# A Quadtree, a Steiner Spanner, and Approximate Nearest Neighbours in Hyperbolic Space

**Sándor Kisfaludi-Bak**
Department of Computer Science, Aalto University, Finland

**Geert van Wordragen**
Department of Computer Science, Aalto University, Finland

## Abstract

We propose a data structure in $d$-dimensional hyperbolic space that can be considered a natural counterpart to quadtrees in Euclidean spaces. Based on this data structure we propose a so-called L-order for hyperbolic point sets, which is an extension of the Z-order defined in Euclidean spaces.

Using these quadtrees and the L-order we build geometric spanners. Near-linear size $(1+\varepsilon)$-spanners do not exist in hyperbolic spaces, but we create a Steiner spanner that achieves a spanning ratio of $1+\varepsilon$ with $\mathcal{O}_{d,\varepsilon}(n)$ edges, using a simple construction that can be maintained dynamically. As a corollary we also get a $(2+\varepsilon)$-spanner (in the classical sense) of the same size, where the spanning ratio $2+\varepsilon$ is almost optimal among spanners of subquadratic size.

Finally, we show that our Steiner spanner directly provides an elegant solution to the approximate nearest neighbour problem: given a point set $P$ in $d$-dimensional hyperbolic space we build the data structure in $\mathcal{O}_{d,\varepsilon}(n \log n)$ time, using $\mathcal{O}_{d,\varepsilon}(n)$ space. Then for any query point $q$ we can find a point $p \in P$ that is at most $1+\varepsilon$ times farther from $q$ than its nearest neighbour in $P$ in $\mathcal{O}_{d,\varepsilon}(\log n)$ time. Moreover, the data structure is dynamic and can handle point insertions and deletions with update time $\mathcal{O}_{d,\varepsilon}(\log n)$. This is the first dynamic nearest neighbour data structure in hyperbolic space with proven efficiency guarantees.

**2012 ACM Subject Classification** Theory of computation → Computational geometry; Theory of computation → Approximation algorithms analysis; Theory of computation → Nearest neighbor algorithms

**Keywords and phrases** hyperbolic geometry, Steiner spanner, dynamic approximate nearest neighbours

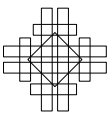
**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2024.68

**Related Version** *Full Version*: `https://arxiv.org/abs/2305.01356` [31]

## 1 Introduction

Hyperbolic spaces have special properties that set them apart from the more familiar Euclidean spaces: they expand exponentially and are tree-like, which makes them the natural choice to represent hierarchical structures. Hyperbolic geometry has applications in several fields, including special relativity, topology, visualisation, machine learning, complex network modelling, etc. [45, 43, 34, 39, 33]. With the growing interest in the larger scientific community, there are growing computational and graphical/visualisation needs. It is becoming increasingly important to develop basic data structures and algorithms for hyperbolic spaces. Despite clear and growing interest in machine learning [39, 25] and the random graph/graph modelling communities [8, 24, 13, 7], the data structures and algorithms in this very natural geometric setting have been largely overlooked. If we wish to process and analyse data in hyperbolic spaces in the future, the basic theory for this processing needs to be established.

Quadtrees in Euclidean spaces [22] are among the simple early geometric data structures that have proven to be useful both in practical algorithms and in theory [1, 19, 27]. They form the basis of various algorithms by being able to "zoom in" efficiently. Quadtrees

provide a hierarchical structure, as well as a way to think of ordering points of the plane (or higher-dimensional spaces) using the so-called Z-order curve [36]. They can be used as a basis for nearest neighbour algorithms [27].

Given a point set $P$ in the Euclidean plane (henceforth denoted by $\mathbb{R}^2$), a quadtree of $P$ can be defined as follows. Let $\sigma_0$ be a minimal axis-parallel square containing $P$, and let $T$ be a tree graph whose root corresponds to $\sigma_0$. Then consider a square $\sigma$ and the corresponding vertex $v_\sigma$ of $T$ where $|\sigma \cap P| \geq 2$ (starting with $\sigma = \sigma_0$). We subdivide $\sigma$ into four squares of half the side length of $\sigma$. Each smaller square is associated with a new vertex that is connected to $v_\sigma$. This procedure is repeated for each square $\sigma$ where $\sigma \cap P \geq 2$ exhaustively, until all leaves of $T$ correspond to squares that contain at most one point from $P$. The squares are called the *cells* of the quadtree, and we can speak of parent/child and ancestor/descendant relationships of cells based on $T$. The *level* of a cell or vertex is its distance to the root of $T$ along the shortest path in $T$ (i.e., the root has level 0).

Some crucial properties of Euclidean quadtrees (slightly relaxed) include the following.

1. *Diameter doubling.* If $C'$ is a child cell of $C$, then $c_1 < \operatorname{diam}(C')/\operatorname{diam}(C) < c_2$ where $0 < c_1 < c_2 < 1$ are fixed constants, and $\operatorname{diam}(C)$ denotes the diameter of the cell $C$.
2. *Fatness.* Each cell $C$ contains a ball that has diameter at least constant times the diameter of $C$. Thus cells are so-called *fat* objects in $\mathbb{R}^2$ [20].
3. *Bounded degree.* Each cell has at most $k$ children cells for some fixed constant $k$.
4. *Same-level isometry.* Cells of the same level are isometric, that is, any cell can be obtained from any other cell of the same level by using a distance-preserving transformation.

Could the above four properties be replicated by a quadtree in the hyperbolic plane? Unfortunately this is not possible: the volume of a ball in hyperbolic space grows exponentially with its radius (thus hyperbolic spaces are not *doubling spaces*). Consequently, for large cells, a cell of constant times smaller diameter than its parent will only cover a vanishingly small volume of its parent. This rules out having properties 1, 2, and 3 together. Property 4 also poses a unique challenge: while the hyperbolic plane provides many types of *tilings* one could start with, there is no transformation that would be equivalent to scaling in Euclidean spaces. This is unlike the scaling-invariance exhibited by Euclidean quadtrees. Moreover, in small neighbourhoods hyperbolic spaces are locally Euclidean, meaning that a small ball in hyperbolic space can be embedded into a Euclidean ball of the same radius with distortion infinitesimally close to 1. Thus a hyperbolic quadtree needs to operate at two different scales: an almost-Euclidean metric at small distances and a non-doubling metric at larger distances.

Quadtrees in Euclidean spaces give rise to *spanners* through so-called well-separated pair decompositions. Spanners are a way of representing approximate distances among the points of a point set $P$ without taking up quadratic space (i.e., without storing all $\binom{n}{2}$ pairwise distances). A spanner is a geometric graph, that is, a graph where vertices correspond to $P$, and edges are straight segments (geodesic segments) between some pairs of points with edge lengths being equal to the distances of the underlying space. A geometric graph $G$ is called a $t$-spanner if for any pair of points $p, q \in P$ their distance in $G$ is at most $t$ times longer than their distance in the underlying space.

Geometric spanners have a vast literature. We encourage the interested reader to look at the book of Narasimhan and Smid [38] for an overview. The simplest $(1+\varepsilon)$-spanner is the greedy spanner, which considers all pairs of points sorted on increasing distance and adds an edge between a pair when their current distance in the graph is too large. This gives a spanner that is optimal in many aspects, but constructing it takes $\mathcal{O}\big(n^2 \log n\big)$ time [12]. For constant dimension $d$, a $\Theta$-graph can be constructed in $\mathcal{O}(n \log n)$ time and is a $(1+\varepsilon)$-spanner with $\mathcal{O}\big(n/\varepsilon^{d-1}\big)$ edges. As shown by Le and Solomon [35], this edge count is optimal. Well-separated pair decompositions [14] in Euclidean spaces are built upon quadtrees and naturally give rise to $(1+\varepsilon)$-spanners that have linearly many edges.

Unfortunately well-separated pair decompositions with the required properties do not exist in hyperbolic spaces (due to Lemma 8). In fact, spanners that are not complete graphs cannot achieve a spanning ratio less than 2 in any hyperbolic space. Luckily a spanning ratio of $1+\varepsilon$ can be achieved if one does not insist on a geometric graph whose vertices are exactly $P$, but allows more vertices. Such spanners are called Steiner spanners, i.e., these are geometric graphs on a set $P \cup Q$ that satisfy the spanner property for pairs of points from $P$. The points of $Q$ are referred to as *Steiner points*. In Euclidean space, these Steiner points allow spanners to be sparser: Steiner spanners can be constructed that only use $\mathcal{O}\left(n/\varepsilon^{(d-1)/2} \cdot \log^2 \frac{1}{\varepsilon}\right)$ edges [35] and there is a lower bound of $\Omega\left(n/\varepsilon^{(d-1)/2}\right)$ edges [6].

Quadtrees are also applicable as a basis of *nearest neighbour search*: for a fixed point set $P$ and a query point $q$, can we find the point $p \in P$ that is closest to $q$ among all points in $P$? Nearest neighbour search is a well-studied and fundamental problem with numerous applications in machine learning, data analysis, and classification. Exact solutions to queries are typically only feasible in very low dimensional spaces; indeed the Euclidean methods carry over to the hyperbolic setting [40, 10]. However, in dimensions 3 and above, even in the Euclidean setting, it is much more feasible to compute *approximate nearest neighbours*, i.e., to find a point $p$ that is at most $1+\varepsilon$ times farther from $q$ than the nearest neighbour of $q$. Arya et al. [3] give a data structure for this that is constructed in $\mathcal{O}(dn \log n)$ time, requires $\mathcal{O}(dn)$ space and allows for queries in $\mathcal{O}\left(\lceil 1 + 6d/\varepsilon \rceil^d \log n\right)$ time. One can increase this space requirement to decrease the query time [2]. Well-separated pair decompositions [14] and locality-sensitive orderings [17] can also be used here, with similar guarantees in query time but a worse dependence on $\varepsilon$ and $d$ in preprocessing time and space requirements. For higher dimensions, the exponential dependence on $d$ becomes a problem, so there one can sacrifice the optimal dependence on $n$ [28]. The main question we wish to answer is as follows.

> *Is there a data structure for approximate nearest neighbour search in hyperbolic space with similar guarantees as in Euclidean space?*

**Our contribution.** We propose a hyperbolic quadtree that satisfies properties 1, 2, as well as property 4 in case of cells of super-constant diameter. Moreover, our hyperbolic quadtree resembles a Euclidean quadtree for cells of sub-constant diameter. Based on the quadtree we are able to construct a new order and space-filling curve, named the L-order, which serves as a hyperbolic extension of the Euclidean Z-order. We show that a few hyperbolic quadtrees (and corresponding L-orders) can create a useful cover of $\mathbb{H}^d$ in the following sense.

▶ **Theorem 1.** *For any $\Delta \in \mathbb{R}_+$, there is a set of at most $3d+3$ infinite hyperbolic quadtrees such that any two points $p, q \in \mathbb{H}^d$ with $\operatorname{dist}_{\mathbb{H}^d}(p,q) \leq \Delta$ are contained in a cell with diameter $\mathcal{O}\left(d\sqrt{d}\right) \cdot \operatorname{dist}_{\mathbb{H}^d}(p,q)$ in one of the quadtrees.*

Krauthgamer and Lee [32] achieve a similar decomposition in a more general setting (on visual geodesic Gromov-hyperbolic spaces), but their construction is more implicit as it is based on a decomposition of the so-called Gromov boundary. For example, it does not immediately lend itself to an easily computable L-order. Theorem 1 matches the Euclidean result given by Chan, Har-Peled and Jones [17, Lemma 3.7]. Note however that one cannot create their locality sensitive orderings in hyperbolic spaces due to Lemma 8. We take a different route: we construct a Steiner spanner using the quadtrees of Theorem 1.

▶ **Theorem 2.** *Let $P \in \mathbb{H}^d$ be a given set of $n$ points and $\varepsilon \in (0, 1/2]$. We can construct a Steiner $(1+\varepsilon)$-spanner for $P$ with Steiner vertex set $Q$ that has $n \cdot d^{\mathcal{O}(d)} \log(1/\varepsilon)/\varepsilon^d$ edges. The constructed spanner is bipartite with parts $P$ and $Q$, where each $p \in P$ has degree at most*

$d^{\mathcal{O}(d)} \log(1/\varepsilon)/\varepsilon^d$. *Furthermore, a point can be inserted or deleted in* $\log n \cdot d^{\mathcal{O}(d)} \log(1/\varepsilon)/\varepsilon^d$ *time, where each insertion or deletion creates or removes at most* $d^{\mathcal{O}(d)} \log(1/\varepsilon)/\varepsilon^d$ *Steiner points and edges.*

Apart from the Steiner points, this again matches the results from locality sensitive orderings [17], which were until recently the best results for dynamic Euclidean spanners. Very recently Gao and Har-Peled [26] developed a variant of locality sensitive orderings that gives a spanner with $\mathcal{O}_d\big(\frac{n}{\varepsilon^{d-1}} \log \frac{1}{\varepsilon}\big)$ edges and allows updates in $\mathcal{O}_d\Big(\frac{\log n}{\varepsilon^{d-1}} \log^2 \frac{1}{\varepsilon}\Big)$ time. With Steiner points, the current best Euclidean result has $\mathcal{O}_d\big(\frac{n}{\varepsilon^{(d-1)/2}} \log^2 \frac{1}{\varepsilon}\big)$ edges [35] but cannot be maintained dynamically. In the hyperbolic setting, Krauthgamer and Lee [32] gave a static Steiner spanner with comparable properties, but their construction is not published.

The most important step in constructing a spanner based on a cover such as the one in Theorem 1 is to consider connections for pairs of points in a fixed quadtree cell $C$ whose distance is of the same magnitude as the cell diameter $\operatorname{diam}(C)$. This is typically done by subdividing this cell into cells that are of diameter $\varepsilon\operatorname{diam}(C)$, and choosing hub points in each of these subcells. In our setting however the number of these subcells is unbounded, so we need a different technique. The crucial insight in our Steiner spanner construction is that the geodesics connecting pairs of points in $C$ of distance $\Omega(\operatorname{diam}(C))$ all go through a small region of $C$. By placing a Steiner point in this region and connecting it to the relevant points, we can approximate a large number of pairwise distances efficiently.

Our spanner has a nice bipartite structure, and with a simple trick it can be made into a $2+\varepsilon$ spanner with the same properties (that is, a spanner without Steiner points.) Moreover, we can use this spanner to answer dynamic approximate nearest neighbour queries.

▶ **Theorem 3.** *We can construct a data structure that uses* $n \cdot d^{\mathcal{O}(d)} \log(1/\varepsilon)/\varepsilon^d$ *space and can answer queries for a* $(1+\varepsilon)$*-approximate nearest neighbour in* $\log n \cdot d^{\mathcal{O}(d)} \log(1/\varepsilon)/\varepsilon^d$ *time, and perform updates (point insertions and removals) in* $\log n \cdot d^{\mathcal{O}(d)} \log(1/\varepsilon)/\varepsilon^d$ *time.*

To our knowledge, this is the first data structure for *dynamic* approximate nearest neighbour search in hyperbolic space with a rigorous analysis. Krauthgamer and Lee [32] gave a static data structure[1] that has comparable query time but exponential storage. They also have variant with $\mathcal{O}\big(n^2\big)$ storage and $\mathcal{O}\big(\log^2 n\big)$ query time (each for constant $d$), but this latter construction is not published. Our construction is comparatively simple, more efficient, and dynamic. Once again our result matches the original Euclidean result from locality sensitive orderings [17]. However, some Euclidean data structures designed specifically for approximate nearest neighbours only require $\mathcal{O}(dn)$ space and allow updates in $\mathcal{O}(d \log n)$ time [3]. Getting similar results in the hyperbolic setting remains an open problem.

**Further related work.** In addition to Krauthgamer and Lee [32], approximate nearest neighbour search in hyperbolic spaces has been studied by Wu and Charikar [47] and by Prokhorenkova et al. [42]. Wu and Charikar [47] describe various methods of using any algorithm for Euclidean nearest neighbour search to find exact and approximate nearest neighbours in hyperbolic space. Their algorithms perform well in practice, but logically cannot outperform their Euclidean counterparts and they perform worse the further from Euclidean the data set is. More concretely, they give exact and $(1+\varepsilon)$-approximate nearest neighbours

[1] The data structure of Krauthgamer and Lee [32] achieves constant additive error, but they note that it can be extended to our current setting of a multiplicative $(1+\varepsilon)$-approximation.

using queries to an exact algorithm for Euclidean nearest neighbours, and $\sqrt{w}(1+\varepsilon)$-approximate nearest neighbours using queries to an algorithm for $(1+\varepsilon)$-approximate Euclidean nearest neighbours, where $w > 1$ is a parameter that influences the query time.

In general metric spaces, graph-based nearest neighbour search can be used. These are based on a *k-nearest neighbour graph*, a geometric graph where each point is connected to its $k$ nearest neighbours. To answer a query, algorithms will start from some vertex in the graph and travel along edges that decrease the distance to the query point, until reaching a local minimum. Prokhorenkova et al. [42] show that for hyperbolic space these algorithms perform well in practice and give theoretical guarantees under some assumptions (for example, the points are uniformly distributed in a ball of radius $R$). Their data structure has size $\mathcal{O}\big(n \log n \cdot M^d\big)$ and when $R \ll 1/\sqrt{d}$ allows queries in $\mathcal{O}\big(n^{1/d} \cdot M^d\big)$ time, where $M$ is a constant related to the approximation factor. When $R \gg \log d$, the query time becomes $\mathcal{O}\big(n^{1/d} \cdot M^d R / e^{R(d-1)/d}\big)$: queries become faster when points are further apart.

Some decompositions similar to our quadtree have been considered in the context of hyperbolic random graphs. Von Looz, Meyerhenke and Prutkin [46] introduced a *polar quadtree*. For this, they represent points in the hyperbolic plane by polar coordinates: the angle $\phi$ and distance $r$ w.r.t. a fixed direction and origin. Each cell of the quadtree is then of the form $[\phi_{\min}, \phi_{\max}) \times [r_{\min}, r_{\max})$. As in a Euclidean quadtree, a cell can be split into four children. Cells of the same level can be set to have the same area, but their diameters will vary significantly, which also means they are not isometric and can get arbitrarily thin.

Various papers about hyperbolic random graphs [9, 23, 37] use a discretisation similar to the binary tiling (introduced in the next section and the basis of our quadtree), but based on the polar coordinate system instead of the half-space model. Eppstein [21] uses Euclidean quadtrees for approximating the hyperbolic plane at small scales, another key part of our quadtree. Contemporaneously with this paper's publication, Park and Vigneron [41] showed that the problems we consider can also be solved efficiently with constant *additive* error.

## 2 Preliminaries

The reader should be able to grasp the main ideas and the structure of our constructions even if they are new to hyperbolic geometry; we suggest thoroughly understanding binary tilings (see below), the basic shapes (shortest paths, hyperplanes), and probing the distance formula to get some basic intuition. A more thorough reader will need some familiarity with the basics of hyperbolic geometry and trigonometry, as well as the Poincaré half-space model. For more background on hyperbolic geometry, please see [15] and the textbooks [29, 44, 4]. Note that our results are presented in the half-space model, but are in fact model-independent. Our algorithms are also presented for point sets whose coordinates are given in the half-space model. Apart from numerical challenges – something we will not tackle in this article –, working in the half-space model is not restrictive as conversion between various models of hyperbolic geometry is straightforward.

Let $\mathbb{H}^d$ denote the $d$-dimensional hyperbolic space of sectional curvature $-1$. We denote points as $(x, z)$ for $x \in \mathbb{R}^{d-1}$ and $z \in \mathbb{R}^+$, with the distance

$$\operatorname{dist}_{\mathbb{H}^d}((x,z),(x',z')) = 2 \operatorname{arsinh}\left(\frac{1}{2}\sqrt{\frac{\|x-x'\|^2 + (z-z')^2}{zz'}}\right),$$

where $\|x\|$ refers to the Euclidean norm of $x$. When $x = x'$ this reduces to $\left|\ln\left(\frac{z}{z'}\right)\right|$.

For the rest of the paper, we fix a particular half-space model. We will think of the $z$ direction as going "up", and the $d-1$ other axes are going "sideways".

The transformations $T_{\sigma,\tau}(x,z) = \sigma \cdot (x+\tau, z)$, where we translate $x$ with a vector $\tau \in \mathbb{R}^{d-1}$ and then scale all coordinates by $\sigma \in \mathbb{R}^+$, are isometries of $\mathbb{H}^d$. This can be verified by applying $T_{\sigma,\tau}$ to both arguments in the distance formula. Consider now all the transformations $T_{\sigma,\tau}$ where $\sigma = 2^k$ for some integer $k$ and $\tau$ is an integer vector. One can observe that acting upon the Euclidean unit cube with corners $(0,\ldots,0,1)$ and $(1,\ldots,1,2)$ these transformations together create a tiling of the half-space model with isometric tiles. This tiling has been named the *binary tiling*, and it was introduced by Böröczky [11], see also [15]. The binary tiling is the basis of our quadtree construction. The 2-dimensional binary tiling is illustrated in Figure 1.

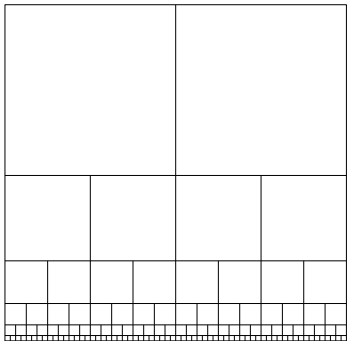


**Figure 1** A portion of the binary tiling in the half-plane model.

In the half-space model, the shortest path (*geodesic*) between two points only matches the Euclidean line segment between them when this segment is vertical. Otherwise, the geodesic is an arc of a Euclidean circle that has its centre at $z = 0$. For hyperplanes the situation is similar: vertical Euclidean hyperplanes are hyperbolic hyperplanes, but the other hyperbolic hyperplanes are Euclidean spheres with their centre somewhere on the plane $z = 0$. This paper will often also use horizontal Euclidean hyperplanes, so it is worth noting that these are *not* hyperplanes in the hyperbolic sense (for example, they can be intersected twice by the same geodesic), but a different shape called a horosphere. Finally, the half-space model is conformal: Euclidean and hyperbolic angles are the same. Additionally it means that any Euclidean sphere that does not intersect $z = 0$ is also a hyperbolic sphere, but its hyperbolic centre will lie below its Euclidean centre.

## 3 A hyperbolic quadtree

The Euclidean quadtree is a tree whose vertices are associated with axis-parallel hypercubes. Correspondingly, our hyperbolic quadtree will be a tree whose vertices are associated with so-called *cube-based horoboxes*. For this paper, it will be useful to define an *axis-parallel horobox* as the shape that corresponds to a Euclidean axis-parallel box. Such a horobox $B$ can be defined by its corner points $(x_{\min}(B), z^{\downarrow}(B))$ and $(x_{\max}(B), z^{\uparrow}(B))$, where one is minimal and the other maximal in all coordinates. Notice that a horobox is bounded by 2 horospheres and $2(d-1)$ hyperplanes. The cube-based horobox is defined as follows.

▶ **Definition 4.** *In a fixed half-space model, a* cube-based horobox $C$ *is an axis-parallel horobox with* $\frac{z^{\uparrow}(C)}{z^{\downarrow}(C)} = 2^h$ *and* $\frac{x_{\max}(C) - x_{\min}(C)}{z^{\downarrow}(C)} = (w,\ldots,w)$*, where* $w = w(C)$ *is called the* width *of* $C$ *and* $h = h(C)$ *is called the* height *of* $C$*.*

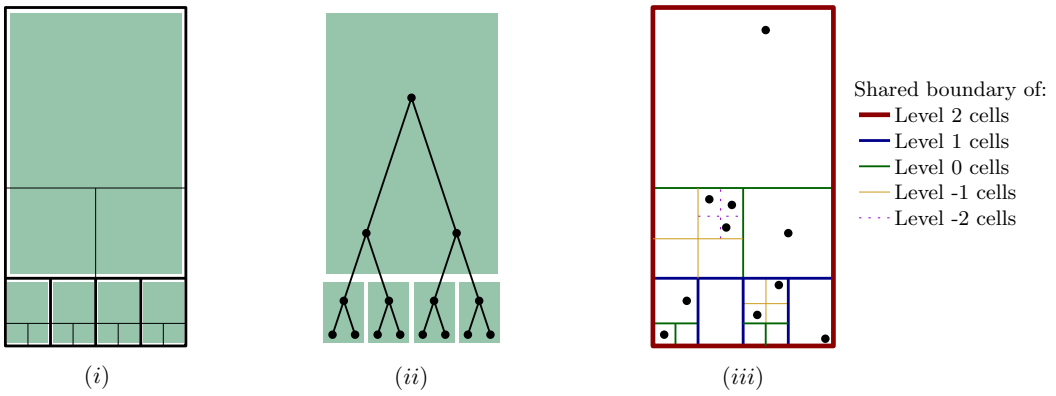



**Figure 2** (i) A quadtree cell of level 2 is split into 5 cells of level 1. (ii) The binary tree of depth 4 is split into 5 isomorphic binary trees of depth 2. (iii) A hyperbolic quadtree where the root (red) is a level 2 cell, which is split into 5 isometric cells of level 1 that are separated by blue Euclidean segments.

It is worth noting that for $z^{\downarrow}(C) = 1$, the width corresponds to the Euclidean width of $C$. On top of that, defining the width and height in this way ensures that two cube-based horoboxes $C', C$ with the same width and height are congruent to one another: setting $\sigma = \frac{z^{\downarrow}(C')}{z^{\downarrow}(C)}$ and $\tau = x_{\min}(C')/\sigma - x_{\min}(C)$ gives $T_{\sigma,\tau}(C) = C'$.

**Hyperbolic quadtree construction.** One property of hyperbolic space that makes quadtrees more complicated is that it behaves differently at different scales: in small enough neighbourhoods the distortion compared to Euclidean space becomes negligible, but at larger scales the hyperbolic nature becomes more and more pronounced. This means that the quadtree also has to work differently at different scales.

For a point set $P \subset \mathbb{H}^d$, the hyperbolic quadtree $\mathcal{Q}(P)$ is a graph whose vertices are regions of hyperbolic space. We can construct $\mathcal{Q}(P)$ as follows. First, we find the Euclidean minimum bounding box of $P$ in the half-space model. From this we can get a minimum bounding cube-based horobox $C_{\text{bound}}$ where $z^{\downarrow}(C_{\text{bound}}) = \min_{p \in P} z(p)$ (i.e., we shift the horobox up as much as possible).

In case of $d = 2$, our goal is to ensure that quadtree cells of level $\ell \geq 0$ correspond to horoboxes whose vertices come from a fixed binary tiling. Note that unlike the Euclidean setting where the quadtree levels are usually defined based on a point set, our quadtree has its levels defined based on the absolute size of the cells. Higher levels will correspond to larger cells, and level 0 will act as the transition between the hyperbolic and Euclidean ways of splitting cells. The levels can be constructed starting at level $\ell = 0$, where cells are the tiles of a binary tiling. The binary tiling is closely related to binary trees: we get a binary tree from the tiling by making a graph where each vertex corresponds to a horobox and edges correspond to them being vertically adjacent. When we have a binary tree, we can partition it into a small number of identical subgraphs by "cutting" at half its depth, see Figure 2(ii). This gives a natural way to split cells of level $\ell \geq 1$ into $1 + 2^{\ell}$ isomorphic cells of level $\ell - 1$, one corresponding to the "top" part of the binary tree, and the rest corresponding to the subtrees defined by the vertices at depth $\ell$. When splitting cells of level $\ell \leq 0$, we are already in a setting where the distortion is very small compared to the Euclidean setting; here we simply use Euclidean dissection into four smaller cells, each with the same Euclidean width and hyperbolic height. Figure 2(iii) shows an example of the resulting hyperbolic quadtree.

We can also define the quadtree for general $d$. First, we define its root cell $C_{\text{root}}$ with $x_{\min}(C_{\text{root}}) = x_{\min}(C_{\text{bound}})$ and $z^{\downarrow}(C_{\text{root}}) = z^{\downarrow}(C_{\text{bound}})$, and furthermore

- If[2] $w(C_{\text{bound}}) \leq \frac{1}{\sqrt{d-1}}$ and $h(C_{\text{bound}}) \leq 1$, we find the smallest integer $\ell$ such that $w(C_{\text{bound}}) \leq \frac{2^\ell}{\sqrt{d-1}}$ and $h(C_{\text{bound}}) \leq 2^\ell$, then let $w(C_{\text{root}}) = \frac{2^\ell}{\sqrt{d-1}}$ and $h(C_{\text{root}}) = 2^\ell$.
- Otherwise, we find the smallest integer $\ell$ such that $w(C_{\text{bound}}) \leq \frac{2^{2^\ell-1}}{\sqrt{d-1}}$ and $h(C_{\text{bound}}) \leq 2^\ell$, then let $w(C_{\text{root}}) = \frac{2^{2^\ell-1}}{\sqrt{d-1}}$ and $h(C_{\text{root}}) = 2^\ell$.

We then subdivide cells to get their children, but unlike with the Euclidean quadtree this subdivision depends on the size of the cell. If we have a cell $C$ with $h(C) \leq 1$, then we split it into $2^d$ smaller ones using the axis-parallel Euclidean hyperplanes through $\left(\frac{x_{\min}(C)+x_{\max}(C)}{2}, \sqrt{z^{\downarrow}(C)z^{\uparrow}(C)}\right)$. For larger cells, we also use the Euclidean hyperplane $z = \sqrt{z^{\downarrow}(C)z^{\uparrow}(C)}$. This gives two horoboxes with height $h(C)/2$, where the top one has width $w(C)/2^{\frac{h(C)}{2}}$ but the bottom one still $w(C)$. Thus, we also split the bottom horobox into a grid of $2^{\frac{h(C)}{2}(d-1)}$ horoboxes of width $w(C)/2^{\frac{h(C)}{2}}$ so that in total we have $2^{\frac{h(C)}{2}(d-1)}+1$ cells of the same size.

▶ **Lemma 5.** *At any level $\ell$, cells are cube-based horoboxes with height $2^\ell$. For $\ell \geq 0$, the width is $\frac{2^{2^\ell-1}}{\sqrt{d-1}}$ and the diameter is $2\operatorname{arsinh}(2^{2^\ell-2})$. For $\ell < 0$, the width is $\frac{\alpha \cdot 2^\ell}{\sqrt{d-1}}$ and the diameter is $2\operatorname{arsinh}\left(\frac{1}{2}\sqrt{\frac{\alpha^2 \cdot 4^\ell + (2^{2^\ell}-1)^2}{2^{2^\ell}}}\right)$, where $\alpha \in \left(\frac{1}{2}, 1\right]$ is a cell-specific value. Moreover, if a cell $C$ of level $\ell$ has corresponding value $\alpha$ and a child cell $C'$ of $C$ has corresponding value $\alpha'$, then $\alpha'/\alpha \in \{1, 2^{-2^{\ell-1}}\}$.*

The final claim in this lemma shows that sibling cells will become extremely close to being isometric, as $2^{-2^{\ell-1}}$ rapidly converges to 1 as $\ell$ goes to $-\infty$. The proofs for this lemma and Theorem 6 are straightforward but calculation-heavy and can be found in the full version [31].

▶ **Theorem 6.** *The hyperbolic quadtree of $P \subset \mathbb{H}^d$ has the following properties:*

- **(i)** *If $C'$ is a child cell of $C$, then $0.42 < \operatorname{diam}(C')/\operatorname{diam}(C) < 0.561$.*
- **(ii)** *If $C$ is a cell at level $\ell$, then $\operatorname{diam}(C) = \Theta\left(2^\ell\right)$.*
- **(iii)** *Cells are $\Omega(1/\sqrt{d})$-fat.*
- **(iv)** *A quadtree cell $C$ has $\max(2^d, 2^{\mathcal{O}(d \cdot \operatorname{diam}(C))})$ children; in particular, the root has $\max(2^d, d^{\mathcal{O}(d \cdot \operatorname{diam}(P))})$ children.*
- **(v)** *Cells of the same level $\ell \geq 0$ are isometric, and cells of level $\ell < 0$ are cube-based horoboxes with the same height whose width differs by less than a factor two.*

**Covering with hyperbolic quadtrees.** Euclidean quadtrees are useful in computing nearest neighbours and other related problems because of a particular distance property: there is a small collection of quadtrees one can define such that any pair of points at distance $\delta$ will be contained in a cell of diameter $\mathcal{O}(\delta)$ in one of the quadtrees. Moreover, the quadtrees can be generated by simply taking one quadtree and translating (*shifting*) it with different vectors. We will prove an analogous property for our hyperbolic quadtrees. For this, we consider the mappings $\tilde{\pi}_z(p) = \log z(p)$ and $\tilde{\pi}_x(p) = x(p)\sqrt{d-1}$. Under both of these, hyperbolic quadtree cells appear as Euclidean quadtree cells and the transformations $T_{\sigma,\tau}$ can be used as translations. Thus, we can combine shifting results from Chan et al. [17] in one dimension and Chan [16] in $d-1$ dimensions to get Theorem 1; this is shown in the full version [31].

[2] Notice that the $1/\sqrt{d-1}$ terms in this definition simplify the diameter formula of quadtree cells, making them independent of $d$: see Lemma 5.

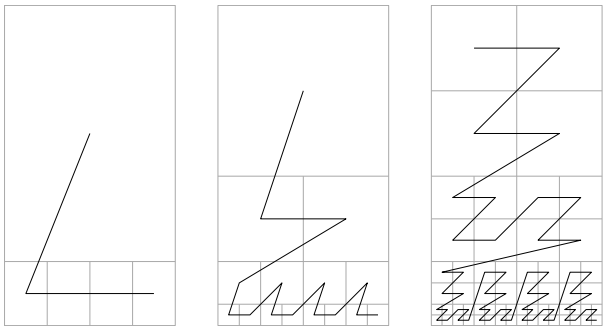


**Figure 3** The curve defined by the L-order, with increasing detail.

Theorem 1 is defined for *infinite quadtrees*: the cells of the infinite quadtree $\mathcal{Q}_\infty^d$ at level 0 correspond to a fixed binary tiling and for each integer level we get cells by subdividing the cells one level higher or taking the union of the cells one level lower, in the same way as defined earlier for a quadtree $\mathcal{Q}(P)$. A full definition is also given in the full version [31].

▶ **Theorem 1.** *For any $\Delta \in \mathbb{R}_+$, there is a set of at most $3d+3$ infinite hyperbolic quadtrees such that any two points $p, q \in \mathbb{H}^d$ with $\operatorname{dist}_{\mathbb{H}^d}(p,q) \leq \Delta$ are contained in a cell with diameter $\mathcal{O}\left(d\sqrt{d}\right) \cdot \operatorname{dist}_{\mathbb{H}^d}(p,q)$ in one of the quadtrees.*

**L-order.** When we have the Euclidean quadtree for a set of points, we can do a depth-first traversal of the tree and note in which order the points are visited. This gives rise to the Z-order. As it turns out, adding or removing points does not change the Z-order and for a pair of points we can determine which comes first without ever constructing a quadtree. The only thing to specify is which infinite quadtree their quadtree would be a subset of, because a differently shifted quadtree can give different results.

We can do the same to get the L-order from a hyperbolic quadtree. Here, we first need to define how exactly we do the depth-first traversal. For levels $\ell > 0$, we first visit the top child and then visit the bottom children in Z-order. For lower levels, the split is the same as for Euclidean quadtrees so we visit the children in the same order as the Z-order. Figure 3 shows the resulting curve for $d = 2$.

▶ **Lemma 7.** *For two points $p, p' \in \mathbb{H}^d$, we can check which comes first in the L-order for $\mathcal{Q}_\infty^d$ by using $\mathcal{O}(d)$ floor, logarithm, bitwise logical and standard arithmetic operations.*

## 4 Steiner spanners and $(1+\varepsilon)$-approximate nearest neighbours

Using Theorem 1 and Lemma 7 it is already possible to find constant-approximate nearest neighbours with the same methods as in Euclidean space (see the full version [31]). On the other hand, we will see that better approximations require new methods.

**Lower bound on hyperbolic spanners.** In constant-dimensional Euclidean space we can get $t$-spanners with $o(n^2)$ edges for any constant $t > 1$. However, this is already impossible in $\mathbb{H}^2$, because (as in high-dimensional Euclidean space) we can construct arbitrarily large sets of points where the distance between any pair of points is approximately the same.

▶ **Lemma 8.** *For any $n \geq 2$ and any $\varepsilon \in (0,1]$, the point set $P(n,\varepsilon) \in \mathbb{H}^2$ of $n$ points equally spaced around a circle of radius $r = \frac{1}{\varepsilon} \ln n$ has the property that any distance between a pair of points is in $(2r(1-\varepsilon), 2r]$.*

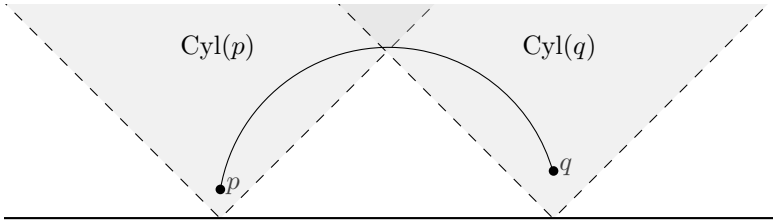



**Figure 4** The geodesic through $p$ and $q$ intersects $\mathrm{Cyl}(p) \cap \mathrm{Cyl}(q)$.

These point sets show that it is often not possible to approximate distances with the same techniques as in constant-dimensional Euclidean space. Low-stretch spanners, and generally any technique that can induce such spanners, such as well-separated pair decompositions and locality-sensitive orderings will fail; see the full version [31] for a proof of this and Lemma 8.

We will get around this lower bound by using *Steiner points* and also match it by giving sparse $t$-spanners for any $t > 2$ (leaving the case $t = 2$ open).

**Steiner spanners.** If we add the centre of the circle to the point set $P(n, \varepsilon)$, suddenly we can make sparse spanners. These are *Steiner spanners* for the original point set: a Steiner $t$-spanner for a point set $P$ is a geometric graph on $P \cup S$ that is a $t$-spanner for the points of $P$, where $S$ are the *Steiner points*. Theorem 2 gives a result for Steiner spanners similar to that given by locality-sensitive orderings for spanners in Euclidean space.

▶ **Theorem 2.** *Let $P \in \mathbb{H}^d$ be a given set of $n$ points and $\varepsilon \in (0, 1/2]$. We can construct a Steiner $(1+\varepsilon)$-spanner for $P$ with Steiner vertex set $Q$ that has $n \cdot d^{\mathcal{O}(d)} \log(1/\varepsilon)/\varepsilon^d$ edges. The constructed spanner is bipartite with parts $P$ and $Q$, where each $p \in P$ has degree at most $d^{\mathcal{O}(d)} \log(1/\varepsilon)/\varepsilon^d$. Furthermore, a point can be inserted or deleted in $\log n \cdot d^{\mathcal{O}(d)} \log(1/\varepsilon)/\varepsilon^d$ time, where each insertion or deletion creates or removes at most $d^{\mathcal{O}(d)} \log(1/\varepsilon)/\varepsilon^d$ Steiner points and edges.*

Because this Steiner spanner has a nice (bipartite) structure, we can also use it to immediately get a result for normal spanners.

▶ **Corollary 9.** *We can construct in $n \log n \cdot d^{\mathcal{O}(d)} \log(1/\varepsilon)/\varepsilon^d$ time a $(2+\varepsilon)$-spanner that has at most $n \cdot d^{\mathcal{O}(d)} \log(1/\varepsilon)/\varepsilon^d$ edges.*

**Proof.** Start by constructing a Steiner $(1+\varepsilon/2)$-spanner according to Theorem 2. To get a normal spanner, we have to remove the Steiner points. For each Steiner point $s$, find the closest point $q_s$ that is connected to it. Now, connect each point that was connected to $s$ to $q_s$ instead. This increases the graph distance between any two points connected to $s$ by at most $2\,\mathrm{dist}_{\mathbb{H}^d}(s, q_s)$, which means their distance at most doubles. Thus, overall the graph distances at most double as well, giving a $(2+\varepsilon)$-spanner. ◀

The remainder of this section will focus on proving Theorem 2. First, we need two properties that significantly limit where the geodesics through a given point can go.

A *cylinder* of radius $r$ around a given geodesic $\ell$ is the set of points with distance at most $r$ to $\ell$. We will use $\mathrm{Cyl}(p)$ to denote the (infinite) cylinder of radius $\mathrm{arsinh}(1)$ around the vertical line through $p$, i.e. $\mathrm{Cyl}(x, z) = \{(x', z') \in \mathbb{H}^d \mid \|x - x'\| \leq z'\}$. This cylinder appears as a Euclidean cone with its apex at $(x, 0)$ and aperture $\frac{\pi}{2}$.

▶ **Lemma 10.** *For any $p, q \in \mathbb{H}^d$, the geodesic between them will go through $\mathrm{Cyl}(p) \cap \mathrm{Cyl}(q)$.*

**Proof.** See Figure 4. The geodesic between $p$ and $q$ is an arc from some Euclidean circle normal to the hyperplane $z = 0$. Using $T_{\sigma,\tau}$ we can always isometrically transform this circle to a unit circle centred at the origin, so without loss of generality we will only look at that case. Now, let let $t$ be the highest point on the geodesic. Notice that $\|x(t)\| \leq 1$. Then,

$$\|x(p) - x(t)\|^2 < (1 - \|x(t)\|)^2 \leq 1 - \|x(t)\|^2 = z(t)^2.$$

Therefore $t \in \mathrm{Cyl}(p)$ and we can similarly argue that $t \in \mathrm{Cyl}(q)$. ◀

The cells of a hyperbolic quadtree are not convex, unlike their Euclidean counterparts. However, we show in the full version [31] that they are still *star-shaped*: a cell $C$ has a non-empty subset $K \subseteq C$ (its *kernel*), such that any geodesic between a point in $K$ and a point in $C$ is fully contained in $C$.

▶ **Lemma 11.** *For any $p, q \in \mathbb{H}^d$ and any hyperbolic quadtree cell $C$ that contains both, if $p$ lies below $z = z^{\downarrow}(C) \cdot 3^{h(C)/2}$ then the geodesic between $p$ and $q$ is completely contained in $C$.*

**Spanner construction.** For each of the $3d+3$ infinite hyperbolic quadtrees from Theorem 1, we sort $P$ based on the corresponding L-order. Then, for each pair of points $p, q$ adjacent in an L-order, we find the cell $C$ at the lowest level $\ell$ that contains both $p$ and $q$ in the corresponding infinite hyperbolic quadtree. Let $c$ be the constant hidden by the big-O notation in Theorem 1. We apply quadtree splits to $C$ until we get a set of cells we will call $\mathrm{child}_\varepsilon(C)$, where each element has diameter at most $\frac{\varepsilon}{c \cdot d\sqrt{d}}$ times that of $C$. By Theorem 6(ii), this requires $\log(c \cdot d\sqrt{d}/\varepsilon) + \Theta(1)$ quadtree splits. Both $p$ and $q$ are then connected to a Steiner point in each cell of $\mathrm{child}_\varepsilon(C)$ that intersects $\mathrm{Cyl}(p)$, respectively $\mathrm{Cyl}(q)$. We repeat this procedure for all ancestor cells $C'$ of $C$ where there is a $\hat{C} \in \mathrm{child}_\varepsilon(C')$ such that $\hat{C} \subseteq C \subseteq C'$. Note that this gives a bipartite graph: all edges are between an input point and a Steiner point. We show in the full version [31] that the constructed graph is sparse:

▶ **Lemma 12.** *A point $p \in P$ gets connected to $2^{\mathcal{O}(d)} d^{\frac{3}{2}d} \log(d/\varepsilon)/\varepsilon^d$ Steiner points. When the distance $\delta$ from $p$ to its nearest neighbour in $P$ is large enough this improves to*

- $\frac{2^{\mathcal{O}(d)} d^{2d} \log(d/\varepsilon)}{\delta^{d-1}\varepsilon^d}$ *Steiner points if $\sqrt{d} < \delta < d^2/\varepsilon$,*
- $\frac{2^{\mathcal{O}(d)} \log(d/\varepsilon)}{\varepsilon}$ *Steiner points if $\delta \geq d^2/\varepsilon$.*

▶ **Lemma 13.** *This construction gives a Steiner $(1+7\varepsilon)$-spanner when $\varepsilon \leq \frac{1}{14}$.*

**Proof.** Let $\mathrm{dist}_G$ denote the distance in the graph. Given $p, q \in P$ we want to prove $\mathrm{dist}_G(p,q) \leq (1+7\varepsilon)\,\mathrm{dist}_{\mathbb{H}^d}(p,q)$. From Theorem 1 we know that $p$ and $q$ are contained in a cell $C$ in one of the infinite hyperbolic quadtrees where $\mathrm{diam}(C) = \mathcal{O}\left(d\sqrt{d}\right)\mathrm{dist}_{\mathbb{H}^d}(p,q)$. By construction, the cells of $\mathrm{child}_\varepsilon(C)$ have diameter at most $\varepsilon\,\mathrm{dist}_{\mathbb{H}^d}(p,q)$. Let $p' \in P$ be the point in the same cell from $\mathrm{child}_\varepsilon(C)$ as $p$ closest in the L-order to $q$, and $q'$ defined analogously. Now, $p'$ and $q'$ must both be connected to a Steiner point in each cell of $\mathrm{child}_\varepsilon(C)$ that intersects $\mathrm{Cyl}(p')$, respectively $\mathrm{Cyl}(q')$. By the combination of Lemma 10 and Lemma 11, this means that one of the Steiner points they are both connected to lies in a cell of $\mathrm{child}_\varepsilon(C)$ that is intersected by the geodesic between $p'$ and $q'$. Therefore, $\mathrm{dist}_G(p',q') \leq \mathrm{dist}_{\mathbb{H}^d}(p',q') + 2\varepsilon\,\mathrm{dist}_{\mathbb{H}^d}(p,q)$. We can get $\mathrm{dist}_G(p',q') \leq (1+4\varepsilon)\,\mathrm{dist}_{\mathbb{H}^d}(p,q)$ from this by noticing that $\mathrm{dist}_{\mathbb{H}^d}(p',q') \leq \mathrm{dist}_{\mathbb{H}^d}(p,p') + \mathrm{dist}_{\mathbb{H}^d}(p,q) + \mathrm{dist}_{\mathbb{H}^d}(q,q') \leq (1+2\varepsilon)\,\mathrm{dist}_{\mathbb{H}^d}(p,q)$.

We will now use induction to prove $\mathrm{dist}_G(p,q) \leq (1+7\varepsilon)\,\mathrm{dist}_{\mathbb{H}^d}(p,q)$. For $\mathrm{dist}_{\mathbb{H}^d}(p,q) = 0$ this holds trivially. As induction hypothesis, we now assume that for any pair $v, w \in P$ closer together than $p$ and $q$, we have $\mathrm{dist}_G(v,w) \leq (1+7\varepsilon)\,\mathrm{dist}_{\mathbb{H}^d}(v,w)$. In particular, this gives

us $\operatorname{dist}_G(p,p') \leq (1+7\varepsilon)\operatorname{dist}_{\mathbb{H}^d}(p,p')$. Because $p$ and $p'$ lie in a cell of diameter at most $\varepsilon \operatorname{dist}_{\mathbb{H}^d}(p,q)$ and $\varepsilon \leq \frac{1}{14}$, we have $\operatorname{dist}_G(p,p') \leq (1+7\varepsilon)\varepsilon \operatorname{dist}_{\mathbb{H}^d}(p,q) \leq \frac{3}{2}\varepsilon \operatorname{dist}_{\mathbb{H}^d}(p,q)$. For the same reason, $\operatorname{dist}_G(q,q') \leq \frac{3}{2}\varepsilon \operatorname{dist}_{\mathbb{H}^d}(p,q)$ and thus $\operatorname{dist}_G(p,q) \leq (1+7\varepsilon)\operatorname{dist}_{\mathbb{H}^d}(p,q)$. ◀

After replacing $\varepsilon$ by $\varepsilon/7$, Lemma 12 and Lemma 13 together prove most of Theorem 2. It remains to show how we handle point insertions and deletions.

**Dynamic manipulation and wrap-up of the proof of Theorem 2.** To maintain the construction dynamically, we only need self-balancing binary search trees (e.g. red-black trees [18]), henceforth called BST. For each L-order $i = 1, \ldots, 3d+3$ we maintain the points of $P$ in that order using a BST denoted by $P_i$. Additionally, we maintain a BST $S$ containing the Steiner points that functions as an associative array: each Steiner point $s \in S$ is associated with a set of points $E_s$, containing the input points connected to it. In other words, these represent the edge set of the Steiner spanner. These sets can again be implemented as BSTs. Each tree $P_i$, $S$ or $E_s$ will contain less than $n^2$ points[3] and has comparisons that take $\mathcal{O}(d)$ time, so searching, adding and removing take $\mathcal{O}(d \log n)$ time.

For a point $p \in P$, we can determine in $\mathcal{O}\big(d^2 \log n\big) + d^{\mathcal{O}(d)} \log(1/\varepsilon)/\varepsilon^d$ time which Steiner points it should be connected to by the following procedure. For each $i = 1, \ldots, 3d+3$, find (in $\mathcal{O}(d \log n)$ time) its neighbours in $P_i$. Then, for each neighbour we consider the smallest cell $C$ in the $i^{\text{th}}$ infinite hyperbolic quadtree that it shares with $p$. The cells to connect to are those in $\operatorname{child}_\varepsilon(C')$ that intersect $\operatorname{Cyl}(p)$, where $C'$ starts as $C$ and goes up its ancestors until $C \in \operatorname{child}_\varepsilon(C')$. For each cell, we take its centre as its corresponding Steiner point. Enumerating all these Steiner points takes $d^{\mathcal{O}(d)} \log(1/\varepsilon)/\varepsilon^d$ time.

Finding a Steiner point $s$ in $S$ takes $\mathcal{O}(d \log n)$ time, as does adding/removing a connection in $E_s$ (or adding/removing $E_s$ itself, if it did not exist yet or has no connections left).

Adding and removing a point $p$ only affects the points adjacent to $p$ in the L-orders and $p$ itself, so $\mathcal{O}(d)$ points altogether. For each of the affected points, we can remove all connections to Steiner points it had before and then add connections to Steiner points in the new configuration. This can affect their in total $d^{\mathcal{O}(d)} \log(1/\varepsilon)/\varepsilon^d$ connections to Steiner points, so the update takes $\log n \cdot d^{\mathcal{O}(d)} \log(1/\varepsilon)/\varepsilon^d$ time.

**Approximate Nearest Neighbours.** Using Theorem 2 we can also get a data structure for dynamic $(1+\varepsilon)$-approximate nearest neighbours. We only need to modify the data structure slightly: for each Steiner point $s$, the points in $E_s$ will now be sorted based on their distance to $s$. To query for the nearest neighbour of $p$, we then find all Steiner points $p$ would get connected to if it were to get inserted into the data structure. For each of these Steiner points, we retrieve the input point closest to it. This gives a small set of candidate points, from which we return the point closest to $p$. The returned point is the exact nearest neighbour to $p$ using the spanner distances, thus it will be an $(1+\varepsilon)$-approximate nearest neighbour in the actual space.

The sets $E_s$ together store all the edges of the Steiner spanner exactly once and $S$ contains the ($d$-dimensional) Steiner points, so these together take $n \cdot d^{\mathcal{O}(d)} \log(1/\varepsilon)/\varepsilon^d$ space. The size of the trees $P_i$ sums up to $\mathcal{O}\big(d^2 n\big)$, thus the final data structure still requires $n \cdot d^{\mathcal{O}(d)} \log(1/\varepsilon)/\varepsilon^d$ space. The full version [31] applies a similar method to dynamically maintain an approximate bichromatic closest pair.

[3] If $\varepsilon$ is small enough to lead to $\Omega(n^2)$ Steiner points, we can use the complete graph on $P$ as spanner.

▶ **Theorem 3.** *We can construct a data structure that uses* $n \cdot d^{\mathcal{O}(d)} \log(1/\varepsilon)/\varepsilon^d$ *space and can answer queries for a* $(1+\varepsilon)$*-approximate nearest neighbour in* $\log n \cdot d^{\mathcal{O}(d)} \log(1/\varepsilon)/\varepsilon^d$ *time, and perform updates (point insertions and removals) in* $\log n \cdot d^{\mathcal{O}(d)} \log(1/\varepsilon)/\varepsilon^d$ *time.*

## 5 Conclusion

We have presented a quadtree and a dynamically maintained Steiner spanner that work in hyperbolic space. The constructions are relatively simple and closely correspond to Euclidean versions, so we hope that this will spark further research on algorithms in hyperbolic space. As an example of this, we were able to give similar results for approximate nearest neighbour search as are known in Euclidean space, which is the best one can hope for as a point set inside a small hyperbolic neighbourhood has a nearly Euclidean metric. One would hope that better results are possible for point sets that are more spread out [30]. While our result does not show strong signs of improvement for spread-out point sets, we can observe improved behaviour for larger dimensions for such point sets: in Lemma 12 and near the end of the proofs of Theorem 1 and Theorem 6(iii), it can be seen that the dependence on $d$ is milder for longer distances. This is in line with the strong results on dimension reductions in hyperbolic spaces by Benjamini and Makarychev [5]. Thus, an open question is how well Steiner spanners and approximate nearest neighbour data structures designed specifically for sparse hyperbolic point sets can perform.

For both spanners and approximate nearest neighbours, our results are optimal in $n$ and match those provided by locality sensitive orderings [17] in Euclidean space, but in Euclidean space there are more complex algorithms for both that give better guarantees. For the spanner specifically, we come close to the bound for classical spanners but do not match the improved bounds for Steiner spanners [6]. Is there a hyperbolic Steiner $(1+\varepsilon)$-spanner with only $\mathcal{O}\big(n/\varepsilon^{(d-1)/2}\big)$ edges? On a related note, we do not consider spanner lightness (the total weight of the edges) and with our current construction this is unbounded. Is there a hyperbolic Steiner $(1+\varepsilon)$-spanner whose total weight is $f(\varepsilon, d)$ times the weight of a minimum spanning tree? These are all avenues for further research.

## References


1. Srinivas Aluru. Quadtrees and octrees. In *Handbook of Data Structures and Applications*. Chapman and Hall/CRC, 2004. doi:10.1201/9781420035179.ch19.
2. Sunil Arya, Theocharis Malamatos, and David M. Mount. Space-time tradeoffs for approximate nearest neighbor searching. *J. ACM*, 57(1), November 2009. doi:10.1145/1613676.1613677.
3. Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45(6):891–923, November 1998. doi:10.1145/293347.293348.
4. Riccardo Benedetti and Carlo Petronio. *Lectures on Hyperbolic Geometry*. Springer Science & Business Media, 1992.
5. Itai Benjamini and Yury Makarychev. Dimension reduction for hyperbolic space. *Proceedings of the American Mathematical Society*, 137(2):695–698, 2009.
6. Sujoy Bhore and Csaba D. Tóth. Euclidean Steiner spanners: Light and sparse. *SIAM J. Discret. Math.*, 36(3):2411–2444, 2022. doi:10.1137/22m1502707.
7. Thomas Bläsius, Tobias Friedrich, Maximilian Katzmann, Ulrich Meyer, Manuel Penschuck, and Christopher Weyand. Efficiently generating geometric inhomogeneous and hyperbolic random graphs. *Netw. Sci.*, 10(4):361–380, 2022. doi:10.1017/nws.2022.32.
8. Thomas Bläsius, Tobias Friedrich, and Anton Krohmer. Hyperbolic random graphs: Separators and treewidth. In *24th Annual European Symposium on Algorithms, ESA 2016*, volume 57 of *LIPIcs*, pages 15:1–15:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ESA.2016.15.

9 Michel Bode, Nikolaos Fountoulakis, and Tobias Müller. On the largest component of a hyperbolic model of complex networks. *The Electronic Journal of Combinatorics*, pages P3–24, 2015.

10 Mikhail Bogdanov, Olivier Devillers, and Monique Teillaud. Hyperbolic Delaunay triangulations and Voronoi diagrams made practical. In *XIV Spanish Meeting on Computational Geometry,*, 2011.

11 Károly Böröczky. Gömbkitöltések állandó görbületű terekben I. *Matematikai Lapok (in Hungarian)*, 25(3-4):265–306, 1974.

12 Prosenjit Bose, Paz Carmi, Mohammad Farshi, Anil Maheshwari, and Michiel H. M. Smid. Computing the greedy spanner in near-quadratic time. *Algorithmica*, 58(3):711–729, 2010. doi:10.1007/s00453-009-9293-4.

13 Karl Bringmann, Ralph Keusch, and Johannes Lengler. Geometric inhomogeneous random graphs. *Theor. Comput. Sci.*, 760:35–54, 2019. doi:10.1016/j.tcs.2018.08.014.

14 Paul B. Callahan and S. Rao Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *J. ACM*, 42(1):67–90, 1995. doi:10.1145/200836.200853.

15 James W Cannon, William J Floyd, Richard Kenyon, Walter R Parry, et al. Hyperbolic geometry. *Flavors of Geometry*, 31(59-115):2, 1997.

16 Timothy M. Chan. Approximate nearest neighbor queries revisited. *Discrete & Computational Geometry*, 20(3):359–373, October 1998. doi:10.1007/PL00009390.

17 Timothy M. Chan, Sariel Har-Peled, and Mitchell Jones. On locality-sensitive orderings and their applications. *SIAM Journal on Computing*, 49(3):583–600, 2020. doi:10.1137/19M1246493.

18 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*. The MIT Press and McGraw-Hill Book Company, 2001.

19 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition*. Springer, 2008. URL: https://www.worldcat.org/oclc/227584184.

20 Alon Efrat, Matthew J. Katz, Frank Nielsen, and Micha Sharir. Dynamic data structures for fat objects and their applications. *Computational Geometry*, 15(4):215–227, 2000. doi:10.1016/S0925-7721(99)00059-0.

21 David Eppstein. Squarepants in a tree: Sum of subtree clustering and hyperbolic pants decomposition. *ACM Trans. Algorithms*, 5(3):29:1–29:24, 2009. doi:10.1145/1541885.1541890.

22 Raphael A. Finkel and Jon Louis Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Informatica*, 4:1–9, 1974. doi:10.1007/BF00288933.

23 Nikolaos Fountoulakis and Tobias Müller. Law of large numbers for the largest component in a hyperbolic model of complex networks. *The Annals of Applied Probability*, 28(1):607–650, 2018. doi:10.1214/17-AAP1314.

24 Tobias Friedrich and Anton Krohmer. On the diameter of hyperbolic random graphs. *SIAM J. Discret. Math.*, 32(2):1314–1334, 2018. doi:10.1137/17M1123961.

25 Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, pages 5350–5360, 2018. URL: https://proceedings.neurips.cc/paper/2018/hash/dbab2adc8f9d078009ee3fa810bea142-Abstract.html.

26 Zhimeng Gao and Sariel Har-Peled. Almost optimal locality sensitive orderings in euclidean space. *CoRR*, abs/2310.12792, 2023. To appear in SoCG 2024. doi:10.48550/arXiv.2310.12792.

27 Sariel Har-Peled. *Geometric Approximation Algorithms*. American Mathematical Society, USA, 2011.

**28** Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory Comput.*, 8(1):321–350, 2012. doi:10.4086/toc.2012.v008a014.

**29** Birger Iversen. *Hyperbolic geometry*. Number 25 in London Mathematical Society Student Texts. Cambridge University Press, 1992.

**30** Sándor Kisfaludi-Bak. A quasi-polynomial algorithm for well-spaced hyperbolic TSP. *J. Comput. Geom.*, 12(2):25–54, 2021. doi:10.20382/jocg.v12i2a3.

**31** Sándor Kisfaludi-Bak and Geert van Wordragen. A quadtree, a steiner spanner, and approximate nearest neighbours in hyperbolic space. *CoRR*, abs/2305.01356, 2023. doi:10.48550/arXiv.2305.01356.

**32** Robert Krauthgamer and James R. Lee. Algorithms on negatively curved spaces. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006)*, pages 119–132. IEEE Computer Society, 2006. doi:10.1109/FOCS.2006.9.

**33** Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguná. Hyperbolic geometry of complex networks. *Physical Review E*, 82(3):036106, 2010.

**34** John Lamping, Ramana Rao, and Peter Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 401–408, 1995.

**35** Hung Le and Shay Solomon. Truly optimal euclidean spanners. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1078–1100, 2019. doi:10.1109/FOCS.2019.00069.

**36** Guy M Morton. A computer oriented geodetic data base and a new technique in file sequencing. Technical report, International Business Machines Company New York, 1966.

**37** Tobias Müller and Merlijn Staps. The diameter of KPKVB random graphs. *Advances in Applied Probability*, 51(2):358–377, 2019. URL: http://www.jstor.org/stable/45277962.

**38** Giri Narasimhan and Michiel Smid. *Geometric spanner networks*. Cambridge University Press, 2007.

**39** Maximilian Nickel and Douwe Kiela. Learning continuous hierarchies in the Lorentz model of hyperbolic geometry. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3776–3785. PMLR, 2018. URL: http://proceedings.mlr.press/v80/nickel18a.html.

**40** Frank Nielsen and Richard Nock. Hyperbolic Voronoi diagrams made easy. In *Prodeedings of the 2010 International Conference on Computational Science and Its Applications, ICCSA 2010*, pages 74–80. IEEE Computer Society, 2010. doi:10.1109/ICCSA.2010.37.

**41** Eunku Park and Antoine Vigneron. Embeddings and near-neighbor searching with constant additive error for hyperbolic spaces, 2024. arXiv:2402.14604.

**42** Liudmila Prokhorenkova, Dmitry Baranchuk, Nikolay Bogachev, Yury Demidovich, and Alexander Kolpakov. Graph-based nearest neighbor search in hyperbolic spaces. In *International Conference on Learning Representations*, 2022. URL: https://openreview.net/forum?id=USIgIY6TNDe.

**43** William P Thurston. Three dimensional manifolds, Kleinian groups and hyperbolic geometry. *Bulletin (New Series) of the american mathematical society*, 6(3):357–381, 1982.

**44** William P Thurston. Three-dimensional geometry and topology, volume 1. *Princeton Mathematical Series, Vol. 35*, 1997.

**45** Abraham A Ungar. Einstein's special relativity: The hyperbolic geometric viewpoint. *arXiv preprint*, 2013. arXiv:1302.6961.

**46** Moritz von Looz, Henning Meyerhenke, and Roman Prutkin. Generating random hyperbolic graphs in subquadratic time. In *Algorithms and Computation - 26th International Symposium, ISAAC*, volume 9472 of *Lecture Notes in Computer Science*, pages 467–478. Springer, 2015. doi:10.1007/978-3-662-48971-0_40.

**47** Xian Wu and Moses Charikar. Nearest neighbor search for hyperbolic embeddings. *CoRR*, abs/2009.00836, 2020. arXiv:2009.00836.