Hexagonal Hierarchies in Space-Filling Curves: A Novel Approach to Urban Geocoding Challenges

Kudzaishe G Zharare Tatenda Rongonda

February 14, 2024

Abstract

Numerous solutions have been proposed to overcome the absence of precise street addresses affecting half the world's urban population, a key hurdle in advancing service delivery, ecommerce, navigation, emergency response, disaster relief, and urban planning. Current solutions grapple with three main challenges: balancing the system's adaptability with user-friendliness; reconciling universal applicability with local proximity indicators; and striking a balance between ensuring offline functionality and maintaining ease of system implementation. Space-filling curves (SFCs), widely used in big data, image processing, and computer graphics, offer an efficient means to transform n-dimensional space into a one-dimensional sequence. This paper introduces a novel use of the Naive Node Gosper Curve in indexing, clustering, and geocoding points from a 3 dimensional sphere. We explain the implementation of a new hexagonal hierarchical structure for SFCs, utilizing the Gosper fractal to enhance hexagonal grid benefits and overcome issues of non-convex shapes and recursive transformations. Our geocoding method ensures unique, bijective mapping for each point along the curve, maintaining proximity in multidimensional space and sequential code continuity for effective memory addressing at all hierarchy levels.

1 Introduction

In large parts of the world, homes and buildings either do not have street addresses or have addresses that cannot be precisely located without specific local knowledge. This is due to a variety of factors, including:

1. Streets that are unnamed or have been renamed a few times.
2. Unofficial roadways (small and large paths) and settlements (e.g., slums of Harare, or Soweto).
3. Newly constructed streets whose names are not widely known.
4. Areas that have the same or similarly named street in close proximity (e.g., there is more than 10000 streets named Main Street in the US alone).
5. Locally used names of streets that differ from the official names. These could be completely different names or abbreviations of the official names.
6. Unusual ordering of building numbers - non-consecutive or not aligned with the street.

Often, the most precise and easily accessible street maps are online. Updating online mapping services with the road layout is relatively easy, but adding street names takes detailed local knowledge. This can take years, and until it is complete, the maps are of limited use.

It is estimated that half the world's population lives in urban areas and that half of those have no street address. In many cases, people are fully connected to and accessible by the internet, but without an address, they are disconnected physically. They have

laptops, and mobile phones. They can, for example, order goods online, but are unable to receive deliveries at home and must use a delivery address, or a poste restante. Typically, if they know a package has been sent to them, they have to repeatedly travel to the nearest post office to check if it has arrived. In some cases, local mail, or packages are sent to a nearby school, so people have to go there and check. But this is usually unsafe since anyone can claim any mail. Businesses have similar problems with making and receiving deliveries, and delivery services need to allow for extra time to locate destinations. This effectively limits many businesses to operate only in their immediate locality.

Another challenge businesses face is in their supply chain management. They have no clear way of communicating where different goods, services, vendors, and clients are physically on their supply chain. In most informal economies, a single building can be occupied by 100s of people running different businesses making the use of street addresses useless.

There are cases where street addresses are not useful, even when they are available. For example, locations inside parks may be hundreds of meters away from the nearest address. Even for buildings that do have unique addresses, there is no practical way, for example, to specify the location of different entrances. This particularly affects large entities like factories, hospitals, and sporting facilities. Shopping malls also typically have a single address but may have several entrances in addition to each shop having its own location.

Many European countries have small villages where the streets have no official names and buildings are not numbered. There may also be buildings far from the nearest road, such as huts and refuges in mountains. In addition, street furniture (e.g., benches, vending machines, fountains) don't have street addresses but it would still be useful to be able to refer to their location.

How can a mobile business (e.g., restaurant on wheels) accurately communicate their business address for the day?

Three approaches are usually used to provide location in these circumstances. The most common solution is to provide simplified directions instead of an address. This results in addresses such as:

- 11th km of the Dusty Road from C to D.
- In front of the park entrance.
- Up the winding road, 600 meters from the bridge, Past lazy Dog, past English Bakery on the left at Hotel Mountain Dew.
- Using any other landmarks.

These directions rely on detailed local knowledge and are difficult or impossible to geocode algorithmically. If the directions cannot be followed, there is no alternative other than to ask around and hope to find a guide.

Where there are nearby street addresses, an option is to use the best available street address. This can lead to a building having an address of the named street, which could be some distance away.

Finally, latitude and longitude provide an exact location, and are used internally by GPS and satellite navigation devices, and are sometimes printed on paper maps. However, they are rarely seen on city or street maps and are difficult for people to use. To express a location to roughly 10-meter accuracy (five decimal places), latitude and longitude require between 15 and 20 text characters ("0.39122,9.45225" to "-43.95134,-176.55053"). This is around double the length of a typical telephone number.

Latitude and longitude express a point location, and there is no universally accepted way to provide a location of something that is not a point, such as a football field, park, or lake, other than by providing multiple points to enclose an area. Truncation of latitude and longitude doesn't make any real sense, since it just moves the location.

Cheap GPS devices have existed for at least 15 years, and yet latitude and longitude coordinates are still not widely used by people to specify locations. We think that this

shows latitude and longitude have too many disadvantages to be adopted for a street addressing solution.

A system of encoding location information into a short and easy-to-use code would solve these problems. As smartphones become cheaper and more widely used, they could provide a way to convert easily communicated identifiers to locations. Such codes could be used and exchanged over email, phone, in prints, and handwritten.

We believe that these codes need to meet the following requirements for widespread adoption:

1. Easy to use.
2. Complete.
3. Flexible precision.
4. Indicate proximity.
5. To prevent confusion, a place should have only one code.
6. Cultural independence.
7. Functions offline.
8. Easy to implement.

1.1 Easy to use

The codes must be short enough to be remembered and used. It should be easy for someone to communicate the code over the phone, in person, and in print. This means that they need to be shorter than the latitude and longitude.

1.2 Complete

The codes must have enough information on their own. Extra information (such as street number, locality, or country) could be helpful, but should not be required.

1.3 Flexible Size

Shortness is the key differentiating factor. More densely populated areas are designed with shorter codes. This idea, especially, yields very good results. Codes need only be accurate enough for human everyday use. If you're within a few meters of a destination, you're there. The location precision required for a sports field is less than that required to locate a utility meter. Locations should be expressed to an appropriate degree of precision. For example, the location of an apartment building may only need a precision of +/- 10 meters. But locating a smaller house may require a precision for different situations, and the precision of a code should be visually apparent.

1.4 Indicate Proximity

It should be possible to determine if two codes are near each other by looking at them. Ideally, it should also be possible to determine directions, and even to have a rough estimate of distance.

1.5 Functions Offline

The codes will be used in both built-up and rural areas, so must be able to be created and decoded without a data network. This also applies to users who are roaming or who live in areas where data networks are expensive.

2 Background

In this section, we provide a theoretical and mathematical foundation essential for comprehending and defining our Node-Gosper Space-Filling Curve (SFC). We draw a lot of inspiration from the works of Uher et al, 2019.

2.1 Hexagonal Grids

Regular hexagons, closely resembling circles, are an ideal shape for the regular tessellation of a plane and offer additional symmetrical advantages compared to squares. These properties contribute to several benefits:

2.1.1 Reduced Edge Effects

A hexagonal grid boasts the lowest perimeter-to-area ratio among all regular tessellations of the plane. In practical terms, this minimizes edge effects when working with hexagonal grids. This principle aligns with the construction of beehives from hexagonal honeycomb structures, as it minimizes the material required to create a lattice of cells with a specific volume.

2.1.2 Uniform Neighbors

Hexagonal grids provide a consistent arrangement of neighbors. Unlike square grids, which have two classes of neighbors (cardinal directions sharing an edge and diagonal directions sharing a vertex), hexagonal grid cells have six identical neighboring cells, each sharing one of the six equal-length sides. Additionally, the distance between centroids is consistent for all neighbors.

2.1.3 Adaptation to Curved Surfaces

When dealing with large areas where the curvature of the Earth becomes significant, hexagons offer a better fit to this curvature compared to squares. This is exemplified by the use of hexagonal panels in the construction of soccer balls.

2.1.4 Visual Appeal

It's undeniable that hexagonal grids possess an impressive and visually striking appearance, making them a popular choice over square grids.

2.2 Space-Filling Curves (SFC)

Space-Filling Curves (SFCs) constitute a family of point hashing algorithms that establish fundamental principles for linearizing space [1]. These principles play a pivotal role in the implementation of the Node-Gosper SFC (by uher et al 2019). SFCs transform n-dimensional points into one-dimensional (1D) indices by localizing them within multi-resolution grids . For each point, a set of hierarchical indices is computed to address the sub-clusters containing the hashed point. The resulting hash code is typically a bit string assembled from the indices in a top-down fashion, with the root index occupying the most significant bits. Sorting points by these codes constructs an SFC, aligning points from the same cluster (with identical hashes) in memory. The order of clusters in an SFC reflects a depth-first traversal of the corresponding tree (Figure 1), resulting in nearby points being stored close to each other in memory. However, it's important to note that some points may appear close on the curve but are actually distant in reality (see Figure 5a). Various shapes and indexing methods exist for clusters, leading to the creation of numerous different curves. Notable SFCs include Z-order, Hilbert, and Sierpinski space-filling curves [1, 2–4].

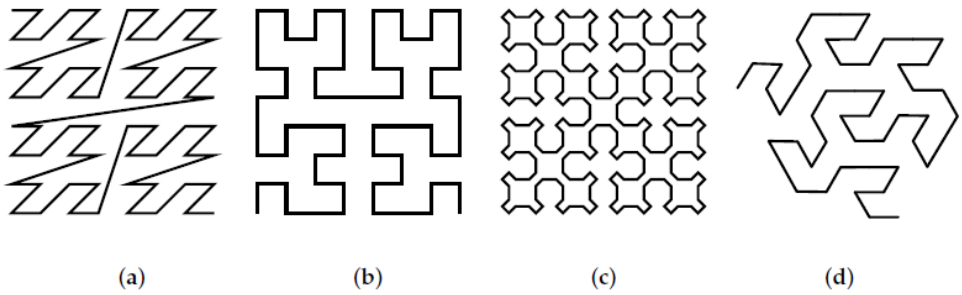

Figure 1: The examples of space-filling curves: (a) Z-order, (b) Hilbert, (c) Sierpi ´ nski, (d) Gosper.
Image source: [Uher V., Gajdoš P., and Snášel V. 2017]

2.2.1 Space-Filling Curves Criteria

SFCs vary in their properties, making it challenging to find a single SFC that efficiently suits all types of applications. Nevertheless, there are several criteria that should be satisfied to define a high-quality SFC [1], with conditions 1–3 being mandatory, followed by quality criteria:

1. Mapping between a point and an SFC code must be bijective, ensuring that each point is assigned a unique hash code that defines its position along the SFC. This bijective mapping should be possible at greater depths of hierarchy.
2. The curve must be space-filling, implying that every point in space should lie along the curve.
3. Total ordering must be maintained, ensuring that each cluster is visited by an SFC exactly once.
4. Mapping between points and codes should be computationally straightforward.
5. Points that are close along the curve should also be close in multidimensional space.
6. Sequentialization of a contiguous data set should result in a contiguous sequence of code numbers, facilitating efficient memory addressing.

2.3 Hexagonal Curve Selection

2.3.1 Properties of the Gosper Curve

The Gosper curve, a hierarchical curve based on the hexagonal grid, possesses several remarkable properties [10, 11, 12]. It satisfies the criteria of good SFCs, including continuity, uniformity, and scale invariance. Every point in space is visited exactly once, with uniform distances between adjacent clusters along the curve.

Hexagons (or Gosper islands) share edges with six identical shapes, facilitating straightforward addressing using the same coordinate system as regular hexagons. The entire dataset lies on the curve, and the point cloud can be contained within the inscribed circle of the Gosper island. The mapping remains bijective for sufficiently deep hierarchies, with the complexity of indexation and mapping depending on the specific algorithm.

The Gosper curve also excels in theoretical metrics. The worst-case locality value (WL), where smaller values are better, compares favorably with non-hexagonal SFCs such as Hilbert and Sierpi ´ nski curves. Additionally, the Arrwwid number, where smaller values are better, indicates the Gosper curve's suitability for circular neighborhood querying, outperforming other SFCs.

These properties make the Gosper curve a valuable choice for various spatial applications.

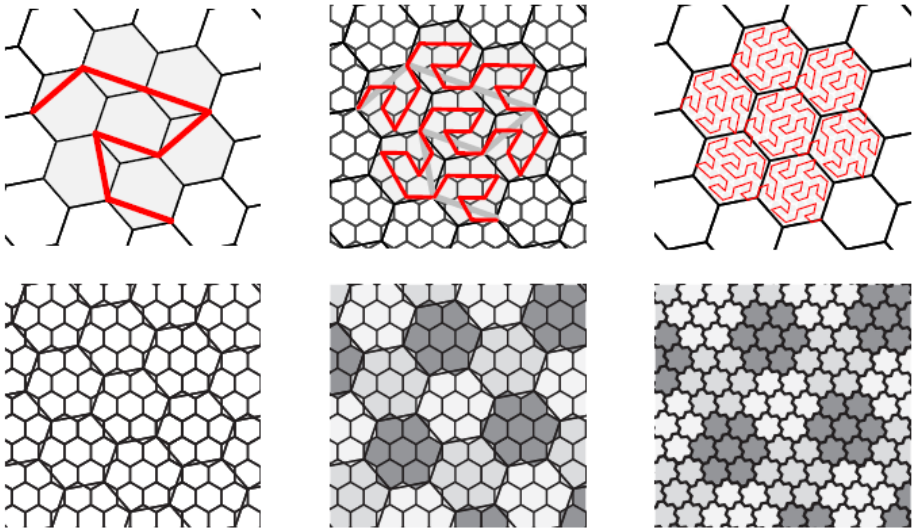

Figure 2: Row 1: Three iterations of the Gosper curve. Row 2: Tiling with Gosper islands. Seven joined hexagons represent the basic level-1 island.
Image source: [Uher V., Gajdoš P., and Snášel V. 2017]

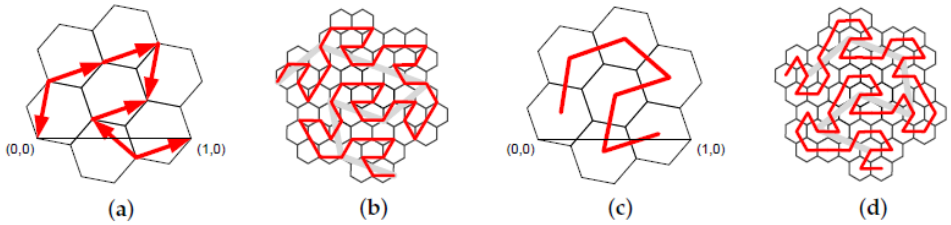

Figure 3: The basic patterns of the Gosper curve: (a) 7-pattern of vertex representation, (b) the second iteration of vertex representation, (c) 7-pattern of node representation, (d) the second iteration of node representation.

2.4 Node-Gosper Indexation

The Node-Gosper Space-Filling Curve (SFC) draws inspiration from the Gosper fractal drawing algorithm, which we will briefly revisit here for context. The Gosper fractal is built upon a pattern of seven hexagons, as depicted in Figure 3a. The fractal's recursive construction involves replacing the oriented red arrows with properly rotated and scaled patterns in a top-down manner, as shown in Figure 3b. These arrows connect the vertices of the hexagonal grid, ensuring the geometrical continuity of the patterns and producing an accurate representation of the fractal.

Alternatively, the same procedure can be applied to the node-based pattern, as illustrated in Figure 3c. It is particularly well-suited for point indexing purposes compared to the vertex-based approach.

To define the Node-Gosper SFC, it is essential to understand the precise mathematical properties of the Gosper fractal, which we will briefly recap here. For more in-depth information, please refer to papers [7,8].

Figure 4 describes the transformation of the pattern replacing the base hexagon inscribed in the unit circle of size $r = 1$ (black dashes) by seven smaller hexagons (level-1 Gosper island). A hexagon itself practically represents a level-0 Gosper island. The fig-

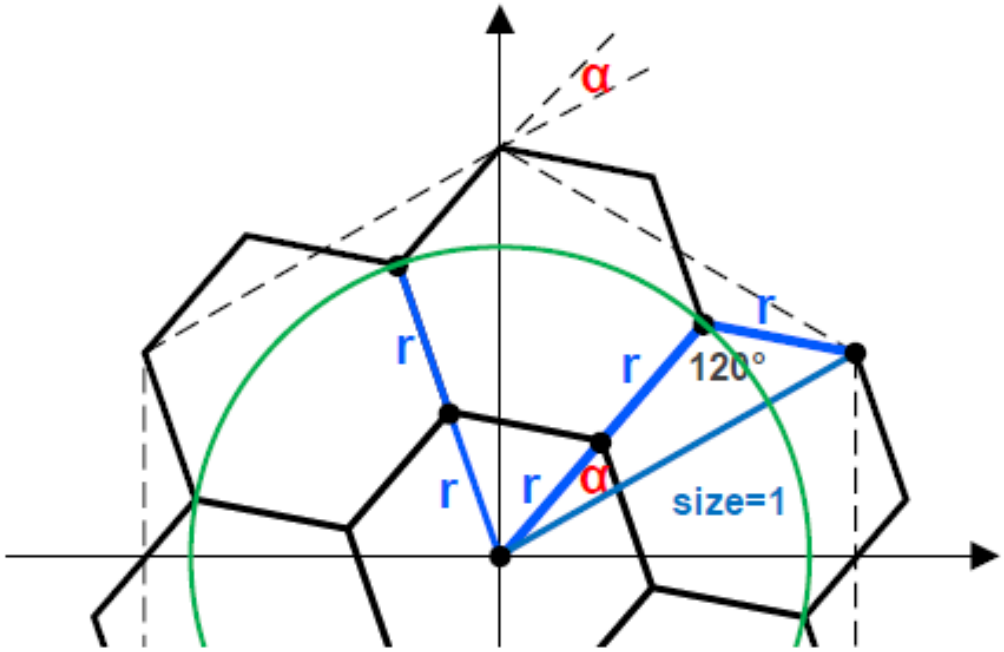

Figure 4: Grid transformation for Gosper islands construction.
Image source: [Uher V., Gajdoš P., and Snášel V. 2017]

ure also shows two features of the Gosper islands: The Gosper island preserves the area of the parent hexagon, but its perimeter limit is infinity. The scale factor $r = \frac{1}{\sqrt{7}}$ can be computed according to the blue triangle (Figure 8) using the Law of Cosine and it also represents the size of child hexagons. The angle $\alpha = \arcsin\left(\frac{\sqrt{3}}{2\sqrt{7}}\right)$ can be computed using the Law of Sine and the scale factor r. Figure 7a shows that the patterns replacing differently oriented arrows have to be additionally rotated. The angles of orientation are defined by function $f(y)$ in (1). Going through the basic Gosper pattern (Figure 3a) from the first vertex $(0, 0)$ to the last vertex $(1, 0)$, the hexagons 0 and 3 are rotated by $-120°$ and the hexagon 5 by $120°$. The transformations can be easily computed using the well-known matrices for rotation (2), scale and translation.

$$f(y) = \begin{cases} -120° & \text{if } y \in \{0, 3\} \\ +120° & \text{if } y = 5 \\ 0° & \text{else} \end{cases}$$

$$R_y = \begin{bmatrix} \cos\gamma & -\sin\gamma \\ \sin\gamma & \cos\gamma \end{bmatrix}$$

3 Proposed Methodology

3.1 Localization with Node-Gosper Curves

Consider the scenario where the foundational hexagon has a size set to 1, as depicted in Figure 11. This hexagon is iteratively replaced by Node-Gosper patterns, referenced in Figure 9, ascending up to level n. For a vector of indices denoted as $\mathbf{k} = (k_1, k_2, \ldots, k_{n-1}, k_n)$ that specifies a hexagon within a level-n Gosper island, the algorithm calculates the centroid of said hexagon. This procedure embodies the reverse mapping process and serves to confirm the bijective nature of the Gosper mapping. The

The recursive approach outlined is primarily utilized for rendering a graphical representation of the Gosper fractal, ensuring geometric coherence. However, the indexing framework is required to explicitly delineate the sequence of clusters along the space-filling curve (SFC). Every point, denoted as $\tilde{p}$, within the point cloud is pinpointed on the hexagonal lattice, and the pertinent hexagon is allocated a binary hash code that signifies its sequence number. This discussion introduces our hexagon indexing scheme, arranged according to the Node-Gosper SFC.

Figure 9 illustrates the fundamental designs that determine both rotation and indexing. A critical aspect is that the subsidiary patterns manifest divergent indexing sequences, which hinge on the indexing configuration of the overarching pattern. For the purposes of indexing, merely maintaining geometric continuity does not suffice.

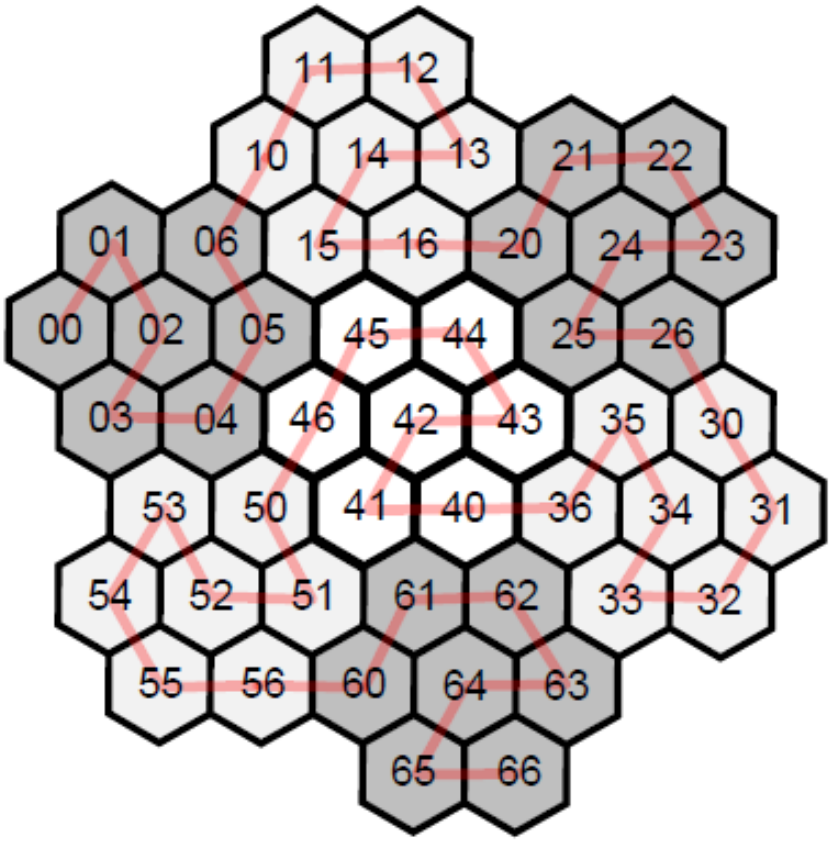

Figure 5: Indexation of the level-2 Node-Gosper curve

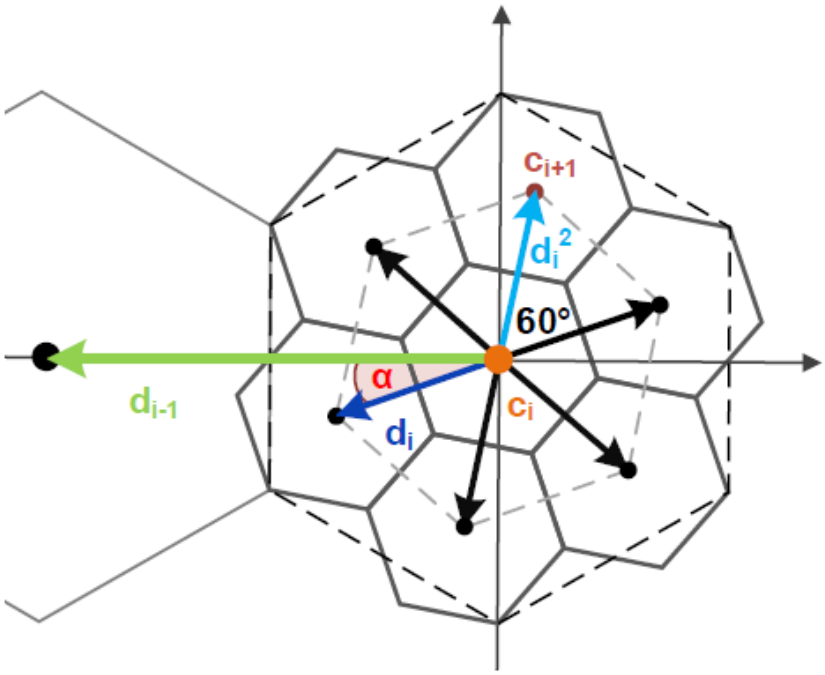

Figure 6: The transformation of directional di and position ci vectors.

subordinate patterns can be indexed in the forward (F) or backward (B) manner (Figure 9), which affects the addressing order of the hexagons by $\mathbf{k}$. A pattern passing order F/B of the i-th recursion can be determined depending on $\mathbf{k}$ as follows:

$$Order(i) = \begin{cases} F & \text{if } i = 1 \\ B & \text{if } i > 1 \text{ and } Order(i-1) = F \text{ and } k_{i-1} \in \{0, 4, 5\} \\ B & \text{if } i > 1 \text{ and } Order(i-1) = B \text{ and } k_{i-1} \notin \{1, 2, 6\} \\ F & \text{otherwise} \end{cases} \tag{1}$$

Each k_i index has to be recalculated to index $k'_i \in \mathbb{N}$ in the F order (Figure 9) to unify addressing of the corresponding Gosper islands in the top-down manner. The calculation of $\mathbf{k}' = (k'_1, k'_2, \ldots, k'_{n-1}, k'_n)$ is defined as follows:

$$k'_i = \begin{cases} 6 - k_i & \text{if } Order(i) = B \\ k_i & \text{otherwise} \end{cases} \tag{2}$$

$\tilde{\mathbf{k}}_\mathbf{0}$ is utilized to calculate the centroid of the targeted hexagon. The algorithm iteratively modifies two vectors: the position vector $\tilde{\mathbf{c}}_\mathbf{i}$ and the directional vector $\tilde{\mathbf{d}}_\mathbf{i}$ (refer to Figure 11), where i signifies the current recursion depth. The vector $\tilde{\mathbf{c}}_\mathbf{i}$ denotes the initial center location of the current Gosper island, whereas $\tilde{\mathbf{d}}_\mathbf{i}$ characterizes the rotation and scaling applied to the grid at the i-th recursion stage. The initial directional vector is aimed at the adjacent hexagon and is given by $\tilde{\mathbf{d}}_\mathbf{0} = (w, 0)$, with w representing the breadth of the base hexagon. The vector $\tilde{\mathbf{d}}_\mathbf{0}$ establishes the primary orientation and spacing between the centers of two neighboring hexagons before undergoing the initial transformation.

3.2 Node-Gosper Point Mapping

In the preceding section, the algorithm for determining the center of a hexagon based on $\tilde{\mathbf{k}}$ was outlined. We now introduce the converse process: mapping a point onto the Node-Gosper curve, whereby $\tilde{\mathbf{k}}$ is ascertained and the corresponding hash code is produced.

The projection onto the Node-Gosper curve must be executed in a bottom-up fashion, as hexagons are not capable of being subdivided into smaller hexagons (as illustrated in Figure 12). As elucidated in prior sections, the hexagonal 7-pattern, also known as a Gosper island, undergoes rotation and scaling to substitute a larger, parent hexagon, thereby establishing the essential hierarchical structure of hexagons. The emergent Gosper islands are non-convex fractal shapes, necessitating precise alignment with adjacent islands to completely tessellate the plane. The top-down approach for point localization is flawed due to the fact that the irregular boundaries of islands extend beyond the confines of the parent hexagons. Consequently, a point situated near the edges may be ascribed to a different branch of the hierarchy in subsequent iterations (as demonstrated in Figure 12). An essential aspect of the hexagonal hierarchy is that the centers of the seven hexagons comprising each Node-Gosper pattern invariably lie within the bounds of their respective parent hexagon. This indicates that the process of localizing hexagon centers in an upward direction constitutes a numerically stable reverse operation to the top-down method, ensuring accurate hierarchical categorization in spite of the irregular boundaries. However, several challenges remain to be addressed:

1. The relative positioning of hexagons between two consecutive grids at levels i and $i-1$ is not immediately apparent, since the full extent of the parental hierarchy remains undefined at level i (refer to Figure 12).

2. The rotation of the grid differs across various hierarchical branches. This particular transformation of the grid is recursive in nature and remains undetermined at level i (see Section 3.2).

3. Correspondingly, the indexation and sequence of traversal are not discernible at level i for analogous reasons.

In order to preserve the integrity of hierarchical localization while adhering to the recursive nature of the Gosper fractal, the hashing algorithm is bifurcated into two distinct phases. Initially, a point $\tilde{\mathbf{p}}$ is localized within the multi-resolution hexagonal grid employing a bottom-up strategy, utilizing inverse calculations pertinent to the Gosper fractal's construction (as illustrated in Figure 12). This hierarchical localization yields a vector of indices which demarcate the relative positioning of a point $\tilde{\mathbf{p}}$ within the grid. This phase ensures accurate placement within the Gosper islands' fractal contours, and the methodology is expounded upon in Section 3.4.1. Owing to the recursive definition of the Gosper fractal, the explicit pattern transformations and indexations for the disparate recursion branches must be determined in a top-down fashion. Consequently, the indices procured from the initial phase necessitate subsequent top-down recursive recalibration (refer to Section 3.4.2). The latter phase of the algorithm furnishes the precise indices requisite for the computation of the ultimate hash code, which establishes the sequential positioning of the point along the Node-Gosper SFC.

4 Node-Gosper Curve Application

5 Conclusion

Recap the benefits of the proposed addressing system and its potential impact on service delivery, e-commerce, navigation, and emergency response in regions lacking conventional addresses.

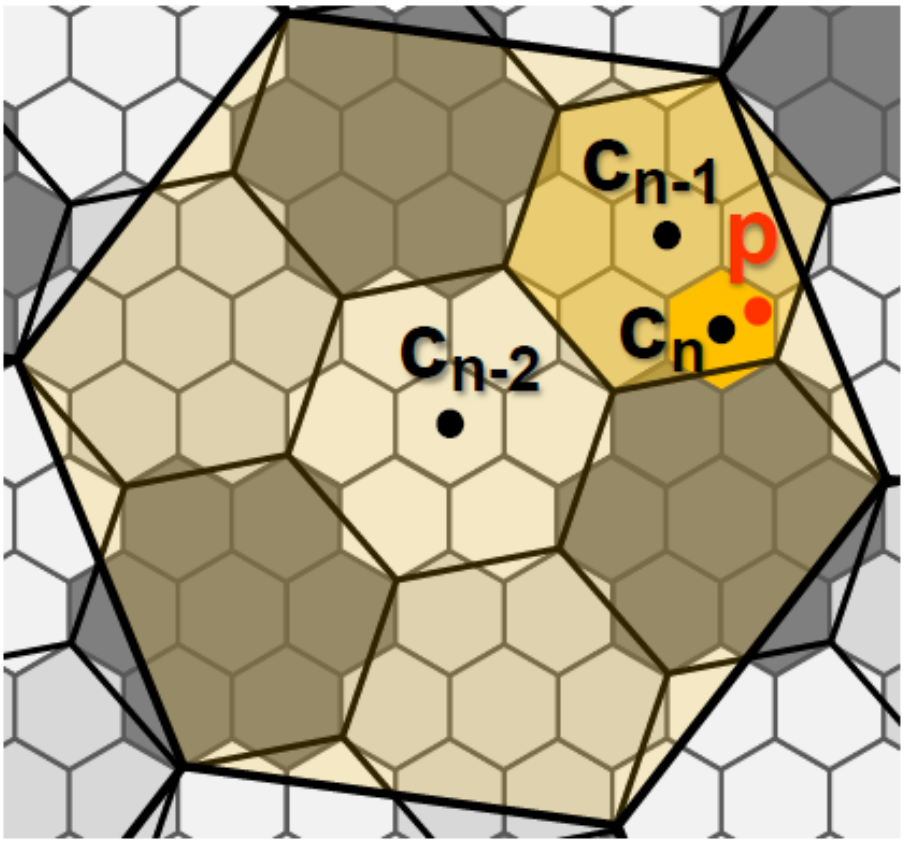

Figure 7: The localization of p in level-n Gosper island and the search of the corresponding centers

Algorithm 1 Hierarchical point mapping

Require:

$\tilde{\mathbf{v}} = (x, y, z)$: cube coordinates of input point $\tilde{\mathbf{p}}$

$B(\tilde{d})$: transfers relative coordinates $\tilde{d}$ to index b_i of the default pattern (16)

T: transformation matrix (13)

n: number of hierarchical levels

Ensure:

returns the indices $\tilde{\mathbf{k}} = (k_0, \ldots, k_n)$ of a point $\tilde{\mathbf{p}}$

coordinate transformation from cube $\tilde{\mathbf{v}}$ to axial $\tilde{\mathbf{a}}$

$\tilde{q} \leftarrow \tilde{v}_x$, $\tilde{l} \leftarrow \tilde{v}_z$

for all levels

for $i \leftarrow 1$ to n **do**

    transformation of axial coordinates $\tilde{\mathbf{a}}$

    $\tilde{\mathbf{a}} \leftarrow T \cdot (\tilde{\mathbf{a}})^T$

    coordinate transformation from axial $\tilde{\mathbf{a}}$ to cube $\tilde{\mathbf{v}}$

    $\tilde{v}_x \leftarrow \tilde{q}$, $\tilde{v}_z \leftarrow \tilde{l}$, $\tilde{v}_y \leftarrow -\tilde{v}_x - \tilde{v}_z$

    $\tilde{q} \leftarrow (\mathrm{INT})\mathrm{round}(\tilde{a}_q)$, $\tilde{l} \leftarrow (\mathrm{INT})\mathrm{round}(\tilde{a}_l)$

    decimal part $\tilde{d}$ computation

    $\tilde{d}_x \leftarrow \tilde{v}_x - (\mathrm{INT})\mathrm{round}(\tilde{v}_x)$

    $\tilde{d}_y \leftarrow \tilde{v}_y - (\mathrm{INT})\mathrm{round}(\tilde{v}_y)$

    $\tilde{d}_z \leftarrow \tilde{v}_z - (\mathrm{INT})\mathrm{round}(\tilde{v}_z)$

    transfer to final index k_i

    $k_{n-i} \leftarrow B(\tilde{d})$

end for

Algorithm 2 Node-Gosper index computation

Require:

$\mathbf{b}$: vector of default indices of input point $\tilde{\mathbf{p}}$
$E(t, b_i)$: index transfer function defined in Table 1
$order = F$: passage order
$rotDir = 0$: rotation of the pattern ($rotDir \in \{-1, 0, 1\}$)
n: number of hierarchical levels

Ensure:

returns the indices $\mathbf{k} = (k_1, \ldots, k_n)$ of a point $\tilde{\mathbf{p}}$

for all levels **do**
  for $i \leftarrow 1$ to n **do**
    if $b_i \neq 0$ and $rotDir \neq 0$ **then**
      $b_i \leftarrow b_i + 2 \cdot rotDir$
    end if
    if $b_i < 1$ **then**
      $b_i \leftarrow b_i + 6$
    else if $b_i > 6$ **then**
      $b_i \leftarrow b_i - 6$
    end if
    $k_i^0 \leftarrow E(P, b_i)$
    if $k_i^0 \in \{0, 3\}$ and $(-rotDir < -1)$ **then**
      $rotDir \leftarrow 1$
    else if $k_i^0 = 5$ and $(+rotDir > 1)$ **then**
      $rotDir \leftarrow -1$
    end if
    if $order = B$ **then**
      $k_i \leftarrow 6 - k_i^0$
    else
      $k_i \leftarrow k_i^0$
    end if
    if $k_i \in \{0, 4, 5\}$ **then**
      $order \leftarrow (order = B \,?\, F : B)$
    end if
  end for
end for