On Curve Rasterization in n-Dimensional Space

Charles Wüthrich

Faculty of Media
Bauhaus-University Weimar
D-99421 Weimar (GERMANY)
E-Mail: caw@informatik.uni-weimar.de

Abstract : The rapid development of scanning and measuring hardware for medical imaging and for scientific experiments, the introduction of animation techniques into common use have created the need to understand n-dimensional raster geometry, where $n > 3$. This paper presents therefore a discrete regular structure called *hyperlattice* used for defining n-dimensional raster geometry. Discrete curves are then defined on hyperlattices. A general definition of rasterization onto hyperlattices is then given, and algorithms for rasterizing generic curves and straight lines onto n-dimensional lattices are presented.

1 Introduction

The rapid development of scanning and measuring hardware for medical imaging and for scientific experiments, the introduction of animation techniques for the visualization of data, and the need for examining and manipulating data of growing complexity and dimensionality have created new problems regarding sampling, representing and rendering data defined in n-dimensional spaces for use in the discrete world representable in computers.

Rasterization is the operation that allows passing from the human continuous representation of the world to the discrete digital world of computing devices. While the study of rasterization in two-dimensional spaces is well advanced [12, 8, 25] and many algorithms for performing discretization onto the plane are readily available, the study of this operation in n-dimensional spaces, with $n \geq 3$, are a rarity and generally limited to the three-dimensional case [23, 11, 1, 8, 17]

Due to the recent progress in scientific visualization and in medical imaging, there has been a renewed interest in new methods for the visualization of n-dimensional spaces [28, 4, 16, 22, 6] and for methods for integer interpolation in higher-dimensional spaces, [3, 20]. To avoid the development of mutually incompatible theories, a general framework for the development of the theory for n-dimensional rasterization is needed.

This paper tries to address exactly these needs: it provides a general mathematical model for multidimensional raster devices, and extends currently available rasterization schemes to the n-dimensional case. The paper also shows the strict interdependence of the type of neigbourhood relations chosen in the discrete model space (which influences the connectivity of a discrete curve) with the rasterization scheme that has to be adopted. Finally, an algorithm for the rasterization of generic curves, and its application to n-dimensional straight lines are presented. These algorithms allow to generate curves having all examined types of connectivity, and are therefore a generalization of readily available algorithms from the literature [3, 20].

2 Raster I/O devices

As the support environment for this paper we take the real n-dimensional space $\mathbf{R}^n$, with the common geometric definitions of point, straight line, plane, flat, hyperplane and polytope [7, 21].

The most common type of output device nowadays connected to a computer is a raster device. By raster device we understand here a device which can represent a set of discrete values $i_1, i_2, \ldots, i_m$ on a discrete subset L of the n-dimensional real space. In symbols, a raster device A is a device that can represent a discrete-valued function

$$D : L \longrightarrow I_1 \times I_2 \times \ldots I_m ,$$

where $I_1, I_2, \ldots I_m$ are discrete (usually finite) subsets of the real numbers. Typical examples of raster output devices in two-dimensional space are traditional Computer Graphics square-based output devices, such as raster screens and dot-matrix, inkjet or laser printers, the domain of which (L) is usually a rectangular subset[1] of $\mathbf{Z}^2$ and the codomain of which is the set $\{0, 1\}$ for black and white raster screens and printers,, the set $\{0, \ldots, 255\}$ for 8 bit greyscale display devices, the set $RGB = \{0, \ldots, 255\} \times \{0, \ldots, 255\} \times \{0, \ldots, 255\}$ for 24 bit display devices, or a set of colours $c_1, \ldots, c_N$ [2] for colour lookup table (CLUT) devices. Typical raster input devices are two-dimensional scanners, which usually can output any of the formats discussed above. Examples of three-dimensional raster input devices are medical imaging devices, such as Magnetic Resonance and Computerized Tomography devices, which have $\mathbf{Z}^3$ as a domain, and output one value per position. Recently, four-dimensional medical input devices have appeared, e.g. Positron Emitting Tomography scanners, which scan the variations of three-dimensional data in time, and thus data in four dimensions. On the output device side, the first three-dimensional output devices are making their appearance, for example the work at the University of Braunschweig [14]. They are capable of projecting a given colour defined as above onto a raster point in 3D space.

The use of a unified model for raster devices allows the same notation both for raster input devices, such as scanners, and for raster output devices, such as printers and raster screens. In particular, the operation of scanning, i.e. of sampling a continuous signal through a raster device into a discrete dataset for its elaboration through a computer, looks similar, in principle, to the operation of rasterizing, i.e. of discretizing the representation of data through continuous curves, polygons, and all the components of a geometric model, into a discrete set of data representable on a raster output device. In both cases, due to the discrete nature of the device, the sampling process can only be done at fixed frequencies, thus causing a noticeable loss of detail as well as aliasing problems. The similarities in the two sampling processes end here, though. Scanning devices are limited by their resolution, and data can be sampled only at fixed frequencies. The resolution limits of the device cannot be overcome without an increase of the sampling resolution, i.e. without buying a higher resolution device, or through new devices that place the sampling sets more efficiently, as has been proposed recently [18, 7, 26]. Raster output devices, instead, permit a different approach, since the continuous model that has to be sampled is readily available in the computer. The sampling frequency of the model can thus be increased at will (super-sampling) and sampling errors can be averaged, or partially corrected through anti-aliasing techniques before displaying on the physical

[1] The range of which, of course, depends on the resolution of the device.

[2] Here each colour c_i is a triplet of integers in the set of all possible colours representable by the device, which, in general, is a set of type $\{0, \ldots, N_1\} \times \{0, \ldots, N_2\} \times \{0, \ldots, N_3\}$.

device [10]. However, such techniques are in general very expensive, and do not address the core problems caused by the rasterization process. As many authors have pointed out [12, 13, 8], there is a real lack of study on the mathematical foundations of rasterization, and the notation used is either derived from the theory of sampling, and thus is too heavy, or is too incomplete for productive use.

3 Rasterization in n-dimensional space

In this section we will focus on the introduction of a model for rasterization based on regular discrete structures called n-dimensional lattices.

The operation of sampling a continuous m-dimensional signal $f : \mathbf{R}^n \longrightarrow \mathbf{R}^m$ to obtain an m-dimensional discrete signal of a discrete variable $D : L \longrightarrow I_1 \times I_2 \times \ldots I_m$ which can be output on a raster display device is called *discretization* or *rasterization* of the signal f. A discretization function ρ therefore associates with a function f which has to be discretized its discretization D, *i.e.* $\rho : \mathcal{F} \longrightarrow \mathcal{D}$, where $\mathcal{F} = \{f : \mathbf{R}^n \longrightarrow \mathbf{R}^m\}$ and $\mathcal{D} = \{D : L \longrightarrow I_1 \times \ldots \times I_m\}$. From the above definitions it can be observed that discretization is, in fact, a particular sampling process in the n-dimensional space that can be characterized by the fact that also the output set $I = I_1 \times I_2 \times \ldots I_m$ is also discrete.

In general, for Computer Graphics purposes, both L and I are subsets respectively of $\mathbf{R}^n$ and $\mathbf{R}^m$, and the discretization is done so that some visually important characteristics of the function f are preserved. In particular, discretization is performed in such a way that the embedding of $\rho(f)$ in $\mathcal{F}$ is an "acceptable" approximation of f. The criteria for the definition of acceptable vary widely, and depend mainly on the function ρ used.

Some rasterization schemes can be directly derived as the n-dimensional extension of existing rasterization schemes for two-dimensional rasterizations. Let $\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_{n-1}$ be n linearly independent vectors of $\mathbf{R}^n$, and let $\Lambda = \Lambda(\mathbf{v}_0, \ldots, \mathbf{v}_{n-1})$ be an n-dimensional lattice, i.e. the set of all linear combinations with integer coefficients of the vectors $\mathbf{v}_i$, or, in other terms, the points λ of $\mathbf{R}^n$ such that $\lambda = c_0\mathbf{v}_0 + \ldots + c_{n-1}\mathbf{v}_{n-1}$ (where $c_i \in \mathbf{Z}$). The vectors $\mathbf{v}_i$ are usually called *lattice generators*. Consider the Voronoi sets associated to the lattice points, i.e. the sets of points V_λ closer to the point $\underline{\lambda}$ that to any other point of the lattice Λ. Each Voronoi set V_λ is in a natural correspondence with the lattice point λ. Depending on the mutual orthogonality of the lattice generators, neighbouring Voronoi sets share a single point, a straight line segment, up to an $(n-1)$-dimensional polytope. Two lattice points λ and λ' are said to be *l-neighbours* if the closures of the corresponding Voronoi sets V_λ and $V_{\lambda'}$ share an l-dimensional polytope. Two lattice points λ and λ' are said to be *$l_\geq$-neighbours* if the closures of the corresponding Voronoi sets V_λ and $V_{\lambda'}$ share a k-dimensional polytope (where $0 \leq l \leq k \leq (n-1)$). In the traditional notation used for the three-dimensional case (voxel space), for mutually orthogonal lattice generators, $2_\geq$-neighbours are 6- (or face-) connected neighbours, $1_\geq$-neighbours are 18- (or line-) connected neighbours, and $0_\geq$-neighbours are 26- (or vertex-) connected neighbours.

Arcs can then be introduced on an n-dimensional lattice with the same procedure used on infinite graphs: given two lattice points A and B, an *$l_\geq$-arc* from A to B is a finite sequence of lattice elements $P_0, P_1, \ldots, P_n$ such that $\forall i$, P_i and P_{i+1} are $l_\geq$-neighbours and such that $P_0 = A$ and $P_n = B$ [19]. The points A and B are called *endpoints* of the arc. Note how this definition corresponds to the definition of the path between two nodes of a graph. An arc is said to be *closed* if the two endpoints coincide. An *$l_\geq$-curve* is an infinite arc, i.e. a sequence of lattice points $\{P_i\}_{i=-\infty}^{+\infty}$ such that P_i and P_{i+1} are $l_\geq$-neighbours $\forall i$.

Consider now the Voronoi set associated with the origin V_O, and a simply connected set K_ρ such that $K_\rho \subseteq V_O$. The set K_ρ is called the *basic domain of the rasterization scheme*. Let $\lambda \in \Lambda$, and let $K_\rho(\lambda)$ be the translated set of K_ρ through the vector λ, in other words, let $K_\rho(\lambda) = \{\mathbf{y} \in \mathbf{R}^n \; : \; \mathbf{y} = \lambda + \mathbf{x}, \text{where}\, \mathbf{x} \in K_\rho\}$. The rasterization of a point and of a set can be defined in the following way:

Definition 3.1 *Let $\mathbf{v} \in \mathbf{R}^n$. The rasterization $Dig_{K_\rho}(\mathbf{v})$ of $\mathbf{v}$ is defined by the following relation*

$$Dig_{K_\rho}(\mathbf{v}) = \begin{cases} \{\lambda\} & \textit{iff}\ (\lambda \in \Lambda) \ \wedge \ (\mathbf{v} \in K_\rho(\lambda)) \\ \\ \emptyset & \textit{otherwise.} \end{cases} \tag{1}$$

Definition 3.2 *Let $A \subseteq \mathbf{R}^n$. The rasterization of A is the set*

$$Dig_{K_\rho}(A) = \bigcup_{\mathbf{x} \in A} Dig_{K_\rho}(\mathbf{x}) \,. \tag{2}$$

The rasterization function introduced by these definitions is called *nearest neighbour* rasterization. Although it is defined exactly in the same way as two-dimensional rasterization is, the operation of rasterizing in n-dimensional space is, of course, more complicated, and requires particular care in the choice of the set K_ρ. If $K_\rho = V_O$, then the rasterization function is called *cellular rasterization*. The major drawback of cellular rasterization is the fact that in the case of n orthogonal lattice generators, whenever a curve has to be rasterized, its cellular rasterization in Λ is a $0_\geq$-curve. Although this is a desirable feature for hypervoxel[3] traversal algorithms, in some cases [2] it is more desirable to compute $l_\geq$-curves for rasterization in the n-dimensional lattice, where $l > 0$, just as in the two-dimensional case, where 8-connected curves are usually preferred to 4-connected curves.

As in the two-dimensional case, there is a particular choice of the set K_ρ that generates $(n-1)$-connected curve digitalizations in the case of orthogonal lattice generators. If the vectors $\mathbf{v}_i$ are orthogonal, by letting one of the axes coincide with the straight line containing one generator, we can always lay an orthogonal cartesian coordinate system in $\mathbf{R}^n$ such that all the coordinates of the vector $\mathbf{v}_i$ are zero except the i-th coordinate, $v_{i,i}$. Let A_i be the hyperplane passing through the origin and perpendicular to the vector $\mathbf{v}_i$. Consider the Voronoi set associated with the origin V_O, and let $A'_i = A_i \cap V_0$. Consider now as basic domain of the rasterization scheme the set

$$K_\rho = \bigcup_{i=0}^{n-1} A'_i \,.$$

The scheme defined in this way computes the intersections of the curve γ to be rasterized with the hyperplanes of the form $x_i = j v_{i,i}$, where j is an integer, and then approximates the resulting points to their nearest lattice point[4]. The resulting lattice points build the rasterization of the curve. This last type of rasterization is called *grid intersection* rasterization.

Grid intersection rasterization can be extended to non-orthogonal lattices by substituting the hyperplanes A_i with the hyperplanes $\overline{A}_i$ passing through the origin and containing all vectors $\mathbf{v}_j$ for all $j \neq i$. In the lattice coordinate system, such hyperplanes have the form $x_i = 0$, with k integer. To rasterize a curve γ in this case, its intersections with the grid of hyperplanes of the form $x_i = j$ (in the lattice coordinate system) have to be computed, and the resulting points are then approximated to the nearest lattice point on the hyperplane.

[3] Where a hypervoxel is by definition the set V_λ.

[4] Such a point is not always unique. However, this ambiguity can be trivially overcome by forcing uniqueness.

In general, cellular rasterization schemes guarantee that the rasterization of any given point P exists, since the Voronoi sets associated with a lattice tessellate $\mathbf{R^n}$, and should be therefore used to rasterize single points, whereas grid intersection rasterizations are more convenient whenever a curve has to be rasterized, since they perform intersections with a set of parallel hyperplanes, the equations of which are easy to handle.

Nearest neighbour rasterization schemes provide a fairly simple mechanism for the definition of rasterization. However, they are not extensible to take into account the concept of $l_{\geq}$-neighbourhoods. In order to generate the discretization of a set A, such schemes use replicas of one single (fixed) set, and compute the intersections of the set to be rasterized with these copies. The result of this procedure is then a connected set of lattice points that represent the set A on the lattice, and that lies the closest possible to the set to be rasterized, within the limits imposed by the discrete nature of lattices and by the relation of neighbourhood involved. With the schemes introduced above, there is one and only one optimal choice of the resulting rasterization set[5].

4 Curve rasterization onto n-dimensional lattices

To extend the operation of rasterization to lattices onto which a generic $l_{\geq}$-neighbourhood is defined, from now on in this paper we restrict ourselves to the definition of *curve rasterization* onto n-dimensional lattices, since a definition of rasterization for generic sets would imply a thorough analysis of the topology of n-dimensional lattices, and this would be well beyond the scope of this paper. The notation presented above for the definition of rasterization, although more general, thus, turns out to be too heavy: the operation of curve rasterization can also be seen as the rasterization of subsets of $\mathbf{R}^{n+1}$. In fact, a curve is a particular subset of $\mathbf{R} \times \mathbf{R}^n$, namely a continuous function $\gamma : \mathbf{R} \longrightarrow \mathbf{R}^n$, and therefore its rasterization must be a curve in a lattice $\Lambda \subset \mathbf{R}^n$, i.e. a sequence $S : \mathbf{Z} \longrightarrow \Lambda$ such that $S(i)$ and $S(i+1)$ are neighbours. The discrete curve resulting from the rasterization process will therefore be a connected subset of points of Λ that lies the closest possible to the curve to be rasterized.

The keywords for the definition are the words *connected* and *closest*. Connectivity is the characteristic which is most important from an observer's standpoint: a rasterization of a curve must be connected. As far as closeness is concerned, an observer is usually less categorical about it, and depending on the requirements of the system this closest criterion can be relaxed [9, 15]. For example, in two-dimensional rasterizations, for speed reasons a polygon inscribed in a curve might be rasterized instead of the curve itself [24]. The rasterization scheme, therefore, must ensure curve connectedness first.

The choice of the closest $l_{\geq}$-path to the curve γ to be rasterized is, in general, more complicated than the choice of a closest $0_{\geq}$-path or $(n-1)_{\geq}$-path to γ, because $0_{\geq}$- and $(n-1)_{\geq}$-rasterizations involve only local decisions in the rasterization process, whereas generic $l_{\geq}$-rasterizations have also to take into account conditions for the neighbours of the current point.

Similarly to grid intersection rasterizations, the starting point for the definition of an $l_{\geq}$-path generating scheme, is calculating the intersection of the continuous curve γ with a grid of lines, so as to partition the rasterization procedure into an (enumerable) sequence of steps. However here the choice of the next rasterization point has to take into account the local neighbourhood configurations, which impose additional conditions on the choice

[5]In fact, all the schemes introduced to date avoid in one way or another the ambiguities derived from the points on the border of V_λ, which inherently are points shared by more than one K_λ set, by arbitrarily attributing these points to one and only one K_λ

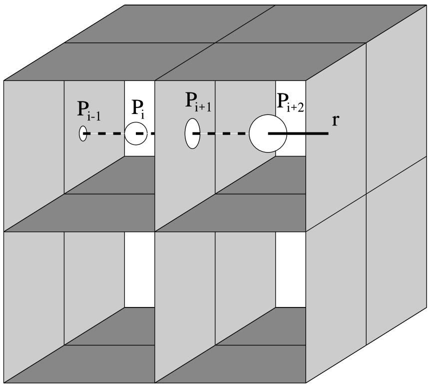

Figure 1: Intersection of a curve r with the grid hyperplanes.

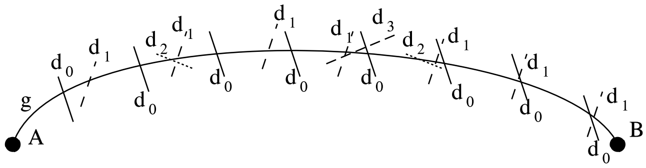

Figure 2: Grouping 1-steps into $l_{\geq}$-steps for producing the rasterisation of a curve r.

of the points of the discrete curve.

Let $\Lambda(\mathbf{v}_0, \ldots, \mathbf{v}_{n-1})$ be the lattice onto which we want to perform rasterization, let $\gamma : [0,1] \longrightarrow \mathbf{R^n}$ be the curve to be rasterized, and let A and B be its endpoints. Let the coordinates of the generator $\mathbf{v}_i$ be $(v_{i,0}, \ldots, v_{i,n-1})$ in the orthogonal coordinate system. Consider the coordinate system introduced by the lattice generators. In this coordinate system, the coordinates of the generator $\mathbf{v}_i$ will all be zero, except the i-th coordinate which will be 1. In the coordinate system introduced by the lattice generators in $\mathbf{R^n}$, consider the hyperplanes of the form $x'_i = k$. In other words, consider the hyperplanes $H_{k\mathbf{v}_i}$ whose equation in the orthogonal coordinate system is

$$v_{i,0}x_0 + \ldots + v_{i,n-1}x_{n-1} = k(v_{i,0}^2 + \ldots + v_{i,n-1}^2)\,,$$

where $k \in \mathbf{Z}$. The hyperplanes $H_{k\mathbf{v}_i}$ form a grid over $\mathbf{R^n}$, which will be intersected with the curve to be rasterized to obtain the lattice rasterization points. Note that, in the case of rectangular lattices, the grid defined here coincides with the grid which derives from grid intersection rasterization.

Let λ_A and λ_B respectively be the grid point digitalizations of A and B,[6] and consider the intersections of γ with the grid of the hyperplanes $H_{k\mathbf{v}_i}$. For each point P, intesection between γ and the hyperplane $H_{k_P\mathbf{v}_i}$, let P' be its nearest lattice point on the hyperplane $H_{k_P\mathbf{v}_i}$. The paramenter t induces an ordering on the points of the curve γ, and thus also on the intersections of γ with the grid. Let $S = \{P_0, \ldots, P_r\}$ be the ordered sequence of intersections of γ with the hyperplanes $H_{k\mathbf{v}_i}$. This sequence represents the order in which the hyperplanes $H_{k\mathbf{v}_i}$ will be intersected by the curve γ for growing values of t, as illustrated in Figure 1.

The sequence S induces also an ordering on the sequence S' of the P'_i, i.e. of the nearest neighbours defined above. Note that the P'_i are lattice points, and will be used to build the $l_{\geq}$-path, rasterization of γ. From this sequence, it is easy to extract an $l_{\geq}$-connected path: up to $n-l$ steps along the directions of the generators can be grouped into a single diagonal step, provided that a single ("combined") step is not composed of more than one step in each direction.

To illustrate this a little, imagine an observer travelling in the n-dimensional space from A to B along the curve γ. Such an observer would "bump" into the hyperplanes

[6] From a logical point of view, if we denote the rasterization of γ with Γ, both λ_A and λ_B should belong to Γ. However, in order to maintain consistency, in traditional rasterization algorithms this is not always the case, and here we shall follow the same approach.

of the grid at the points P_i in a certain order, which defines a sequence S of points on the grid hyperplanes. Each time the observer encounters a hyperplane in a point P, he scrupulously takes note of the point encountered, looks for the closest lattice point to it, say P', and takes note of it too. Once the observer arrives at B, he has built two sequences of points S and S', respectively containing the points P and P', from which it is possible to build a shortest $l_{\geq}$-path from A to B. Let us call this path Γ. In order to do this, it suffices to note that each point P' in S' represents a step in the direction d perpendicular to the grid hyperplane containing P'. The sequence S' therefore defines a sequence D of single steps in the directions of the axes which builds a 1-path from A to B. From the sequence D we can obtain an $l_{\geq}$ path Δ by simply grouping together whenever possible in each hop up to $n-l$ different steps in the directions of the generators.

Let $D = \{d_{i_0}, \ldots, d_{i_m}\}$. Here, naturally, the elements of D are directions parallel to the lattice generators. Let i_{h_0} be the minimum index of the sequence D that has appeared before in D. Let $k_0 = Min(l, h_0)$. In the first hop, the first $k_0 \leq l$ steps of D in different directions of the single generators can be grouped in a single hop. Let us denote this fact by rewriting D and enclosing grouped steps in brackets. We have $D = \{(d_{i_0}, \ldots, d_{i_{k_0}}), d_{i_{k_0+1}}, \ldots, d_{i_m}\}$. Consider now the subsequence extracted from D starting from i_{k_0+1}, i.e. the subsequence $D_{k_0} = \{d_{i_{k_0+1}}, \ldots, d_{i_m}\}$ and, again, let i_{h_1} be the minimum index of the sequence that has appeared before in D_{k_0}. Let $k_1 = Min(l, h_1)$. In the second hop of the $l_{\geq}$ path, the steps from $d_{i_{k_0+1}}$ to $d_{i_{k_1}}$ can be grouped together in a single step. We can thus rewrite D as

$$D = \{(d_{i_0}, \ldots, d_{i_{k_0}}), (d_{i_{k_0+1}}, \ldots, d_{i_{k_1}}), d_{i_{k_1+1}}, \ldots, d_{i_m}\}\,.$$

This can be repeated until all the directions of D are grouped in hops. The result of this procedure is the grouping of the directions in D into $l_{\geq}$ hops which define an $l_{\geq}$ path from A to B. Such a path represents the rasterization of the curve γ.

Figure 2 illustrates this in an example. The curve $g \in \mathbf{R^4}$ has to be rasterized onto an orthogonal $1_{\geq}$-connected[7] lattice Λ from the point A to the point B. Since we are working in 4-dimensional space, there will be four lattice generators, and therefore four parallel sheafs of hyperplanes (perpendicular to the lattice generators) which will be used to compute the rasterization of g. In the figure, the projection of g onto a convenient plane is shown. The diagonal segments intersecting the curve represent the intersections of the curve with the sheafs of hyperplanes of equation $x_i = k$ $(i = 0, \ldots, 3)$, where parallel hyperplanes have been represented by parallel segments. The intersections found define a 1-connected path from A to B, the steps of which build the sequence of directions

$$D = \{d_0, d_1, d_0, d_2, d_1, d_0, d_1, d_0, d_1, d_3, d_0, d_2, d_1, d_0, d_1, d_0, d_0, d_1\}\,.$$

The directions of D are grouped to form a $1_{\geq}$ path in the following way:

$$D = \{(d_0, d_1), (d_0, d_2, d_1), (d_0, d_1), (d_0, d_1, d_3),\ (d_0, d_2, d_1), (d_0, d_1), (d_0), (d_0, d_1)\}\,.$$

Note that here one step in the direction d_0 cannot be grouped together with any other step, due to the fact that d_0 occurs twice in a row. Note also that the chosen path is not the only $3_{\geq}$-path possible: for example also the path

$$D = \{(d_0, d_1), (d_0, d_2, d_1), (d_0, d_1), (d_0, d_1, d_3),\ (d_0, d_2, d_1), (d_0, d_1), (d_0), (d_0, d_1)\}\,.$$

is an $l_{\geq}$-connected path from A to B, and represents g accurately. This non-uniqueness is due to the nature of $l_{\geq}$ paths. However, all possible paths derived from D by grouping

[7] Note that orthogonality is required here to allow $l_{\geq}$-connectedness.

together up to 3 different directions will be close enough to the original curve to represent it "well". A more precise definition of rasterization which would compute a unique path would have to compromise greatly in speed, and is thus not desirable, since all possible legal groupings within D would have to be analyzed. The algorithm defined above follows the curve from A to B and groups directions as soon as the occasion arises.

To recapitulate, several rasterization techniques have been introduced. The use of each of them depends on the object to be rasterized, on the lattice onto which rasterization is to be performed and on the neighbourhood relation defined onto it. If, as in most cases, there is no requirement for $l_{\geq}$ connectedness, then it is much easier to use nearest neighbour rasterizations. Cellular rasterization is preferred if single points or generic objects have to be rasterized, while curves are preferably rasterized through grid intersection rasterization schemes. However, there are cases, for example when conditions on the smoothness of the rasterized object are imposed, or in interconnection network routing, where the $l_{\geq}$-connectedness of the rasterization is required. In these cases the last method presented is capable of finding an $l_{\geq}$-connected rasterization of the curve. In the next section, we will define an algorithm that rasterizes straight line segments.

5 Straight line rasterization

Although there is plenty of literature on straight line rasterization onto square planar lattices [5], only recently has line rasterization onto hexagonal lattices been introduced [25, 27]. In three-dimensional space, straight line drawing algorithms have been developed in the context of ray-tracing for following a ray through the scene to be rendered [1]. Lately there has been growing interest on line segment rasterization algorithms in n-dimensional space. Badouel and Wüthrich [2] presented a face connected algorithm which rasterized straight line segments onto n-dimensional hypercubes. In an independent work, Slater [20] published two algorithms that generated both the $(n-1)_{\geq}$-connected and the $0_{\geq}$-connected rasterization of straight line segments onto a hyperrectangular lattice. The purpose of this section is to illustrate the definition of rasterization presented in the last chapter through the development of an algorithm for the rasterization of a straight line onto an $l_{\geq}$-connected hypercubic lattice. We shall start by introducing one such algorithm for the hypercubic unit lattice Λ_u, i.e. such that its generators are all of unit length and mutually orthogonal.

Let $P = (p_0, p_1, \ldots, p_{n-1})$ and $Q = (q_0, q_1, \ldots, q_{n-1})$ be two distinct hyperlattice points. Let $n_i = q_i - p_i$. The straight line from P to Q is the set of points $X = (x_0, x_1, \ldots, x_{n-1})$ such that $x_i = (q_i - p_i)t + p_i$, where $t \in [0, 1]$. As seen in the previous section, the parameter t introduces an ordering on the points of the straight line. For each dimension i, consider the straight line points P_{j,h_i} obtained for $t = \frac{h_j}{n_i}$, where $h_j = 1, \ldots, n_i$, and order these points in increasing order of their corresponding parameter value. The segment PQ results subdivided into n_i equal parts for each dimension i, and the points obtained on the straight line segment are ordered by increasing values of the parameter t to form a sequence $A = \{A_1, \ldots, A_r\}$ which in turn is used to build the $l_{\geq}$-connected path as in the previous section.

The algorithm outlined above can be efficiently implemented to perform integer calculations only, and to generate one hop for each of its steps. To do this, for each dimension i, an integer counter d_i is allocated ($i = 0, \ldots, n-1$). The values of n_i, together with the least common multiple $L = LCM(\{n_i\})$ are then computed. Let $n''_i = \frac{L}{n_i}$, and let $n'_i = 2n''_i$. The n'_is represent the increment that will be used for the counter d_i throughout the whole algorithm. The counters d_i are initialized to n''_i. At each step, the set D of the l smallest counter cells d_i is considered. Let d_{i_0} be the smallest counter cell, and let n'_{i_0} be

Initialization:
  $\forall i,\ n_i \leftarrow q_i - p_i$ END$\forall$
  $L \leftarrow \mathrm{LCM}_i(n_i)$
  $\forall i\ n''_i \leftarrow \frac{L}{n_i}\ n'_i \leftarrow 2 \cdot n''_i$ END$\forall$.
Translation:
  $\forall i,\ d_i \leftarrow n''_i$ END$\forall$
Loop:
  WHILE $(\forall i,\ d_i \leq 2L)$
    Consider the l minimum d_i,
    and order them in the sequence
    $d_{i_0} \leq \ldots \leq d_{i_{l-1}}$
    IF$(d_{i_k} + n'_{i_k} \geq d_{i_l},\ \forall k < l)$
      $\forall k < l$,
        $d_{i_k} \leftarrow d_{i_k} + n'_{i_k}$
        $x[i_k] \leftarrow x[i_k] + 1$
      END$\forall$
    ENDIF
    ELSE
      Let m be the minimum $d_{i_r} + n'_{i_r}$,
      and let l' be the maximum i_k
      such that $d_{i_{l'}} \leq m$:
      $\forall k \leq l'$,
        $d_{i_k} \leftarrow d_{i_k} + n'_{i_k}$
        $x[i_k] \leftarrow x[i_k] + 1$
      END$\forall$
    ENDELSE
    Discard directions such that $d_i > 2L$
    Write $X = (x[0], \ldots, x[n-1])$
  ENDWHILE

Figure 3: An $n_{\geq}$-connected straight line rasterization algorithm

the increment corresponding to it. First n'_{i_0} is added to d_{i_0}. Then the set D is reordered by increasing values of the counter cells. Again, the corresponding increment n'_{i_1} is added to the smallest d_i, until either all l elements of D have been incremented, or the smallest d_i is a counter cell that has already been incremented in the current step. Once one of these two conditions is fulfilled, the directions corresponding to the elements of D that have been incremented in the step are grouped together in one single hop. The steps are repeated until all counters d_i have reached the value of $2L + n''_i$, which is equivalent to the condition $\forall i, d_i \geq 2L$. To avoid missing the endpoint Q, whenever a counter d_i is greater than or equal to $2L$, it cannot contribute any more to the hops, and it is therefore removed from the list of the incrementable d_i counters. This condition is unnecessary if there is the need to draw an "infinite" line. Whenever two counter cells d_i have equal values in a certain step, an arbitrary choice can be made: here the direction corresponding the lowest dimension (i.e. corresponding to the smallest i) is considered first (as if it was smaller), unless it has already been incremented in the current step in which case it is considered to be bigger than all non-incremented counter cells equal to it.

Note that the initialization of the counters d_i is important, since otherwise in the first step all d_i would be equal to zero, and this would imply an arbitrary choice of the directions of the first step.

The algorithm translates into the pseudocode listed in Figure 3. It is a little slower than traditional algorithms developed for the 0- and $(n-1)$-connected cases, which are to be preferred whenever possible. This is due to the inherent complexity of $l_{\geq}$-connectedness.

The algorithm presented above can be extended trivially to a generic hypercubic lattice Λ_c by using an affine transform for transforming the points of the generic hypercubic lattice into points of a unitary hypercubic lattice Λ_u and transforming back the resulting rasterization through the inverse affine transform. More generally, the algorithm above can be trivially extended to hyperrectangular lattices. In fact, given a hyperrectangular lattice Λ_r and a hypercubic lattice Λ_c, there is an affine transformation A that converts the generators of Λ_r into the generators of Λ_c. To compute a hyperrectangular $l_{\geq}$-connected straight path between two points P and Q in Λ_r, it suffices again to compute an $l_{\geq}$-connected straight path from $A(P)$ to $A(Q)$, and to transform the resulting rasterization points through the inverse of A.

Finally, a small remark has to be made on the generation of the rasterization of a straight line segment from P to Q onto a non-orthogonal lattice Λ. In this case also, there exists a simple affine transformation $\overline{A}$ mapping the generators of Λ into the generators of Λ_u. As for all affine transformations, this transformation guarantees that lattice points will be mapped into lattice points, and that both grid hyperplanes and segment midpoints will be preserved. To find the 1-connected rasterization of a segment onto a generic lattice Λ, the 1-connected rasterization of the segment from $\overline{A}(P)$ to $\overline{A}(Q)$ can be computed and transformed back through the inverse of $\overline{A}$. The resulting rasterization points coincide with the points that would have been reached by applying grid intersection rasterization directly on the grid hyperplanes defined previously.

6 Conclusions

This paper has presented a first definition of rasterization in n-dimensional spaces using hyperlattices as a model for n-dimensional raster output devices. The model used does not require the lattice generators to be orthogonal, allowing thus non-orthogonal rasterization and sampling operations in higher order spaces. Rasterization schemes have been linked to the degree of connectivity required by the rasterized object in the discrete space. Such links will permit in future some control of the smoothness of the generated curve before the rasterization process begins. Both a generic algorithm for the rasterization of a curve, and an algorithm for the generation of a straight line segment have been presented: each of these algorithms generates the degree of smoothness required in the output.

The immediate application of this work in Computer Graphics lies in the visualization of n-dimensional datasets from scientific and experimental data, as well as in the development of the theory necessary for the display and representation of n-dimensional virtual worlds. In Image Processing the main contribution of this study lies in the allowed non-orthogonality of the data acquisition process through the scanning device. For example, the theory developed allows scanning devices, such as NMR and PET (Positron-Emission Tomography) devices, to acquire data along non-orthogonal directions, and to display such data in computer visualization systems, thus granting more flexibility to the data analysis.

The definition of a model for n-dimensional raster output devices and for the operation of rasterization allows the application of rasterization to new fields, whenever continuous objects have to be represented through discrete ones. To be fully usable, however, the theory presented in this paper has to be refined: although curve rasterization is defined here, to the author's knowledge there are no studies available on surface and hypersurface rasterization in n-dimensional space. There are not even algorithms for the rasterization of hyperplanes in n-dimensional spaces. The link between lattice connectivity and the smoothness of rasterized objects is still on an intuitive level, and deserves further study.

As the power of computing devices increases, so does the quest for visualizing progressively more complex spaces. Advances can be made only if the concepts of discrete n-dimensional geometry are clear. It is thus reasonable to assume that in the near future many of the unresolved issues mentioned above will be tackled.

Acknowledgments

This work has been supported by the *ERCIM* fellowship programme during stays at SERC, Rutherford Appleton Laboratory (U.K.), at INESC (Instituto de Engenharia de Sistemas e Computadores) (P), and at CWI (Centrum voor Wiskunde en Informatica) (NL), and by Hewlett-Packard Germany. Special thanks are due to John Ashby, David Duce, Mario Rui Gomes, Paul ten Hagen, Fons Kujik, Robert van Liere, Chris Wadsworth, and Olivia Wüthrich-Martone for their help and readiness to discuss the various issues of this paper.

References

[1] J. Amanatides and A. Woo. A fast voxel traversal algorithm for ray tracing. In *Proceedings of Eurographics '87*, pages 3–9, Amsterdam, 1987. North Holland.

[2] D. Badouel, C. A. Wüthrich, and E. L. Fiume. An analysis of connectivity of k-ary n-cube m-diag interconnection networks. Technical Report CSRI-266, Computer Systems Research Institute, University of Toronto, Toronto, 1992.

[3] D. Badouel and C.A. Wüthrich. Face-connected line segment generation in an n-dimensional space. In D. Kirk, editor, *Graphics Gems III*, pages 89–91,460. Academic Press, Toronto, 1992.

[4] J. Beddow. Shape coding of multidimensional data on a microcomputer display. In *Proceedings of IEEE Visualization '90*, pages 238–246, Los Alamitos, CA., 1990. IEEE Computer Society Press.

[5] J. E. Bresenham. Algorithm for computer control of a digital plotter. *IBM System Journal*, (4):25–30, 1965.

[6] D. Cohen-Or, A. E. Kaufman, and T. Y. Yong. On the soundness of surface voxelizations. In T. Y. Kong and A. Rosenfeld, editors, *Topological Algorithms for Digital Image Processing*, pages 181–204. Elsevier Science BV, Amsterdam, 1996.

[7] E. Dubois. The sampling and reconstruction of time-varying imagery with application in video systems. *Proceedings of the IEEE*, 73(4):502–522, April 1985.

[8] S. Eker. *Formal Foundations for the Design of Rasterization Algorithms and Architectures*. PhD thesis, University of Leeds, School of Computer Studies, Leeds, September 1990.

[9] E. L. Fiume. *The Mathematical Structure of Raster Graphics*. Academic Press, New York, N.Y., 1989.

[10] A.R. Forrest. Antialiasing in practice. In R.A. Earnshaw, editor, *Fundamental Algorithms for Computer Graphics*, volume F-17 of *NATO ASI Series*, pages 113–133. Springer Verlag, Berlin, 1985.

[11] C. E. Kim. Three-dimensional digital planes. *IEEE Transactions on Pattern Anasysis and Machine Intelligence*, PAMI-6(5):639–645, April 1984.

[12] R. Klette. The m-dimensional grid point space. *Computer Vision, Graphics and Image Processing*, 30:1–12, 1985.

[13] V. A. Kovalevsky. Finite topology as applied to image analysis. *Computer Vision, Graphics and Image Processing*, 46:141–161, 1989.

[14] K. D. Linsmeier. Raumbilder auf dreidimensionalem Monitor. *Spektrum der Wissenschaften*, pages 24–25, January 1996.

[15] M. G. Luby. Grid geometries which preserve properties of euclidean geometry: A study of line drawing algorithms. In R.A. Earnshaw, editor, *Theoretical Foundations of Computer Graphics and CAD*, volume F-40 of *NATO ASI Series*, pages 397–432. Springer Verlag, Berlin, 1988.

[16] T. Mihalisin, J. Schweiger, and J. Timlin. Hierarchical multivariate visualization. In *Proceedings of Interface 92*, 1992.

[17] C. Montani and R. Scopigno. Rendering volumetric data using the sticks representation scheme. *ACM Computer Graphics*, 24(5):87–93, November 1990.

[18] C. A. Rogers. *Packing and Covering*. Cambridge University Press, Cambridge, 1964.

[19] A. Rosenfeld. Arcs and curves in digital pictures. *Journal of the ACM*, 20(1):81–87, 1973.

[20] M. Slater. Tracing a ray through uniformly subdivided n-dimensional space. *The Visual Computer*, 9:39–46, 1992.

[21] D.M.Y. Somerville. *An Introduction to the Geometry of N Dimensions*. Dover Publications, Inc., New York, 1958.

[22] J.J. van Wijk and R. van Liere. Hyperslice - visualization of scalar functions of many variables. In *Proceedings of IEEE Visualization '93*, 1993.

[23] Li-De Wu. On the chain code of a line. *IEEE Transactions on Pattern Anasysis and Machine Intelligence*, PAMI-4(3):347–353, 1982.

[24] C. A. Wüthrich. An analysis of digitalization algorithms for non-linear curves. In I. Herman, B. Falcidieno, and C. Pienovi, editors, *Computer Graphics and Mathematics*, Proceedings of the Eurographics Workshop on Computer Graphics and Mathematics, pages 285–297. Springer Verlag, Berlin, 1992.

[25] C. A. Wüthrich and P. Stucki. An algorithmic comparison between square- and hexagonal-based grids. *CVGIP: Graphical Models and Image Processing*, 53(4):324–339, July 1991.

[26] C. A. Wüthrich, P. Stucki, and A. Ghezal. A frequency domain analysis of square and hexagonal based images. In J. Andrè and R.D. Hersch, editors, *Raster imaging and digital typography, Proceedings of RIDT'89*, pages 261–270, Cambridge, 1989. Cambridge University Press.

[27] Liu Yong-Kui. The generation of straight lines on hexagonal grids. *Computer Graphics Forum*, 12(1):27–31, 1993.

[28] F.W. Young, D.P. Kent, and W.F. Kuhfeld. *Dynamic Graphics for Exploring Multivariate Data*. Wadsworth Inc., 1988.