# Space-Filling Trees: A New Perspective on Incremental Search for Motion Planning

James J. Kuffner[†‡]

[†]*The Robotics Institute*
*Carnegie Mellon University*
*Pittsburgh, PA, 15213 USA*
kuffner@cs.cmu.edu

[‡]*Google, Inc.*
*1600 Amphitheatre Pkwy.*
*Mountain View, CA, 94043 USA*
kuffner@google.com

Steven M. LaValle[§]

[§]*Dept. of Computer Science*
*Univ. of Illinois at Urbana-Champaign*
*Urbana, IL, 61801 USA*
lavalle@uiuc.edu

***Abstract*— This paper introduces space-filling trees and analyzes them in the context of sampling-based motion planning. Space-filling trees are analogous to space-filling curves, but have a branching, tree-like structure, and are defined by an incremental process that results in a tree for which every point in the space has a finite-length path that converges to it. In contrast to space-filling curves, individual paths in the tree are short, allowing any part of the space to be quickly reached from the root. We compare some basic constructions of space-filling trees to Rapidly-exploring Random Trees (RRTs), which underlie a number of popular algorithms used for sampling-based motion planning. We characterize several key tree properties related to path quality and the overall efficiency of exploration and conclude with a number of open mathematical questions.**

## I. Introduction

We define and analyze an iterative process whereby a single point in a continuous space is connected via a continuous path to every other point in the space. The result is called a *space-filling tree*, in which every path has finite length and for every point in the space, there is at least one path that converges to it. We are inspired by *space-filling curves*, which started in the 19th century with Peano [1]. Some well-known examples are the Hilbert curve [2], Morton curves [3], and Sierpinski curves [4]; see [5]. The primary mathematical motivation was to illustrate one of the bizarre consequences of $[0, 1]$ and $[0, 1]^2$ having the same cardinality. Space-filling curves have also gained much recognition due to their fractal properties. Space-filling curves have a wide-range of interesting applications in mathematics and geometry [6], [7], biology and computer graphics [8], [9], cryptography [10], image compression [11], and in indexing large datasets [12], [13]. We believe there may be similar potential for space-filling tree constructions.

Structures resembling space-filling trees are common in nature. Mandelbrot showed many examples of naturally-occurring fractal structures [14], such as the vascular networks of trees, ferns, leaves, and river deltas. Examples in animal biology include the pulmonary network of the lungs, and the blood vessel networks of animal circulatory systems [15]. Lindenmayer systems (L-systems) [16] are used in computer graphics for representing and generating models of organic objects such as trees, bushes, flowers, and seashells [8], [9] and in crop sciences for modeling root systems and analyzing their absorption properties [17]. There are also potentially useful connections to industrial applications, such as designing electrical power, gas, or water distribution systems for cities. Engineers already use the "H-tree" or *H-fractal* in VLSI design as a clock distribution network for routing timing signals to all parts of a chip with equal propagation delays [18], as well in the analysis of worst-case bounds for traveling salesman problems [19]. Fundamentally, searching continuous spaces is inherent to computer science problems, such as optimization and data mining. In robotics, path planning algorithms search a continuous space for a path that enables a robot to move from one configuration to another while avoiding collisions. One successful family of approaches is based on Rapidly-exploring Random Trees (RRTs) [20], which can be considered as a Monte Carlo or stochastic variant of the trees mainly studied here. Our work may provide additional insights to recent studies on improved path qualty in RRTs[21], [22], by introducing a new measure of overall *tree* quality, as opposed to individual path quality. We believe that asymptotically optimizing the overall tree length is important to efficient exploration in motion planning. Although this paper does not propose methods to apply space-filling constructions directly to motion planning algorithms, we suspect that there may be several interesting possibilities in that direction. At the very least, space-filling trees may provide additional new tools for analyzing tree-based incremental search algorithms.

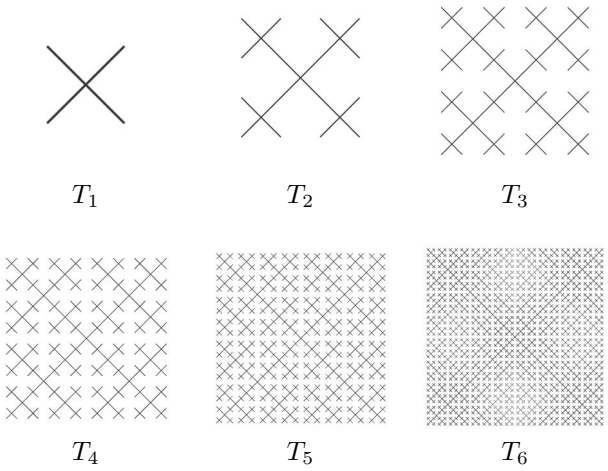


Fig. 1. Six iterations of the square space-filling tree.

Section II fully covers the basic cases such as trees that fill square and triangular planar regions. These are the easiest examples to understand and analyze. Section III will then generalize it to other space-filling trees that have self-similar, fractal structure. Section IV introduces a more general construction, which iteratively extends the tree by connecting each point in a dense sequence to the nearest tree vertex, and provides a comparison to RRTs. Section V concludes with questions and open problems for further consideration.

## II. Basic Space-Filling Trees

### A. Filling a Square

We will "fill" the $2 \times 2$ square, given by the set $[-1,1]^2 \subset \mathbb{R}^2$. Figure 1 shows six iterations of the construction that will be described. We will show that path lengths from the start to arbitrary points in the space are finite, unlike linear space-filling curves such as the Hilbert curve. The construction we use is also more efficient at covering the space in terms of total path distances and maximum path length as compared to an H-tree construction.

We define a *tree* $T$ to consist of a set $V$ of *vertices*, each of which is a point in $[-1,1]^2$, and a set $E$ of *edges*, each of which is a line segment that connects a pair of vertices.

A space-filling tree is actually defined as an infinite sequence of trees, $\mathcal{T} = \{T_0, T_1, T_2, \ldots\}$, in which each $T_i$ has associated vertices $V_i$ and edges $E_i$. Furthermore, the trees in $\mathcal{T}$ monotonically grow so that $V_i \subset V_j$ and $E_i \subset E_j$ for all $i < j$.

Now consider the following process, which was used for Figure 1. Initially, $V_0 = \{(0,0)\}$, yielding the *root* of the tree, and $E_0 = \emptyset$. Next, divide the $2 \times 2$ square into four quadrants and place a vertex at the center of each $1 \times 1$ square. Let $V_1'$ denote this set of new vertices, $V_1' = \{(1/2, 1/2), (-1/2, 1/2), (-1/2, -1/2), (1/2, -1/2)\}$. An edge is formed from every vertex in $V_1'$ to the root, yielding four edges in $E_1$. The tree $T_1$ after one iteration is given by $V_1 = V_0 \cup V_1'$ and $E_1$.

In the next iteration, each of the four $1 \times 1$ squares is divided into quadrants, and a new vertex is placed at the center of each quadrant. This is perfect symmetry with respect to the previous iteration; the structure is simply scaled by a factor of $1/2$ and shifted. The set $V_2'$ of new vertices contains all 16 ways to make points in which each coordinate is $\pm 1/4$ or $\pm 3/4$. Let $E_2'$ be the set of new edges. For each vertex $v \in V_2'$, an edge is placed in $E_2'$ that connects $v$ to the vertex in $V_1'$ that lies at the center of the $1 \times 1$ square that contains $v$. The tree $T_2$ is given by $V_2 = V_1 \cup V_2'$ and $E_2 = E_1 \cup E_2'$.

Each subsequent iteration proceeds in the same way. The tree $T_{i+1}$ is constructed from $T_i$ as follows. Each vertex in $v \in V_i'$ lies at the center of a square of width $1/2^{i-1}$. That square is divided into quadrants, with a vertex being placed into $V_{i+1}'$ for the center of each quadrant. An edge is placed in $E_{i+1}'$ for each of these four vertices, connecting it to $v$. The tree $T_{i+1}$ is given by $V_{i+1} = V_i \cup V_{i+1}'$ and $E_{i+1} = E_i \cup E_{i+1}'$. Note that some edges in $E_{i+1}'$ may overlap with parts of edges in $E_i$.

Now that the incremental process has been defined, suppose it is iterated indefinitely, resulting in a sequence $\mathcal{T}_{square}$ of trees. We will next study the properties of $\mathcal{T}_{square}$.

For any sequence $\mathcal{T}$ of trees, let $V^* = \cup_{i \in \mathbb{N}} V_i$, in which $\mathbb{N}$ is the set of natural numbers. A tree sequence $\mathcal{T}$ is called *space-filling* in a space $X$ if for every $x \in X$, there exists a path in the tree that starts at the root and converges to $x$. This convergence can be stated in terms of the vertices along the path from the root to $x$. For many $x \in X$, this convergence may occur only in the limit, rather than actually reaching $x$.

*Theorem 1:* $\mathcal{T}_{square}$ is a space-filling tree.

***Proof:*** To establish the theorem, we argue that for any point $p \in [-1,1]^2$, there exists a sequence of vertices in $V^*$ that converges to it. Without loss of generality, assume $p \in [0,1]^2$. The arguments below extend by symmetry to the other three quadrants.

For any $(x, y) \in V^*$, let $b_x$ and $b_y$ denote the binary decimal representations of $x$ and $y$, respectively. For example, if $x = 3/16$, then $b_x = .0011$. Assume that $b_x$ and $b_y$ are written canonically so that the rightmost bit is always 1. Let $|b_x|$ denote the length of the representation, which is 4 in the case of $x = 3/16$. Note that if $p \in V_i'$, then $|b_x| = |b_y| = i$. Every $V_i'$ contains $2^{2i}$ vertices, of which $2^{2i-2}$ are in $[0,1]^2$. These are all first-quadrant points that can be expressed as $(j/2^i, k/2^i)$ for any odd $j, k \in \{1, 2, 3, \ldots, 2^i\}$.

The tree already reaches any $p \in V^*$; therefore, consider any point $p \in [0,1]^2 \setminus V^*$. Let $v_i$ be the vertex in $V_i'$ that is closest to $p$. Consider the sequence $\tilde{v} = (v_1, v_2, \ldots)$. Each $v_i$ can be considered as the closest approximation to $p$ for which the number of bits needed for each coordinate is $i$. We observe that $\|p - v_i\| < 1/2^{i-1}$ for every $i$. Therefore, $\tilde{v}$ is a sequence that converges to $p$. ∎

For any point $p \in [-1,1]^2$, we use $\mathcal{T}_{square}$ and explicitly define a continuous path from $(0,0)$ to $p$. The path will be parameterized by distance along the curve. Suppose $p \notin V^*$. From the proof of Theorem 1, use the sequence $\tilde{v} = (v_1, v_2, \ldots)$ of vertices that converge to $p$. We will define a path $\pi : [0, \sqrt{2}) \to (-1,1)^2$. Let $v_0 = (0,0)$. For each $i \in \mathbb{N}$ the path is defined over the interval $[\sqrt{2}(2^{i-1}-1)/2^{i-1}, \sqrt{2}(2^i - 1)/2^i]$ as

$$\pi(s) = (1 - \alpha) v_{i-1} + \alpha v_i \tag{1}$$

in which $\alpha = (s/\sqrt{2} - (2^{i-1} - 1)/2^{i-1})/2^i$. Note that the path does not actually "reach" $p$, but instead converges to it. We will nevertheless say that $\tau$ is a *path to* $p$. For the case in which $p \in V^*$, we imagine that $\tilde{v}$ is truncated to a finite sequence that actually reaches $p$ for some $v \in V_k'$. In this case, the function (1) and its domain are limited to $i$ from 1 to $k$, rather than using an infinite number of segments.

Now that the space filling property has been established, it is next interesting to consider how efficiently this is accomplished. For example, the classical Hilbert space-filling curve has the property that the length of the curve doubles (asymptotically) in each iteration. Also in each iteration, the distance from the furthest away points in $[-1,1]^2$ from the

curve is cut in half. As the iterations increase, the path length tends to infinity.

The next theorems establish remarkable properties of $\mathcal{T}_{square}$. Theorem 2 says that no paths are longer than $\sqrt{2}$, rather than tending to infinity as in the Hilbert curve. Theorem 3 implies that the total length of all edges grows asymptotically in each iteration by a factor of 2, just as in the case of the Hilbert curve.

*Theorem 2:* For every $p \in [-1,1]^2$, there exists a path in $\mathcal{T}_{square}$ that converges to $p$ and has length no more than $\sqrt{2}$. Furthermore, the path length equals $\sqrt{2}$ if and only if $p \notin V^*$.

***Proof:*** Suppose $p \notin V^*$. In that case, consider the path $\tau$, defined in (1), using the sequence $\tilde{v}$ of vertices that converges to $p$. Note that each segment of that path is exactly half the length of the previous segment. The total length is therefore expressed as an infinite sum

$$\sum_{i\in\mathbb{N}} \sqrt{2}/2^i = \sqrt{2}\sum_{i\in\mathbb{N}} 2^{-i}, \tag{2}$$

in which the right sum is the classical geometric series with ratio $1/2$. The sum converges to 1, and the path length is $\sqrt{2}$.

For the case in which $p \in V^*$, the path stops at some vertex $v \in V'_k$ for some $k$. The total length is expressed using only the first $k$ terms of (2), which is strictly less than $\sqrt{2}$. ∎

Now consider the total length of all edges in $\mathcal{T}_{square}$. For overlapping edges, we count them only once. Imagine that the tree is built from electrical wire and we would like to know how much total wire is used. Clearly, an infinite amount of wire is needed; however, it is interesting to know the rate of wire consumption with respect to the iterations. The next theorem characterizes this.

*Theorem 3:* The combined length of the union of all edges in $T_i$ is $\frac{4\sqrt{2}}{3}\left(2^i - 2^{-i}\right)$.

***Proof:*** Let $\ell_i$ denote the combined path length in $T_i$. By exploiting symmetry over the four quadrants of $[-1,1]^2$, the length in each quadrant is $\ell_i/4$. We therefore derive an expression for $\ell_i/4$:

$$\ell_i/4 = \frac{\sqrt{2}}{2} + 3\left(\frac{\sqrt{2}}{4}\right) + 11\left(\frac{\sqrt{2}}{8}\right) + \cdots + N_i\left(\frac{\sqrt{2}}{2^i}\right),$$

in which $N_i$ is the number of new branches that are added at iteration $i$. The expression for $N_i$ can be derived by recursion. The first three iterations are illustrated in Figure 2.

The new branches at each iteration (*illustrated in red*) are half the length of the branches added in the previous iteration. From $T_1$ to $T_2$, three new branches of length $\sqrt{2}/4$ are added. From $T_2$ to $T_3$, a total of eleven new branches of length $\sqrt{2}/8$ are added. Because of edge overlap, the number of branches $N_i$ added at iteration $i$ is one less than the nominal four branches for each quadrant of each branch added at the previous iteration $N_{i-1}$. This yields the

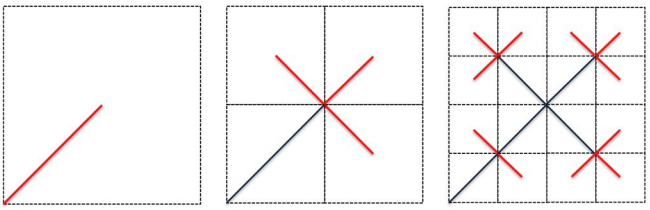

Fig. 2. Successive iterations ($T_1$, $T_2$, and $T_3$) of a square space-filling tree. Only the quadrant $[0,1]^2$ with the root at the bottom-left corner is illustrated. The remaining quadrants are symmetrical. New branches (*red*) are half the length of the branches added in the previous iteration.

recurrence $N_i = 4N_{i-1} - 1$ with base case $N_1 = 1$. Solving the recurrence in closed form yields

$$N_i = \frac{2^{2i-1}+1}{3}.$$

The combined path length $\ell_i/4$ can be expressed as the sum

$$\ell_i/4 = \sum_{j=1}^{i} N_j \frac{\sqrt{2}}{2^j}.$$

This sum represents adding up all edges of a fixed height in the tree. In the first iteration, a single edge of length $\sqrt{2}/2$ is produced. In the second iteration, 3 edges of length $\sqrt{2}/4$ are formed, and in general, at iteration $i$, $N_i$ branches of length $\sqrt{2}/2^i$ are added.

Substituting the closed form of $N_i$ into the sum yields

$$\ell_i/4 = \sum_{j=1}^{i}\left(\frac{2^{2j-1}+1}{3}\right)\frac{\sqrt{2}}{2^j}.$$

By factoring out the constant $\sqrt{2}/3$, we obtain

$$\ell_i/4 = \frac{\sqrt{2}}{3}\sum_{j=1}^{i}\left(\frac{2^{2j-1}+1}{2^j}\right).$$

Splitting the sum into two terms gives

$$\ell_i/4 = \frac{\sqrt{2}}{3}\left(\sum_{j=1}^{i} 2^{j-1} + \sum_{j=1}^{i}\frac{1}{2^j}\right).$$

Each of the two sums can be simplified to their known closed-form equivalents:

$$\sum_{j=1}^{i} 2^{j-1} = 2^i - 1, \qquad \textit{and} \qquad \sum_{j=1}^{i}\frac{1}{2^j} = 1 - 2^{-i}.$$

Finally, by substitution and simplification we obtain:

$$\ell_i/4 = \frac{\sqrt{2}}{3}\left(2^i - 2^{-i}\right).$$

∎

Note that the construction in this section could be transformed into another space by using a well-behaved mapping, such as a Lipschitz continuous function from the square into the desired space. The same applies to constructions in Section III. These transformations, however, may destroy self-similarity and distort path lengths.

## III. Other Self-Similar Space-Filling Trees

The space-filling tree $\mathcal{T}_{square}$ of Section II-A clearly has self-similarity, making it a *fractal*, much like the Sierpinski triangle, Cantor sets, and numerous other constructions. This section introduces several other self-similar space-filling trees.

### A. Filling a cube

We can easily generalize $\mathcal{T}_{square}$ to fill a cube $[-1,1]^n \subseteq \mathbb{R}^n$ for any positive integer $n$. Whereas the square was divided into quadrants in each iteration, we now divide $[-1,1]^n$ into $2^n$ orthants. Once again, we initially have $V_0 = \{(0,0)\}$. The vertices in $V_1'$ are the centers of the $2^n$ orthants. These are all points of the form $(\pm 1/2, \pm 1/2, \cdots, \pm 1/2)$. This results in $2^n$ edges in $E_1$ (there were $2^n = 4$ for $\mathcal{T}_{square}$). Proceeding incrementally, each $v \in V_i'$ lies at the center of a cube of width $1/2^{i-1}$. The result is a sequence $\mathcal{T}_{cube}$ that fills $[-1,1]^n$. Figure 3 illustrates the space-filling tree for the case of $n = 3$.

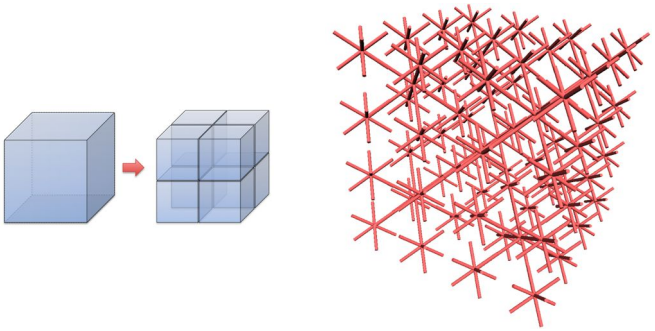

Fig. 3. Subdivision scheme (*left*) and the third iteration of the space-filling tree (*right*) for the 3-dimensional cube, $[-1,1]^3$.

The theorems of Section II-A cleanly generalize:

*Theorem 4:* $\mathcal{T}_{cube}$ is a space-filling tree.

***Proof:*** The proof proceeds in the same manner as the proof of Theorem 4. The sequence $\tilde{v}$ is formed by the sequence of closer and closer approximations to $p \in [0,1]^n$ by considering binary representations for all $n$ coordinates. ∎

*Theorem 5:* For every $p \in [-1,1]^n$, there exists a path in $\mathcal{T}_{cube}$ that converges to $p$ and has length no more than $\sqrt{n}$. Furthermore, the path length equals $\sqrt{n}$ if and only if $p \notin V^*$.

***Proof:*** The proof is nearly identical to that of Theorem 2. The edge length is again divided by 2 in each iteration. Therefore, the same geometric series is obtained. It is scaled by $\sqrt{n}$ (generalized from $\sqrt{2}$), which can be seen by considering the path length from the root to any corner of the cube. ∎

*Theorem 6:* The combined length of the union of all edges in $\mathcal{T}_{cube}$ increases asymptotically as $O(2^{(n-1)i})$, with the combined length of the union of all edges in $T_i$ given exactly by $\frac{2^n\sqrt{n}}{2^n-1}\left(2^{(n-1)i} - 2^{-i}\right)$.

***Proof:*** Let $\ell_i$ denote the combined path length in $T_i$. Following the proof of Theorem 3, we exploit symmetry across the orthants. First, we derive the number of branches $N_i$ added at each iteration $i$.

Because of edge overlap, $N_i$ is one less than the nominal $2^n$ branches for each orthant of each branch added at the previous iteration $N_{i-1}$. The general recurrence is $N_i = bN_{i-1} - c$, in which $b = 2^n$ is the constant nominal branching factor of the tree and $c$ is the constant number of overlapping branches discounted at each iteration. For the case of our method of constructing $\mathcal{T}_{cube}$, we have $c = 1$. As before, the base case for the recursion is $N_1 = 1$, and solving in closed form yields

$$N_i = \frac{(b-c-1)b^{i-1} + c}{b-1}.$$

For the square ($n = 2$), we have $b = 4$ and $c = 1$, which yields a formula that matches the result of the recursive derivation in Theorem 2. For the case of $n = 3$, we have $b = 8$ and $c = 1$; therefore,

$$N_i = \frac{6 * 8^{i-1} + 1}{7}.$$

For the general case, we have $b = 2^n$ and $c = 1$, and the general formula for the number of branches added to $\mathcal{T}_{cube}$ at iteration $i$ is

$$N_i = \frac{(2^n - 2)2^{n(i-1)} + 1}{2^n - 1}.$$

We now derive the combined path length $\ell_i/2^n$ as the combined sum of all edges in $T_i$ as

$$\frac{\ell_i}{2^n} = \sum_{j=1}^{i} N_j \frac{\sqrt{n}}{2^j}.$$

This sum represents adding up all edges of a fixed height in the tree. At iteration $i$, $N_i$ branches of length $\sqrt{n}/2^i$ are added. Substituting the closed form of $N_i$ into the sum yields

$$\frac{\ell_i}{2^n} = \sum_{j=1}^{i} \left( \frac{(2^n - 2)2^{n(i-1)} + 1}{2^n - 1} \right) \frac{\sqrt{n}}{2^j}.$$

By factoring out the constant $\sqrt{n}/(2^n - 1)$, we obtain

$$\frac{\ell_i}{2^n} = \frac{\sqrt{n}}{2^n - 1} \sum_{j=1}^{i} \left( \frac{(2^n - 2)2^{n(i-1)} + 1}{2^j} \right).$$

Splitting the sum into two terms gives

$$\frac{\ell_i}{2^n} = \frac{\sqrt{n}}{2^n - 1} \left\{ \sum_{j=1}^{i} \left( \frac{(2^n - 2)2^{n(i-1)}}{2^j} \right) + \sum_{j=1}^{i} \frac{1}{2^j} \right\}.$$

Each of the two sums can be simplified to the closed-form equivalents:

$$\sum_{j=1}^{i} \left( \frac{(2^n - 2)2^{n(i-1)}}{2^j} \right) = 2^{(n-1)i} - 1$$

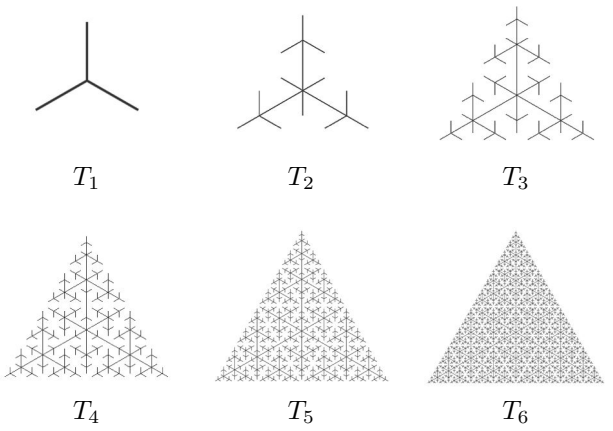


Fig. 4. Six iterations of the triangle space-filling tree.

and

$$\sum_{j=1}^{i} \frac{1}{2^j} = 1 - 2^{-i}.$$

Finally, through substitution and simplification we obtain:

$$\frac{\ell_i}{2^n} = \frac{\sqrt{n}}{2^n - 1}\left(2^{(n-1)i} - 2^{-i}\right).$$

We verify that for $n = 2$, we obtain the same formula derived in Theorem 3. This general result proves that the combined length of the union of all edges in $\mathcal{T}_{cube}$ increases asymptotically as $O(2^{(n-1)i})$. ∎

### B. Filling a Triangle

Figure 4 shows a space-filling tree $\mathcal{T}_{tri}$ over a triangular region. To define the incremental construction, we divide the triangular region into smaller triangles, instead of quadrants. Initially, the root vertex is placed at the triangle center, yielding $V_0$. The triangle is then subdivided into four triangles. For $V_1'$, we obtain four vertices, one for the center of each smaller triangle; however, one of the vertices coincides with the root. The other three vertices in $V_1'$ are connected to the root to form $E_1$. This process continues in the same way by exploiting the symmetries of the subdivision. An interesting difference can be observed in Figure 4 in comparison to Figure 1. In each step of the subdivision, the central triangle appears "upside down" with respect to the others. This causes an interesting orientation change in that part of the tree. This was not obtained for the square case because all smaller squares are axis aligned.

Theorems 1 to 3 can again be adapted. To establish that $\mathcal{T}_{tri}$ is space-filling, a sequence $\tilde{v}$ is constructed by using the sequence of refined triangles that contain $p$, rather than squares. This results in a converging sequence of vertices for any point in the triangle.

Now consider the length of the path to each point. If the height of the equilateral triangle $h = 1$ and the tree is defined to be rooted at the center of the triangle, then the length of each edge in $T_1$ is $1/3$. In general, for a triangle of side length $L$, the length of added edges in $T_1$ will be $\frac{L}{2\sqrt{3}}$, or $h/3$ (exactly $1/3$ of the height). As with the case of the square, each of the subtriangles is exactly half the size of the triangle in the previous iteration. Thus, subsequently added edges are reduced in length by a factor of two at each iteration.

*Theorem 7:* For every $p$ contained in an equilateral triangle of height $h$, there exists a path in $\mathcal{T}_{tri}$ that converges to $p$ and has length no more than $2h/3$. Furthermore, the path length equals $2h/3$ if and only if $p \notin V^*$.

***Proof:*** The proof follows that of Theorem 2, as the edge length is again divided by 2 in each iteration. The distance $d_i$ from the root at the center of the tree to a leaf vertex of $T_i$ is bounded by the sum:

$$d_i \leq \frac{h}{3}\left(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots + \frac{1}{2^i}\right).$$

Let $d_{max}$ be the upper bound on the length of the path to any point $p$ inside the triangle from the root. Note that $d_{max}$ is the limit of the sum:

$$d_{max} = \frac{h}{3} \lim_{i\to\infty} \sum_{j=0}^{i} \frac{1}{2^i}.$$

The closed-form of the sum is 2; therefore, the result is $d_{max} = 2h/3$. ∎

Note the perfect correspondence with the $\sqrt{2}$ limit for the square case, which was the distance from the center of the square to a corner. For the general case of a cube of dimension $n$ and side length $2L$, the upper bound on $d_{max}$, the length of the path to any point $p \in [-L, L]^n$ inside the cube from the root asymptotically approaches $\sqrt{n}L$. In the case of an equilateral triangle of height $h$, the distance from the center to a corner is exactly $2h/3$, which is analogous to this result.

We now derive the sum total length of all edges in $T_i$ for the equilateral triangle case.

*Theorem 8:* The combined length of the union of all edges in $\mathcal{T}_{tri}$ increases asymptotically as $O(2^i)$, with the combined length of the union of all edges in $T_i$ given exactly by $h\left(\frac{5}{3}2^{i-1} - \frac{2}{3}2^{1-i}\right)$ for a triangle of height $h$.

***Proof:*** Let $\ell_i$ denote the combined path length in $T_i$. First, we derive the number of branches $N_i$ added at each iteration $i$. Each triangle is divided into four equal subtriangles, with the terminal vertex of each edge added in the previous iteration becoming the center point of four new subtriangles at the next iteration. Thus, the total number of subtriangles at iteration $i$ is given by $4^i$. Although $\mathcal{T}_{tri}$ has a nominal constant branching factor of 4, due to the coincident subsequent vertex at the center and overlap between edges, the structure of $T_i$ can be expressed entirely with vertices of degree 1, 3, 4, or 6. The initial tree $T_0$ contains only a single vertex in the center. The first three iterations are illustrated in Figure 5.

The new branches at each iteration (*illustrated in red*) are half the length of the branches added in the previous iteration. The first iteration adds three branches of length $h/3$ yielding $T_1$ (thus $N_i = 3$). The subsequent iteration, $T_2$, adds

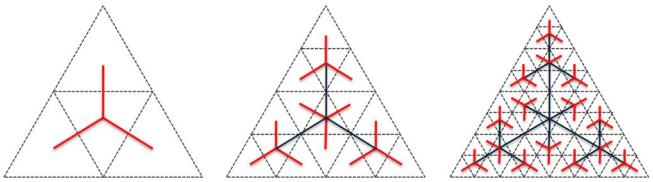

Fig. 5. Successive iterations ($T_1$, $T_2$, and $T_3$) of a triangle space-filling tree. New branches (*red*) are half the length of the branches added in the previous iteration. From $T_1$ to $T_2$, twelve new branches are added, and from $T_2$ to $T_3$, twenty-two new branches are added.

three branches of length $h/6$ to each of the four vertices of $T_1$ for a total of twelve additional edges ($N_2 = 12$). From $T_2$ to $T_3$, a total of forty-two new branches of length $h/12$ are added ($N_3 = 42$). All subsequent iterations follow the general recurrence

$$N_{i+1} = 3N_i + 2(4^i - N_i - 4^{i-2}),$$

which is valid for $i > 2$. The first term represents the 3 edges added to each terminal vertex of each edge added at the previous iteration, whereas the second term represents the 2 edges added to the remaining vertices ($4^i - N_i$) minus the number of vertices that have already achieved degree 6 ($4^{i-2}$). Solving the recurrence in closed form in terms of $N_i$ and simplifying gives

$$N_i = 10(4^{i-2}) + 2,$$

in which $i \geq 2$. We now derive the combined path length $\ell_i$ in $T_i$ as the sum length of all edges added at each iteration:

$$\ell_i = T_1 e_i + \sum_{j=2}^{i} N_j e_j.$$

At iteration $i$, $N_i$ branches of length $e_i = \frac{2h}{3}2^{-i}$ are added. Substituting $T_1 e_1 = h$ and the closed form of $N_i$ and $e_i$ into the sum and splitting the sum into two terms yields

$$\ell_i = h + \frac{2}{3}h\left(10\sum_{j=2}^{i} 2^{j-4} + 2\sum_{j=2}^{i} 2^{-j}\right).$$

Each of the two sums can be simplified to the closed-form equivalents:

$$\sum_{j=2}^{i} 2^{j-4} = \frac{2^{i+1}-4}{2^4} \qquad \textit{and} \qquad \sum_{j=2}^{i} 2^{-j} = \frac{1}{2} - 2^{-i}.$$

Finally, by substitution we obtain

$$\ell_i = h + \frac{2}{3}h\left(10\left(\frac{2^{i+1}-4}{2^4}\right) + 2\left(\frac{1}{2} - 2^{-i}\right)\right),$$

which after further simplification reduces to

$$\ell_i = h\left(\frac{5}{3}2^{i-1} - \frac{2}{3}2^{1-i}\right).$$

We see that the combined length of the union of all edges in $\mathcal{T}_{tri}$ increases asymptotically as $O(2^i)$. ∎

### C. Honeycombs and Spatial Subdivisions

It is not surprising that many space-filling tree constructions are possible given the number of shapes that can be used to tile regions of space. In the previous examples, we relied on base shapes (square, cube, and triangle) that could easily be subdivided into smaller, self-similar regions.

Patterns of space-filling or close-packing polyhedral or higher-dimensional cells without gaps are called *honeycombs*. Although the cubic tiling is notable as the only regular honeycomb in $\mathbb{R}^n$ for $n > 2$, there are numerous non-regular honeycomb subdivisions [23], [24]. By connecting the centers of honeycomb cells to recursively subdivided close-packing shapes, it is possible to construct a rich variety of space-filling trees. We illustrate this concept with the examples of the regular tetrahedron and the regular octahedron in $\mathbb{R}^3$.

Tetrahedral and octahedral shapes yield alternating recursive spatial subdivisions, which can be generalized into space-filling tree construction techniques for both shapes. A regular tetrahedron of side length $L$ can be subdivided into four smaller tetrahedra and a single octahedron, all of uniform side length $L/2$. Figure 6 illustrates this subdivision scheme. A regular octahedron of side length $L$ can be subdivided into six smaller octahedra and eight tetrahedra all of uniform side length $L/2$, which is illustrated in Figure 7. Note that each of these subdivision schemes *requires the other* in order to be iterated. The alternating recursive iterations allow us to start with any size base shape and construct a space-filling tree. We begin by defining a single root vertex at the center of the original shape. At each iteration, we add branches that connect each vertex to new vertices defined at the centers of each of the subshapes. Smaller tetrahedra use the tetrahedron subdivision scheme, whereas smaller octahedra use the octahedron subdivision scheme. The process can be iterated indefinitely.

Intuitively, if the vertices of each $T_i$ lie at the centers of the subshapes that are completely contained within the original shape and form a close-packing, increasingly fine-grained subdivision will gradually fill the original shape. As before, Theorems 1 to 3 can be adapted to yield a converging sequence of vertices for any point inside the original shape for which the total length from the root is finite. Although we do not derive the exact formulas here, the scaling factors for both the tetrahedron and octahedron are again $1/2$, which produces a similar exponential growth rate for the combined length of the union of all edges.

## IV. Other Space-Filling Trees

In addition to the basic constructions presented in Sections II and III, an even larger variety of space-filling trees can be imagined. Suppose that we start with any bounded path-connected region $R$ of some metric space. Consider a countably infinite set $Y$ of points that is dense[1] in $R$.

[1]From topology, a set $Y$ is *dense* in $X$ if the closure of $Y$ is $X$. This implies that there are no open sets in $X$ that do not contain at least one point in $Y$.

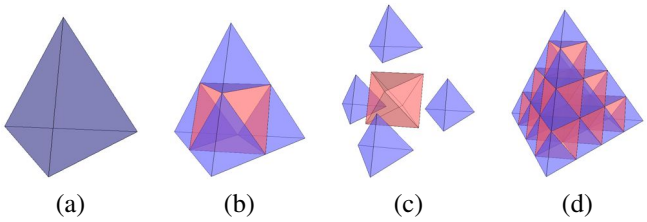


Fig. 6. Subdivision of the Tetrahedron: (a) base primitive; (b) first subdivision into one central octahedron and four corner tetrahedra; (c) exploded view of first subdivision; (d) second subdivision iteration.

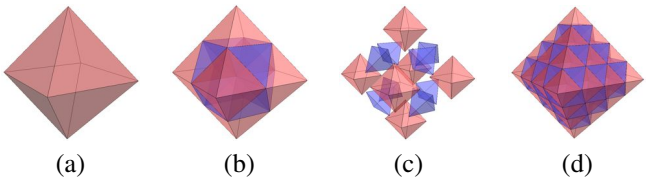


Fig. 7. Subdivision of the Octahedron: (a) base primitive; (b) first subdivision into six octahedra and eight tetrahedra; (c) exploded view of first subdivision; (d) second subdivision iteration.

Suppose the points in $Y$ have been ordered into a sequence $\gamma : \mathbb{N} \to Y$, in which $\gamma(i)$ yields the $i^{th}$ point in the sequence. We refer to $\gamma$ as a *dense sequence* of points.

Now designate any point $x \in R$ as the *root vertex*, and then connect $x$ to every point in $\gamma$ via any finite-length path that maps into $R$. As a simple example, suppose $R$ is a unit disc in $\mathbb{R}^2$, centered at the origin. Using a dense set $Y$ and sequence $\gamma$ of points in $R$, we connect every $\gamma(i)$ to $(0, 0)$ by a continuous path. Clearly this makes a tree. If possible (i.e. if $R$ is a *geodesic metric space*.), we may connect $x$ to each $\gamma(i)$ along the shortest possible path. This yields optimal distance to every point in the dense set, which seems wonderful. However, this tree is not space-filling because there is not a path that converges to points outside of $Y$. We can present a sequence of entire paths whose endpoints get closer and closer to some $p \in R \setminus Y$; however, it is not achieved by a single path.

There nevertheless exists a simple way to convert any dense sequence into a space-filling tree. Rather than connecting every point in $Y$ to the root, we connect each $\gamma(i)$ to the nearest vertex in $V_{i-1}$. Ideally, this connection should be along the shortest possible path (if it exists). In this case, the $i^{th}$ vertex $v_i$ is simply $v_i = \gamma(i)$. The $i^{th}$ edge $e_i$ is produced by connecting $v_i$ to $v_j$ in which

$$j = \underset{k \in \{1,\ldots,i-1\}}{\operatorname{argmin}} \rho(v_i, v_k) \tag{3}$$

and $\rho$ is the distance metric on $R$.

Using the general process described in (3), numerous constructions can be imagined, leading to many open questions regarding path lengths. The properties of the resulting space-filling tree depend heavily on the particular sequence $\gamma$. The tree will generally not have self-similarity unless the appropriate symmetries exist in $\gamma$.

The method of (3) can even be applied as an alternative way to construct $\mathcal{T}_{square}$. We need only to define $\gamma$ as the centers of the squares that are constructed in each iteration. Hence, $\gamma(1) = (0, 0)$, and $\gamma(2)$ through $\gamma(5)$ are the four points of the form $(\pm 1/2, \pm 1/2)$ (the particular order these four does not matter). The next iteration is simulated by assigning the centers of the 16 squares of width $1/2$ from $\gamma(6)$ to $\gamma(21)$. The process continues in this way indefinitely.

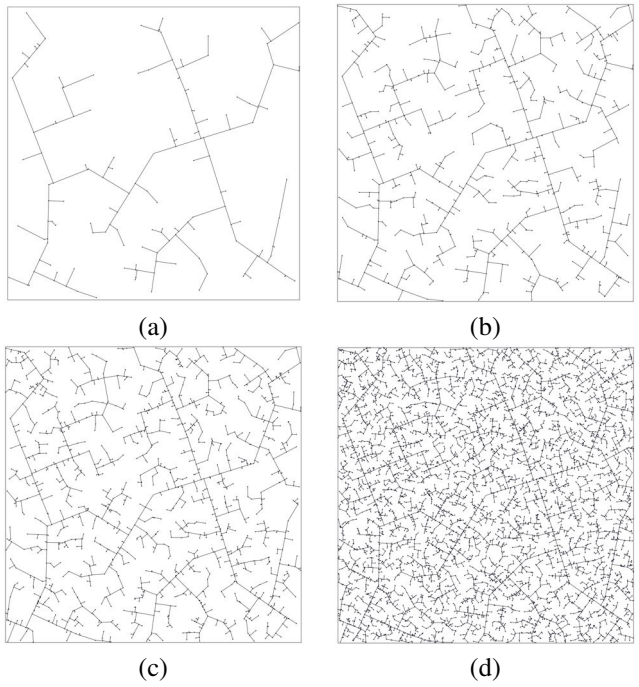


Fig. 8. Example of a Rapidly-exploring Random Tree (RRT) in $[-1, 1]^2$ after: (a) 100 samples; (b) 500 samples; (c) 1000 samples; (d) 5000 samples.

Now suppose that $\gamma$ is obtained as the result of uniform random sampling in $R$. The sequence is dense *almost surely*. Using this to construct the tree generates what is referred to as a *Rapidly-exploring Random Tree* (RRT) [20]. Figure 8 shows an example. The RRT is space-filling with probability one. The RRT has been particularly useful in recent years for searching high-dimensional spaces for path planning, optimization, and control problems.

As in previous sections, consider the lengths of paths in the tree. Figure 9 illustrates the distribution of values for the average path length and the maximum path length for an RRT rooted at the center of $[-1, 1]^2$ after 1000 iterations. The path length is defined as sum of the total edge lengths to reach any given point on the tree from the root. The histograms were computed over 1000 independent trials using point-to-edge distances for RRT growth which removes any slight geometric variations in the tree structure due to discretization of the RRT step size [25].

The average path length from the center of the square to any point on the RRT had a mean of 1.3028 and a median of 1.2766, whereas the maximum from among all possible paths in the tree had a mean of 2.7206 and a median of 2.6736 over 1000 trials. This compares to the worst-case path length of $\sqrt{2} \approx 1.4142$ for the square space-filling tree. Although the maximum path length for an RRT in the worst-case is

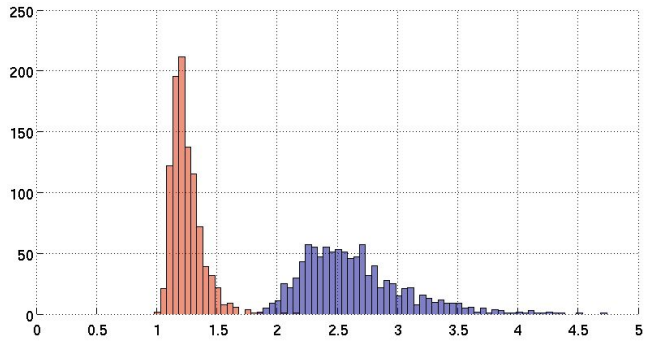


Fig. 9. Distribution of values for the average path length *(red)*, and maximum path length *(blue)* for a Rapidly-exploring Random Tree (RRT) in $[-1, 1]^2$ after 1000 samples (N = 1000 trials).

theoretically unbounded, these data show that the average or expected value is rather small (less than twice $\sqrt{2}$).

## V. Conclusions

We introduced an incremental construction called a *space-filling tree* and provided several examples. Although it is straightforward to construct trees for which the set of vertices is dense, the distinguishing feature of a space-filling tree is that for every point in the space, there is a finite-length path that converges to it. The two main properties to analyze are: 1) the lengths of individual paths in the tree, and 2) how the total length of all edges increases incrementally. We have presented a basic comparative analysis to an RRT, which can be considered as stochastic variants of space-filling trees. Many open problems remain, primarly exploring how space-filling tree concepts can be applied to designing and analyzing motion planning algorithms. One key challenge is how to incorporate obstacles into space-filling constructions. There are a number of classical planning algorithms that rely on recursive spatial subdivisions, such as quadtrees and octrees. Perhaps there exist methods for constructing space-filling trees applied recursively on cells of free space that result in deterministic planning methods with useful path quality or resolution-completeness properties. For example, could a sampling-based planner like an RRT be improved using space-filling tree constructions? How would such an algorithm scale with dimension, in terms of the expected number of samples required to get within an epsilon distance to an arbitrary point in free space? Can the basic constructions for the square or triangle be generalized to a decomposition of Voronoi cells or simplicial complexes of $n$-dimensional tetrahedral cells? Are there additional heuristics or efficient distance metrics that could be leveraged for goal-directed search using a space-filling tree?

In addition to these questions specific to applications for motion planning, there are numerous possibilities for other space-filling tree constructions, leading to a wide range of open questions from a pure mathematics perspective. For example, what shapes can be filled with self-similar structures? Which constructions are optimal with respect to path lengths or total edge lengths? What constructions can be made that optimize vertex degree? In summary, we hope that space-filling tree concepts will become useful in many diverse fields, and this paper has attempted to provide a solid introduction and framework from which to build upon.

**Acknowledgments:** Kuffner has been supported in part by Google Research, NSF grant EEC-0540865, the CMU Quality of Life Technology Center, and the Digital Human Research Center (AIST Tokyo) Japan. LaValle is supported in part by NSF grant 0904501 (IIS Robotics), NSF grant 1035345 (CNS Cyberphysical Systems), DARPA SToMP grant HR0011-05-1-0008, and MURI/ONR grant N00014-09-1-1052.

## References

[1] G. Peano, "Sur une courbe, qui remplit toute une aire plane," *Math. Ann*, vol. 36, pp. 157–160, 1890.

[2] D. Hilbert, "Über die stetige abbildung einer linie auf ein flächenstück," *Math. Ann*, vol. 38, pp. 459–460, 1891.

[3] G. Morton, "A computer oriented geodetic data base and a new technique in file sequencing," 1966.

[4] W. Sierpinski, "Sur une nouvelle courbe countinue qui remplit toute une aire plane," *Bull. Acad. Sci. de Cracovie (Sci. math. et nat., Serie A)*, pp. 462–478, 1912.

[5] H. Sagan, *Space-Filling Curves*. Springer-Verlag, 1994, iSBN 0387942653.

[6] A. Butz, "Space filling curves and mathematical programming," *Information and Control*, vol. 12, pp. 314–330, 1968.

[7] M. Mokbel, W. Aref, and I. Kamel, "Analysis of multi-dimensional space-filling curves," *GeoInformatica*, vol. 7, pp. 179–209, 2003.

[8] P. Prusinkiewicz, "Simulation modeling of plants and plant ecosystems," *Communications of the ACM*, vol. 43, no. 7, pp. 84–93, 2000.

[9] P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants*. Springer-Verlag, 1990, iSBN 978-0387972978. Online at: http://algorithmicbotany.org/.

[10] Y. Matias and A. Shamir, "A video scrambling technique based on space filling curves," in *Advances in Cryptology (CRYPTO'87)*, 1987.

[11] B. Moghaddam, K. Hintz, and C. Stewart, "Space-filling curves for image compression," in *Proc. of the SPIE*, 1991, pp. 414–421.

[12] T. Asano, D. Ranjan, T. Roos, and E. Welzl, "Space-filling curves and their use in the design of geometric data structures," in *Theoretical Computer Science*. Elsevier, 1997.

[13] J. Lawder and P. King, "Using space-filling curves for multi-dimensional indexing," in *Lecture notes in computer science*. Springer, 2000.

[14] B. Mandelbrot, *The Fractal Geometry of Nature*. W.H. Freeman, 1982.

[15] T. Nonnenmacher, G. Losa, and E. Weibel, Eds., *Fractals in Biology and Medicine*. Birkhäuser, 1993.

[16] A. Lindenmayer, "Mathematical models for cellular interaction in development, parts i and ii," *J. Theore. Biol.*, vol. 18, pp. 280–315, 1968.

[17] M. Bohn, J. Novais, R. Fonseca, R. Tuberosa, and T. Grift, "Genetic evaluation of root complexity in maize," *Acta Agronomica Hungarica*, vol. 54, no. 3, pp. 291–303, 2006.

[18] S. Browning, "The tree machine: A highly concurrent computing environment," Ph.D. dissertation, California Inst. of Technology, 1980.

[19] M. Bern and D. Eppstein, "Worst-case bounds for subadditive geometric graphs," in *9th ACM Symp. on Comp. Geometry*, 1993, pp. 183–188.

[20] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int'l Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, May 2001.

[21] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," in *Proceedings Robotics: Science and Systems*, 2010.

[22] O. Nechushtan, B. Raveh, and D. Halperin, "Sampling-diagram automata: A tool for analyzing path quality in tree planners," in *Proceedings Workshop on Algorithmic Foundations of Robotics*, December 2010.

[23] M. Deza and M. Shtogrin, "Uniform Partitions of 3-space, their Relatives and Embedding," *European J. of Combinatorics*, vol. 21, no. 6, pp. 807–814, 2000.

[24] B. Grünbaum, "Uniform tilings of 3-space," *Geombinatorics*, vol. 4, pp. 49–56, 1994.

[25] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, also available at http://planning.cs.uiuc.edu/.