An Exact Weaving Rasterization Algorithm For Digital Planes*

C. Lincke [1], C. A. Wüthrich [1,2], P. Guitton [2]

[1] CoGVis/MMC†
Faculty of Media
Bauhaus University Weimar
99421 Weimar, Germany
{lincke,caw}@medien.uni-weimar.de

[2] LaBRI
Université Bordeaux I
351, Cours de la Liberation
33405 Talence, France
guitton@labri.u-bordeaux.fr

ABSTRACT

In this paper a new 3D plane rasterization algorithm based on weaving techniques is presented. This algorithm uses the coherence present in 3D parallel lines within the plane. Parallel lines have the same cyclic Freeman chain code but shifted by a parameter. This article provides the theory to determine this shift parameter.

Keywords: Rasterization, digital planes, weaving

1 Introduction

The recent development of three-dimensional scanning devices, such as MRI devices, the introduction of volume visualization software and systems, and the development of real 3D visualization devices have evidenced the need for algorithms which perform rasterization in 3D discrete spaces.

One of the central problems for the rasterization in 3D is the rasterization of linear surfaces, i.e. subsets of planes, onto a three-dimensional discrete space. The first generic studies on plane and surface rasterization appeared in the mid-eighties [Kim84]. Rasterization algorithms for voxel spaces were developed instead for pioneering voxel-based hardware [Kaufm87a, Kaufm87b], and have received for a long time much less attention than they deserve [Cohen91]. The recent study of new algorithms [Delfo93] on plane rasterization testifies the renewed interest in the subject.

There are basically two ways of performing the rasterization of a plane in voxel space. Cohen and Kaufman [Cohen91] introduced a three-dimensional extension of 2D the scan-line algorithm [Foley90] for polygon rasterization. This algorithm is fast and works on integer arithmetics, but takes no advantage of the coherence of line rasterization. Recently, weaving techniques have been proposed in order to take advantage of rasterization coherence [Cohen91]. In short, weaving algorithms rasterize cylindrical sur-

*This work has also been supported under the cooperation contract between Silicon Graphics Germany and the Bauhaus University Weimar.

†Computer Graphics, Visualization, Man-Machine Communication Group

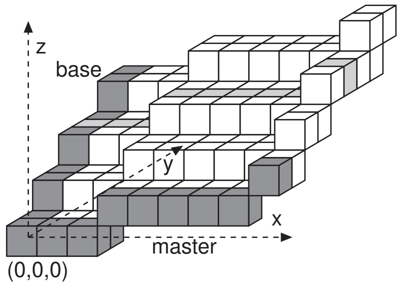

Figure 1: Weaving of the plane given by $x+2y-5z=0$.

faces by decomposing the rasterization process into two orthogonal (or, at least non-parallel) curve rasterizations: first the rasterization of the higher order planar section of the surface with a plane is computed. In a second step parallel discrete lines are rendered starting at the rasterization of the higher order curve to complete the rasterization. Delfosse [Delfo93] and later authors of this paper [Guitt95] extended these methods for planes as shown in Fig. 1.

Plane weaving sweeps a discrete line, called *master* along another discrete line, called *base*. This idea is attractive, since all digital lines swept along the base are copies of the master line, and have the same Freeman chain code sequences. Copying these chain code sequences is a fast operation since it involves just increment operations and may be used on specially developed hardware, for example on a parallel machine. Unfortunately, the weaving algorithms proposed do not compute the exact grid intersection rasterization of the plane, as defined in [Klett85] and [Wüt98].

This paper introduces a new exact weaving algorithm for planar rasterization. We will prove that in a discrete plane parallel discrete lines have the same cyclic chain code sequence shifted by some positions and we will compute this shift. The exact weaving algorithm takes this shift into consideration when copying the code sequence of the master line. The paper is structured as follows: section 2 introduces some basic notation, section 3 deals with the mathematical proof of the algorithm and derives some important properties of straight lines in 2D. Section 4 uses these results for line rasterization. The last sections of the paper present the new algorithm and compare it with Cohen and Kaufman's approach.

2 Basic Definitions

The discrete space considered is the set $\mathbb{Z}^3$ of all points of the three-dimensional space having integer coordinates. The rasterization of a surface is its grid intersection rasterization, that is the set of integer points which are closest to the intersections of the surface with the grid of lines parallel to the main axes and passing through points having integer coordinates [Wüt98]. Rasterization algorithms in two dimensions such as the DDA or Bresenham's algorithm generally assume that the end points of a line segment have integer coordinates. More general theories have been developed by Dorst and Smeulders [Dorst84] and by Pham [Pham87]. However, their notations are too difficult. In this paper we will develop a new, simpler notation.

We restrict our considerations to two-dimensional straight lines with rational slope. By symmetric transformation all lines can be transformed into straight lines having a slope in the interval $[0,1]$. To exclude trivial cases, we study only lines $L=\{(x,y)\in\mathbb{R}^2 : y=\frac{p}{q}\cdot x+e\}$ with $e\in\mathbb{R}$, $p,q\in\mathbb{N}^*(=\mathbb{N}\setminus\{0\})$, $p\le q$, where the greatest common divisor of p and q is 1 ($\gcd(p,q)=1$). The rasterization of a line L is $\mathrm{Dig}(L)=\{(i,y)\in\mathbb{Z}^2 : y=[\frac{p}{q}\cdot i+e]\}$ where the rounding operator is $[x]=k$, if $k\in\mathbb{Z}$ and $k-\frac{1}{2}<x\le k+\frac{1}{2}$.

A discrete curve γ is a sequence of points such that consecutive points are neighbours and it can be represented by its beginning point and by a sequence of direction codes [Freem61]. In $\mathbb{Z}^2$ the directions of this *chain code* are encoded as shown in Fig. 2.

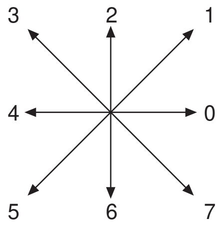

Figure 2: Direction codes

3 Theoretical Background

The chain code of the grid intersection rasterization of a straight line defined as above consists only of the elements 0 and 1, where the i-th element is $c_i = [y(i)] - [y(i-1)]$, i.e. $c_i = [\frac{p}{q} \cdot i + e] - [\frac{p}{q} \cdot (i-1) + e]$.

To compute the exact rasterization of a line segment, the value at the beginning point, say $([y(i_0)], i_0)$ for $i_0 \in \mathbb{Z}$, and the chain code beginning at this point are required. For any $k \in \mathbb{N}^*$, $[y(i_0+k)]$ is determined by $[y(i_0+k)] = [y(i_0)] + \sum_{i=1}^{k} c_{i_0+i}$.

The chain code of a digital straight line $[y(i)] = [\frac{p}{q} \cdot i + e]$ with $p, q \in \mathbb{N}$, $p \leq q$, $0 < q$ is periodic, i.e. $c_i = c_{i+q}$ for all $i \in \mathbb{Z}$. For $\gcd(p,q) = 1$ the length of the minimal periodic string is q [Pham87]. This cyclic behaviour of the chain code can be used to speed up rasterization algorithms. In this section, we prove that the chain code depends only on the slope of the straight line. Furthermore, we derive a formula to determine the position in this cyclic chain at any given point of the straight line.

The next two lemmas are results from number theory. A detailed discussion of the following notions and the proofs of these lemmas can be found in most introductory books on this subject, such as [Jones98].

Definition. For $p, r \in \mathbb{Z}$ and $q \in \mathbb{N}^*$, p is said to be *congruent to r modulo q*, written $p \equiv r \bmod (q)$, if p and r leave the same remainder when divided by q.

r is called a *residue* of p modulo q. Since $\bmod (q)$ is an equivalence relation for fixed q, $\mathbb{Z}$ can be partitioned into equivalence classes, called *congruence classes modulo q*.

Definition. The class $[p]_q$ is the set of all residues of p modulo q and the set of all classes $\bmod (q)$ is denoted by $\mathbb{Z}_q$.

Recall that the addition, subtraction and multiplication of congruence classes $[a]_q, [b]_q \in \mathbb{Z}_q$ are defined by their representatives, that is $[a]_q + [b]_q = [a+b]_q$ and similarly for other operation.

Definition. The *multiplicative inverse* of $[a]_q \in \mathbb{Z}_q$ is a class $[b]_q \in \mathbb{Z}_q$, such that $[a]_q \cdot [b]_q = [1]_q$. $[a]_q \in \mathbb{Z}_q$ is called a *unit* in $\mathbb{Z}_q$ if it has a multiplicative inverse.

Lemma 1. *$[a]_q$ is a unit iff $\gcd(a,q) = 1$.*

Definition. A *complete set of residues modulo q* is a set of q integers, containing one representative for each of the classes $[0]_q, [1]_q, \ldots, [q-1]_q$.

Lemma 2. *Let M be a complete set of residues modulo q and let p and k be integers with $\gcd(p,q) = 1$. Then $\{p \cdot i + k : i \in M\}$ is also a complete set of residues modulo q.*

The theory of congruence classes $\bmod (q)$ is useful for the study of special lines L_k.

Definition. For $k \in \mathbb{Z}$ the straight line L_k is defined as $L_k = \{(x,y) \in \mathbb{R}^2 : y = \frac{p \cdot x + k}{q}\}$.

From now on, p and q are fixed. To prove that every line L can be represented by a line L_k, we need the following lemma.

Lemma 3. *For $q \in \mathbb{N}^*$ and $p, r \in \mathbb{Z}$ holds $[\frac{p}{q}] = \frac{p-r}{q}$ if and only if $-\frac{q}{2} < r \leq \frac{q}{2}$.*

Proof. Using the definition of the rounding operator, we have $[\frac{p}{q}] = \frac{p-r}{q}$. This is equivavalent to $\frac{p-r}{q} - \frac{1}{2} < \frac{p}{q} \leq \frac{p-r}{q} + \frac{1}{2}$ and finally to $-\frac{q}{2} < r \leq \frac{q}{2}$. $\square$

Definition. Let $\sim$ can be a relation for straight lines defined by $L \sim L'$ if L and L' *have the same rasterization.*

Obviously, $\sim$ is reflexive, symmetric and transitive and thus an equivalence relation. Hence, equivalence classes can be constructed. From now on, p and q are fixed.

Definition. Let $\overline{L_k}$ ($k \in \mathbb{Z}$) be the set of all straight lines L, such that $L \sim L_k$.

Theorem 1. *For all $k \in \mathbb{Z}$ the set $\overline{L_k}$ is an equivalence class and it contains all lines $y = \frac{p}{q} \cdot x + e$ with*

$$\frac{k}{q} - \frac{1}{2q} < e \leq \frac{k}{q} + \frac{1}{2q} \quad \text{if } q \text{ is odd and}$$
$$\frac{k-1}{q} < e \leq \frac{k}{q} \quad \text{if } q \text{ is even.}$$

Proof. For an arbitrary line L, there exists a unique $k \in \mathbb{Z}$ such that $\frac{k}{q} - \frac{1}{2q} < e \leq \frac{k}{q} + \frac{1}{2q}$ or $\frac{k-1}{q} < e \leq \frac{k}{q}$. To prove $L \sim L_k$ we need to show $[\frac{p}{q} \cdot i + e] = [\frac{p \cdot i + k}{q}]$ for all $i \in \mathbb{Z}$. For every $i \in \mathbb{Z}$ there exists an integer $r_i = p \cdot i + k - q \cdot [\frac{p \cdot i + k}{q}]$. This can be rewritten as $[\frac{p \cdot i + k}{q}] = \frac{p \cdot i + k - r_i}{q}$. Because of Lemma 3, $r_i \in (-\frac{q}{2}, \frac{q}{2}]$ holds for every $i \in \mathbb{Z}$. The length of this half-open intervall is q. Consequently, it contains exactly q integers. Note that r_i is congruent $p \cdot i + k$ modulo q. Using Lemma 2 the set $\{p \cdot i + k : i = 0, 1, \ldots, q-1\}$ is a complete set of residues $\bmod (q)$. Hence, $\{r_i : i \in \mathbb{Z}\}$ consists all integers of the interval $(-\frac{q}{2}, \frac{q}{2}]$. We have $\frac{p \cdot i + k - r_i}{q} - \frac{1}{2} < \frac{p}{q} \cdot i + e \leq \frac{p \cdot i + k - r_i}{q} + \frac{1}{2}$ for all $i \in \mathbb{Z}$, which can also be written as $\frac{k - \min_{i \in \mathbb{Z}}\{r_i\}}{q} - \frac{1}{2} < e \leq \frac{k - \max_{i \in \mathbb{Z}}\{r_i\}}{q} + \frac{1}{2}$. For even q, we have $\max_{i \in \mathbb{Z}}\{r_i\} = \frac{q}{2}$ and $\min_{i \in \mathbb{Z}}\{r_i\} = 1 - \frac{q}{2}$ and thus $\frac{k-1}{q} < e \leq \frac{k}{q}$. Analogously, for odd q holds $\max_{i \in \mathbb{Z}}\{r_i\} = \frac{q-1}{2}$ and $\min_{i \in \mathbb{Z}}\{r_i\} = -\frac{q-1}{2}$. Finally we have $\frac{k}{q} - \frac{1}{2q} < e \leq \frac{k}{q} + \frac{1}{2q}$. $\square$

Let $Q = \{0, 1, \ldots, q-1\}$. Because of Lemma 2, the set $\{p \cdot i + k : i \in Q\}$ is a complete set of residues, and $p \cdot i + k$ is a bijection from $\mathbb{Z}_q$ onto $\mathbb{Z}_q$. It maps every congruence class $[i]_q$ onto a class $[p \cdot i + k]_q$.

Definition. Let $r_k(i)$ be the remainder of the division of $p \cdot i + k$ by q.

Trivially, the remainder $r_k(i)$ is a representative for $[p \cdot i + k]_q$ an element of Q. Thus, the function r_k is a bijection from Q onto Q. Consequently, for a given remainder $r \in Q$, the corresponding $i \in Q$ with $r = r_k(i)$ is uniquely determined.

Lemma 4. *For any two lines L_j and L_k ($j, k \in \mathbb{Z}$) the sequences $\{r_j(i)\}_{i \in \mathbb{Z}}$ and $\{r_k(i)\}_{i \in \mathbb{Z}}$ are equal up to a shift.*

Proof. $\{r_j(i) : i \in \mathbb{Z}\} = \{r_k(i) : i \in \mathbb{Z}\} = Q$. Thus, for arbitrary $i \in \mathbb{Z}$ there exists an integer s such that $r_j(i) = r_k(i+s)$. The two sequences differ by the same shift s, since $r_j(i+1) = r_k(i+s+1)$. This holds because $[r_j(i+1)]_q = [p \cdot (i+1) + j - q]_q = [r_j(i)]_q + [p]_q$ and similarly $[r_k(i+s+1)]_q = [r_k(i+s)]_q + [p]_q$. $\square$

As a conclusion, not only the remainders but also the chain codes of two lines L_j and L_k are equal up to the shift s. Finally, we want to compute this shift parameter at an arbitrary position of a straight line L_k with respect to the line L_0 through the origin.

Definition. *The shift of a line* L_k at the position i with respect to L_0 is defines as an integer s such that $r_k(i) = r_0(s)$.

Lemma 1 states that the multiplicative inverse $[p^*]_q$ of $[p]_q$ exists if $\gcd(p, q) = 1$.

Theorem 2. *Let $r = r_k(i)$ be the remainder of the division of $p \cdot i + k$ by q. The shift s of a line L_k at the position i with respect to L_0 is determined by $[s]_q = [r \cdot p^*]_q$.*

Proof. Because of Lemma 4 there exists an s such that r is the remainder of the division of $p \cdot s$ by q, i.e. $r = r_0(s)$. In other words, there exists an s with $[s * p]_q = [r]_q$. By multiplying this equation with the inverse $[p^*]_q$, we obtain $[s]_q = [r \cdot p^*]_q$. $\square$

Since any representative of $[s]_q$ is valid shift parameter, we use the remainder of the division of $r \cdot p^*$ by q. Again, p^* can be any representative of the class $[p^*]_q$. It can be calculated using Euler's function ϕ as $[p^*]_q =$

```
int ComputeChain(int c[], int p, int q)
{  int pInverse, i, d=2*p-q,
       inc0=2*p, inc1=2*(p-q);
/* compute c[1]...c[q] */
   for (i=1; i<q+1; i++) {
      if (d<=0) {
         d+=inc0;
         c[i]=0;
      } else {
         d+=inc1;
         c[i]=1; }
/* check if i is inverse of p */
      if (((p*i)%q)==1)
         pInverse=i; }
   c[0]=c[q];
/* compute c[q]...c[q+kSize] */
   for (i=1; q*i<q+kSize; i++)
       memcpy(&c[q*i], &c[0],
         q*sizeof(c[0]));
   return (pInverse); }
```

Figure 3: Computation of the chain code

$[p^{\phi(q)}]_q$. The calculation of Euler's function $\phi(q)$, which is the number of all k with $1 \leq k \leq q$ and $\gcd(k,q) = 1$, is computationally expensive, since it requires a decomposition of q into its prime factors. See [Jones98] for details. Combining the results of Theorem 1 and 2, the shift of any line L at position $i \in \mathbb{Z}$ can be computed from the exact value of $y(i)$.

Conclusion. *Let L be a straight line given by $y(x) = \frac{p}{q} \cdot x + e$ with $\gcd(p,q) = 1$. $[p^*]_q$ denotes the inverse of $[p]_q$. The shift s of L at position $i \in \mathbb{Z}$ is given by $[s]_q = [r \cdot p^*]_q$ with*

$$r = \begin{cases} [q \cdot y(i)] & \textit{if } q \textit{ is odd,} \\ \left[q \cdot y(i) + \frac{1}{2}\right] & \textit{if } q \textit{ is even,} \end{cases}$$

$$\textit{or } r = \begin{cases} [p \cdot i + q \cdot e] & \textit{if } q \textit{ is odd,} \\ \left[p \cdot i + q \cdot e + \frac{1}{2}\right] & \textit{if } q \textit{ is even.} \end{cases}$$

Proof. It has to be proven that $[r]_q = [r_k(i)]_q$, where $k \in \mathbb{Z}$ is determined by Theorem 1. For odd q, k is given by the inequality $\frac{k}{q} - \frac{1}{2q} < e \leq \frac{k}{q} + \frac{1}{2q}$, which is equivalent to $k = [e \cdot q]$. Thus $[r_k(i)]_q = [p \cdot i + k]_q = [p \cdot i + [e \cdot q]]_q = [[p \cdot i + q \cdot e]]_q = [r]_q$. The proof for even q is similar. □

4 Line Rasterization

The new 3D plane weaving algorithm is based on a 2D line rasterization. Next, we show how the theory presented above can be used in a rasterization algorithm for 2D line segments. We use standard C for the algorithms. For clarity reasons these routines are not fully optimized. We assume that the system provides functions `PutPixel(x,y)` and `PutVoxel(x,y,z)` writing pixel and voxel into the frame and volume buffer respectively. Its size is `kSize` in each dimension. The line segment to be rasterized is defined as in the previous section. Its end points are determined by the integers x_0 and x_1 with $x_0 < x_1$. The whole line segment should lie within the frame buffer.

In the first step a sufficiently large subsequence of the chain code of L_0 is generated and stored in an array `c`. Since s is an integer between 0 and $q-1$ and since x will not be greater than the size of the frame buffer, it suffices to precompute the chain code between 0 and `kSize+q`. This is done in a function `ComputeChain(c,p,q)` (see Fig. 3) that is based on the midpoint line scan-conversion algorithm as presented in [Foley90]. The first loop generates the chain code of q consecutive elements. For every i it is checked whether $[p \cdot i]_q = [1]_q$. In this case $[i]_q$ is the inverse of $[p]_q$. Because of Lemma 1, one of q consecutive integers must be the inverse. This is simpler than a time-consuming computation using Euler's function. The second loop uses a fast memory copy operation to replicate q elements in one step. The function `ComputeChain` returns `pInverse`.

In the second step the exact value of $y(x_0)$ is computed. The shift at the x_0 can be obtained using the main result from the previous section. Finally the whole line segment is incrementally generated in a fast loop for all x, since the y-value at any position $x_0 + i$ ($i \in \mathbb{N}^*$) is $\lfloor y(x_0 + i) \rfloor = \lfloor y(-1) \rfloor + c_{s+i}$. This step is similar to the 3D routine in Fig. 4.

The extension of this 2D algorithm to 3D is straightforward. It is based on the fact that a rasterization of 3D line can be split into two 2D line rasterizations [Cohen91, Fujim86].

The computation of the minimal periodic string of the chain code is computationally as expensive as the traditional rasterization of a line with the same length. In fact, using periodic behaviour for the rasterization of a line segment is only effective if the length of the period is smaller than the size of a line segment. It becomes more effective if many lines with the same slope have to be rasterized. In the next section this result is used to speed up plane rasterizations.

5 Plane Rasterization

Consider the 3D plane given by $F(x,y,z) = A \cdot x + B \cdot y - C \cdot z + D = 0$ $(A,B,C,D \in \mathbb{Z})$. To exclude trivial cases we assume $A,B,C \neq 0$. Without loss of generality, we only consider planes with $0 < A,B \leq C$. The implicit formula can be transformed into $z = z(x,y) = \frac{A \cdot x + B \cdot y + D}{C}$. The grid intersection rasterization of this plane is $\{(x,y,[z(x,y)]) : x,y \in \mathbb{Z}\}$. For clarity reasons, we explain the algorithm for a rectangular subset of the plane which does not exceed the volume buffer.

As master line we choose the intersection of the plane to be rasterized with the plane orthogonal to the x-axis through the integer point $(x_0,0,0)$. The base line is the intersection with the plane orthogonal to the y-axis through $(0,y_0,0)$. The original plane weaving algorithm first rasterizes the master line between x_0 and $x_1 \in \mathbb{Z}$, which is essentially a 2D rasterization in the plane orthogonal to the y-axis through $(0,y_0,0)$. Then the base line is rasterized looping through all integers y between y_0 and $y_1 \in \mathbb{Z}$. In every step of the loop a copy of the master line is translated to the current point of the base line. The new algorithm does not only translate the master line, it also shifts the chain codes.

Lines in the plane with fixed y-coordinate are parallel to the master. Such lines are given by $z = z(x) = \frac{A}{C} \cdot x + \frac{B \cdot y + D}{C}$, so the slope $\frac{A}{C}$ does not depend on y.

```
void Plane3D(int A, int B, int C, int D,
     int x0, int x1, int y0, int y1)
{  int x, y, z, s, p, q, g, pInvers;
   int c[kMaxChain];
   int d, f, inc0=2*B, inc1=2*(B-C);
   g=gcd(A,C); p=A/g; q=C/g;
   pInvers=ComputeChain(c, p, q);
   z=Round((A*x0+B*(y0-1)+D)/(float)C);
   d=2*A*x0+2*B*y0-2*C*z-C+2*D;
   if (q%2) f=3*C; else f=3*C+g;
   for (y=y0; y<y1+1; y++) {
      s=(pInvers*Round((d+f)/(2.0*g)))%q;
      if (d<=0){
         d+=inc0;
      } else {
         d+=inc1;
         z++; }
      CopyLine3D(x0, y, z,
           x1-x0, &c[s]); } }

void CopyLine3D(int x0, int y, int z,
                int l, int *cPtr)
{  int x;
   PutVoxel (x0, y, z);
   for (x=x0+1; x<x0+l+1; x++) {
      z+= * ++cPtr;
      PutVoxel (x, y, z); } }
```

Figure 4: 3D plane rasterization algorithm

In the first step of the plane weaving algorithm, the chain code of the master line and `pInverse` are precomputed. $\frac{A}{C}$ has to be substituted by a reduced slope $\frac{p}{q}$ with $p = \frac{A}{\gcd(A,C)}$ and $q = \frac{C}{\gcd(A,C)}$. Then, the base line $z = z(y) = \frac{B \cdot y}{C} + \frac{A \cdot x_0 + D}{C}$ is rasterized using the midpoint 2D line scan-conversion algorithm [Foley90].

For every $y \in \mathbb{Z}$ a decision variable d, defined as $d = 2 \cdot F(x_0, y, [z(y-1)] + \frac{1}{2}) = 2 \cdot A \cdot x_0 + 2 \cdot B \cdot y - 2 \cdot C \cdot [z(y-1)] - C + 2 \cdot D$, indicates whether the value of $[z(y)]$ is incremented. Since the algorithm begins at the position y_0, the decision variable is initialized with $d = 2 \cdot F(x_0, y_0, [z(y_0 - 1)] + \frac{1}{2})$. To calculate the shift of the translated master line beginning at the point $(x_0, y, [z(y)])$

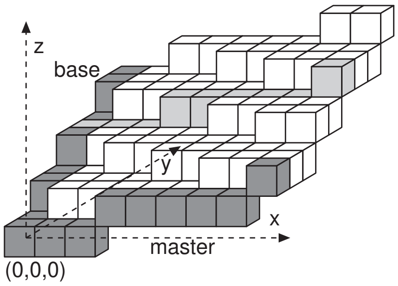

Figure 5: Grid intersection rasterization of the plane $x+2y-5z=0$.

we have to determine $r=[q\cdot z(y)]$ for odd q and $r=[q\cdot z(y)+\frac{1}{2}]$ for even q.

We actually need only a residue of r modulo q. Hence we obtain the same shift s if we ignore the integer part of $z(y)$ or if we add an arbitrary integer to $z(y)$. Recall that $\frac{d}{2\cdot C}$ is the distance to the next mid-line. So we can use $\frac{d}{2\cdot C}+\frac{1}{2}$ instead of $z(y)$ to calculate r. In C the result of modulo operation `k%q` is negative for negative k. To avoid a negative r, we add 1 to $\frac{d}{2\cdot C}+\frac{1}{2}$. Consequently, we obtain $r=[q\cdot\frac{d+3\cdot C}{2\cdot C}]$ for odd q and $r=[q\cdot\frac{d+3\cdot C}{2\cdot C}+\frac{1}{2}]$ for even q. Finally, the shift s is determined by $[s]_q=[p^*\cdot[\frac{d+f}{2\cdot g}]]_q$ with $g=\gcd(A,C)$, $f=3\cdot C$ for odd q and $f=3\cdot C+g$ for even q.

Fig. 4 contains the complete C source code for the exact plane weaving algorithm. The function `CopyLine3D` copies chain code. Its parameters are the beginning point of the discrete line segment, which is $(x_0,y,z(y))$, its length l and the pointer `&c[s]` to the chain code at the beginning point.

6 Perfomance discussion

The exact weaving algorithm computes the grid intersection rasterisation of the plane. Fig. 5 shows the exact rasterisation of the plane of Fig. 1. Notice the shift in the chain code of the translated master lines. On generic hardware the exact weaving algorithm and Cohen and Kaufman's, are linear in the number of voxel to be generated. If a square plane segment of side legth n has to be rasterized, the time complexity is $O(n^2)$.

In practice, since the resolution is limited, the length of the periodic chain code will not exceed the size of the volume buffer. For big plane segments, i.e. if the number of voxels to be generated is very large compared to the size of the period, the computation of the chain code and the shift for every y can be neglected. All operations in the loop in `CopyLine3D` are repeated for every voxel to be generated. Thus, we only compare `CopyLine3D` with the inner loop of Cohen and Kaufman's algorithm, which consists of a DDA step. For every step a comparison and an incrementation of the decision variable has to be performed. In contrast, the exact weaving algorithm increments the value of z by the precomputed chain code.

Both algorithms were profiled on a SGI Indigo^2 R4400. The size of the volume buffer was 256^3 and 1000 square plane segments with side length n and random parameters $A,B,C,D\leq 256$ were generated. On an average, `CopyLine3D` was about 35% faster than the inner loop of Cohen and Kaufman's algorithm. The modulo operation in `ComputeChain` can optimized to a DDA step. So in the worst case, `ComputeChain` performs $2\cdot 256$ DDA steps and a memory copy. In our tests both algorithms generated plane segments with side length $n=64$ in almost the same time. For smaller plane segments the exact weaving algorithm is significantly slower, whereas for $n=256$ it is about 30% faster.

The performance of the exact weaving algorithm may be sped up when the hardware provides a operation to copy chain code in constant time. Chen proposed a simple hardware design for a line segment replicator for 2D lines [Chen97]. It consists of a cyclic shift register, which has to be initialized with the first q elements of the chain code. To copy a line, only the beginning point of the line, its length and shift parameter need to be passed to the device. In this case, the complexity will drop to $O(n)$.

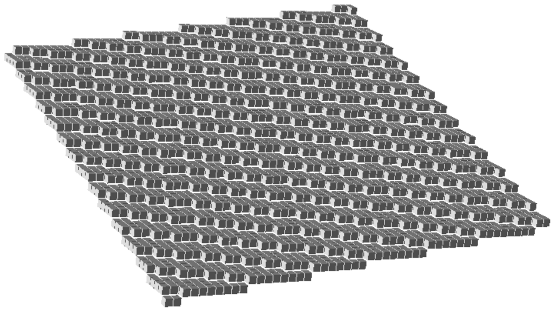

Figure 6: Rasterized plane segment $(5x + 11y - 38z = 0)$

7 Conclusions

In this paper we have presented a new plane rasterization algorithm based on weaving techniques that uses the coherence present in parallel 3D lines within the plane. The new algorithm is faster than the traditionally used Cohen and Kaufman algorithm.

The weaving technique presented here can be easily extended to produce planes defined through generic incident lines and for higher order surfaces having lines as their intersection with planes. This approach has been suggested by Cohen and Kaufman [Cohen91] for example for cylindrical surfaces. The authors are currently working on this problem.

REFERENCES

[Chen97] J. X. Chen. Multiple segment line scan-conversion. *Computer Graphics Forum*, 16(5):257–268, 1997.

[Cohen91] D. Cohen and A. Kaufman. Scan conversion algorithms for linear and quadratic objects. In A. Kaufman, editor, *Volume Visualization*, pages 280–300. IEEE Computer Science Press, 1991.

[Delfo93] J. Delfosse. Discrétisation d'objets graphiques. Master's thesis, Université des sciences et techniques de Lille, Lille, July 1993.

[Dorst84] L. Dorst and A. W. M. Smeulders. Discrete representation of straight lines. *PAMI*, 6(4):450–463, July 1984.

[Foley90] J. D. Foley, A. Van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley, Reading, MA., 1990.

[Freem61] H. Freeman. On the encoding of arbitrary geometric configurations. *IRE Transactions on Electronic Computing*, EC-10(2):260–268, June 1961.

[Fujim86] A. Fujimoto, T. Tanaka, and K. Iwata. Arts: Accelerated ray-tracing system. *IEEE Computer Graphics & Applications*, 6(4):16–26, April 1986.

[Guitt95] P. Guitton and C. A. Wüthrich. Fast arbitrarily oriented 3D discrete slicing for medical imaging. Technical Report 1063-95, LaBRI, University of Bordeaux I, Bordeaux, 1995.

[Jones98] G.A. Jones and J.M. Jones. *Elementary Number Theory*. Springer, London, Berlin, 1998.

[Kaufm87a] A. Kaufman. An algorithm for 3d scan-conversion of polygons. In *Proceedings of Eurographics '87*, Amsterdam, 1987. North Holland.

[Kaufm87b] A. Kaufman. Efficient algorithms for 3d scan-conversion of parametric curves, surfaces, and volumes. *ACM Computer Graphics*, 21(4):171–179, July 1987.

[Kim84] C. E. Kim. Three-dimensional digital planes. *IEEE Transactions on Pattern Anasysis and Machine Intelligence*, PAMI-6(5):639–645, September 1984.

[Klett85] R. Klette. The m-dimensional grid point space. *Computer Vision, Graphics and Image Processing*, 30:1–12, 1985.

[Pham87] S. Pham. Equations of digital straight lines. In T. L. Kunii, editor, *Computer Graphics 1987, Proceedings of CG International 1987*, pages 221–248, Tokio, 1987. Springer Verlag.

[Wüt98] C. A. Wüthrich. A model for curve rasterization in n-dimensional space. *Computer & Graphics*, 22(2–3):153–160, 1998.