Discrete Mathematics and Theoretical Computer Science **AB(DMCS)**, 2003, 167–178

Cellular Lines: An Introduction

Panama Geer[1] † and Harry W. McLaughlin[2] and Keith Unsworth[3]

[1] *Bryn Mawr College, Department of Mathematics and Computer Science, PA, USA, pgeer@brynmawr.edu,*
[2] *Rensselaer Polytechnic Institute, Department of Mathematical Sciences, NY, USA, mclauh@rpi.edu,*
[3] *Lincoln University, Applied Management and Computing Division, Canterbury, NZ, unsworth@lincoln.ac.nz,*

This paper provides a definition of a cellular line in a cellular array that is independent of the notion of a line in $\Re^2$. It also presents a way of determining whether or not a cell set is a cellular line. Brief statements about existence, uniqueness, and properties of cellular lines are included.

Keywords: cellular line definition, cellular array, string representation, derived string, line drawing algorithm

1 Introduction

Our work is motivated by the question: Can one find, completely within the world of cellular arrays, a geometry with which to model cellular phenomena? In particular, does there exist an indigenous cellular geometry completely independent of Euclidean notions? Or are all possible cellular geometries inherently tied to Euclid?

It is our belief that finding a Euclidean-free geometry will open up new modeling and computational techniques in the cellular world. In particular it will open up the possibility of capturing geometries as they emerge from complex processes rather than designing them with an engineering perspective.

We wish to define cellular curves in a two-dimensional cellular array that are meaningful, yet fundamentally distinct from those that are defined in Euclidean two-space. In particular we investigate this question for the special case of cellular lines.

There has been considerable attention given to discretizing curves in Euclidean two-space. The results presented here are not derived from a discretization process, but rather built solely within the medium of cellular arrays.

The need to define a cellular line arose in recent studies in which we attempted to use *ant algorithms* to construct straight lines, (GMU01). The work was modeled on the work of (DMC96) in which ant algorithms are used to find solutions for the traveling salesman problem. Initial experiments with reasonable metrics produced "optimal paths" that did not appear straight. The investigations of this paper are partially motivated by a need to determine when a virtual ant has traversed a "straight" line.

†Partially supported by the National Science Foundation's VIGRE Program (grant DMS-9983646) and a grant from the W. M. Keck Foundation (#001746).

Our goals are:

(1) to define the notion of a Euclidean-free cellular line (Fig. 1), and

(2) to state an algorithm for determining whether or not a given cell set is a cellular line.

While not a focus of this paper, we note that an algorithm for the construction of cellular lines has been developed and is discussed formally in (Gee02) and (GMU03). The algorithm takes as input two integers and produces a representation of a cellular line.

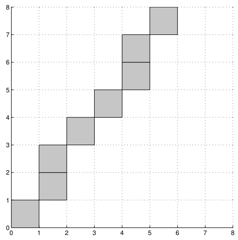


Fig. 1: A cellular line represented by the string (1 2 1 1 2 1).

2 Related Work

There has been much work in the area of line drawing algorithms since the 1960's.[1] Recent work includes (BB00), (RK01), and (SL00). The work can generally be categorized in terms of two distinct approaches: conditional algorithms and structural algorithms. A survey of these approaches is presented in (Bro85). Conditional algorithms take two points as input. Beginning at one of these lattice points, additional lattice points are found according to whether a specified "condition" is satisfied (this condition is dependent on a Euclidean line model). In this case, the line can be displayed as it is being generated. One of the early conditional algorithms was presented by (Bre65). Bresenham's algorithm is often thought of as a benchmark against which other line generation algorithms are compared.[2] Structural algorithms are those based on structural properties of string representations of lines. Typically, the line cannot be displayed until the algorithm has finished running. One of the early structural algorithms was presented by (Bro74).

(Fre61) introduced a method of encoding a geometric configuration in a lattice as a sequence, called *Freeman chain coding*. He stated three properties of chain codes that represent lines. These properties were later formalized using the *chord property*, (Ros74).

The algorithm presented by (HKV81) and the one presented here are similar in that both accept as input a string representation of a cell set and both give as output a determination of whether or not the string represents a "line."[3] The determination provided by the algorithm in (HKV81) is based on the chord property, while the determination provided by the algorithm here is not.

3 Setting the Stage

We work with two-dimensional rectangular cellular arrays. In defining a cellular line we limit our attention to certain cell sets, called *candidate cell sets*. Candidate cell sets are not necessary for the definition of a cellular line given below, however, they are introduced to help motivate this definition.

A cell located in the i^{th} column and j^{th} row of the second octant of a two-dimensional cellular array has coordinates, (i, j), where $1 \le i \le j$. These cells are ordered lexicographically first by i (by column) and then by j (by row). For example $(2,3) < (3,2)$ and $(2,2) < (2,3)$. A two-dimensional *candidate cell set* is a finite cell set, ordered by the above lexicographical ordering such that (i) the first cell has coordinates

[1] A substantial list of references can be found in (KRS98).

[2] See (Gee02) for a comparison of the cellular lines defined here and Bresenham's algorithm.

[3] In the case of (HKV81), the "line" is a digital straight segment; while in this work, the "line" is a cellular line.

$(1,1)$, and (ii) for each cell, (i, j), in the cell set, its immediate successor in the cell set, is a cell with one of the following forms: $(i, j+1)$, an edge connection, or $(i+1, j+1)$, a vertex connection.

Thus, a candidate cell set is a lexicographically ordered, "connected" cell set, where "connected" indicates that each cell shares either its northeast vertex or its northern edge with its successor. We assume that a rectangular coordinate system is imposed on the cellular array so that the chosen initial cell lies in the first column and row of the cellular array and that the chosen terminal cell lies in the c^{th} column and the r^{th} row, with $1 \leq c \leq r$. This convention is used without loss of generality because one can impose such a coordinate system after choosing initial and terminal cells. The cell sets shown in Figs. 1 and 2 are candidate cell sets.

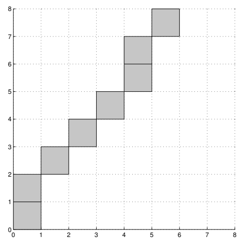


Fig. 2: This is a candidate cell set.

Cellular lines, as they are defined below, are special candidate cell sets. In what follows we justify the choice of the candidate cell set depicted in Fig. 1 as a cellular line, while not admitting the candidate cell set shown in Fig. 2.

4 Modeling Tool: Strings

In lieu of vector-valued functions that are frequently used as modeling tools to define curves in the Euclidean plane, we use strings (sequences of positive integers) as modeling tools to define cellular lines in cellular arrays. An example is given by the string, (1 2 1 1 2 1), which records, reading left-to-right, the number of cells in each of the columns of the cell set shown in Fig. 1.[4]

Using the convention illustrated in the above example, it can be seen that the candidate cell sets are exactly the cell sets that have string representations.[5] Analogous to defining a Euclidean straight line in terms of a function that has the property of linearity, a cellular line is defined below as a cell set having a string representation with particular properties.

In what follows we work with strings and provide a characterization of string representations of cell sets that we later define to be cellular lines. Preliminary definitions are needed to formalize the notion of a cellular line. To motivate these definitions, we provide an example of the selection of a cellular line with a given terminal cell.

5 Example 1

For the cell set in Fig. 1 the terminal cell is in the 6^{th} column and the 8^{th} row i.e., $c = 6$ and $r = 8$. The length of the corresponding string is 6, i.e., it has six components. The sum of the components of the string is 8. There are 21 such strings and hence 21 corresponding cell sets satisfying the four conditions in section 3.[6] Exactly one of them, see Fig. 1, is selected as a cellular line.

---

4 In the string (1 2 1 1 2 1), we represent two cells by the number 2. The concept of representing multiple cells by one symbol is analogous to the notion of a "span" in (BB99) or a "run" in (Ste98).

5 A mathematically precise definition of string representations is found in (GMU03).

6 If the string contains six components that sum to eight, then the string has either two 2's and four 1's, or one 3 and five 1's. There are 21 such strings, i.e. $21 = \binom{6}{2} + \binom{6}{1}$.

Three of the 21 strings are: (1 2 1 1 2 1), (2 1 1 1 2 1), and (1 1 3 1 1 1). They represent the cell sets shown in Figs. 1, 2 and 3 respectively. There are two primary distinctions between the cell sets depicted in these figures.

- The third column in Fig. 3 is significantly longer than the other columns in that figure; while the column lengths in Figs. 1 and 2 are nearly equal. In both Figs. 1 and 2 column lengths vary by no more than one cell.
- Even though the column lengths in Fig. 2 are nearly equal, the longer columns in Fig. 2 are not as evenly distributed among the shorter columns as they are in Fig. 1.[7]

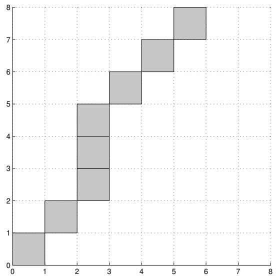


Fig. 3: (1 1 3 1 1 1)

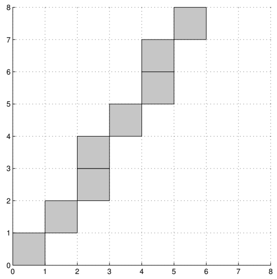


Fig. 4: (1 1 2 1 2 1)

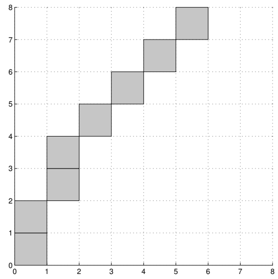


Fig. 5: (2 2 1 1 1 1)

The definition of a cellular line presented here reflects the above two observations; additionally it reflects notions of even distributions in higher order constructs.[8] The human eye can readily determine when column lengths vary substantially, Fig. 3. The human eye can also frequently determine when columns of different lengths are not evenly distributed, Fig. 2. However, the human eye cannot generally detect structural properties represented by the higher order string constructs developed below. This is analogous to higher order derivatives that characterize structural properties of planar curves, properties not detectable with the human eye.

Just as some work is needed to develop the notions of derivatives and derivatives of derivatives in order to define special curves in the plane, we invest some effort in developing the notion of derived strings of different orders to define cellular lines.[9]

Construction of Candidate Strings. Given $c = 6$ and $r = 8$, the *candidate strings* are the strings: (i) over the alphabet of positive integers, (ii) of length six, (iii) whose components sum to eight. There are 21 candidate strings. The candidate cell sets corresponding to three of the candidate strings are shown in Figs. 1, 2, and 3.

---

[7] (Fre70), stated three properties for strings that represent straight lines. Requiring columns to be "nearly equal" and "evenly distributed" satisfies Freeman's properties.

[8] The structural algorithms presented by (Bro74), (HKV81), and (RK01), use analogous hierarchical approaches to generate a string representation of a straight line with Freeman coding. The definition of a cellular line given here does not arbitrarily place "the least occurring symbol" at "the end of the period" as in (Bro74). The notion of a fully partitioned string (discussed later) differs from the notion of a symbol that is "singular" in a string, as in (RK01) and (HKV81).

[9] Certain self-similarity properties discussed by (RK01) and (HKV81) are related to this work. In particular, the notion of a derived string is referred to as a "reduction operation" by (RK01) and elsewhere it is referred to as replacing a "run" by its "run length."

Construction of Zero Order Derived Strings. Requiring that the column lengths of the candidate cell sets be nearly equal, means that two components of each of the corresponding strings must be 2 and four of the components must be 1, resulting in 15 such strings, i.e. $15 = \binom{6}{2}$.

The *zero order derived strings* are those candidate strings over the alphabet $\{1, 2\}$. Three of the 15 zero order derived strings are: (1 2 1 1 2 1), (1 1 2 1 2 1), and (2 2 1 1 1 1). For the zero order derived strings, the symbol 2 is called the minority symbol and the symbol 1 is called the majority symbol because the symbol 2 appears less frequently than the symbol 1.

The zero order derived strings listed above represent the cell sets shown in Figs. 1, 4 and 5 respectively.[10] Notice that the alphabet underlying the zero order derived strings, consisting of two consecutive positive integers, models the desired property that column lengths be nearly equal.

Construction of First Order Derived Strings. These are generated by requiring that columns of different lengths be evenly distributed. This leads to the following considerations. Among the 15 zero order derived strings there are some that represent candidate cell sets with columns of length 2, evenly distributed among the columns of length 1. That is, there are zero order derived strings having three substrings of 1's (majority symbols) partitioned by the two 2's (minority symbols). There are three such strings, i.e. $3 = \binom{3}{2}$, namely (1 2 1 1 2 1), (1 1 2 1 2 1), and (1 2 1 2 1 1). For these strings, we say that the 2's *fully partition* the strings.[11]

Each *first order derived string* is formed from a zero order derived string that is fully partitioned by the 2's (the minority symbol). A fully partitioned string is transformed into a first order derived string in two steps. (i) Each substring of 1's (majority symbol) is replaced by the integer that counts the number of 1's in the substring. (ii) The minority symbols (the original 2's) are deleted.

For example, $(1\ 1\ \underline{2}\ 1\ \underline{2}\ 1) \rightarrow (2\ \underline{2}\ 1\ \underline{2}\ 1) \rightarrow (2\ 1\ 1)$, where $\underline{2}$ denotes a 2 appearing in the zero order derived string. The three first order derived strings are shown below along with their predecessor zero order derived strings. The first order derived strings reflect an even distribution of columns of different heights in candidate cell sets.

First Order Derived String	Zero Order Derived String
(1 2 1)	(1 2 1 1 2 1)
(2 1 1)	(1 1 2 1 2 1)
(1 1 2)	(1 2 1 2 1 1)

Construction of Second Order Derived Strings. Geometric motivation for the construction of second order derived strings is discussed by (Gee02). Each first order derived string contains two 1's and one 2. Following the same procedure as above, the 2's partition each first order derived string into substrings of 1's. This first order derived string, (1 2 1), is said to be fully partitioned by the 2, because the number of substrings of 1's is maximal.

Every *second order derived string* is formed from a first order derived string that is fully partitioned by the symbol 2. The transformation from a first order derived string to a second order derived string occurs in two steps. (i) Each substring of majority symbols (1's) is replaced by the integer that counts the number of majority symbols (1's) in the substring. (ii) The minority symbols (original 2's) are deleted. There is 1 second order derived string, i.e. $1 = \binom{2}{2}$. It is shown below with its predecessor first and zero order derived strings.

---

10 The string (2 1 1 1 2 1) is also a zero order derived string, Fig. 2.

11 A string is fully partitioned when the number of substrings of majority symbols is maximal.

Second Order	First Order	Zero Order
(1 1)	(1 2 1)	(1 2 1 1 2 1)

Construction of Third and Higher Order Derived Strings. Using the above process, there is exactly one third order derived string.[12] It is shown below with its predecessor second, first and zero order derived strings.

Third Order	Second Order	First Order	Zero Order
(2)	(1 1)	(1 2 1)	(1 2 1 1 2 1)

Fourth and higher order derived strings will all exist and be (1).[13] Generating them will not identify other zero order derived strings.

The string, (1 2 1 1 2 1), is *defined* as the string representation of the cellular line determined by the terminal cell in the 6th column and 8th row, (Fig. 1).

6 Definitions

The steps used in Example 1 can be described as a search among 21 candidate strings, for the one string that has all four (0, 1st, 2nd, 3rd) order derived strings. In order to identify different order derived strings, the notion of fully partitioning a string by one of its elements was used. In this section, we formalize the two notions of *fully partitioned* and *derived strings*. Using these definitions we offer a definition of a cellular line in terms of different orders of derived strings and conclude with several examples.

Fully Partitioned Strings. In a string over a two-symbol alphabet one of the symbols occurs not more frequently than the other. It is designated as the string's *minority symbol*. The other symbol is designated as the string's *majority symbol*.[14]

A string of integers is said to be *fully partitioned* when the following conditions hold.

i. It is a finite string over an alphabet of two consecutive positive integers.

ii. When the two symbols do not occur the same number of times, the minority symbol partitions the string into $|m|+1$ non-empty substrings of majority symbols, called the *majority symbol substrings*, where $|m|$ denotes the number of times the minority symbol occurs in the string.

iii. When the two symbols occur the same number of times, the minority symbol partitions the string into $|m|$ non-empty substrings of majority symbols, also called majority symbol substrings.[15,16]

---

[12] Notice that there is no minority symbol in the second order derived string, so none is deleted. We simply replace the two 1's with the integer 2.

[13] This is analogous to the fact that the $(n+1)^{st}$ and higher order derivatives of an n^{th} degree polynomial are zero.

[14] When the two symbols occur the same number of times, either can be designated as the minority symbol.

[15] In this case each symbol alternates and the substrings are of length 1. It is also understood that when both symbols appear the same number of times, either one of them can be selected as the minority symbol.

[16] The string (4 3 4 4 3 4) is fully partitioned. Its minority symbol is 3, which occurs twice. The symbol 3 partitions the string into three ($|m|+1$) substrings of majority symbols, 4's. The string (4 3 4 3 4 3) is fully partitioned. Either of the two symbols, 3 or 4 can be designated as the minority symbol. In each case, the minority symbol partitions the string into three ($|m|$) substrings of majority symbols.

It is understood that if the string contains only one of the two symbols from the alphabet, the missing one is designated the minority symbol. In this case the minority symbol partitions the string into one substring, namely the whole string. Thus any finite string with only one distinct symbol is fully partitioned.

j^{th} Order Derived Strings. Let S denote a string over some alphabet. If S is a string over an alphabet consisting of two consecutive positive integers, denote its *zero order derived string* by $S^{(0)}$ and define it by $S^{(0)} = S$.

For all integers, $j \geq 1$, if $S^{(j-1)}$ is fully partitioned,

i. replace each of the majority symbol substrings of $S^{(j-1)}$, by its length,

ii. delete the minority symbols of $S^{(j-1)}$,

iii. define the resulting string to be the j^{th} *order derived string* of S and denote it by $S^{(j)}$.

If for some n, $length(S^{(n)}) = 1$, then for all $k > n$, $S^{(k)}$ exists and $S^{(k)} = (1)$. Further, if all derived strings of S exist, then for some n, $length(S^{(n)}) = 1$. This follows from the observation that if $length(S^{(j-1)}) > 1$ and if $S^{(j)}$ exists, then $length(S^{(j-1)}) > length(S^{(j)})$.

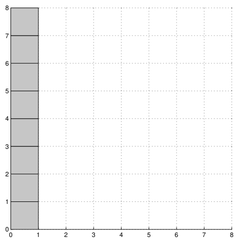
If S is not a string over an alphabet consisting of two consecutive positive integers, then $S^{(0)}$ is not defined. Further, if $S^{(j-1)}$ is not fully partitioned for some $j \geq 1$, then $S^{(j)}$ and higher order derived strings are not defined.

Cellular Line. A *cellular line* is a cell set that is represented by a string, S, such that the j^{th} order derived string of S exists for all $j \geq 0$.[17,18]

A cellular line may be defined in terms of strings only, omitting the geometry of cell sets in cellular arrays.[19] The definition implicitly defines an algorithm for determining whether or not a given cell set is a cellular line. If a cell set can be represented by a string, S, then for all $j \geq 0$ determine whether or not $S^{(j)}$ exists. This is a finite process since there is a j such that either $S^{(j)}$ does not exist or $length(S^{(j)}) = 1$.

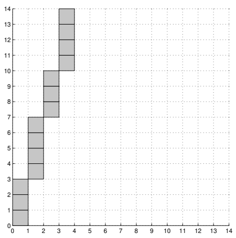
Examples. The following examples show how the definition of a cellular line implicitly defines an algorithm for determining if a given cell set is a cellular line. We use the abbreviation "fp" for fully partitioned.

Example 2



$S = (8)$.
$S^{(0)} = (8)$.
$length(S^{(0)}) = 1$.
The cell set is a cellular line.

Example 3



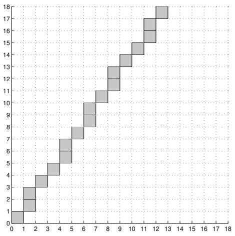
$S = (3\,4\,3\,4)$.
$S^{(0)} = (3\,4\,3\,4)$, $S^{(0)}$ is fp.
$S^{(1)} = (1\,1)$, $S^{(1)}$ is fp.
$S^{(2)} = (2)$, $length(S^{(2)}) = 1$.
The cell set is a cellular line.

17 There exists an integer, n, such that for $n \leq k$, $S^{(k)}$ exists and $length(S^{(k)}) = 1$.

18 The term "cellular" was used by (Kim82), in referring to a digitization process.

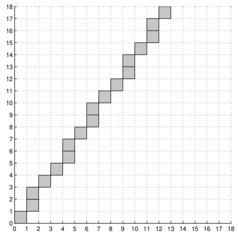
19 This is analogous to the classical definition of a curve in a topological space; a curve is a continuous function from the unit interval into the topological space.

Example 4



$S = (1\,2\,1\,1\,2\,1\,2\,1\,2\,1\,1\,2\,1)$.
$S^{(0)} = (1\,2\,1\,1\,2\,1\,2\,1\,2\,1\,1\,2\,1)$,
$S^{(0)}$ is fp.
$S^{(1)} = (1\,2\,1\,1\,2\,1)$, $S^{(1)}$ is fp.
$S^{(2)} = (1\,2\,1)$, $S^{(2)}$ is fp.
$S^{(3)} = (1\,1)$, $S^{(3)}$ is fp.
$S^{(4)} = (2)$. $length(S^{(4)}) = 1$.
The cell set is a cellular line.

Example 5



$S = (1\,2\,1\,1\,2\,1\,2\,1\,1\,2\,1\,2\,1)$.
$S^{(0)} = (1\,2\,1\,1\,2\,1\,2\,1\,1\,2\,1\,2\,1)$
$S^{(0)}$ is fp.
$S^{(1)} = (1\,2\,1\,2\,1\,1)$, $S^{(1)}$ is fp.
$S^{(2)} = (1\,1\,2)$, $S^{(2)}$ is not fp.
The cell set is *not* a cellular line.

7 Existence, Uniqueness, and Properties

Existence. It is clear that, given a string representation of a cellular line, one can determine the terminal cell of the cellular line. Less clear is the fact that there exists a cellular line with an arbitrarily selected terminal cell. The verification of this assertion appears in (GMU03). There it is shown that:

> For integers c and r, $1 \le c \le r$, there exists a cellular line with an initial cell in the first column and first row and a terminal cell in the c^{th} column and r^{th} row. It is represented by a string:
>
> i. over the alphabet $\{\lfloor \frac{r}{c} \rfloor, \lfloor \frac{r}{c} \rfloor + 1\}$,
> ii. whose length is c
> iii. whose components sum to r.

For example, when $c = 13$ and $r = 31$, the string representation of the corresponding cellular line is a string of length 13 over the alphabet $\{2,3\}$. Using a construction one finds the string representation to be (2 3 2 2 3 2 3 2 3 2 2 3 2).

Uniqueness. The above definition leaves open the possibility that there exists more than one cellular line with a given terminal cell. The strings (1 2 1 2) and (2 1 2 1) both represent cellular lines determined by the cell in the 4^{th} column and 6^{th} row.

> It is shown in (GMU03) that there exist at most two cellular lines with the same initial and terminal cells.

Palindromes. It follows from the Uniqueness Theorem, (GMU03), that when there is only one cellular line determined by a terminal cell, the string representation is a palindrome, (Gee02), (GMU03). Further, when there are two cellular lines, determined by a terminal cell, the string representations are not palindromes. The two strings, determined by fixed values of c and r, are dual to each other. For example the cell set shown in Fig. 6 can be represented by both the strings (1 2 1 2) and (2 1 2 1) depending upon the choice of coordinate system.

Finally it is observed that the string (1 2 2 1) is a palindrome but does not represent the cellular line determined by $c = 4$ and $r = 6$.

Comparisons. The following is a list of comparisons between cellular lines in a cellular array and chords in $\Re^2$.

- Chords (straight line segments) in $\Re^2$ have the property that their connected subsets are also straight lines. The chord determined by two points of a chord in $\Re^2$ is a subset of the original chord. This is

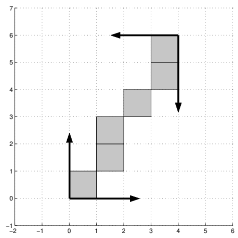


Fig. 6: (1 2 1 2) or (2 1 2 1)

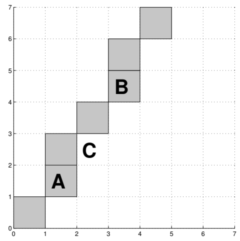


Fig. 7: (1 2 1 2 1)

not always the case for cellular lines. For example, the cellular line shown in Fig. 7 is represented by the string, (1 2 1 2 1). It includes cells *A* and *B* but not cell *C*. However, the cellular line with initial cell *A* and terminal cell *B*, represented by the string (1 2 1), does include cell *C*.

- Chords in $\Re^2$ have the property that they belong to exactly one (infinite) straight line. Cellular lines do not have such a property. Every cellular line has an infinite number of extensions that are cellular lines, (Gee02).
- Between every two points in $\Re^2$ there exists a unique chord in $\Re^2$. Cellular lines do not have this property. For example, the strings (1 2 1 2) and (2 1 2 1) represent two cellular lines with their terminal cell in the 4^{th} column and the 6^{th} row.
- A chord in $\Re^2$ is the unique geodesic between its two end points. There does not appear to be a direct cellular line analogue.

8 Ongoing Studies and Conclusions

The following list shows some ongoing studies that have emerged from this study of cellular lines.

- The questions of existence, uniqueness, and construction of cellular lines have been put on secure mathematical foundations, (GMU03), (Gee02).
- A notion of curvature of strings, used to characterize cellular lines, has been formulated, (Gee02).
- One can "integrate" string representations of cellular lines, in two dimensional cellular arrays, in order to define higher order cellular curves (quadratic, cubic, etc.), (Gee02).
- There is not a unique method of extending a cellular line, (Gee02). Ways of defining infinite cellular lines are under investigation.
- There are symmetries within the set of all cellular lines, (Gee02). Hexadecant symmetry is mentioned by (BB99).
- The construction of cellular lines reminds one of a growing process. This idea leads to the possibility of being able to grow "arbitrary" cellular curves. An algorithm that models the growing process has been formulated but substantial experimentation is needed.

- An "ant algorithm" for finding cellular lines has been formulated, (GMU01).
- A definition of a cellular line in three-dimensions has been formulated. It builds on the definition in two-dimensions presented in this paper and is discussed in (GMSU03).
- A definition of a convex cell set has been formulated and will be reported.

We began with the idea of defining geometric objects in a cellular array that are completely independent of Euclidean notions. The work presented here provides a starting point: a definition of a cellular line without the aid of Euclidean constructs and a way of determining whether or not a cell set is a cellular line.

Acknowledgements

The authors would like to thank Ken Brodlie for his helpful comments during the writing of this paper. The authors would also like to thank Reinhard Klette and Azriel Rosenfeld for their remarks on an earlier version of this paper.

References

[BB99] Vincent Boyer and Jean-Jacques Bourdin. Fast lines: A span by span method. *Computer Graphics Forum, Eurographics 99*, 18, 1999.

[BB00] Vincent Boyer and Jean-Jacques Bourdin. Auto-adaptive step straight-line algorithm. *IEEE Computer Graphics and Applications*, pages 67–69, Sep./Oct. 2000.

[Bre65] Jack E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4:25–30, 1965.

[Bro74] Reyer Brons. Linguistic methods for the description of a straight line on a grid. *Computer Graphics and Image Processing*, 3:48–62, 1974.

[Bro85] Reyer Brons. Theoretical and linguistic methods for describing straight lines. In Rae A. Earnshaw, editor, *Fundamental Algorithms for Computer Graphics*, volume 17, pages 19–57. Springer-Verlag, Heidelberg, 1985.

[Bru89] Alfred M. Bruckstein. Self-similarity properties of digitized straight lines. In Robert A. Melter, Azriel Rosenfeld, and Prabir Bhattacharya, editors, *Contemporary Mathematics: Vision Geometry*, volume 119, pages 1–20. Proceedings of an AMS Speical Session, American Mathematical Society, Oct. 1989.

[DMC96] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on System, Man and Cybernetics-Part B*, 26(1):1–13, 1996.

[Fre61] Herbert Freeman. On the encoding of arbitrary geometric configurations. *The Institute of Radio Engineers Inc. (IRE) Transactions on Electronic Computers*, EC-10:260–268, 1961.

[Fre70] Herbert Freeman. Boundary encoding and processing. In *Picture Processing and Psychopictorics*, pages 241–266. Academic Press, New York - London, 1970.

[Gee02] Panama Geer. *On Cellular Lines*. Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, New York, May 2002. http://mainline.brynmawr.edu/∼pgeer/Research/.

[GMSU03] Panama Geer, Harry W. McLaughlin, David Scherzer, and Keith Unsworth. 3D Cellular lines: A definition. In preparation, April 2003.

[GMU01] Panama Geer, Harry W. McLaughlin, and Keith Unsworth. Discrete lines and ant algorithms. Technical report, Lincoln University, Canterbury, New Zealand, 2001.

[GMU03] Panama Geer, Harry W. McLaughlin, and Keith Unsworth. Cellular lines: Existence, uniqueness, and construction. In preparation, April 2003.

[HKV81] Albrecht Hübler, Reinhard Klette, and Klaus Voss. Determination of the convex hull of a finite set of planar points within linear time. *Elektronische Informationsverarbeitung und Kybernetik (EIK, Journal of Information Processing and Cybernetics)*, 17:121–139, 1981.

[Kim82] Chul E. Kim. On cellular straight line segments. *Computer Graphics and Image Processing*, 18:369–381, 1982.

[KRS98] Reinhard Klette, Azriel Rosenfeld, and Fridrich Sloboda. *Advances in Digital and Computational Geometry*. Springer, Singapore, 1998.

[RK82] Azriel Rosenfeld and Chul Kim. How a digital computer can tell whether a line is straight. *American Mathematics Monthly*, 89:230–235, 1982.

[RK01] Azriel Rosenfeld and Reinhard Klette. Digital straightness. *Electronic Notes in Theoretical Computer Science*, 46, 2001.

[Ros74] Azriel Rosenfeld. Digital straight line segments. *IEEE Transations on Computers*, C-23:1264–1269, 1974.

[SL00] Peter Stephenson and Bruce Litow. Why step when you can run? Iterative line digitization algorithms based on hierarchies of runs. *IEEE Computer Graphics and Applications*, pages 76–84, Nov./Dec. 2000.

[Ste98] Peter Stephenson. *The Structure of the Digitised Line: With Applications to Line Drawing and Ray Tracing in Computer Graphics*. Department of Computer Science, James Cook University of North Queensland, North Queensland, Australia, 1998.