PERIODICA POLYTECHNICA SER. ARCHITECTURE VOL. 35, NO. 1, PP. 59-77 (1991)

COMPUTER REPRESENTATION OF PLANE FIGURES AND 3D SOLIDS USING ADDITIVE ALGORITHMS

J. PEREDY

Department of Descriptive Geometry II
Technical University of Budapest

Received April 20, 1991

Abstract

The paper presents a definition of figures and curves in the plane and solids and surfaces in the 3D space based on the notion of (generalized) pixels and voxels. The corner points of the pixels/voxels form a point grid in the plane or in the space and a grid value is attached to each grid point by means of additive algorithms. A plane figure or a 3D solid consists of those pixels/voxels for which the grid values ib all four/eight corner points are negative. Some basic transformations of the algorithms generating the grid values (and thus the plane figures and 3D solids) are ivestigated useful in constructing axonometric and perspective projections. The paper is a first one of a series of publications intending to present a new philosophy for solving a wide range of problems in computational geometry, such as representation and manipulation of different types of curves, surfaces, plane figures and solids; finding points/curves of intersection and contour lines; finding different 2D projections of 3D objects, etc.

Introduction

This paper is the first in a series of papers intended to present an essentially new philosophy for solving such problems as representation and manipulation of a wide range of different types of curves, surfaces, plane figures and solids; findig points/curves of intersection and contour lines; findig different 2D projections of 3D objects, etc. The proposed method is founded on the notion of pixels and voxels (volume pixels) and aims at constructing possibly efficient additive algorithms for solving the aforementioned problems. In this paper some basic ideas will be presented for the 2D case.and then will be generalized for 3D. Many special problems (e.g. those of free formed curves and surfaces) are left for consecutive papers.

Basic Concepts in 2D

Pixels and Grid Points

Let us divide the plane into pixels by two sets (say an X set and a Y set) of lines. Most frequently, the lines of each set are among themselves parallel and equidistant straight lines, any two lines from different sets are perpendicular, and the pixels are congruent squares. An other important arrangement is where the subdivision lines are parallel in one of the two sets only, while the other set consists of a pencil of straight lines having a common central point and, consequently, the pixels are (in general) trapezoids with two parallel sides. (This latter case is important from the point of view of 3D generalisations, namely of constructing the central projections of 3D solids.)

A set of pixels forming a finite (regularly rectangular or trapezoidal) part of the plane will be considered, bordered by two X lines and by two Y lines, containing $(m + 2)$ lines of the X set and $(n + 2)$ lines of the Y set (boundary lines included). The division lines of the X set and of the Y set can be numbered by the integers 0 ...m+1 and 0 ... n+1, resp. The points of intersection of any two division lines of different sets are the grid points. The grid points can be identified by the serial numbers of the two intersecting division lines, for example *(i,j)* is the grid point at the intersection of the i-th line of the X set and the j-th line of the Y set (of course $0 \leq i \leq m+1$ and $0 \leq j \leq n+1$). The i and j values may be considered as the X and Y coordinates resp. of the grid points.

There are four neighbouring grid points associated with a pixel, namely its four corner points $(i,j),(i+1,j),(i,j+1),(i+1,j+1)$. A pixel will be identified by the coordinates (i,j), i.e. by the grid point with the lowest coordinate values out of its four corner points. Thus the finite part of the plane under consideration is composed of the pixels with coordinates from (0,0) to (m,n).

Grid Values. Definition of Curves and Figures

Let us assign a positive or negative (possibly integer) number to each grid point, the grid value $R_{i,j}$. A pixel (i,j) is characterized by the four grid values of the four adjacent grid points: $R_{i,j}, R_{i+1,j}, R_{i,j+1}, R_{i+1,j+1}$. A pixel is negative homogeneous if all four of its associated grid values are negative; a pixel is positive homogeneous if all four of its associated grid values are zeros or positive numbers; in any other case the pixel is non-homogeneous.

Our investigations are founded on the following definition: *a plane figure is composed of the negative homogeneous pixels, the non-homogeneous pixels form its boundary curve, while the positive homogeneous pixels are outside the plane figure.*

On the one hand, it is clear from this definition, that any set of the $(m+2)*(n+2)$ grid values defines unambiguously a single plane figure (if there is at least one negative homogeneous pixel). On the other hand, a plane figure does not define unambiguously a set of corresponding grid values, there are always many different sets of grid values describing the same boundary curve and the same figure. For example if the grid values (or some of them) are multiplied by any positive number, this modified set of grid values still describes the original figure.

It is possible to define more than one set of grid values simultaneously. This is necessary, for instance, if we want to study some plane figures together, say in order to compute the point(s) of intersection of their boundary curves, or to define the common part of the figures, etc. When working with multiple grid values the different sets will be distinguished by using primes or other convenient symbols.

If the grid values are assigned to the grid points randomly, the definition still works, but the result hardly resembles anything what might be called a plane figure or its boundary curve in the 'normal' sense of these words. Nevertheless one can find those rules of assigning the grid values by means of which the important objects of the plane geometry (or more precisely their 'pixel counterparts') can be described.

Additive Algorithms

Let us construct now algorithms for assigning grid values to the grid points in a possible simple way: using the operation of addition (possibly of integer addition) only. The basic form of these algorithms should produce the grid values step by step, i.e. starting from the grid value $R_{i,j}$ of any grid point should compute the $R_{i+1,j}$ or $R_{i-1,j}$ or $R_{i,j+1}$ or $R_{i,j-1}$ value of a neighbouring grid point. Thus if we want to define such an algorithm we have to specify how the grid value varies when stepping from a given grid point to one of its four neighbours (X step or Y step in positive or negative direction). Of course, it is expedient if the algorithm is of the same structure for any grid-point, only the numerical values appearing in it vary from point to point. Such an algorithm can yield the grid value of any grid point, because any grid point can be attained from any other one by an appropriate number of consecutive X and Y steps.

The possible simplest additive algorithm is the following:

$$\begin{array}{lll} (X+\text{step}:) & i=i+1, & R=R+X; \\ (X-\text{step}:) & j=j-1, & R=R-X; \\ (Y+\text{step}:) & i=i+1, & R=R+Y; \\ (Y-\text{step}:) & j=j-1, & R=R-Y. \end{array}$$

In these formulae i and j are the grid point coordinates, R is a 'register' containing the grid value. X and Y are the modifying registers; by adding them to or subtracting them from the 'main' register R computes the algorithm the grid values corresponding to the neighbouring grid points. This simplest additive algorithm produces a 'straight line' as a boundary, on one side of which is the 'half plane' forming the plane figure defined by the algorithm, while the other 'half plane' contains those pixels which are outside of this plane figure.

Additive Algorithm of Second Order

Let us consider now a somewhat more complicated additive algorithm:

$$\begin{array}{llllll} (X+\text{step}:) & i=i+1, & R=R+X, & X=X+XX, & Y=Y+YX; \\ (X-\text{step}:) & i=i-1, & Y=Y-YX, & X=X-XX, & R=R-X; \\ (Y+\text{step}:) & j=j+1, & R=R+Y, & Y=Y+YY, & X=X+XY; \\ (Y-\text{step}:) & j=j-1, & X=X-XY, & Y=Y-YY, & R=R-Y. \end{array}$$

The main difference between this variant and the previous one is that X and Y are not constants any more, but they themselves are modified from step to step. The X and Y, influencing the content of the main register R directly, are the first order modifying registers, while XX, YX, XY and YY are the second order modifying registers being added to or subtracted from the first order modifiers. The previous variant is the additive algorithm of first order, while the present one is the additive algorithm of second order.

Some Characteristics of Additive Algorithms

The second order additive algorithm gives the opportunity to study some important properties of additive algorithms in general.

Notice that in the formulae of the second order additive algorithm a special notation has been adopted for the modifying registers. The 'name' of a first order modifier is a single letter (X or Y), while the names (identifiers) of the second order modifying registers consist of two letters: XX, YX, YY, XY. The last letter always indicates whether the register is

active (has to be added or subtracted) in the X or in the Y steps, while the letters preceding the last one (if there is any) form the name of the 'target register' to which it has to be added or from which it has to be subtracted. If the name of the register is a single letter and therefore there is no letter preceding the last one, the target register is the main register R. Using this convention the whole additive algorithm can be reconstructed from the names (identifiers) of the constant registers and this notation lends itself for further generalisations.

In the algorithm the consecutive order of execution of the additive operations is of importance. In the positive steps the first order modifier has to be added to the main register first, and the second order modifying registers are added to the first order ones after that. In the negative steps just in contrary the second order modifiers are subtracted from the first order ones first and the subtraction of the first order modifier from the main register follows after that. This structure of the algorithms ensures that if one proceeds n steps in one (say positive) sense and n steps in the opposite (say negative) sense then the original grid value of the starting point is exactly restored.

Let us consider now what happens if starting from a given grid point

a) we make first an $X+$ step and after that a $Y+$ step, or

b) we make first a $Y+$ step and after that an $X+$ step.

In both cases the same grid point is arrived at but, on two different routes. The values of the registers vary in the followig way:

a) 1st step:

$$i = i_0 + 1, j = j_0, R_1 = R_0 + X_0, X_1 = X_0 + XX, Y_1 = Y_0 + YX;$$

2nd step:

$$i = i_0 + 1, j = j_0 + 1, R_2 = R_1 + Y_1, Y_2 = Y_1 + YY, X_2 = X_1 + XY;$$

i.e. $R_2 = R_0 + X_0 + Y_0 + YX, X_2 = X_0 + XX + XY, Y_2 = Y_0 + YY + YX.$

b) 1st step:

$$i = i_0, j = j_0 + 1, R_1 = R_0 + Y_0, Y_1 = Y_0 + YY, X_1 = X_0 + XY;$$

2nd step:

$$i = i_0 + 1, j = j_0 + 1, R_2 = R_1 + X_1, X_2 = X_1 + XX, Y_2 = Y_1 + YX;$$

i.e. $R_2 = R_0 + X_0 + Y_0 + XY, X_2 = X_0 + XX + XY, Y_2 = Y_0 + YY + YX.$

In these formulae R_0, X_0, Y_0 denote the content of the corresponding registers before the first step, while R_1, X_1, Y_1 and R_2, X_2, Y_2 represent these values after the first step and after the second step, resp. The second order modifying registers do not need such a specification because they are the constants of the second order additive algorithm, i.e. they preserve their initial values during the whole process.

Note that the two formulae for R_2 are not perfectly identical in cases a) and b); they give the same grid value for the grid point $(i + 1, j + 1)$ if

and only if

$$XY = YX.$$

This is the necessary condition for the second order additive algorithm to produce a single, well defined grid value in each grid point, not depending on the actual path used to approach the grid point in question. An additive algorithm, which assignes a single, unambiguously defined grid value to each grid point independently of the route used to approach it, is said to be conservative. The description of plane figures and their boundary curves usually needs conservative algorithms. Nevertheless, there are plane curves e.g. spirals which do not define any plane figure and can be described by non-conservative algorithms. A non-conservative algorithm assigns in general more than one grid value to the same grid point depending on the route used to approach the grid point. All the same, a non-conservative algorithm can be used to describe a curve unambiguously if, for example, those grid values are considered only, which arise walking along the curve itself.

In the previous paragraphs the first order additive algorithm and the second order additive algorithm have been presented separately. But it is clear from the structure of the additive algorithms that the first order additive algorithm is nothing else than a special case of the second order one. If in the second order additive algorithm the constant registers are set to zero ($XX = XY = YX = YY = 0$), it works as a first order algorithm. The results won by the investigation of a higher order algorithm apply to the lower order ones, too.

Pixel Curves, Pixel Plane Figures and their Analytic Counterparts

The second order additive algorithm (at least in its conservative form) defines plane figures bounded by conic sections such as circles, ellipses, parabolas and hyperbolas. We have already mentioned that the first order additive algorithm (which is by its nature always conservative) defines straight lines. These statements are a bit loosely formulated and without proof, so their exact meaning needs some explanation.

The general equation of conic sections has the form:

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0.$$

The left member of this equation can be considered to define the register values for the integer coordinates i and j

$$R_{i,j} = Ai^2 + Bij + Cj^2 + Di + Ej + F.$$

Consequently, the difference of two neighbouring grid values in the X and Y direction respectively are given by the formulae:

$$X_{i,j} = R_{i+1,j} - R_{i,j} = 2Ai + A + Bj + D,$$
$$Y_{i,j} = R_{i,j} + 1 - R_{i,j} = Bi + 2Cj + C + E,$$

and one can compute the variation (from grid point to neighbouring grid point) of the $X_{i,j}$ and $Y_{i,j}$ values too:

$$XX = X_{i+1,j} - X_{i,j} = 2A,$$

$$XY = X_{i,j+1} - X_{i,j} = B, \quad YX = Y_{i+1,j} - X_{i,j} = B,$$
$$YY = Y_{i,j+1} - Y_{i,j} = 2C.$$

The notation employed for the differences of the register values and for the differences of these differences corresponds to the notation of the first and second order modifying registers in the second order additive algorithm. Really, if we start from the grid point (i, j) and the initial values of the registers in the second order additive algorithm are set to

$$R = Ai^2 + Bij + Cj^2 + Di + Ej + F,$$

$$X = 2Ai + A + Bj + D, \quad Y = Bi + 2Cj + C + E,$$
$$XX = 2A, \quad XY = YX = B, \quad YY = 2C,$$

the second order additive algorithm produces the grid values prescribed by the left member of the general equation of the conic sections. The left member of this equation takes zero value regularly inside those (and only those) pixels, which are not sign homogeneous. Therefore the analytic curve of the conic section runs regularly inside the non-homogeneous pixels and if a pixel contains a segment of the analytic curve it is regularly not homogeneous.

The train of thoughts presented in the above paragraph explains and justifies the statement that the second order additive algorithm produces conic sections. But the formulation of the results in the last sentences of the previous paragraph is conditional. There are two types of conditions on their validity.

On the one hand, the mesh of the division lines used to form the pixels should be in accordance with the dimensions of the plane figure in question. Objects (or details of objects) smaller (or at least thinner) than the mesh width cannot be represented properly by the given resolution and, consequently, it may happen that the proposed method based on additive

algorithms 'does not take notice of them'. However, it is always possible to represent by means of the proposed method these 'small' objects (or details) too, only a finer mesh has to be used.

On the other hand, the second order additive algorithm describes conic sections in the sense of Euclidean plane geometry, if the pixels are formed by two perpendicular sets of parallel and equidistant straight lines. This is expressed in the above verification by the fact that the i, j pixel coordinates have been simply identified with the integer values of the x, y Cartesian coordinates. No doubt, this is by far the most natural and most frequently used arrangement of pixels, but because in some cases (e.g. for constructing perspective images) other forms of pixels can be expedient too, this condition is not trivial either.

Multiple Steps

The basic form of the additive algorithm (as used in the previous chapters) works 'step by step': knowing the initial values $R_{0,0}, X_{0,0}, Y_{0,0}, XX, XY = YX, YY$ at a grid point (i_0, j_0) the grid value R of any other grid point $(i_0 + i, j_0 + j)$ can be determined but only by means of computing the grid values of many other intermediate grid points along a route leading from (i_0, j_0) to $(i_0 + i, j_0 + j)$. We want to construct now a direct algorithm for computing the grid value of an arbitrary point (on the basis of known register values in any other grid point) without involving explicitly intermediate points. For this purpose it is expedient to consider the effect of the individual modifying registers on the value of the main register separately and choose the simplest possible route, say marching first i times X step then j times Y step.

- If all but $X_{0,0}$ initial values are zeros, then $R = iX_{0,0}$.
- If all but $Y_{0,0}$ initial values are zeros, then $R = jY_{0,0}$.
- If all but XX initial values are zeros, then the X register takes the values $0, XX, 2XX, \ldots iXX$ while walking i steps in the X direction.

These values constitute the arithmetic series and, consequently, the content of the main register is the sum of this arithmetic series:

$$R = \frac{(i-1)i}{2} XX.$$

If all but YY initial values are zeros, then, on the basis of similar considerations,

$$R = \frac{(j-1)j}{2} YY.$$

If all but $XY = YX$ initial values are zeros, then after doing i times X step the Y register has the value $Y = iXY$ and all other registers remain

unchanged. Doing now j steps in Y direction $R = jY = ijXY = ijYX$. (Choosing other routes one arrives at the same result and the conservative character of the second order additive algorithm under the condition $XY = YX$ can be verified.)

Summing up these results the variation of the main register is expressed by the formula:

$$R_{ij} = R_{0,0} + iX_{0,0} + jY_{0,0} + \frac{(i-1)i,}{2}XX + ijXY + \frac{(j-1)j}{2}YY.$$

It helps to express the basic structure of this formula if we apply the usual notation $\frac{(k-1)k}{2} = \binom{k}{2}$. (Anyway, it should be noted that in the present case this expression is valid and sensible for negative integer k values, too.) Using this notation, the multiple step form of the second order conservative additive algorithm is as follows:

$$R_{ij} = R_{0,0} + iX_{0,0} + jY_{0,0} + \binom{i}{2}XX + ijXY + \binom{j}{2}YY$$

$$X_{ij} = X_{0,0} + iXX + jXY, \quad Y_{ij} = Y_{0,0} + jYY + iXY.$$

These formulae comprise all four cases (X or Y steps in positive or negative direction) of the step by step form. While the step by step form is expedient for drawing a curve on the screen of a raster display pixel by pixel, the multistep form is more general and concise.

Additive Algorithms of Higher Order

The multistep form of the additive algorithms lends itself for further generalization. E.g. the third order conservative additive algorithm can be defined by the following formulae:

$$R_{ij} = R_{0,0} + iX_{0,0} + jY_{0,0} + \binom{i}{2}XX_{0,0} + ijXY_{0,0} +$$

$$\binom{j}{2}YY_{0,0} + \binom{i}{3}XXX + j\binom{i}{2}XXY + i\binom{j}{2}XYY + \binom{j}{3}YYY,$$

$$X_{ij} = X_{0,0} + iXX_{0,0} + jXY_{0,0} + \binom{i}{2}XXX + ijXXY + \binom{j}{2}XYY,$$

$$Y_{ij} = Y_{0,0} + iXY_{0,0} + jYY_{0,0} + \binom{i}{2}XXY + jiXYY + \binom{j}{2}YYY,$$

$$XX_{ij} = XX_{0,0} + iXXX + jXXY,$$

$$XY_{ij} = XY_{0,0} + iXXY + jXYY,$$

$$YY_{ij} = YY_{0,0} + iXYY + jYYY.$$

In view of the conservative character of the algorithms to be studied $XY = YX, XXY = XYX = YXX$ and $XYY = YXY = YYX$, therefore out of each group only the first variant appears in the above formulae. The notation of the third order modifying registers strictly follows the rules established for the second order ones, e.g. XXY modifies XX in a Y step or $XY = YX$ modifies Y in an X step. Note that $\binom{k}{3} = (k-2)(k-1)k/6$.

For the sake of a concise formulation of the general pth order conservative additive algorithm let us introduce the notation $qXrY$ for the $(q+r)$ th order modifying register containing qX -es and rY-s in its identifier and let us accept the main register R as the zeroth order register $R = 0X0Y$. With these conventions the general multistep form of the pth order conservative additive algorithm is the following:

$$qXrY_{i,j} = \sum_{k=q}^{p-r} \sum_{l=r}^{p-k} \binom{i}{k-q} \binom{j}{l-r} kXlY_{0,0}$$

with

$$\binom{a}{b} = \frac{a(a-1)...(a-b+1)}{1.2...b}.$$

Note that in the right member of this equation all registers have to be taken with their initial values, while the left member is the register value after iX step and jY step.

P^{th} Order Additive Algorithms and Polynomials of p^{th} Degree

Substituting the 'continuous' real x, y Cartesian coordinates for the i,j integer pixel coordinates in the above general formula of the pth order conservative additive algorithm, it is equivalent to a polynomial of pth degree. One can say, that the general multistep formula of the pth order conservative additive algorithm with continuous cartesian coordinates is nothing else than a special form of writing for a general polynomial of the pth degree $fp(x, y)$.

No doubt, if we want to obtain 'the additive algorithm form' of a general polynomial of the pth degree starting at the origo, the initial value of the main register R has to be equal to the function value of the polinomial at the origo: $R_{0,0} = f_p(0,0)$. The initial values of the first order modifying registers are the first differences of unit distance: $X_{0,0} = f_p(1,0) - f_p(0,0), Y_{0,0} = f_p(0,1) - f_p(0,0)$. The second order modifiers are the first differences of these first differences, i.e. they are the second order differences:

$$XX_{0,0} = X_{1,0} - X_{0,0} = f_p(2,0) - 2f_p(1,0) + f_p(0,0),$$

$$XY_{0,0} = X_{0,1} - X_{0,0} = Y_{1,0} - Y_{0,0} = f_p(1,1) - f_p(1,0) - f_p(0,1) + f_p(0,0),$$

$$YY_{0,0} = Y_{0,1} - Y_{0,0} = f_p(0,2) - 2f_p(0,1) + f_p(0,0).$$

The general formula for the initial values of the registers in the additive algorithm form of a pth degree polynomial:

$$qXrY_{0,0} = \sum_{k=0}^{q}\sum_{l=0}^{r}(-1)^{k+l}\binom{q}{k}\binom{r}{l}f_p(k,l).$$

In case of a polynomial of the pth degree the corresponding additive algorithm may have non-zero modifiers up to the pth order. (The higher order modifiers are, by nature, zeros, consequently, they need not be considered at all.) Note that in case of a polynomial with real coefficients the register values, as given by the above formulae, won't be integers any more. Clearly, this fact does not affect essentially the nature of the additive algorithms, but it deserves some comment (which will be presented in one of the consecutive papers).

Transformations

We want to use the additive algorithms for representing and studying plane figures and their boundary curves. It is necessary for this purpose to describe the same plane figure in different coordinate systems. Let us consider a new (u,v) (general, possibly curvilinear) coordinate system which is connected to the original (x,y) one by the equations $x = x(u,v), y = y(u,v)$. It is possible to describe the same plane figure with the same boundary curve by an appropriate additive algorithm even in the new coordinate system. The register values of the two additive algorithms describing the same plane figure in the new and in the original coordinate systems are

interconnected by the following relation:

$$qUrV_{0,0} = \sum_{k=0}^{q}\sum_{l=0}^{r}(-1)_{k+l}\binom{q}{k}\binom{r}{l}\sum_{s=0}^{p}\sum_{t=0}^{p-s}\binom{x(q,r)}{s}\binom{y(q,r)}{t}kXlY_{0,0},$$

where $qUrV$ are the names (identifiers) of the registers producing the grid values in the new coordinate system (u, v). This formula is nothing else than a combination of formulae a) and b). If in formula a) we take $q = r = 0$ and use instead the integers i, j the Cartesian coordinates x, y , it computes for any x and y the function values of a polynomial as defined by the registers $kXlY$. So it can be used to produce the grid values at the grid points $[x(q, r), y(q, r)]$ of the new coordinate system, too. (Note, here q and r are the integer 'coordinates' describing the grid points of the new (u, v) coordinate system and must not be confused with the q and r in the formula a), which have been set now to zero.) On the basis of the grid values pertaining to the new coordinate system formula b) computes the new register values.

It is important to note that the two additive algorithms describing the same figure in two different coordinate systems are not always of the same order. In a coordinate system 'matching better' some important characteristics of the plane figure to be described an additive algorithm of a lower degree can be sufficient, while in the case of a 'poorer match' higher degree modifiers may have to be considered.

From the point of view of the applications two coordinate transformations are of special importance: the linear transformation and the perspective transformation.

The linear transformation has the form $x = au + bv, y = cu + dv$. This transformation comprises such important geometric manipulations as stretching, rotating, mirroring the plane figures. The linear transformation, as a rule, does not affect the order of the additive algorithms, they are of the same order after a linear transformation as they were before it.

The perspective transformation will be used in the 3D generalizations to facilitate the construction of central projections of 3D geometric objects. If the original coordinate system was a Cartesian one it is transformed by a perspective transformation into a coordinate system in which both sets of the coordinate lines are straight lines, but only one of them remains unchanged (i.e. remains a set consisting of parallel lines), the other one is transformed into a radial set. We will use the perspective transformation in the following form: $x = u + Tuv, y = v$. This means that the $y=$ const. set of the original Cartesian system does not change, while the other one transforms into a radial set having its centre on the y axis of the original system at a distance $1/T$ from the origo in negative direction. Un-

der 'perspective transformation' this special form will be understood. This perspective transformation doubles the order of the additive algorithm, but only a few higher order modifiers appear with non-zero values. For example in case of a second order additive algorithm with nonzero modifying registers X, Y, XX, XY, YY in the original Cartesian coordinate system, after a perspective transformation only the higher than second order registers UUV, UVV and $UUVV$ will have nonzero values (in addition to the U, V, UU, UV, VV 'normal' ones). Generally speaking, in the perspective transformation of a p^th order conservative additive algorithm above the 'normal' p^th order modifiers appear non-zero higher order modifiers with names won by adding V-s to the names of the normal ones, but the total number of V-s in a name must not exceed p. Thus the highest order modifier in the transformed form is of order $2p$, but there is only a single non-zero register of this order, namely $pUpV$.

Rotation and Perspective Transformation of the Second Order Algorithm

Of course, the general formula for obtaining the initial register values in a transformed coordinate system can be used directly in case of the linear and of the perspective transformations, too. However, it may be expedient to deduce from them simple solutions for the most frequently used special cases. Finally, as a demonstration of the practical possibilities of the methods presented in this paper we deduce now the formulae for the rotation and perspective transformation of the second order conservative additive algorithms.

The rotation is a special linear transformation defined by the equations $x = Cu - Sv, y = Su + Cv$, where S and C are the sine and cosine of the angle of rotation. Substituting this into the general formula the initial values of the transformed registers are as follows:

$$U = CX + SY + .5(C2C)XX + CSXY + .5(S^2S)YY,$$

$$V = -SY + CX + .5(S^2S)XXCSXY + .5(C2C)YY,$$

$$UU = C^2XX + 2CSXY + S^2YY, \quad VV = S^2XX2CSXY + C^2YY.$$

$$UV = CSXX + (C^2S^2)XY + CSYY.$$

The definition of the perspective transformation has been given already as $x = u + Tuv, y = v$. Substituting these into the general formula the following non-zero initial values are obtained for the transformed registers:

$$U = X, \quad V = Y, \quad UU = XX, \quad VV = YY,$$

$$UV = TX + (1+T)XY + .5(T^2+T)XX,$$

$$UUV = (2T+T^2)XX, \quad UVV = 2TXY + T^2XX, \quad UUVV = 2T^2XX.$$

Generalisation for 3D

Voxels and Grid Points

In the previous investigations a finite part of the 2D plane has been divided into pixels. Consequently, as a first step towards the 3D generalisation a finite part of the 3D space has to be divided into voxels. Let us divide the 3D space into voxels by three sets (say an X set, a Y set and a Z set) of planes. Most frequently, the planes of each set are among themselves parallel and equidistant, any two planes from different sets are perpendicular, and the voxels are congruent rectangular parallelepipeds. An other important arrangement is where the dividing planes are parallel in one of the three sets onely, while the other two sets are pencils of planes (i.e. the planes of each set have a common straight line), the two 'carrier' straight lines of the two sets being intersecting and perpendicular. In this latter case the voxels are (in general) truncated pyramids with two parallel and four 'skew' faces.

A set of voxels forming a finite (regularly rectangular or pyramidal) part of the 3D space will be considered, being bordered by two X planes, two Y planes and two Z planes, containing (l+2) planes of the X set,(m+2) planes of the Y set and (n+2) planes of the Z set (boundary planes included). The division planes of the X set, of the Y set and of the Z set can be numbered by the integers $0...l+1, 0...m+1$ and $0...n+1$ resp. The points of intersection of any three division planes of different sets are the grid points. The grid points can be identified by the serial numbers of the three intersecting division planes, for example (i,j,k) is the grid point at the intersection of the ith plane of the X set, the jth plane of the Y set and the kth plane of the Z set (of course $0 \leq i \leq l+1, 0 \leq j \leq m+1$ and $0 \leq k \leq n+1$. The i,j and k values may be considered as the X, Y and Z coordinates of the grid points.

There are eight neighbouring grid points associated with a voxel, namely its eight corner points $(i,j,k), (i+1,j,k), (i,j+1,k), (i+1,j+1,k), (i,j,k+1), (i+1,j,k+1), (i,j+1,k+1), (i+1,j+1,k+1)$. A voxel will be identified by the coordinates (i,j,k), i.e. by the grid point with the lowest coordinate values out of its eight corner points. Thus the finite part of the plane under consideration is composed of the voxels with coordinates from (0,0,0) to (l,m,n).

Grid Values in 3D. Definition of Surfaces and Solids

The 3D Form of Additive Algorithms

Let us assign a positive or negative (possibly integer) number to each grid point, the grid value $R_{i,j,k}$. A voxel (i, j, k) is characterized by the eight grid values of the eight adjacent grid points: $R_{i,j,k}$, $R_{i+1,j,k}$, $R_{i,j+1,k}$, $R_{i+1,j+1,k}$, $R_{i,j,k+1}$, $R_{i+1,j,k+1}$, $R_{i,j+1,k+1}$, $R_{i+1,j+1,k+1}$. A voxel is negative homogeneous if all eight of its associated grid values are negative; a voxel is positive homogeneous if all eight of its associated grid values are zeros or positive numbers; in any other case the voxel is non-homogeneous.

Our investigations are founded on the following definition: a 3D solid is composed of the negative homogeneous voxels, the non-homogeneous voxels form its boundary surface, while the positive homogeneous voxels are outside the solid.

This definition is a straightforward generalisation of the definition given for the plane figures and their boundaries and the role of additive algorithms in constructing 'sensible' solids and surfaces is strictly analogous to the 2D case, too. The basic form of the 3D algorithms should produce the grid values step by step, i.e. starting from the grid value $R_{i,j,k}$ of any grid point should compute the $R_{i+1,j,k}$ or $R_{i-1,j,k}$ or $R_{i,j+1,k}$ or $R_{i,j-1,k}$ or $R_{i,j,k+1}$ or $R_{i,j,k-1}$ value of a neighbouring grid point. Thus if we want to define such an algorithm we have to specify how the grid value varies when stepping from a given grid point to one of its six neighbours (X step or Y step or Z step in positive or negative direction).

The possible simplest 3D additive algorithm is the following:

$$
\begin{array}{lll}
(X+ \text{ step:}) & i = i + 1, & R = R + X; \\
(X- \text{ step:}) & i = i - 1, & R = R - X; \\
(Y+ \text{ step:}) & j = j + 1, & R = R + Y; \\
(Y- \text{ step:}) & j = j - 1, & R = R - Y; \\
(Z+ \text{ step:}) & k = k + 1, & R = R + Z; \\
(Z- \text{ step:}) & k = k - 1, & R = R - Z.
\end{array}
$$

In these formulae i, j and k are the grid point coordinates, R is a 'register' containing the grid value. X, Y and Z are the modifying registers; by adding them to or subtracting them from the 'main' register R computes the algorithm the grid values corresponding to the neighbouring grid points. This simplest 3D additive algorithm produces a 'plane' as a boundary surface, on one side of which is the 'half space' forming the solid defined

grid points. This simplest 3D additive algorithm produces a 'plane' as a boundary surface, on one side of which is the 'half space' forming the solid defined by the algorithm, while the other 'half space' contains those voxels which are outside of this solid.

As another example let us consider now the 3D form of the second order additive algorithm:

$$
\begin{array}{ll}
(X+ \text{ step:}) & i = i+1,\ R = R+X,\ \ X = X+XX,\ Y = Y+YX,\ Z = Z+ZX; \\
(X- \text{ step:}) & i = i-1,\ Z = Z-ZX,\ Y = Y-YX,\ X = X-XX,\ R = R-X; \\
(Y+ \text{ step:}) & j = j+1,\ R = R+Y,\ \ Y = Y+YY,\ X = X+XY,\ Z = Z+ZY; \\
(Y- \text{ step:}) & j = j-1,\ Z = Z-ZY,\ X = X-XY,\ Y = Y-YY,\ R = R-Y; \\
(Z+ \text{ step:}) & k = k+1,\ R = R+Z,\ \ X = X+XZ,\ Y = Y+YZ,\ Z = Z+ZZ; \\
(Z- \text{ step:}) & k = k-1,\ Z = Z-ZZ,\ Y = Y-YZ,\ X = X-XZ,\ R = R-Z.
\end{array}
$$

As introduced earlier, a single well defined grid value R has been assigned to each grid point. Straightforward calculations show that this implies

$$XY = YX,\ XZ = ZX,\ YZ = ZY.$$

This is the necessary and sufficient condition in 3D for the second order additive algorithm to produce a single, well defined grid value in each grid point, not depending on the actual path used to approach the grid point in question, i.e. to be conservative. The description of solids and their boundary surfaces usually needs conservative algorithms. Nevertheless, there are special (e.g. spiral form) surfaces which do not define any solid and can be described by non-conservative algorithms. A non-conservative algorithm assigns in general more than one grid value to the same grid point depending on the route used to approach the grid point. All the same, a non-conservative algorithm can be used to describe a surface unambiguously if, for example, those grid values are considered only, which arise walking on the surface itself.

The second order additive algorithm (at least in its conservative form) defines solids bounded by second order surfaces such as spheres, cones, ellipsoids, paraboloids and hyperboloids. Of course, this statement has to be understood (as the similar statements in the 2D case) again with some restrictions. On the one hand, the spacing of the division planes used to form the voxels should be in accordance with the dimensions of the solid in question. Objects (or details of objects) smaller (or at least thinner) than the mesh width cannot be represented properly by the given resolution of the space and, consequently, it may happen that the proposed method 'does not take notice of them'. However, it is always possible to represent by means of the proposed method these 'small' objects (or details) as well using 'finer' voxels, i.e. a closer spacing of the division planes. On the other hand, the second order additive algorithm describes

second order surfaces in the sense of Euclidean plane geometry, if the voxels are formed by three perpendicular sets of parallel and equidistant planes. This is expressed in the above verification by the fact that the i, j, k voxel coordinates have been simply identified with the integer values of the x, y, z Cartesian coordinates. No doubt, this is by far the most natural and most frequently used arrangement of voxels, but because in some cases (e.g. for constructing perspective images) other forms of voxels can be expedient too, this condition is not trivial either.

The General Form of 3D Additive Algorithms

The second order step-by-step additive algorithm as presented in the previous paragraphs is rather restricted in its range of use. It is possible to generalize it for computing the grid value

- not just in a neighbouring grid point but in a more distant one and
- considering higher than second order modifiers, too.

For the sake of a concise formulation of the general p^{th} order multistep conservative additive algorithm in 3D let us introduce the notation $qXrYsZ$ for the $(q+r+s)$th order modifying register containing qX-es, rY-s and sZ-s in its identifier and let us accept the main register R as the zeroth order register $R = 0X0Y0Z$. The algorithm computes the grid value in a grid point which can be attained from the starting grid point (0,0,0) by stepping i times in X, j times in Y and k times in Z direction. With these conventions the general multistep form of the p^{th} order conservative additive algorithm is the following:

$$qXrYsZ_{i,j,k} = \sum_{u=q}^{p-r-s} \sum_{v=r}^{p-u-s} \sum_{w=s}^{p-u-v} \binom{i}{u-q} \binom{j}{v-r} \binom{k}{w-s} uXvYwZ_{0,0,0}$$

with

$$\binom{a}{b} = \frac{a(a-1)...(a-b+1)}{1.2...b}.$$

Transformations of the 3D Second Order Algorithm

We want to use the additive algorithms for representing and studying 3D solids and their boundary surfaces, therefore it is necessary to describe the same solid in different coordinate systems. Let us consider a new (x', y', z') (general, possibly curvilinear) coordinate system which is connected to the original (x, y, z) one by the equations $x = x(x', y', z')$,

$y = y(x', y', z'), z = (x', y', z')$. It is possible to describe the same solid with the same boundary surface by an appropriate additive algorithm even in the new coordinate system. For this purpose the initial values of the registers have to be modified. The general rules of modification in the 3D case are straightforward generalisations of those presented for the plane figures. Instead of writing out the general formulae the most important three special cases will be presented for the second order case.

a) Stretching in x direction. If $x = cx', y = y', z = z'$ the initial values of the following registers are to be modified:

$$X' = cX + c(c - -1)/2XX, XX' = c^2XX, XY' = cXY, XZ' = cXZ.$$

All other registers remain unchanged. The formulae for stretching in the other coordinate directions can be obtained by an appropriate modification of the letter symbols referring to coordinate directions.

b) Rotation about the z axis. If $x = cx' - -sy', y = sx' + cy', z = z'$ (where $c = \cos\beta$, $s = \sin\beta$) then the initial values of the following registers are to be modified:

$$X' = cX + sZ + c(c1)/2XX + csXZ + s(s1)/2ZZ,$$

$$Z' = sX + cZ + s(s1)/2XXcsXZ + c(c1)/2ZZ,$$

$$XX' = c^2XX + 2csXZ + s2ZZ, ZZ' = s^2XX2csXZ + c^2ZZ,$$

$$XZ' = csXX + (c^2s^2)XZ + csZZ.$$

All other registers remain unchanged. The formulae for rotating about the other coordinate axes can be obtained by an appropriate modification of the letter symbols referring to coordinate directions.

c) Perspective transformation. If $x = x'Tx'y', y = y', z = z'Ty'z'$, the initial values of the following register are to be modified:

$$XY' = T.X + (1T)XY + (T(T1)/2)XX, \quad XXY' = T(T2)XX,$$

$$YZ' = T.Z + (1T)YZ + (T(T1)/2)ZZ, \quad YZZ' = T(T2)ZZ,$$

$$XYY' = -2T.XY + T2XX, \quad YYZ' = 2T.YZ + T2ZZ,$$

$$XXY' = T(T2)XX,$$

$$XYZ' = T(T2)XZ, \quad YZZ' = T(T2)ZZ,$$

$$XXYY' = 2T^2XX, \quad XYYZ' = 2T^2XZ, \quad YYZZ' = 2T^2ZZ.$$

Set Operations

The representation of voxel solids and voxel surfaces in form of additive algorithms facilitates their computer manipulation in many respects. For example one can find relatively efficient and simple algorithms for rendering the curved surfaces or drawing their outlines, the lines of intersection of two surfaces, etc. Concluding this paper let us illustrate now the advantages of the additive algorithm representation by stating the simple rules for set operations.

Let two solids (solid A, and solid B) be represented by the two arrays of grid values $R'_{i.j.k}$ and $R''_{i.j.k}$. Let $R_{i.j.k}$ represent a solid constructed from the two previously defined solids by set operations. The following simple formulae are valid:

If the solid represented by R is the complement of solid A:

$$R_{i.j.k} = -R'_{i.j.k}.$$

If the solid represented by R is the union of the solids A and B:

$$R_{i.j.k} = \mathrm{Min}(R'_{i.j.k}, R''_{i.j.k}).$$

If the solid represented by R is the intersection of the solids A and B:

$$R_{i.j.k} = \mathrm{Max}(R'_{i.j.k}, R''_{i.j.k}).$$

Address:

Dr. József PERÉDY
Department of Descriptive Geometry II
Technical University of Budapest
H-1521, Budapest, Hungary