Accelerating the Ray Tracing of Height Fields

Christian Henning *
University of Rostock

Peter Stephenson †
imedia

Abstract

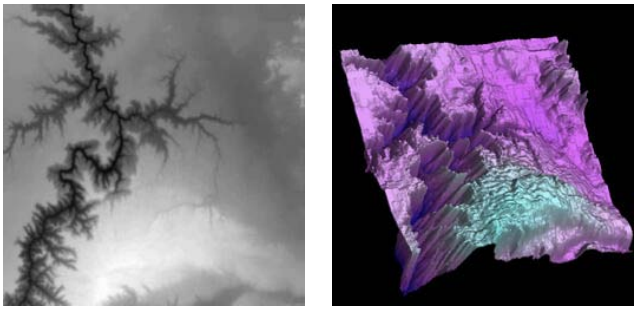
In this paper we apply a run–based ray–traversal algorithm to accelerate the ray tracing of height fields. To intersect the ray and the height field three processes occur: the ray is traversed through the grid that underlies the height field, an intersection between each cell and the ray is sought and if found the intersection point between the ray and the height field is calculated. Run–based ray traversal algorithms, to determine the path of the ray through the height field, have a significant computational advantage over the cell–based traversal algorithms used to date. They also present important structural information that can be used to accelerate the process of determining if and where an intersection has occurred.

CR Categories: I.3.m [Computing Methodologies]: Computer Graphics

Keywords: Ray tracing, height fields, terrain rendering

1 Introduction

Height field rendering has a range of important applications including geographic information systems, scientific visualization, special effects, and terrain rendering. While height fields can be defined in a number of ways, typically a digital height field or elevation map is a uniform grid sampling of the distance of a surface from a plane. Figure 1 provides an example where the height of the field is shown as a linear grey scale value.



(a) Height field (b) Rendering

Figure 1: Example height field from the Grand Canyon.

*e-mail:chhenning@gmx.net
†e-mail: ps@imedia-academy.org

Currently there are two main approaches to render height fields: converting the height field into an alternative representation to utilise a hardware renderer, or to directly render the height field using ray tracing. Currently the most popular approach is to convert the height field into a polygon mesh or point set that can be rendered using a commodity graphics card. For large height fields, level of detail techniques [Lindstrom et al. 1996; Duchaineau et al. 1997; Hoppe 1998] and occlusion schemes [Stewart 1997; Agranov and Gotsman 1995] can be applied to reduce the amount of data processed. A similar idea is to transform the height field into a volumetric representation and to use volume rendering hardware or GPU–based texture slicing.

As opposed to converting the height field to exploit available hardware, ray tracing has a number of advantages. Firstly, the original height field has a more concise representation than a derived geometric or volumetric version and therefore lower memory requirements. Secondly, for large datasets, ray tracing is an image order algorithm that inherently ignores occluded regions of the height field. Lastly, it is simple to include multiple object representations including geometric, volumetric, and procedural models, and global illumination effects such as reflection, refraction, and soft shadowing. The primary disadvantage is that there is no direct hardware support. However due to the recent development of programmable GPUs, several research groups are developing GPU supported ray tracers [Purcell et al. 2002], also for height fields [Qu et al. 2003], and influencing the future design of this hardware.

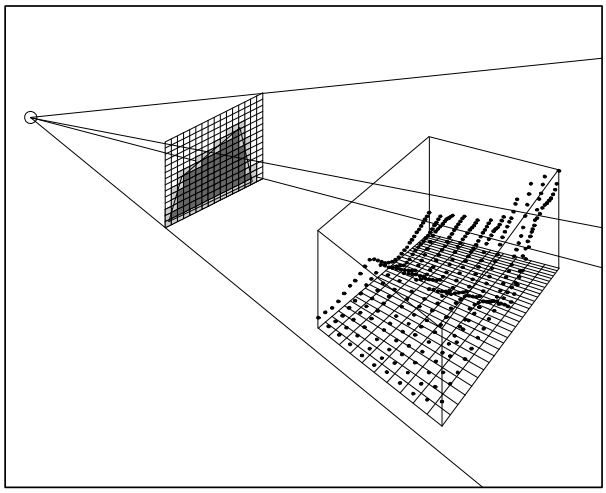


Figure 2: The viewing frustrum of the height field scene.

In a simple ray tracing scheme, one ray per pixel is intersected with the bounding box of the height field (c.f. Figure 2). If an intersection is found, the ray is intersected with the height field. The task of intersecting the ray and the height field is performed most efficiently by projecting the ray onto the base plane of the height field (c.f. Figure 3). The ray can then be iteratively traversed over the two–dimensional base plane using an algorithm such as the digital difference analyser (DDA) algorithm [Musgrave 1988] or the midpoint algorithm [Cohen and Shaked 1993]. While the use

of a line drawing algorithm, such as the midpoint algorithm, may be slightly less expensive than the DDA algorithm due to the use of integer arithmetic, the ray path generated will probably include errors unless the slope of the ray can be represented as a rational number [Stephenson 1998; Stephenson and Litow 2001].

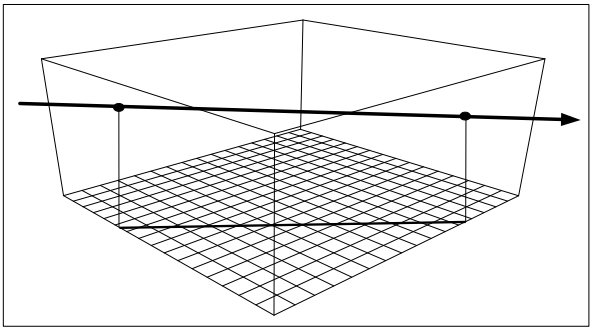


Figure 3: Projection of the ray onto the height–field base–plane.

As the projected ray is traversed over the base plane it is intersected against the neighbouring set of height samples of each cell the ray passes through (c.f. Figure 4). A number of intersection strategies are possible that are suitable for height fields of different sizes and resolutions. The simplest intersection test is to test the ray against the surrounding height samples of a cell. If this test is positive, the intersection point can be taken as the lowest value [Cohen and Shaked 1993; Cohen-Or et al. 1996], or used to engage a more sophisticated and expensive intersection test. The most common intersection test is to construct a polygonal or planar representation of the surface within the cell and to use a plane–ray or polygon–ray intersection test [Musgrave 1988], which can be GPU supported [Qu et al. 2003]. Beyond these techniques surface reconstruction for height fields by ray tracing has not been extensively studied [Qu et al. 2003]. The calculated point of intersection once calculated or assigned is then attributed properties based on a color or texture map and rendered, typically by applying a local illumination model.

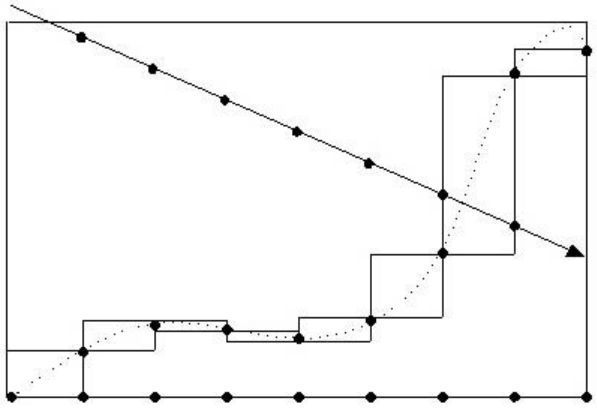


Figure 4: Intersecting the ray with the bounding heights of neighboring cells.

In this paper we describe a more efficient technique to traverse the ray through the height field. We describe the path of the ray as a collection of runs of cells each of which can be calculated with the same cost as each individual cell in a cell–based algorithm. The run-based ray traversal scheme is $O(n/r)$ compared to $O(n)$ for cell based algorithms, where n is the number of cells in the ray path and r is the average run length of the ray. The algorithm has been mathematically proven to produce a correct digitization of the ray path and can be adapted easily to include corner intersections [Stephenson and Litow 2001].

A run of cells contains structural information of the path of the ray that can be used to accelerate secondary processes in the rendering pipeline. We use the structural information presented by the ray traversal technique to very cheaply initiate a linear interpolation of the position of the intersection point from surrounding height field samples. This produces a higher quality reconstruction of the height field surface than traditional techniques. We also discuss the potential for spatial subdivisions that would reduce the bandwidth required for ray tracing operations and may be suitable for level of detail operations.

2 Ray Traversal

We assume the ray is in the coordinate system of the height field and that the height field samples are distributed over a subregion of the integer lattice. Therefore the height field can be represented as an image, array or grid, where every lattice position or cell corner, (x_j, y_j), defines the position of a height sample. To describe the path of the ray through the two–dimensional lattice, we represent the ray by the line $y = \alpha x + \beta$ where α and β are algebraic real numbers. In discussing lines and rays, we will attempt to maintain the distinction that a line is infinite in either direction, a ray is infinite in only one.

The description of the digital ray we will consider is typical for working with the digital line, see for example [Wu 1982; Stephenson and Litow 2001]. We will consider the 8–connected version of the digital ray such that the ray intersects the left face of each cell in the digitised ray path. The ray cell intersection occurs at the point $(x_j, y_j + \beta_j^{[0]})$ where x_j and y_j are integers, $\beta_j^{[0]}$ is real and $0 \leq \beta_j^{[0]} < 1$. To simplify the problem, we assume the slope of the line is within the range $0 \leq \alpha \leq 1$ as all other lines can be generated from this line given the 8–way symmetry of Cartesian space about its origin.

To generate the 4–connected version of the digital ray necessary for ray tracing, the additional cells intersected by the line can be added by extending the beginning of each run of cells within the line by one pixel except where a corner intersection occurs, as shown in Figure 5.

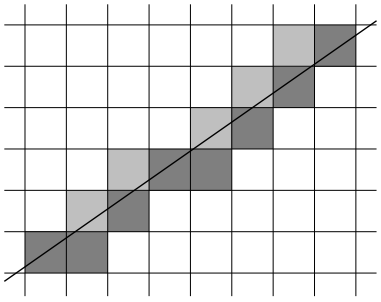


Figure 5: The minimal 8–connected definition of the line.

To describe the structure of the runs within the digital ray, we will concentrate on the line of real slope and zero intercept $y = \alpha x$ because the structure of the line depends only on the slope of the line as can be seen by the definitions presented by Wu [Wu 1982] and Rosenfeld [Rosenfeld 1974]. A non–zero value of the intercept does not serve to alter the structure of the line but causes a shifting of the structure within the line. The incremental algorithm we present constructs the runs within the line based solely on the

value of the intercept at the beginning of each run. Altering the value of the intercept line serves only to alter the placement of the runs within the line.

A run is defined as a set of contiguous cells having the same y coordinate. It is characterised by its length, r_j, the number of composite cells and its start point, (x_j, y_j), the position of the left most cell in the run. To describe the 8–connected digital line using runs, we can use the fact that each run in the line is corner connected. Therefore the line is an ordered set of run lengths where the position of each composite run can be calculated from the position and length of the previous in the set:

$$(x_{j+1}, y_{j+1}) = (x_j + r_j, y_j + 1). \tag{1}$$

This reduces our problem to that of calculating the length of each run in the line. A property of the lengths of runs in the line critical to how we form this calculation is the fact that runs within a digital line occur in a maximum of two lengths which are consecutive integers [Rosenfeld 1974; Stephenson and Litow 2001]. The first run in a digital ray however may be truncated.

As there are only two possible run lengths in the line, we can refer to each run as being either *short* or *long* and to discuss the structure of runs in a line, denote a short run by the symbol s and a long run by l. For example, the run sequence *lslslsslslsslslss* describes the structure of the runs in the digital line $y = \frac{17}{41}x$ shown in Figure 6, our example line. The long runs in the figure are designated by light grey and the short runs, dark grey. The upper collections of runs we will describe in a possible next publication. We should note that we choose a line of rational slope to allow a simpler discourse as this sequence of run lengths displayed in the line segment is repeated throughout the entire line.

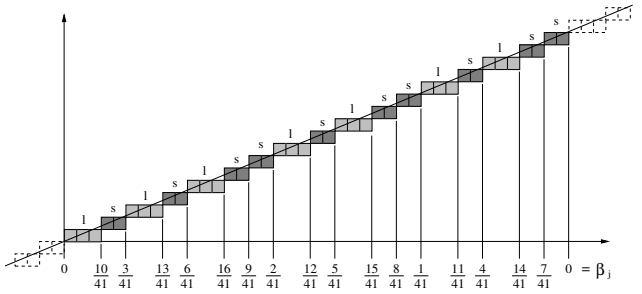


Figure 6: The structure of runs within the digital line $y = \frac{17}{41}x$.

To determine the lengths of runs that can occur in any given line, consider that the slope of the line over a segment of the line is given by $\alpha = \frac{\delta y}{\delta x}$, where δx is the number of cells in the line segment and δy is the number of runs. Therefore, it can be said, the slope of the line defines the ratio of runs to cells in the line.

As the possible run lengths are at most two, integral and measured in cells, the length of a short run must therefore be $\lfloor \frac{1}{\alpha} \rfloor$ and the length of a long run $\lceil \frac{1}{\alpha} \rceil$ as $\frac{1}{\alpha}$ is the average length of a run in the line. For lines of slope $\frac{1}{k}$ where $k = 2, 3, \ldots$, only one run length actually exists. We will however ignore this case for now and assume that two run lengths exist: short, $r = \lfloor \frac{1}{\alpha} \rfloor$, and long, $r+1 = \lceil \frac{1}{\alpha} \rceil$. From our example in Figure 6, we have 41 cells and 17 runs in this line segment, therefore the possible run lengths are $r = 2$ and $r+1 = 3$.

Within Figure 6 we have also described the sequence of values of the fractional component of the intercept of the line at the start of each run, β_j for $j = 0, 1, 2, \ldots$, in the upper sequence of numbers. We shall refer to this sequence of values as the *intercept sequence*. From Figure 6, it can be seen the values of the intercept sequence are all less then the slope of the line, $\alpha = \frac{17}{41}$, and the long runs within the line correspond to the intercept values less than $\frac{7}{41}$. The short runs coincide with the values of the intercept sequence greater than or equal to $\frac{7}{41}$. The ratio of long runs to cells in the line is also $\frac{7}{41}$, which is not coincidental.

Where (x_j, y_j) is the coordinate of the first cell in a run in the line, the intercept sequence can be calculated by

$$\beta_j = \alpha x_j - y_j \tag{2}$$

hence the magnitude of each value of the intercept sequence is less than one, $0 \leq \beta_j < 1$. From the definition of the connectivity of runs within the line given by Equation 1 and the definition of the intercept sequence given by Equation 2, we have the difference between two successive values in the intercept sequence:

$$\beta_{j+1} - \beta_j = \alpha r_j - 1. \tag{3}$$

Figure 7 describes the intercept geometry of a run and the continuous representation of the line. The line is set such that it intersects the end–point of a short run, $(x_{j+1}, y_{j+1}) = (x_j + r, y_j + 1)$, therefore $r_j = r$ and $\beta_{j+1} = 0$. This position of the line is the critical position for deciding the length of the run. If the line lies below this position, the run must be long. If the line lies on or above this position, the run must be short. These are the only two possibilities. Therefore when $\beta_j \geq \nu$ the length of the run is short and when $\beta_j < \nu$, long. As $r_j = r$, $\beta_j = \nu$ and $\beta_{j+1} = 0$, from Equation 3 we have

$$\nu = 1 - \alpha r. \tag{4}$$

As α is the ratio of runs to cells and r is the length of a short run, ν is also the ratio of long runs to cells in the line. Similarly we can define μ to be the ratio of short runs in the line to cells:

$$\mu = \alpha - \nu = \alpha(r+1) - 1. \tag{5}$$

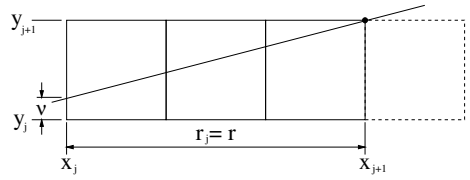


Figure 7: The geometry of the intercept of a run and the line.

The ratios of long and short runs to cells also play a central role in the iterative calculation of the intercept sequence. The next value of the intercept sequence, β_{j+1}, can be calculated from the current, β_j, if the current run is short, $r_j = r$, from Equations 3 and 4 by

$$\beta_{j+1} - \beta_j = -\nu. \tag{6}$$

If the run is long, $r_j = r+1$, by Equations 3 and 5, we have

$$\beta_{j+1} - \beta_j = \alpha(r+1) - 1 = \mu. \tag{7}$$

We therefore have a complete iterative definition for the structure of runs within the line $y = \alpha x$. To use this definition to traverse a ray through a lattice, the essential difference between the line and ray is that the path of the ray through the lattice has a start point and is 4–connected except where a corner intersection occurs.

As the path of the ray is bounded by a starting position, the first run in the path may be truncated. We can assume the starting point of the ray within the lattice is defined by $(x_0, y_0 + \beta)$ where x_0 and y_0 are integers and β is real such that $0 \leq \beta < 1$. For the first run in the line to be truncated, its length must be less than that of

a short run, $r_0 < r$. The determining position of the continuous line for truncation to occur is such that the line intersects the point $(x_0 + r - 1, y_0 + 1)$ and

$$\beta \geq 1 - (r-1)\alpha = \alpha + \nu \tag{8}$$

Therefore if the line lies on or above this point the initial run is truncated.

If the first run is truncated, its length must be calculated as well as the the first intercept sequence value, which corresponds to the start of the first full length run in the path of the ray. From geometry, the length of a truncated run, where $\Delta r = \left\lfloor \frac{\beta - \nu}{\alpha} \right\rfloor$, is

$$r_0 = \left\lceil \frac{1-\beta}{\alpha} \right\rceil \tag{9}$$

$$= r - \Delta r. \tag{10}$$

To calculate the first value in the intercept sequence from the value of the intercept of the line, we have that $\beta_1 = \beta$ if $\beta < \alpha + \nu$ as the first run is not truncated. If the first run is truncated however from Equation 3

$$\beta_1 = (\beta - \nu) - \Delta r \alpha. \tag{11}$$

To make the decision of run length and the initialisation of the initial truncated run length and initial intercept value more efficient, we can translate the values of the intercept sequence by the value $-\nu$ by initially subtracting ν from β before initialising the algorithm. Therefore, the initial run is not truncated if $\beta - \nu < \alpha$ and hence $r_0 = 0$ and $\beta_1 = \beta - \nu$. If the initial run is truncated, our calculations of r_0 and β_1 are simplified, as is our decision of which run length is next in the line or path of the ray, which can be made against zero and implemented as a sign check. Also if $\frac{1}{2} \leq \alpha < 1$, a truncated run is not possible as the length of a short run is always one. Therefore we need only consider the possibility of an initial truncated run for slopes in the range $0 < \alpha < \frac{1}{2}$.

To convert the definition of the path of the ray from being 8–connected to 4–connected, we can extend each run back by one cell with no overhead to the algorithm. However this extension should only take place if there is no corner intersection, which can only occur at the beginning of a run coinciding with an intercept value of zero (or $-\nu$).

3 Surface Reconstruction

As the ray is traversed through the grid underlying the height, within each cell the bounding box of the four height samples can be tested very simply against the ray to determine if an intersection of the height field can occur. When a bounding box intersection is discovered, we must determine if an height field intersection has occurred and if so, determine the point of intersection. Four approaches has been used in the literature as described in Figure 8.

The simplest and most common approach described in Figure 8(a) chooses one of the four sample values from the cell as the intersection point, equivalent to a nearest neighbor strategy. An intersection is said to have occurred if the axis-aligned plane (voxel plane) is intersected by the ray [Cohen and Shaked 1993; Cohen-Or et al. 1996; Wan et al. n. d.].

Our approach, as shown in Figure 8(b), constructs a plane by three out of the four height samples. In a first step, the two surface heights are linearly interpolated at the positions the projected ray intersects the cell faces using parameters from the run-based traversal technique. The second step now decides if there is an intersection of ray and the reconstructed height field surface. An intersection must happen if the algebraic signs of the two distances of ray and surface

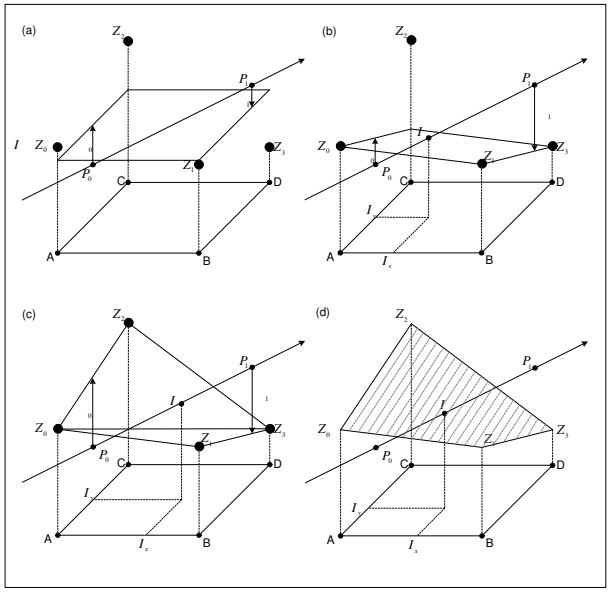


Figure 8: Approaches to surface reconstruction: (a) Nearest neighour or average approximation, (b) Linear plane approximation, (c) Polygon splitting, (d) Bilinear surface construction.

are different. This is accomplished if one ray point lies below and the other lies above the surface, or vice versa. If the signs are different the intersection point is linearly interpolated by the connecting lines of the two surface heights and the two ray heights. Now a rendering function can calculate the colour of the current pixel, based on the point of intersection.

Other more expensive approaches have also bee used. In Figure 8(c) the four height samples define two triangles and a standard ray/polygon intersection calculation is used to determine the point of intersection [Musgrave 1988; Qu et al. 2003]. In Figure 8(c), a bilinear surface can also interpolated from the height samples.

4 Results

A prototypical system based on the ideas presented has been developed in software to compare the use of run–based to cell–based ray traversal techniques. Results of two datasets are presented. The first dataset is a Grand Canyon elevation map from the U.S. Geological survey (USGS) containing 512×512 samples (c.f. Figure 9). The rendered image was generated using linear interpolation to calculate the position of the intersection point, texture mapping and a Phong illumination model.

The seocnd dataset is from a neuroscience experiment by Arnold Heynen of the Massachusetts Institute of Technology (MIT) on thalamocortical transmission in the adult visual cortex in which the neocortical response is measured after stimulation (c.f. Figure 10) [Heynen and Bear 2001]. The dataset has 512×512 samples and was rendered using linearly interpolated intersection points, a height–based colour map, and a Phong illumination model.

The run–based ray–traversal scheme is $O(n/r)$ compared to $O(n)$ for cell based algorithms, where n is the number of cells in the ray path and r is the average run length of the ray. The advantage of using a run–based over a cell–based ray traversal algorithm therefore depends heavily on the orientation of the height field to the ray. An average acceleration of 125% was measured over all orientations. Where the average run length was close to one, no improvement was measured. For average run lengths of

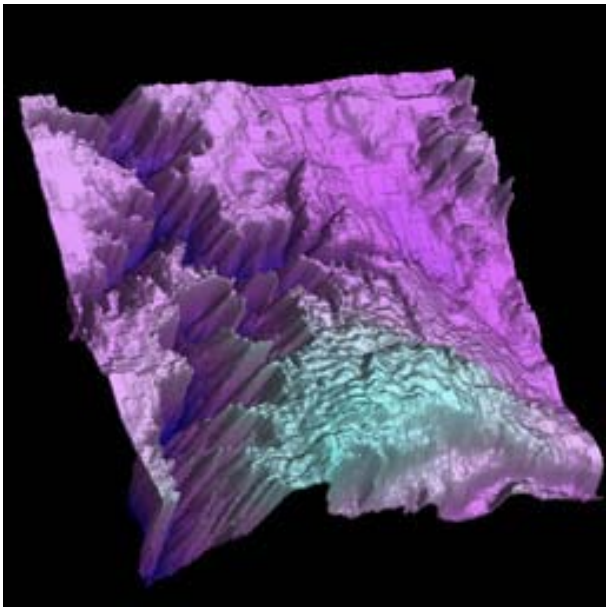


Figure 9: Digital elevation dataset.

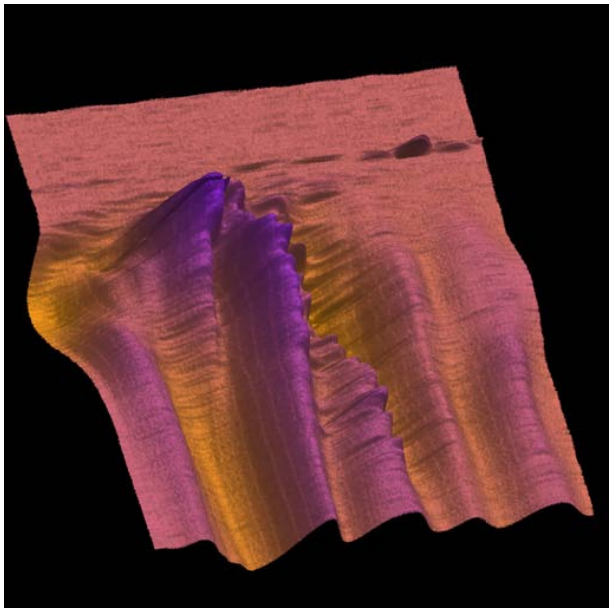


Figure 10: Neuroscience dataset.

two an improvement of 110% was found, and for 25, an improvement of 180%. Because shorter run lengths are more prevalent, using symmetry of the line around a slope of $1/2$ to ensure that the run lengths are always at least two and using a higher order run algorithm [Stephenson and Litow 2001] are being considered.

5 Conclusion

Each of the stages of ray tracing a height field, ray traversal, intersection testing and intersection point rendering, incurs a significant contributing cost to the overall ray tracing process. In this paper we have presented a technique that not only accelerates the ray traversal process but offers an efficient and higher quality reconstruction of the height field surface based on the run–based traversal algorithm.

The structure of the run–based definition of the ray path suggests further possibilities. Traditionally spatial subdivisions such as quadtrees and octrees are used in occlusion maps and level of detail hierarchies. They are also used to skip the ray over uninteresting regions of the height field. How to best combine run–based techniques and these spatial data structures is currently being investigated. New run–based subdivisions other than run length encoding are also being considered.

References

AGRANOV, G., AND GOTSMAN, C. 1995. Algorithms for rendering realistic terrain image sequences and their parallel implementation. *The Visual Computer 11*, 9, 455–464. ISSN 0178-2789.

COHEN, D., AND SHAKED, A. 1993. Photo-realistic imaging of digital terrains. *Computer Graphics Forum 12*, 3 (September), 363–373.

COHEN-OR, D., RICH, E., LERNER, U., AND SHENKAR, V. 1996. A Real-Time Photo-Realistic Visual Flythrough. *IEEE Transactions on Visualization and Computer Graphics 2*, 3 (September), 255–264.

DUCHAINEAU, M. A., WOLINSKY, M., SIGETI, D. E., MILLER, M. C., ALDRICH, C., AND MINEEV-WEINSTEIN, M. B. 1997. ROAMing terrain: Real-time optimally adapting meshes. In *IEEE Visualization '97*, 81–88.

HEYNEN, A., AND BEAR, M. 2001. Long–term potentiation of thalamocortical transmission in the adult visual cortex in vivo. *Journal of Neuroscience 21*, 24 (December), 9801–9813.

HOPPE, H. 1998. Smooth view-dependent level-of-detail control and its application to terrain rendering. In *Proceedings IEEE Visualization'98*, IEEE, 35–42.

LINDSTROM, P., KOLLER, D., RIBARSKY, W., HUGHES, L. F., FAUST, N., AND TURNER, G. 1996. Real-Time, continuous level of detail rendering of height fields. In *SIGGRAPH 96 Conference Proceedings*, Addison Wesley, H. Rushmeier, Ed., Annual Conference Series, ACM SIGGRAPH, 109–118. held in New Orleans, Louisiana, 04-09 August 1996.

MUSGRAVE, F. K. 1988. Grid tracing: Fast ray tracing for height fields. Technical Report YALEU/DCS/RR-639, Yale University Dept. of Computer Science Research.

PURCELL, T. J., BUCK, I., MARK, W. R., AND HANRAHAN, P. 2002. Ray tracing on programmable graphics hardware. In *Proceedings of the 29th Conference on Computer Graphics and Interactive Techniques (SIGGRAPH-02)*, ACM Press, New York, S. Spencer, Ed., vol. 21, 3 of *ACM Transactions on Graphics*, 703–712.

QU, H., QIU, F., ZHANG, N., KAUFMAN, A., AND WAN, M. 2003. Ray tracing height fields. In *Computer Graphics International*, 202–209.

ROSENFELD, A. 1974. Digital straight line segments. *IEEE Transactions on Computers C-23*, 12 (December), 1264–1269.

STEPHENSON, P. D., AND LITOW, B. 2001. Making the DDA run: Two–dimensional ray traversal using runs and runs of runs. *Australian Computer Science Communications (24th Australian Computer Science Conference Proceedings) 23*, 1 (February), 177–183.

STEPHENSON, P. D. 1998. *The Structure of the Digitised Line: With Applications to Line Drawing and Ray Tracing in Computer Graphics*. Ph.d. thesis, James Cook University.

STEWART, A. J. 1997. Hierarchical visibility in terrains. In *Eurographics Rendering Workshop 1997*, Springer Wien, New York City, NY, J. Dorsey and P. Slusallek, Eds., Eurographics, 217–228. ISBN 3-211-83001-4.

WAN, M., ZHANG, N., KAUFMAN, A., AND QU, H. Interactive stereoscopic rendering of voxel-based terrain. In *IEEE Virtual Reality*.

WU, L. 1982. On the chain code of a line. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-4*, 347–353.