Digital Straight Line Segment Recognition on Non-Standard Point Lattices

Kelly Hubble

Examensarbete i matematik, 30 hp
Handledare: Robin Strand
Examinator: Andreas Strömbergsson

Mars 2010

Department of Mathematics
Uppsala University
Digital Straight Line Segment Recognition on Non-Standard Point Lattices

Kelly Hubble

March 12, 2010

Abstract

OC-DSSr is a digital straight line segment (DSS) recognition method in 3D non-Cartesian point lattices. A brief overview of image analysis is given, along with its relationship to digital geometry and non-Cartesian cubic lattices. The Body-Centered Cubic (BCC) and Face-Centered Cubic (FCC) lattices are reviewed. A digitization method-dependent definition of a DSS is used to develop OC-DSSr. The supercover digitization is used to digitize curves on non-Cartesian lattices. An extension to the supercover is proposed to achieve $\alpha$-connectivity in non-Cartesian lattices. A new independent definition of a DSS is proposed, based on the presented recognition method.
Acknowledgments

A special thanks to my advisor, associate professor Dr. Robin Strand, for allowing me the time to explore my many ‘interesting’ ideas and for the time to get married during the course of the project.

For his valuable input and for permitting the project, I am grateful to my examiner, associate professor of mathematics Dr. Andreas Strömbergsson.

For making the project possible, my sincere gratitude to professor emeritus of mathematics Dr. Christer Kiselman.

For financial support and encouragement, I thank my wife, Linnea.
# Contents

1 Introduction  
1.1 Image Analysis ......................... 1  
1.2 Basic Definitions ..................... 2  
1.3 3D Non-Cartesian Lattices .............. 3  
1.4 Motivation ............................. 5  
1.5 The Problem ............................ 6

2 Digital Straight Lines  
2.1 Connections ............................ 6  
2.2 Digital Objects ....................... 8

3 Survey of Recognition Methods in $\mathbb{Z}^2$  
3.1 Chord Property ........................ 8  
3.2 Diophantine Inequalities ............... 9  
3.3 Chain-Codes, Words, and Pattern Recognition ..... 10

4 Digitization  
4.1 Bresenham’s algorithm .................. 12  
4.2 Generalization of Bresenham’s algorithm .... 13  
4.3 Cover and Supercovers ................. 14  
4.4 The $\alpha$-supercovers .................. 16

5 Orthogonal Complement DSS Recognition  
5.1 Input .................................. 19  
5.2 Description ............................ 19

6 Implications  
6.1 Image of a line .......................... 20  
6.2 Rewording .............................. 21  
6.3 Conclusion .............................. 22
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A simple digital image</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Overview of Image Analysis</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Truncated Octahedron</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>Rhombic Dodecahedron</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>Chain-Code Encoding</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>Illustration of the Bresenham’s algorithm</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>Examples of Covers in 2D</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>Supercovet</td>
<td>16</td>
</tr>
<tr>
<td>9</td>
<td>$\alpha$-Supercovet</td>
<td>18</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Image Analysis

The digital age has brought the world copious digital images along with the tools to obtain information from these images. Coming from a mathematical background, one will want some explanation of these concepts.

Digital Image

A digital image is a sample of a continuous or real world object into a discrete set.

Example 1.1. Taking a picture with the now common digital camera is sampling the 3D world into a finite subset of \( \mathbb{Z}^2 \). The elements of the subset are called pixels. The number of pixels can range from one to millions.

Example 1.2. Figure 1 shows the capturing a digital image of a line in \( \mathbb{R}^2 \) into a low resolution digital image. The image in Figure 1c can be stored as an \( 8 \times 8 \) matrix of ones and zeros.

\[
I = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 
\end{bmatrix}
\]

The convention is to label the upper left hand corner as the point (0,0). The object, the digital image of the line, is the set \( L = \{(0,7), (0,8), (1,6), (1,7), (2,5), (2,6), (3,4), (3,5), (4,3), (4,4), (5,2), (5,3), (6,1), (6,2), (7,0), (7,1), (8,0)\} \).

The production of digital images goes beyond 2D digital photography. In the medical field, Computed Tomography scans and Magnetic Resonance Imaging are sampled in 3D. On a much smaller scale, electron microscopy such as scanning electron microscopes and transmission
electron microscopes sample images on the scale of nanometers. Electron microscopes capture images of chemical molecules, individual or groups of cells, organelles, and viruses. Another direction is to store more information at each pixel, such as color or density. For example, a satellite image of a city could store at each pixel the color, distance from the satellite and amount of heat present. These three types of data are thought of as three different images or just one with the combined information, depending on how the image will be used.

**Image Analysis**

The act of gathering information about the object or scene in an image is called image analysis. For example, obtaining statistical information about a scene, such as counting objects. Given a binary image as in Example 1.2, image analysis techniques exist to determine if the object $L \subset I$ is a digital image of a line. Image analysis techniques take advantage of algorithms from machine learning and other high level statistical methods. For example, face finding and recognition algorithms in photographs use these methods.

Figure 2 is an overview of the image analysis model used in this report.

Image acquisition is the method of sampling. Image acquisition is an active area of study that addresses the question how to best sample? Once the image is acquired it needs to be prepared for analysis. Preparation is done in the image processing and segmentation steps of Figure 2. This preparation will be particular to the type of algorithms being used in the final step. Image processing is preparing an image either to be analyzed or to be viewed by the human eye. It should be noted that this is distinct from image analysis. Image processing is not a simple problem, and is a field of study on its own. Different analysis techniques and image types require unique image processing. Image processing includes segmentation, which is given as a separate step because of its complexity. Segmentation is the separation of the image into unique components.

The final step, and focus of this work, is object recognition. Recognition of an object in an image is done by determining if an object meets a set of criteria. Thus the development of an object recognition algorithm has two parts: determining the appropriate criteria and developing a way to test if an object meets the criteria.

1.2 Basic Definitions

**Lattices**

Let $v_1, v_2, \ldots, v_n$ be independent vectors in $\mathbb{R}^n$. A point lattice with basis $\{v_1, v_2, \ldots, v_n\}$ is the set of all linear combinations of the $v_i$ with integer coefficients. The standard notation is

$$\Lambda = \Lambda (v_1, v_2, \ldots, v_n) = \left\{ \sum_{i=1}^{n} \alpha_i v_i : \alpha_i \in \mathbb{Z} \right\} [9].$$

(1)
Example 1.3. A familiar example of a lattice is the set of \( \mathbb{Z}^2 \subset \mathbb{R}^2 \)

\[
\Lambda \left( \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) = \left\{ \alpha_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \alpha_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix} : \alpha_i \in \mathbb{Z} \right\}.
\]  

In image analysis, the term point lattice is rarely used and is replaced with grid [12]. In this report the terms point lattice, lattice, and grid are equivalent to the above definition of point lattice. Both lattice and grid are used to describe more general objects, but are found in place of point lattice in mathematical and image analysis literature.

Pixels

The word pixel is the concatenation of the words picture and element. From this concatenation the definition pixel falls out directly. A pixel is the smallest space filling element centered at each sample point of a digital image. In other terms, pixels are the Voronoi cells at each lattice point. The Voronoi cell of a \( n \)-dimensional lattice point \( g \in \Lambda \) is

\[
\{ x \in \mathbb{R}^n : d(x,g) \leq d(x,g') \ \forall g' \in \Lambda \setminus \{g\} \}.
\]

Pixels of digital images on point lattices are consistent in shape. In 2D images the pixels are most commonly squares or hexagons. Figure 1 is an example using square pixels. The cube is the most common pixel in 3D. Three dimensional pixels are often called voxels, for volume elements. Voxels, pixels or more generally Voronoi cells are categorized into two groups, Cartesian and non-Cartesian. The former is exclusive to \( \mathbb{Z}^n \), for example \( \mathbb{Z}^2 \) and \( \mathbb{Z}^3 \) with square and cubic Voronoi cells. The majority of research in image analysis is done on Cartesian lattices. Non-Cartesian pixels are all other possible shapes. A lattice with hexagonal pixels is a 2D non-Cartesian example. Voxels in the shape of truncated octahedra or rhombic dodecahedra are 3D examples of non-Cartesian lattices.

1.3 3D Non-Cartesian Lattices

Body-Centered Cubic lattices

The lattice generated by the basis

\[
\{(1,-1,-1),(-1,1,-1),(-1,-1,1)\}
\]

is called the Body-Centered Cubic (BCC) lattice. The BCC lattice is visualized as two interleaved Cartesian lattices such that one of the lattices has its points in the center of eight points of the other lattice [5]. The BCC lattice points are expressed as a subset of \( \mathbb{Z}^3 \) in \( B \)

\[
B = \{(x,y,z) \in \mathbb{Z}^3 \mid x \equiv y \equiv z \mod 2\}.
\]

Example 1.4. \( b_0 = (0,0,0) \in B, b_1 = (3,3,3) \in B, \) but \( (2,4,5) \notin B \).

It follows from Equation 5 that the B is a quotient group of order four [5]. This leads to the BCC lattice having a density of \( 1/4 \) in \( \mathbb{Z}^3 \) and further that the volume of the Voronoi cells at each point is 4 [5]. The Voronoi cells of a BCC lattice are truncated octahedra, Figure 3, which have 14 faces, 36 edges, and 24 vertices. Of these faces six are square and eight are hexagonal, and neighboring Voronoi cells are connected by these faces.
Figure 3: Truncated octahedron is the BCC voxel.

Figure 4: Rhombic dodecahedron is the FCC voxel.
Face-Centered Cubic lattices

The Face-Centered Cubic (FCC) lattice is a lattice generated by the basis
\[ \{(1,1,0), (1,0,1), (0,1,1)\}. \] (6)

The lattice points of the FCC lattice are obtained by selecting the points that at the corners of a cube, and then adding a point in the center of each face of the cube [5]. The FCC lattice is also known as the D3 checkerboard lattice [5]. The points of the FCC lattice are presented as a subset of \( \mathbb{Z}^3 \)
\[ F = \{(x,y,z) \in \mathbb{Z}^3 \mid x + y + z \equiv 0 \mod 2\}. \] (7)

**Example 1.5.** \( f_0 = (0,0,0) \in F, f_1 = (2,4,6) \in F, \) but \( (3,3,3) \notin F\).

The set \( F \) has a density of 1/2 in \( \mathbb{Z}^3 \) [5]. The Voronoi cell at each point have volume 2 and the shape of a rhombic dodecahedron, Figure 4. Rhombic dodecahedra have 12 rhombic faces, 24 edges, and 14 vertices, and its neighbors share faces and six vertices [11].

Digital Geometry

Digital geometry is a subfield of discrete geometry. In their book titled Digital Geometry [6], Klette and Rosenfeld define digital geometry as follows.

The study of geometric or topologic properties of sets of pixels or voxels. It often attempts to obtain quantitative information about objects by analyzing digitized (2D or 3D) pictures in which the objects are represented by such sets.

This definition likens digital geometry directly to image analysis described above.

1.4 Motivation

Digital Geometry Paradigms

The concept of discrete geometry is to use the tools and definitions in geometry to answer questions about discrete objects. In digital geometry those questions are answered in two ways. One paradigm is using Euclidean geometric objects to study digital geometry. One example is bounding a collection of lattice points with parallel Euclidean lines to define a line in the point lattice, as done in the works [4, 2]. The other “approach is to develop methods [on a point lattice] without resorting to the Euclidean geometry” [12]. This paradigm creates foundations in the structure of the point lattices and builds on the foundations. The current report uses the latter paradigm to build the foundations of what a line is in a point lattice, but does not ignore the Euclidean geometric approach.

Geometric Advantages

The *sphere packing problem* within discrete geometry is one reason to explore alternatives to the Cartesian lattices [5]. A reason for studying non-Cartesian point lattices is that they are the optimal sphere packing lattices. The FCC lattice is the optimal packing of spheres in 3D, with a packing density of 74.05% compared to the packing density of the Cartesian lattice 52.3% [5]. Another geometric advantage of the BCC and FCC lattices is that they have more symmetries than the Cartesian lattices [11].
Sampling Theory

The BCC lattice has been shown (by the Shannon theorem) to be the optimal sampling lattice in 3D and the FCC lattice also performs well [11]. They are both recommended as the optimal point lattices for Mathematical Morphology processing [11].

However with all these benefits, using non-Cartesian point lattices is still rejected [11]. Commonly this rejection occurs because of an increase in algorithmic complexity, but this may only be because of the lack of deep work into the fundamentals of these lattices [11].

1.5 The Problem

One of the fundamental problems in image analysis is recognizing lines in a point lattice. The goal of this work is to develop a method for detecting lines in the FCC and BCC cases. Specifically the problem is to determine if a given digital arc is a digital straight line segment, which is object recognition as described above. This report will achieve two objectives. First, the appropriate criteria for a line in the FCC and BCC lattices is determined. Second, the report will use the criteria to develop a computational method. The novel method derived from these two steps is proposed as OC-DSSr.

The definitions and methods in this report are presented in an arbitrary point lattice whenever possible. A definition of a line in a point lattice is presented historically to achieve the first objective of the report. A consequence of the development of OC-DSSr is a new refined definition of a line in a point lattice. An extension of the supercover digitization method is proposed, called the $\alpha$-supercover. This extension is necessary to achieve proper connectivity in digitization of lines in non-Cartesian lattices.

2 Digital Straight Lines

The definition of a line in $\mathbb{R}^n$ is clear, but in digital geometry there is less consensus. The concept of a digital line must be taken directly from the Euclidean line. Klette and Rosenfeld do this with the statement: "A digital arc is called ‘straight’ if it’s the digitization of a straight line segment" [7]. Before making a precise definition of a digital straight line segment, DSS, some concepts need to be defined.

2.1 Connections

Lines are continuous objects in $\mathbb{R}^3$, and the natural extension of continuity in a lattice is connectivity. Lattice points are connected if they are neighbors. Let $\Lambda$ be a point lattice.

**Definition 1.** $g_1, g_2 \in \Lambda$ are neighbors if $g_1 \cap g_2 \neq \emptyset$.

When a lattice point $g$ is treated as a set, it represents the set of points in the embedded space that lay within the Voronoi cell at $g$. Definition 1 is general and will be restricted to meet the needs of image analysis. For the following, let $dx = |x - x'|$, $dy = |y - y'|$, and $dz = |z - z'|$. 
Cartesian Neighbors

All of the neighbors of a point in $\mathbb{Z}^3$ will make a collection of face, edge, and vertex neighbors that add up to 26. Two points $(x, y, z)$ and $(x', y', z')$ are 26-neighbors in $\mathbb{Z}^3$ if

$$\max\{|dx|, |dy|, |dz|\} = 1.$$  

(8)

Two points $(x, y, z)$ and $(x', y', z')$ are 18(6)-neighbors in $\mathbb{Z}^3$ if Equation 8 holds and

$$dx + dy + dz \leq c$$

(9)

with $c = 2$ for 18-neighbors and $c = 1$ for 6-neighbors. The 18-neighbors of a point in $\mathbb{Z}^3$ are the same as the 26-neighbors minus the eight vertex neighbors. Further, the 6-neighbors of a point are the face neighbors. These neighbors are also defined by a set of vectors, known as connection vectors, but are not defined here for the Cartesian case. Subsets of the connection vectors form a basis for the point lattices.

Non-Cartesian Neighbors

The BCC lattice has two possible connection schemes; 8-connected where only hexagonal face neighbors are considered, and 14-connected, which includes hexagonal and square face neighbors. Two points $b = (x, y, z)$ and $b' = (x', y', z')$ are 14-neighbors in $B$ if

$$dx + dy + dz \leq 3$$

(10)

and $b, b'$ are 8-neighbors in $B$ when only equality holds. These connections are represented by a set of connection vectors too. For 8-neighbors

$$T_{B8} = \{±(1, 1, 1), ±(1, 1, -1), ±(1, -1, -1), ±(1, -1, 1)\},$$

(11)

and for 14-neighbors

$$T_{B14} = T_{B8} \cup \{±(2, 0, 0), ±(0, 2, 0), ±(0, 0, 2)\}.$$ 

(12)

There are two connection types for the FCC point lattice; 12-connected where only face neighbors are considered and 18-connected where the face neighbors and six vertex neighbors are allowed. Two points $f = (x, y, z)$ and $f' = (x', y', z')$ are 18-neighbors in $F$ if

$$dx + dy + dz = 2,$$

(13)

and $f, f'$ are 12-neighbors with the added condition

$$\max\{|dx|, |dy|, |dz|\} = 1.$$ 

(14)

The sets of connection vectors for FCC are, for 12-neighbors

$$T_{F12} = \{±(1, 1, 0), ±(1, -1, 0), ±(0, 1, 1), ±(0, 1, -1), ±(1, 0, 1), ±(1, 0, -1)\},$$

(15)

and for 18-neighbors

$$T_{F18} = T_{F12} \cup \{±(2, 0, 0), ±(0, 2, 0), ±(0, 0, 2)\}.$$ 

(16)
2.2 Digital Objects

The objects of interest are thin connected components of digital images. The next two definitions clarify thin connected components.

**Definition 2.** A simple digital curve is an infinite subset, $C$, of a lattice such that $x$ has exactly two neighbors $\forall x \in C$.

**Definition 3.** A finite subset $R$ of a simple digital curve is called a digital arc if there exist distinct elements $p, q \in R$ such that $p$ and $q$ have exactly one neighbor in $R$, and such that all other elements in $R$ have exactly two neighbors.

### Digital Straight Segments

Let $\Lambda$ be a point lattice embedded in $\mathbb{R}^n$. For a curve $\Gamma \subset \mathbb{R}^n$ denote $I(\Gamma) \subset \Lambda$ as the digitization of $\Gamma$, for some fixed digitization process.

**Definition 4.** A digital arc $S \subset \Lambda$ is a DSS in $\Lambda$ if there exists a line segment $l \in \mathbb{R}^n$ such that $I(l) = S$.

The question of recognition of a DSS becomes: is a digital arc the digitization of a straight line segment? In the next section 2D recognition methods will be reviewed, followed by a section which addresses digitization as applied in Definition 4.

3 Survey of Recognition Methods in $\mathbb{Z}^2$

There are three basic concepts, which lead to methods that determine if a digital arc is the digitization of a straight line. One, Rosenfeld showed that if a digital arc has the chord property, then it is the digitization of a straight line [10]. Two, a set of Diophantine inequalities may be used to determine if an arc is the digitization of a straight line [7]. Three, Bruckstein presented a method based on the chain codes, words, of a digital arc [1]. The chain code method will be used as a 2D recognition method in this report. Below, each of these approaches is described in more detail. Let $S$ be the set of coordinates that make up the digital arc in question for this section.

#### 3.1 Chord Property

Rosenfeld developed a classic recognition method in 1974 [10]. Let $p, q \in S$ and let $pq$ be the real line segment between $p$ and $q$. The segment $pq$ is near $S$ if $\forall (x, y) \in pq \subset \mathbb{R}^2$ there exists $(i, j) \in S$ such that

$$d_\infty((x, y), (i, j)) = \max[|i - x|, |j - y|] < 1.$$ 

The set $S$ has the chord property if $\forall p, q \in S$ the real line segment $pq$ is near $S$. With Theorem 3.1 Rosenfeld proved that the chord property is an appropriate criterion to recognize DSS in $\mathbb{Z}^2$.

**Theorem 3.1.** If a digital arc has the chord property, it is the digitization of a straight line segment.

Applying this theorem as a test on a digital arc leads to a mathematically simple method of DSS recognition, however implementation is
non-trivial. An illustration of the computational drawbacks of the chord property can been seen in the naive implementation Algorithm 3.1.

**Algorithm 3.1: Naive Chord Property Test ($R$)**

(comment: Test if a given digital arc $R$ is a DSS)

```plaintext
p, q ← end.points($R$)
pq ← real.line.segment($p, q$)

(comment: The segment $pq$ will include an infinite number of elements!)

for each $x \in pq$
do  
  for each $r \in R$
do    
    test ← 0
    if $d_\infty(r, x) < 1$ then
      test ← 1
      break
  
  if test = 0 then return (R is not a line)

return (R is a line)
```

Algorithm 3.1 starts by labeling the set of all the real points in the 2D real line segment between the endpoints of the digital arc $R$ as $pq$. Next, for each element in $pq$ the algorithm tests if there exists a point in $R$ that is near the element. Algorithm 3.1 will attempt to test every point on a real line segment, which is an unwieldy problem for a computer to tackle.

### 3.2 Diophantine Inequalities

Diophantine inequalities in recognition methods have been used to achieve linear time algorithms [7, 4, 2]. As discussed in the introduction, objects and techniques from Euclidean geometry such as Diophantine inequalities are used to study digital objects. The below recognition method uses parallel lines from Euclidean geometry to bound a digital arc. The criteria for recognition is based on the distance between the two parallel lines while bounding a digital arc. The distance and lines are defined by a set of Diophantine inequalities. For $a, b, c, d \in \mathbb{Z}$ and $\gcd(a, b) = 1$, the set

$$D_{a, b, c, d} = \{(i, j) \in \mathbb{Z}^2 | c \leq ai - bj < c + d\}. \quad (17)$$

is a *digital bar* with slope $a/b$, lower bound $c$ and arithmetical width $d$. A digital bar is a digital arc when restricted to $c = 1$. Reveillés, according to Klette, proved the Theorem 3.2 [7].

**Theorem 3.2.** Any set of lattice points $D_{a, b, c, max(|a|, |b|)}$ coincides with a set of lattice points assigned to a digital straight line, and conversely, for any rational digital straight line there are parameters $a, b, c$ such that the set of lattice points assigned to this digital straight line coincides with $D_{a, b, c, max(|a|, |b|)}$.

Theorem 3.2 shows that methods developed with the Diophantine inequalities criteria are valid DSS recognition methods.
Figure 5: A Chain-code can be obtained for a digital arc by using one of these diagrams to encode the steps taken from one point to the next. Diagram 5a is used for 4-connected directions, and Diagram 5b is used for 8-connected directions.

3.3 Chain-Codes, Words, and Pattern Recognition

Chain-Codes

Chain-codes of a digital arc can be used to determine if it’s a DSS [1]. The method can be illustrated with a 4-connected digital arc in the 2D Cartesian lattice. An element in a 4-connected arc implies it has at most 2 neighbors. Each neighbor will be in one of the following directions: right, up, left, or down. A chain code is constructed for the arc starting from one endpoint and recording the direction traveled to get to the next point using the directional coding depicted in the 4-connected diagram of Figure 5.

Example 3.1. Consider the image in Figure 1c starting from the lower right corner end point the chain code will read:

10101010101010

Or if the starting end point is the one in the upper right corner the code will read:

23232323232323

A convenient notation appears for the chain-code of a line is

\[ 1000\ldots0100\ldots010000\ldots010 = \cdots 10^K110^K210^K31\cdots \] (18)

where the K’s are the number of times zero is repeated. The chain-code in Equation 18 can be simplified further by considering only the run-length information in the sequence \( \{K_n\} \). With these sequences Bruckstein points out that if the elements of the \( \{K_n\} \) are \( \lfloor \frac{1}{2} \rfloor \) or \( \lfloor \frac{1}{2} \rfloor + 1 \) for a 4-connected digital arc, then the arc is a DSS with slope \( a \) [1].

Words and Letters

A word is a digital arc after it has been coded into a chain-code. The alphabet used for these words are the integers mod \( n \) where \( n \) is the connection number. Elements of the alphabet are called letters.
Pattern Recognition: Self-Similarity

The self-similarity property of digital straight lines was first presented by Bruckstein [1]. He also showed that the self-similarity property holds under lattice transformations [1]. This allows the self-similarity property to be used on any 2D point lattice. According to Klette, Wu proved that a digital arc with the self-similarity property is a DSS in the below theorem [7].

Theorem 3.3. A finite 8-arc is a digital straight segment iff its chain code sequence satisfies the DSS property [7].

DSS Property for Words

The following operations need to be defined and explained before the DSS property can be defined [7]. Let $C$ be a finite word. A letter $a \in C$ is nonsingular if the letter to the right or left in $C$ is equal to $a$, otherwise $a$ is singular. Let $l(C)$ denote the run length of nonsingular letters to the left of the first singular letter in $C$, and let $r(C)$ denote the run length of nonsingular letters to the right of the last singular letter in $C$. Define the reduction operation of a word $C$ to be $R(C)$ where $R(C)$ equals one of the following

- length of $C$ if $C$ is finite and has no singular letters
- the results from $C$ by replacing all factors of nonsingular letters in $C$, which are between two singular letters in $C$, by their run lengths, and by deletion of all other letters in $C$
- the letter $a$ if $C = a\omega$.

Example 3.2. Let $C$ be the chain-code of a digital arc.

$u_0 = C = 11101111101110111110111110111110111101111$.

Then we can find $u_n = R(u_{n-1})$,

$u_1 = 545554$

$u_2 = 3$.

All further iterations will be empty.

A finite word $u$ has the DSS property if $u = u_0$ and any nonempty word $u_n = R(u_{n-1})$ satisfies [7]:

- for all $u_n$ there are at most two different letters $a, b \in u_n$ and if there are two letters, then $|a - b| = 1$ (mod 8 in the case of $u_0$);
- if there are two distinct letters in $u_n$, then at least one must be singular;

and

- if $u_n$ contains only one letter $a$, or two, $a$ and $a + 1$, then $l(u_{n-1}) \leq a + 1$ and $r(u_{n-1}) \leq a + 1$;
- if $u_n$ contains two distinct letters $a$ and $a + 1$ and $a$ is nonsingular in $u_n$, then $l(u_{n-1}) = a + 1$ implies $u_n$ starts with $a$, and $r(u_{n-1}) = a + 1$ implies $u_n$ ends with $a$.

The self-similarity property as described with the DSS property is used for implementation of 2D line recognition in this report.
4 Digitization

In Definition 4 the digitization of a Euclidean curve was denoted $I()$. Below, $I()$ is defined more specifically. First, line tracing will be discussed, which is limited to digitizing Euclidean lines. Line tracing is an established method for digitizing Euclidean lines that will help justify the determination of the appropriate criteria for the proposed line recognition algorithm. After presenting the tracing method, the cover and supercover of a Euclidean curve will be defined and expanded to meet the needs of this report.

4.1 Bresenham’s algorithm

Line tracing is the selection of a set of lattice points forming the best approximation to the continuous Euclidean line segment [8]. Bresenham’s algorithm is a classic approach to line tracing in $\mathbb{Z}^2$, and has been generalized to line tracing in BCC and FCC lattices. Ibáñez’s description of the 2D Bresenham’s algorithm is presented in this subsection.

Let $e_1, e_2 \in \mathbb{R}^2$. Bresenham’s algorithm is best explained through the example of tracing the line segment between $e_1$ and $e_2$. Without loss of generality, let the slope of the line $e_1e_2$ be less than 45°. First, the algorithm finds the nearest digital, lattice, point to $e_1$ and the line segment, and labels the digital point $d_1$. The algorithm is initialized by setting a pointer $(x, y) = d_1$. Since the slope of $e_1e_2$ is less than 45°, then the algorithm moves the pointer in the x-direction to $(x+1, y)$. Next, the algorithm selects and marks either the point $(x+1, y)$ or $(x+1, y+1)$ whichever is closest to the line $e_1e_2$, selecting $(x+1, y)$ when the distance is equal. Call the current position $d_2$. The algorithm reinitializes to the
newly marked point, and repeats until the pointer is the nearest digital point to \( e_2 \). The last point is \( d_n \), where \( n \) is the number of steps taken in the algorithm. The set of marked points forms the digital line segment \( d_1d_n \). Figure 6 is a snapshot of the Bresenham’s algorithm tracing the Euclidean line segment between \( e_1 \) and \( e_2 \).

Below is Ibáñez summary of the Bresenham’s algorithm to trace the Euclidean line segment \( AB \) [8].

1. Determine the dominant direction of the line: \( X \) axis if the slope is less than 45°, \( Y \) axis otherwise.
2. Take one step along the dominant direction.
3. Test if a step should be taken in the secondary direction: if affirmative: take it.
4. If not yet in point \( B \) go to (2).

### 4.2 Generalization of Bresenham’s algorithm

Ibáñez’s generalization works on any point lattice of dimension \( N \) that is characterized by a set of \( N \) linearly independent vectors \( \{H_i\} \), \( i \in \{1, N\} \) and each vector \( H_i \in \mathbb{R}^N \) [8]. The method is presented below in the 3D case.

#### Ibáñez’s terms

Start by defining a grid point, Ibáñez’s term for lattice point, by its position vector \( P = \sum_{i=1}^{3} x_{pi}H_i \), where \( x_{pi} \in \mathbb{Z} \) and the basis \( \{H_i\} \). For example the two points \( A = \sum_{i=1}^{3} x_{ai}H_i \) and \( B = \sum_{i=1}^{3} x_{bi}H_i \). The set of connection vectors, as above, is denoted \( T \). For example, \( T = T_{F12} \). The grid vector, \( V_0 = B - A \), is the vector that represents the line segment \( AB \). The actual work of Ibáñez’s method is done along the grid vector from the origin to the endpoint of the grid vector \( V_t \).

The optimal vector basis \( \{V_i\} \), \( i \in \{1, 2, 3\} \) is chosen from \( T \) such that the coefficients of

\[
V_i = \sum_{i=1}^{3} x_{i}V_i
\]  

(19)

are minimized and nonnegative. The labeling of the optimal basis is done as follows; \( \{V_1, V_2, V_3\} \) such that the coefficients of Equation 19 are ordered \( x_1 \leq x_2 \leq x_3 \). The vector \( V_3 \) is called the dominant direction of \( V_t \). The other two vectors are secondary directions of \( V_t \). In Ibáñez’s method during the tracing process from one point to the next, only steps in one of these three directions is allowed.

Ibáñez calls the column vector matrix constructed from the optimal vector basis \( M_o = [V_1, V_2, V_3] \), and then uses \( M_o \) to construct \( M_u \) as follows

\[
M_u = M_o(M_o^TM_o)^{-1}.
\]  

(20)

The columns of the matrix \( M_u \) are called the reciprocal vector basis, and labeled \( \{U_i\} \), \( i \in \{1, 2, 3\} \). The Subspace projection matrix, \( M_{sp} \), is defined as having columns \( U_{pi} = x_{3}U_i - x_{i}U_3 \), \( i \in \{1, 2\} \). For the 3D case, the subspace projection matrix is

\[
M_{sp} = \begin{bmatrix}
x_3 & 0 \\
0 & x_3 \\
-x_1 & -x_2
\end{bmatrix}
\]  

(21)
The orthogonal subspace matrix is
\[ \Theta = M_{sp}(M_{sp}^T M_{sp})^{-1} M_{sp}^T. \] (22)

The Modified Bresenham’s algorithm

Here is Ibáñez’s modified Bresenham’s algorithm for a point lattice of dimension \( N = 3 \) between points \( A \) and \( B \) verbatim [8].

1. Initialize current point \( P \) at \( A: x_{pi} = x_{ai}, \forall i \in [1, 3] \).
2. Compute the matrix \( \Theta \).
3. Initialize column matrix \( M_R \) to zero.
4. Compute the column matrix \( M_T \) as the sum of \( M_R \) and the diagonal of \( \Theta \).
5. Select \( k \) as the index of the minimum element in \( M_T \)th column \([\ldots]\).
6. Take a step in direction \( k \), that is, increment \( m_k = m_k + 1 \).
7. Update matrix \( M_R \) by making \( M_R = M_R + 2\cdot \text{column}_k(\Theta) \).
8. If point \( P \) is different from point \( B \), go to Step 4, else end.

The notation \( I_G() \) will be used to refer to the set of points selected by the modified Bresenham’s algorithm \( I_G() : \mathbb{R}^3 \rightarrow G \) described above with the point lattice \( G \).

Description

Similar to the original Bresenham’s algorithm, \( I_G \) chooses the next lattice point based on the distance to the line being traced. The extensions of dimension and lattice type make calculating the distances different. It’s done by first finding the optimal vector basis, which selects the three types of connections that will take the least amount of steps from one end point to the next. The dominant direction is the default step in the algorithm, which the other two directions are tested against. At the current point, the algorithm determines which of the three possible steps to take, by examining the projection of the lattice points reachable into the orthogonal complement of \( V_t \) at the current point. In the orthogonal complement, the algorithm selects the projected lattice point that is closest to the projection of \( V_t \). All of this is achieved with the projection matrix \( \Theta \) and is described in great detail by Ibáñez [8]. In summary, this method moves from one lattice point to the next by choosing from the nearest lattice point out of the three possible. The combination of finding the optimal vector basis for the line and the choice of the nearest possible lattice point at each step, leads to a digital arc that is a good approximation of the line being traced.

4.3 Cover and Supercover

A cover of a continuous [curve] is a set of voxels such that every point of the continuous [curve] lies in some voxel of the cover [3].

Let \( \Gamma \) be a curve in \( \mathbb{R}^n \).

Definition 5. A set of voxels \( S \) covers \( \Gamma \) if \( \Gamma \subseteq S \).
The set of lattice points $S$ is called a cover of $\Gamma$. The voxels of a cover completely contain the curve and the curve is continuous, thus there is a connected component that covers the curve. Each element of the connected component of cover has at least one neighbor. A set $S$ is called a minimal cover if there does not exist a proper subset of $S$ that is a cover.

**Definition 6.** An element $b \in S$ is said to be in a bubble if $\Gamma \cap b \neq \emptyset$ and $\Gamma \cap \text{int}(b) = \emptyset$. A subset of $S$ is called a bubble if it is connected and if all its elements are in a bubble.

**Example 4.1.** (a) Consider the vertex of the parabola in Figure 7a. Four lattice points intersect the vertex of the parabola, and are members of the cover. The bottom two of these four lattice points make a bubble. The top two lattice points are not bubble elements, since the parabola intersects their interiors.

(b) Figure 7b has no bubbles.

(c) As another contrived 2D example, consider a line in $\mathbb{R}^2$ that only passes through the edges of elements in a lattice. One valid cover for such a line would be a digital arc of width two. All elements in such a cover would be elements of bubbles.

A supercover of a continuous curve is the set of all voxels that the curve touches [3].

**Definition 7.** Given a curve $\Gamma$, the supercover is the set of all $x \in \Lambda$ such that $x \cap \Gamma \neq \emptyset$ where $\Lambda$ is the set of all voxels.

The supercover is a minimal cover plus all the possible bubbles.
Example 4.2. Consider $S$ the supercover in the FCC lattice of the segment from $a = (0, 0, 0)$ to $b = (4, 0, 0)$ depicted in Figure 8.

$$S = \{(0, 0, 0), (1, 1, 0), (1, -1, 0), (1, 0, 1), (1, 0, -1), (2, 0, 0),
(3, 1, 0), (3, -1, 0), (3, 0, 1), (3, 0, -1), (4, 0, 0)\}.$$

There are two bubbles in this supercover: $\{(1, 1, 0), (1, -1, 0), (1, 0, 1), (1, 0, -1)\}$ and $\{(3, 1, 0), (3, -1, 0), (3, 0, 1), (3, 0, -1)\}$. The minimal cover is the set $\{(0, 0, 0), (2, 0, 0), (4, 0, 0)\}$.

Figure 8: The Supercover of the line segment between the points $(0, 0, 0)$ and $(4, 0, 0)$ in the FCC lattice. The minimal cover is blue. The red and green components are bubbles. See Example 4.2.

4.4 The $\alpha$-supercover

Let $G$ be a lattice embedded in $\mathbb{R}^3$ with $a, b \in \mathbb{R}^3$. Let $S$ be the supercover of the line segment $ab$ in $\mathbb{R}^3$. It’s easy to guess that the trace $I_{G_\alpha}(ab) \subset S$ where $G_\alpha$ is an $\alpha$-connected lattice. However Example 4.3 provides a counterexample.

Example 4.3. Consider $S$ the supercover in the BCC lattice of the segment from $a = (0, 0, 0)$ to $b = (4, 0, 0)$.

$$S = \{(0, 0, 0), (2, 0, 0), (4, 0, 0)\}.$$

Since $S$ is not 8-connected in BCC, then $I_{G_\alpha}(ab) \not\subset S$. Note that $S$ is the blue subset of the object shown in Figure 9.
With Example 4.3 in mind, a new cover must be constructed: the \(\alpha\)-supercover. The concept of the \(\alpha\)-supercover will be to create something similar to a bubble for the problem in the BCC lattice with 8-connectedness exposed in Example 4.3, and to let the \(\alpha\)-supercover equal the supercover in other cases.

The problem arises in the BCC case when 8-connectedness is needed, and when the curve being covered passes from one voxel to the next through the interior of a square face of the truncated octahedra. Call this passing from one voxel to the next through the interior of a square face a long BCC transition. Long because this transition takes place when the distance between neighboring lattice points is the greatest.

**Definition 8.** For two elements in a cover connected by a long BCC transition, the long bubble is the addition of the four common neighbors to the two elements.

The \(\alpha\)-supercover is equivalent to the supercover in all cases, except for when 8-connectedness is needed in the BCC lattice. Then, the long bubbles are added to the supercover.

**Definition 9.** The \(\alpha\)-supercover is the supercover plus long bubbles to achieve \(\alpha\)-connectedness.

**Example 4.4.** The 8-supercover of the line segment \(ab\) in Example 4.3 is

\[
S_8 = \{(0, 0, 0), (2, 0, 0), (4, 0, 0)\} \\
\cup \{(1, 1, 1), (1, 1, -1), (1, -1, 1), (1, -1, -1)\} \\
\cup \{(3, 1, 1), (3, 1, -1), (3, -1, 1), (3, -1, -1)\} .
\] (23)

The \(\alpha\)-supercover, \(S_\alpha\), is depicted in Figure 9.

An \(\alpha\)-supercover of a line contains a subset that is the trace of a line.

**Lemma 4.1.** Given an \(\alpha\)-supercover \(S\) of a line, the trace of the line between the endpoints of \(S\) is contained in \(S\).

Further with Lemma 4.1 the definition of a DSS is made more specific as follows.

**Definition 10.** A digital arc is a digital straight line segment if it is in the \(\alpha\)-supercover of some line.

The rest of the section is a sketch of a proof of Lemma 4.1.

Let \(l\) be a line segment between \(a, b \in \mathbb{R}^3\), and \(\Lambda\) be a point lattice embedded in \(\mathbb{R}^3\) with the set of connection vectors \(T_\alpha\). Call \(S\) the \(\alpha\)-supercover of the line segment \(l\).

Define \(L\) to be the set constructed by the Modified Bresenham’s algorithm, as described above, of the line segment \(l\) with the optimal vector basis \(\{V_1, V_2, V_3\}\). For \(V_i \in \{V_1, V_2, V_3\}\)

\[
L = \{t_1, \ldots, t_n \text{ s.t. } t_{k+1} = t_k + V_i\} .
\] (24)

It needs to be shown that \(L \subset S\). To start, notice that \(a \subset t_1\) and \(b \subset t_n\) and so \(t_1, t_n \in S\).

The Modified Bresenham’s algorithm is designed to select the lattice points that are nearest to the line it traces. Experimentation showed that this is likely true. This concept is not proven in Ibáñez’s presentation, but it is taken as fact in this report [8]. Now taking the above as fact, \(t_{k+1}\) is selected to be the \(\alpha\)-neighbor in the direction from \(a\) to \(b\) of \(t_k\) that is nearest to the line. By the above, \(t_2\) is the \(\alpha\)-neighbor in the direction...
Figure 9: The 8-Supercover of the line segment between the points (0, 0, 0) and (4, 0, 0). The red and green components are bubbles. Notice that the red and green components are not 8-connected. See Example 4.4.

from a to b of t₁ that is closest to l. To review, the α-supercover is α-connected and each element of S either contains part of the line l, or has two α-neighbors that contain part of the line l and are not α-connected to each other. The element t₁ contains part of the line, and the line passes through the boundary of the voxel of t₁ into the voxel of a lattice point in S; call this lattice point s. Case one, if s is an α-neighbor of t₁ and since part of the line is contained in its’ voxel, then s is the α-neighbor in the direction from a to b of t₁ that is closest to l. So s is the same as t₂ and thus t₂ ∈ S. Case two, if s is not an α-neighbor of t₁, then all the lattice points that α-neighbors to both t₁ and s are in S. These neighbors are the α-neighbors of t₁ closest to the line l. It must be that one of these neighbors are t₂, so t₂ ∈ S. It follows that s = t₂ and so t₂ ∈ S too. These two cases can be followed in an induction argument to complete the proof.

5 Orthogonal Complement DSS Recognition

Orthogonal Complement DSS Recognition, OC-DSSr, is a method for determining if a digital arc is a digital straight line in BCC and FCC lattices. The idea is to project a digital arc into two orthogonal complements, and if the projections of the 3D arc are digital lines in both projected lattices, then the 3D arc is a line.
5.1 Input

The input is a digital arc between two lattice points. The arc is labeled \( R \). Assume \( R \) is of length \( n \). By translation, w.l.o.g. one endpoint is the origin, called \( r_1 \). Order the elements as follows: \( r_1 \) and \( r_n \) are the endpoints of \( R \), and \( r_i \) is connected to \( r_{i+1} \). OC-DSSr requires specification of the type of lattice and connection scheme, \( G \) and \( T \) respectively.

5.2 Description

Determine the vector \( V = r_n - r_1 \). The vector \( V \) is used, in the same way as Ibáñez’s grid vector, to find the optimal vector basis for the possible line.

Optimal Vector Basis

The optimal vector basis is found, as in Ibáñez’s algorithm, by solving Equation 19 such that the coefficients are minimized and nonnegative. Notice that if one or more of the coefficients are zero, then the line is not three dimensional, and therefore is a 2D or 1D recognition problem. The concept of dominant and secondary directions are the same as above. Let the vector \( V_3 \) be the vector with the largest coefficient, and \( V_1 \) and \( V_2 \) be the secondary vectors. In the selection of this basis, OC-DSSr limits \( R \) to the use of three connections, meaning that for every element \( r_i \in R \) the only possible neighbors are in directions of \( V_3, V_2, \) or \( V_1 \).

Three Letter Words

Having only three possible types of connections for each element of the digital arc encourages the use of the word notation. Currently, there is no standard diagram for 3D point lattices that corresponds to Figure 5. However, once the basis has been chosen three letter words can be constructed. The algorithm OC-DSSr will not continue if \( R \) is not expressible in the three letters determined by the optimal basis for \( V \).

Orthogonal Complement

The OC-DSSr will evaluate the projection of \( R \) into the lattice of the orthogonal complement of the secondary directions. The orthogonal complement lattice will be the point lattice in the plane orthogonal to one of the secondary directions, with the generating matrix columns equal to the projections of the other two basis vectors. Denote this orthogonal complement lattice \( V_i^\perp \) and the projection \( P_i : G \to V_i^\perp \) for \( i \in \{1, 2\} \). The projection into the orthogonal complement can be express in the three letter word notation as ignoring one of the secondary directions.

OC-DSSr(\( R, G, T \))

Given a digital arc in a lattice with a set of connection vectors as defined above, the OC-DSSr is outlined in five steps:

1. \( V = r_n - r_1 \) where \( R = \{r_1, r_2, \ldots, r_n\} \).
2. Find optimal basis for \( V \): \( V_1, V_2, V_3 \in T \) with \( V_3 \) as the dominate direction.
3. Create chain-code, $C$, for $R$ with letters $1, 2, 3$ corresponding to steps in directions $V_1, V_2, V_3$. If not possible, $R$ is not a line in $G$ with connection vectors in $T$.

4. Map $C$ to $C_1$ and $C_2$ in the following way:

$$ f : C \rightarrow C_1 \text{ s.t. } f(i) = 1, f(3) = 0. \quad (25) $$

Both $C_1$ and $C_2$ are words with only two letters.

5. If $C_1$ and $C_2$ have the DSS Property for Words, then $R$ is a DSS in $G$ with connection vectors $T$.

6 Implications

6.1 Image of a line

The $\alpha$-supercover is a general digitization method of curves. Examination of the $\alpha$-supercover of a line yields an $\alpha$-connected set, and the $\alpha$-supercover of a line is not necessarily a digital arc, see Example 4.4. Further, Lemma 4.1 implies that there exists a subset of the $\alpha$-supercover which is the result of a line tracing. This leads to Definition 10 of a DSS.

The recognition problem in this report is to determine when a digital arc is a DSS. Thus, an arc detected by OC-DSSr is shown to satisfy Definition 10 in the following proposition.

**Proposition 6.1.** An arc $R$ recognized by OC-DSSr is a DSS.

**Proof.** Let $R = \{r_1, \ldots, r_n\}$ be defined as in the algorithm of OC-DSSr with $r_{k+1} = r_k + V_i$ where $V_i$ is in the optimal basis $\{V_1, V_2, V_3\}$. Without loss of generality, $r_1 = 0$. Define $l$ to be the line segment between the endpoints of $R$ and $V$ be the vector of the line. Let $S$ be the $\alpha$-supercover of $l$, and $L$ be the line trace of $l$. By Lemma 4.1 $L \subset S$. Notice $r_1$ and $r_n$ are elements of $S$. For all $y \in l$, $y = \alpha V$ for $\alpha \in [0, 1]$. From the origin starting with $r_1$, $R$ covers $\alpha V$ where $\alpha \in [0, \beta_0]$, and starting from $r_n$, $R$ covers $\alpha V$ where $\alpha \in [\beta_1, 1]$. Denote

$$ R_0 = \{r_{0i}, \ldots \text{ s.t. } r_{0i} \in R \text{ and } \alpha V \cap \alpha V \neq \emptyset \text{ where } \alpha \in (0, \beta_0)\}, $$

$$ R_1 = \{r_{1i}, \ldots \text{ s.t. } r_{1i} \in R \text{ and } r_{1i} \cap \alpha V \neq \emptyset \text{ where } \alpha \in [\beta_1, 1]\}, $$

and

$$ R_2 = R - R_0 - R_1. $$

$R_0$ and $R_1$ are not empty, since $r_1 \in R_0$ and $r_n \in R_1$. The elements of the sets $R_0, R_1$ are contained in the $\alpha$-supercover. Since

$$ R = R_0 \cup R_1 \cup R_2 $$

the proof is complete if $R_2 = \emptyset$.

Assume $R_2$ is not empty. Then $R_0 = \{r_1, \ldots, r_k\}$, $R_1 = \{r_j, \ldots, r_n\}$ and $R_2 = \{r_{k+1}, \ldots, r_{j-1}\}$. Define $P_i$ for $i \in \{1, 2\}$ as the projections $P_i : \mathbb{R}^3 \rightarrow V_i^\perp$. The sets $P_1(L)$ and $P_1(R)$ are 2D digital arcs and 2D digital straight line segments with the same endpoints. Thus $P_1(L) = P_1(R)$.

Suppose w.l.o.g $i = 1$. Let $p_k = P_1(r_k)$ and $p_{k+1} = P_1(r_k)$ be the $\alpha$-neighbor of $p_k$ in the direction away from the origin. Thus $p_{k+1} = p_k + P_1(V_h)$ where $V_h \in \{V_1, V_2, V_3\}$ and $h \neq 1$. The lattice point $r_{k+1} \in R_2$ is equal to $r_k + V_h$ where $V_h$ is an optimal basis vector and $r_{k+1} \notin S$. Define
where $V_j \neq V_k$ is an optimal basis vector and $t \in L$. Therefore $t \in S$.

$$P_1(r_{k+1}) = P_1(r_k + V_k) = P_1(r_k) + P_1(V_k).$$

$$P_1(t) = P_1(r_k + V_j) = P_1(r_k) + P_1(V_j).$$

The below diagram has three rows: the top row is in $V_2^\perp$, the middle row is in $R_3$, and the bottom row is in $V_1^\perp$. The diagram is drawn w.l.o.g. below with $P_2(t) = p''_{k+1}$, but $P_1(t) = p''_k$ would lead to the same contradiction.

The above diagram gives $V_k = V_1$ and $V_j = V_2$. Since $P_2(L)$ is a digital arc, there exists $t' \in L$ such that $P_2(t') = p''_{k+1}$. For $m \in \mathbb{N}$ below are the two paths from $r_k$ to $t'$.

Thus

$$r_k + V_1 + pV_2 = r_k + (p + 1)V_2.$$ 

But this implies $V_1 = V_2$! Therefore $R_2$ must be empty.

6.2 Rewording

Definition 4 of a DSS depends on the digitization method. It would be helpful to have a definition that is independent of the imaging process. A definition is presented with the criteria for DSS from the OC-DSSr method in mind. Let $C$, as constructed in OC-DSSr, be the optimal vector basis chain-code of a digital arc with end points $r_1$ and $r_2$.

**Definition 11.** The word $C$ is a chain-code of DSS in a point lattice if $C$ is a three letter word, and $f_1(C) = C_1$ and $f_2(C) = C_2$ have the DSS Property for Words, where

$$f : C \rightarrow C_i \text{ s.t. } f(i) = 1, f(3) = 0.$$ (26)

The consequence is a quick reasonable way to classify digital arcs in a point lattice. Digital straight line segments in a point lattice are three letter words such that two of the reduced two letter words are 2D digital straight line segments.
6.3 Conclusion

The OC-DSSr method solves the problem of line recognition in a point lattice. Given a digital arc in the BCC or FCC lattices, OC-DSSr will determine if the arc is a DSS. The criteria for a digital arc to be a DSS was defined as being the digitization of a line. The criteria is achieved with OC-DSSr according to Proposition 6.1.

After determining the criteria for a digital arc to be a DSS, a new digitization method was needed to achieve the correct connectedness. The $\alpha$-supercover was the digitization method developed to attain the appropriate connectedness in all point lattices. The collection of $\alpha$-supercover digitization of lines and the collection of line traces are shown to coincide in Lemma 4.1. This coincidence justified both the digitization method and the criteria for a DSS.

The use of the self-similarity property of 2D digital lines in OC-DSSr led to a new definition of DSS, Definition 11. The presented definition leaves room for improvement and extension. Improvements could be made by eliminating the need to project into 2D words. A criteria for three letter words to be DSS could be developed from the current definition. Extending into higher dimensions, one can imagine a cascading definition, similar to Definition 11, of a line in a point lattice embedded in $\mathbb{R}^n$.

**Example 6.1.** $R$ is a digital arc in the point lattice $G \subseteq \mathbb{R}^4$. Suppose $R$ is a four letter word. Suppose $R$ is a DSS in $G$ if all three-letter sub-words are DSS in 3D. There are three such sub-words. Each three-letter sub-word has two two-letter sub-words. If all the two-letter sub-words are DSS, then $R$ is a DSS in $G$.

The complexity of Definition 11 in algorithms when extended to higher dimensions is apparent in Example 6.1 without accounting for the reduction method in the DSS property for words. The algorithm for checking the 2D DSS property for words would have to be performed $(n - 1)!$ times. Seeking the extension of Definition 11 may not be practical, but this report does leave direction for further research.
References


