]\epsilon_m|\nabla_u d_{lt}|_1+o(\epsilon_m C_0^{-1})$$

if $Q_{11}z \leq C_0$, then

$$|\delta_g|_1 \leq \frac{3E_4}{\tanh(d)}\epsilon_m+[\frac{E_2}{\tanh^2(d)}+\frac{E_3}{\tanh(d)}+1]\epsilon_m|\nabla_u d_{lt}|_1+o(\epsilon_m)$$

□

Error for Computing in H-tiling Model Here we provide the error of computing in H-tiling model.

Theorem 8. *The error of computing distance in H-tiling model using float is*

$$\begin{aligned}
&|d_h^{fl}((j_1,\boldsymbol{k}_1,fl(\boldsymbol{z}_1)),(j_2,\boldsymbol{k}_2,fl(\boldsymbol{z}_2))) - d_h((j_1,\boldsymbol{k}_1,\boldsymbol{z}_1),(j_2,\boldsymbol{k}_2,\boldsymbol{z}_2))| \\
=&[C_3(n)(j_2-j_1)+C_4(n)d]\epsilon_m + (3e^{-d}+\frac{1}{e^d\sinh d})[C_1(n)2^{2j_2-2j_1}(1+e^{d/2})^2+C_2(n)]\epsilon_m
\end{aligned}$$

where d is the real distance between two points, $C_i(n)$ are constants only depends on n, d^{fl} means that inside computation like multiplication are performed with machine error ϵ_m.

Remark: Similar to the worst case error of distance computation in L-tiling model, the worst case error in H-tiling model using float only depends on the distance itself, rather than how far points are from the origin, and hence controls the error and solves the "NaN" problem.

Proof. Firstly, note the distance is

$$d = (2s+j_1-j_2)\log(2)+\log(2^{-2s-j_1+j_2}+X+\sqrt{X^2+2^{1-2s-j_1+j_2}X}),$$

where

$$X=\frac{\|I+2^{-s}z_1-2^{j_2-j_1-s}z_2\|^2}{2z_{1n}z_{2n}},\ \ I=2^{-s}(k_1-2^{j_2-j_1}k_2)$$

Note that $\|I\|\leq 1, \|z_1\|, \|z_2\|\leq\sqrt{n+3}$, then we can be bound X in following way:

$$\begin{aligned}
X=&\frac{\|I+2^{-s}z_1-2^{j_2-j_1-s}z_2\|^2}{2z_{1n}z_{2n}} \\
\leq&\frac{\|I\|^2+2^{-2s}\|z_1\|^2+2^{2j_2-2j_1-2s}\|z_2\|^2}{2z_{1n}z_{2n}} \\
&+\frac{2^{1-2s}\|z_1\|\|I\|+2^{1+2j_2-2j_1-2s}\|z_2\|\|I\|+2^{1-2s+j_2-j_1}\|z_1\|\|z_2\|}{2z_{1n}z_{2n}} \\
\leq&\frac{1+2^{-2s}(n+3)+2^{2j_2-2j_1-2s}(n+3)}{2z_{1n}z_{2n}} \\
&+\frac{2^{1-2s}\sqrt{n+3}+2^{1+2j_2-2j_1-2s}\sqrt{n+3}+2^{1-2s+j_2-j_1}(n+3)}{2} \\
\leq&\frac{1}{2}+2^{-2s-1}(n+3)(1+2^{2j_2-2j_1}+2^{1+j_2-j_1})+2^{-2s}(1+2^{2j_2-2j_1})\sqrt{n+3} \\
\leq&\frac{1}{2}+2^{-2s+1}((n+3)+\sqrt{n+3})\leq 1+2^{-2s+2}(n+2)
\end{aligned}$$

now consider the error for computing I

$$|\hat{I}_i-I_i|=|(1+\epsilon_1)2^{-s}(k_{1i}-2^{j_2-j_1}k_{2i})-I_i|\leq\epsilon_m|I_i|=\delta_{i,1}$$

here denote $X_u=I+2^{-s}z_1-2^{j_2-j_1-s}z_2$, then

$$\begin{aligned}
|\hat{X_{ui}}-X_{ui}|=&|[(I_i+\delta_{i,1}+(1+\epsilon_1)2^{-s}z_{1i})(1+\epsilon)-(1+\epsilon_2)2^{j_2-j_1-s}z_{2i}](1+\epsilon)-X_{ui}| \\
=&|[I_i+2^{-s}z_{1i}+\delta_{i,1}+\epsilon_1 2^{-s}z_{1i}+\epsilon(I_i+2^{-s}z_{1i})-(1+\epsilon_2)2^{j_2-j_1-s}z_{2i}](1+\epsilon)-X_{ui}| \\
\leq&|[I_i+2^{-s}z_{1i}-2^{j_2-j_1-s}z_{2i}+\delta_{i,1}+\epsilon_m(I_i+2^{1-s}z_{1i}+2^{j_2-j_1-s}z_{2i})](1+\epsilon)-X_{ui}| \\
\leq&|\epsilon_m X_{ui}+\delta_{i,1}+\epsilon_m(I_i+2^{1-s}z_{1i}+2^{j_2-j_1-s}z_{2i})| \\
\leq&\epsilon_m(|X_{ui}|+2|I_i|+2^{1-s}|z_{1i}|+2^{j_2-j_1-s}|z_{2i})|)=\delta_{i,2}
\end{aligned}$$

next, make use of float arithmetic to get the norm of X_u, the error becomes

$$\begin{aligned}
&|\|\hat{X}_u\|^2 - \|X_u\|^2| \\
\leq &|(1 + n\epsilon_m)\|X_u + \delta_2\|^2 - \|X_u\|^2| = |n\epsilon_m\|X_u\|^2 + \sum_{i=1}^{n}(\delta_{i,2}^2 + 2X_{ui}\delta_{i,2})| \\
\leq &|n\epsilon_m\|X_u||^2 + \sum_{i=1}^{n} 2X_{ui}\delta_{i,2} + o(\epsilon_m)| \\
= &\sum_{i=1}^{n}(4|I_i| + 2^{2-s}|z_{1i}| + 2^{1+j_2-j_1-s}|z_{2i})|)\epsilon_m|X_{ui}| + (n+2)\epsilon_m||X_u||^2 + o(\epsilon_m) \\
\leq &(4\|z_1\| + 2^{1+j_2-j_1}\|z_2\|)\epsilon_m 2^{-s}\|X_u\| + 4\epsilon_m\|I\|\|X_u\| + (n+2)\epsilon_m||X_u||^2 + o(\epsilon_m) = \delta_3
\end{aligned}$$

Note that $X = \frac{X_u^2}{2z_{1n}z_{2n}}$, then we bound the error of computing X here:

$$\begin{aligned}
|\hat{X} - X| = &|\frac{\|\hat{X}_u\|^2}{2z_{1n}z_{2n}(1+2\epsilon)} - X| = |\frac{\|X_u\|^2 + \delta_3}{2z_{1n}z_{2n}}(1 - 2\epsilon) - X| \leq |\frac{\delta_3}{2z_{1n}z_{2n}} - 2\epsilon X| \\
\leq &2|\epsilon_m X| + |\frac{\delta_3}{2z_{1n}z_{2n}}| \leq 2|\epsilon_m X| + |\frac{\delta_3}{2}| \\
\leq &2|\epsilon_m X| + (2\|z_1\| + 2^{j_2-j_1}\|z_2\|)\epsilon_m 2^{-s}\|X_u\| + 2\epsilon_m\|I\|\|X_u\| \\
&+(n/2+1)\epsilon_m||X_u||^2 + o(\epsilon_m) \\
\leq &2\epsilon_m X + (2 + 2^{j_2-j_1})\sqrt{n+3}\epsilon_m 2^{1-s}\sqrt{2X} + 4\epsilon_m\sqrt{2X} + 4(n+2)\epsilon_m X + o(\epsilon_m) \\
\leq &2\epsilon_m X + 2^{2+j_2-j_1-s}\epsilon_m\sqrt{2(n+3)X} + 4\epsilon_m\sqrt{2X} + 4(n+2)\epsilon_m X + o(\epsilon_m) \\
\leq &2\epsilon_m(1 + 2^{-2s+2}(n+2)) + 2^{3+j_2-j_1-s}\epsilon_m\sqrt{(n+3)(1 + 2^{-2s+2}(n+2))} \\
&\quad + 6\epsilon_m\sqrt{1 + 2^{-2s+2}(n+2)} + 4(n+2)(1 + 2^{-2s+2}(n+2))\epsilon_m = \delta_4
\end{aligned}$$

After having these errors, consider the computation error for the second $\log$ in distance:

$$\begin{aligned}
&\log(2^{-2s-j_1+j_2} + (1+\epsilon)(\hat{X} + \sqrt{1+\epsilon}\sqrt{(1+\epsilon)\hat{X}^2 + 2^{1-2s-j_1+j_2}\hat{X}})) + \log(1+\epsilon) \\
= &\log(2^{-2s-j_1+j_2} + (1+\epsilon)(\hat{X} + \sqrt{X^2 + 2^{1-2s-j_1+j_2}X} + \frac{1}{2}\epsilon\sqrt{X^2 + 2^{1-2s-j_1+j_2}X} \\
&+ \frac{\delta_4(2^{-1-2s-j_1+j_2} + X/2)}{\sqrt{X^2 + 2^{1-2s-j_1+j_2}X}})) + \epsilon \\
= &\log(2^{-2s-j_1+j_2} + X + \delta_4 + \sqrt{X^2 + 2^{1-2s-j_1+j_2}X} + \frac{1}{2}\epsilon\sqrt{X^2 + 2^{1-2s-j_1+j_2}X} \\
&+ \frac{\delta_4(2^{-1-2s-j_1+j_2} + X/2)}{\sqrt{X^2 + 2^{1-2s-j_1+j_2}X}})) + \epsilon(X + \sqrt{X^2 + 2^{1-2s-j_1+j_2}}X)) + \epsilon \\
= &\log(2^{-2s-j_1+j_2} + X + \sqrt{X^2 + 2^{1-2s-j_1+j_2}X}) + \epsilon \\
&+ \frac{\delta_4 + \frac{1}{2}\epsilon\sqrt{X^2 + 2^{1-2s-j_1+j_2}X} + \frac{\delta_4(2^{-1-2s-j_1+j_2} + X/2)}{\sqrt{X^2 + 2^{1-2s-j_1+j_2}X}})) + \epsilon(X + \sqrt{X^2 + 2^{1-2s-j_1+j_2}X})}{2^{-2s-j_1+j_2} + X + \sqrt{X^2 + 2^{1-2s-j_1+j_2}X}}
\end{aligned}$$

All in all, we combine all errors to derive the error for distance computation:

$$\begin{aligned}
\delta_0 =&(2s+j_1-j_2)\log(2)\epsilon+\log(1+\epsilon)\\
&+\frac{\delta_5+\frac{1}{2}\epsilon\sqrt{X^2+2^{1-2s-j_1+j_2}X}+\frac{\delta_5(2^{-1-2s-j_1+j_2}+X/2)}{\sqrt{X^2+2^{1-2s-j_1+j_2}X}}))+\epsilon(X+\sqrt{X^2+2^{1-2s-j_1+j_2}X})}{2^{-2s-j_1+j_2}+X+\sqrt{X^2+2^{1-2s-j_1+j_2}X}}\\
=&(2s+j_1-j_2)\log(2)\epsilon+\epsilon\\
&+\frac{\delta_5+\frac{1}{2}\epsilon\sqrt{X^2+2^{1-2s-j_1+j_2}X}+\frac{\delta_5(2^{-1-2s-j_1+j_2}+X/2)}{\sqrt{X^2+2^{1-2s-j_1+j_2}X}}))+\epsilon(X+\sqrt{X^2+2^{1-2s-j_1+j_2}X})}{2^{-2s-j_1+j_2}+X+\sqrt{X^2+2^{1-2s-j_1+j_2}X}}\\
=&(2s+j_1-j_2)\log(2)\epsilon+\epsilon+\frac{\delta_5(\frac{3}{2}+\frac{2^{-1-2s-j_1+j_2}}{\sqrt{X^2+2^{1-2s-j_1+j_2}X}})+\frac{3}{2}\epsilon\sqrt{X^2+2^{1-2s-j_1+j_2}X}+\epsilon X}{2^{-2s-j_1+j_2}+X+\sqrt{X^2+2^{1-2s-j_1+j_2}X}}\\
=&(2s+j_1-j_2)\log(2)\epsilon+\epsilon+\frac{3}{2}\epsilon+2^{2s+j_1-j_2}\frac{(\frac{3}{2}+\frac{1}{2\sinh d})}{\cosh d+\sinh d}\delta_5\\
=&(2s+j_1-j_2)\log(2)\epsilon+\frac{5}{2}\epsilon+2^{2s+j_1-j_2-1}\frac{(3+\frac{1}{\sinh d})}{\cosh d+\sinh d}[2(1+2^{-2s+2}(n+2))\\
&+2^{3+j_2-j_1-s}\sqrt{(n+3)(1+2^{-2s+2}(n+2))}+6\sqrt{1+2^{-2s+2}(n+2)}\\
&+4(n+2)(1+2^{-2s+2}(n+2))]\epsilon_m
\end{aligned}$$

We simplify the constants with $C_i(n)$, also note that $k_1 - 2^{j_2-j_1}k_2$ is an integer vector, then $s = \lceil \log_2(\|k_1 - 2^{j_2-j_1}k_2\|^2)/2\rceil \geq 0$, also we have $j_2 \geq j_1$, then the error for distance computation is

$$\begin{aligned}
\delta_0 \leq&(3+2s+j_1-j_2)\epsilon+2^{2s+j_1-j_2-1}\frac{(3+\frac{1}{\sinh d})}{\cosh d+\sinh d}[C_1(n)+2^{-2s}C_2(n)+2^{j_2-j_1-2s}C_3(n)]\epsilon_m\\
\leq&(3+2s+j_1-j_2)\epsilon+\frac{(3+\frac{1}{\sinh d})}{\cosh d+\sinh d}[2^{2s+j_1-j_2}C_1(n)+2^{j_1-j_2}C_2(n)+C_3(n)]\epsilon_m\\
\leq&(3+2s)\epsilon+\frac{(3+\frac{1}{\sinh d})}{\cosh d+\sinh d}[2^{2s}C_1(n)+C_2(n)]\epsilon_m\\
=&\lceil\log_2(\|k_1-2^{j_2-j_1}k_2\|^2)\rceil\epsilon_m+\frac{(3+\frac{1}{\sinh d})}{\cosh d+\sinh d}[C_1(n)\|k_1-2^{j_2-j_1}k_2\|^2+C_2(n)]\epsilon_m
\end{aligned}$$

To bound the norm in the distance error, consider

$$\begin{aligned}
\|2^{j_1}k_1-2^{j_2}k_2\| =&\|2^{j_1}z_1-2^{j_2}z_2+2^{j_1}k_1-2^{j_2}k_2+2^{j_2}z_2-2^{j_1}z_1\|\\
\leq&\|2^{j_1}z_1-2^{j_2}z_2+2^{j_1}k_1-2^{j_2}k_2\|+\|2^{j_2}z_2-2^{j_1}z_1\|\\
\leq&2^{\frac{j_1+j_2+1}{2}}\sqrt{(\cosh d-1)z_{1n}z_{2n}}|+2^{j_2}\|z_2\|+2^{j_1}\|z_1\|\\
\leq&2^{\frac{j_1+j_2}{2}+2}\sinh{(d/2)}+(2^{j_2}+2^{j_1})\sqrt{n+3}
\end{aligned}$$

then scale it by 2^{-j_1}, we have

$$\|k_1-2^{j_2-j_1}k_2\| \leq 2^{\frac{j_2-j_1}{2}+2}\sinh{(d/2)}+(2^{j_2-j_1}+1)\sqrt{n+3}$$

Therefore, we bound the distance computation error as follows:

$$\delta_0 \leq [C_3(n)(j_2-j_1)+C_4(n)d]\epsilon_m+(3e^{-d}+\frac{1}{e^d\sinh d})[C_1(n)2^{2j_2-2j_1}(1+e^{d/2})^2+C_2(n)]\epsilon_m$$

$\square$