http://www.moderntechno.de/index.php/meit/article/view/meit06-01-065
DOI: 10.30890/2567-5273.2018-06-01-065

УДК 004.925

THE METHOD OF SPLATTING THE FILTER-BASED WEIGHTED AVERAGE

МЕТОД СПЛЕТІНГУ НА ОСНОВІ СЕРЕДНЬОЗВАЖЕНОГО ФІЛЬТРА

Vyatkin S.I. / Вяткин С.И.
Institute of Automation and Electrometry SB RAS
Romanyuk A.N. / Романюк А.Н.
Necheporyk M.L. / Нечипорук Н.Л.
Roptanov V.I. / Роптанов В.И.
Vinnytsia National Technical University

Annotation. *Modern laser and optical scanners require scene rendering techniques that process millions of points with high-resolution textures. This paper describes a method for rendering and filtering textures based on the weighted average filter (EWA) and splatting. The method provides high quality anisotropic texture filtering, removal of hidden surfaces, antialiasing and transparency.*

Keywords: *splatting, rendering, texture, antialiasing*

Introduction

Complex models can be visualized using laser rangefinder and image-based scanning techniques [1]. One of the problems of these methods is the huge amount of points they generate. A commonly used approach is to generate triangular meshes from point data and use mesh reduction algorithms to render them [2, 3]. However, some scanned meshes are too large for interactive rendering [1]. For some applications it is impossible to prevent the loss of precision of the geometric data and texture, through the reduction of polygons. Direct methods for rendering scanned points were developed [4-6]. Textures can be used to increase the visual complexity of objects by [7]. Representation of objects as a set of points and their use for rendering is shown in [6, 8].

In this article, we propose a method of point rendering based on texture filtering splatting (Elliptical Weighted-EWA [9]) for unevenly spaced points without global texture parameterization. The basis of the splatting method is a model for representing continuous texture functions on the surface of graphic objects. Usually three-dimensional points are arranged irregularly, so the weighted sum of radially symmetric basis functions is used. The problem of visualization of point objects is considered as a concatenation with a continuous texture function.

Method description

The proposed method stores an explicit representation of the texture in object space. Point objects are described as a set of unevenly spaced points in the three-dimensional space of objects without connectivity. A point has a position and a normal. It is related to the radial-symmetric basis function and to the coefficients $q^{R}{}_{k}, q_{k}{}^{G}, q_{k}{}^{B}$ that represent continuous functions for red, green, and blue.

We define a continuous surface function as a weighted sum:

$$f_c(\vec{u}) = \sum_{k \in N} q_k R_k(\vec{u} - \vec{u}_k)$$

The General resampling structure for texture mapping and the EWA texture

filter is shown in [10]. Sample continuous output function to obtain a discrete value:

$$g(\vec{x}) = g'_c(\vec{x})i(\vec{x})$$

Explicit expression for a continuous function

$$g'_c(\vec{x}) = \sum_{k \in N} q_k p_k(\vec{x})$$

Where

$$p_k(\vec{x}) = (R'_k \otimes F)(\vec{x} - \vec{m}_{uk}(\vec{u}_k))$$

where R'_k (warped) is the basis function. Although the texture function is defined on an irregular grid, equation (4) States that the kernel of the recalculation in screen space can be written as a convolution of the curved basis function.

For each point Pk, the matrix must be chosen appropriately in order to display the local density of the points around Pk. The basis function Rk is a Gaussian with a matrix $\vec{V}_k^R$. The decision to select $\vec{V}_k^R$:

$$\vec{V}^R{}_k = \begin{pmatrix} 1/F^2 & 0 \\ 0 & 1/F^2 \end{pmatrix}$$

As described in [9], we select an elliptic Gauss filter for both the basic functions and the low pass filter. EWA screen space filtering begins by projecting a radially symmetric Gauss basis function from an object onto the image plane, resulting in an ellipse. An ellipse folded with a Gaussian low pass filter, form contributions accumulating in screen space. The core of the conversion pk (x) is determined by the Jacobian (J) 2D-to-2D mapping, i.e. the transformation of coordinates the local parameterization of the surface in the coordinate system of the observer. This mapping consists of combining an affine transformation of the view that displays the object into camera space and a perspective projection into screen space. In a view transformation, we do not allow for no uniform scaling or shifting.

This means that we preserve the rotation invariance of the basic functions in the chamber space. Therefore, the Jacobian of this transformation can be written as a uniform scaling matrix with a scaling factor. Similarly, for the viewport, you can describe the Jacobian with a scaling factor. To calculate the Jacobian of a perspective projection, you must calculate the local parameterization of the surface. After the view transformation, the objects are defined in camera coordinates, which can be projected by simply dividing by the z coordinate. The center of the projection is at the origin of the camera space, the plane of the projection is the z = 1 plane. We define the parametrization by choosing two orthogonal basis vectors u0 and u1 in the tangent plane. The functions are radially symmetric; the orientation of these vectors is arbitrary. We find the Jacobian J of the inverse map at point Pk by projecting the basis vectors of the screen of space x0 and x1 along the observation beam connecting the center of the projection using Pk to the tangent plane.

Each Pk point for position m (uk) is displayed on the screen. The kernel then the conversion should be centered at m(uk) and is calculated for each pixel. The contributions of all points are accumulated in the buffer. The projected normals of the points are filtered as well. Besides color and normal components, each pixel in the frame buffer contains the amount of the accumulated contributions of the conversion

of the nuclei and the z-value of space camera.

Because the pixel grid in screen space is regular, the kernel can be evaluated in a rectangular area and use a lookup table. In most cases, the depth complexity of the scene is greater than one, a mechanism is required that separates the contributions from the different surfaces in the frame buffer. The z value of the tangent plane at Pk is calculated at each pixel that is covered by the kernel and used to do so. To determine whether the new contribution belongs to the same surface that is already stored in the pixel, you must calculate the difference between the new z value and the z value stored in the frame buffer. If the difference is less than the threshold, the contribution is added to the pixel. Otherwise, the frame buffer data is replaced with a new contribution.

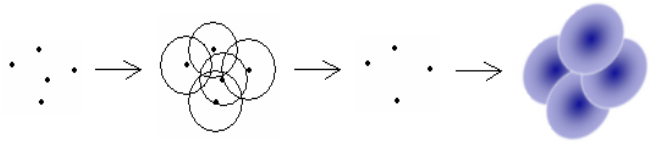

Figure. 1. The point in the center is the z-furthest.

Once projected onto a plane, its radius is completely covered by the radii of other points, hence it is discarded.

The frame buffer is shaded after all points in the scene are projected. This avoids the shading of invisible points (Fig. 1). Instead, each pixel is shaded by a filtered normal. The parameters for the Shader are available through the index of the material properties table. Advanced methods of shading of pixels, such as display of the reflection, you can simply add. There are two different settings when calculating qk coefficients (1), (3): 1. Objects with the color of the point. Method samples the color at the point. 2. The textures of point objects. Graphic or procedural textures from external sources.

Many modern imaging systems, such as laser range scanners or passive vision systems [5] have range and color information. In such cases, there is a color ck at each point. In another case, it is necessary to calculate the continuous approximation fc (u) of the unknown original of the texture function from the irregular set of ck samples. The optimal approximation is the normalization of the basis function. The samples are then used as coefficients. When a graphic or procedural texture is explicitly applied to point geometry, the mapping function from texture space to feature space can be calculated in advance (preprocessing). This allows you to deform (to warp) a continuous texture function from a texture space with s coordinates to an object space with u coordinates.

The rendering method can be extended to handle transparent surfaces. In addition, the approach provides an independent order of processing transparent surfaces using a single rendering pass and a fixed amount of memory buffer frames. The General idea is to use a frame buffer that consists of several layers. A layer stores a fragment at each pixel. The purpose of the fragment-to collect the contribution of one surface per pixel. The order from the farthest points to others.

The method presented in [11] is used to avoid disadvantages both multi-pass [12] and the main A-buffer [13] algorithms. Providing a small fixed number of one

fragments per pixel, the parts are merged when the number of parts exceeds the preset limit of one.

Unlike a single-layer frame buffer, a frame buffer contains several layers, each of which stores a fragment per pixel. Each contribution that is divided into pixels is processed in three stages:

1. Accumulation or separate solution. Using the z threshold, all pixel fragments are checked for data belonging to the same surface as the new contribution.

In this case, the contribution is added to the fragment and the end of the procedure. Otherwise, the new contribution is treated as a separate surface and temporary fragment initializes it with the data.

2. Insert a new fragment. If the number of fragments, including the temporary fragment, is less than the one limit, the temporary fragment is copied to the free slot in the buffer.

3. The fusion fragments. If the above is not true, the two fragments must be combined. The fragments must be shaded before merging.

When merging fragments, some information is inevitably lost and visual artifacts can occur. These effects are minimized using an appropriate merge strategy.

Unfortunately, the situation is complicated by the fact that the decision must be made, without information about the subsequent operational visualization. The main criterion for merging fragments is the difference between their z-values. This reduces the chance that other surfaces that are processed later will lie between two merged surfaces. In this case, the back-to-front is incorrect mixing can result in visual artifacts.

Before combining the fragments, it is necessary to determine their final color, shading them. The setting of their total weight indicates the shaded fragments. This is a negative value to ensure that they are shaded exactly once.

To perform edge smoothing, partial coverage information is required surfaces'. For point representations, one way to approximate coverage is to estimate the density of points per pixel area. The coverage is then calculated by measuring the actual point densities in the fragment and dividing the measured value by the evaluation criterion. Instead of explicitly computing this estimate, a simplifying assumption is made that the Gaussian basis functions are located on a regular grid and have a unit variance.

Conclusion

The splatting method makes the advantages of EWA texture filtering available for point representations of the surface and rendering techniques. The mathematical analysis of the process of constructing a solid textured image from irregular points is provided. An optimized method for selecting images or procedural textures on point objects is described. Modification of the frame buffer data and a simple merge strategy provide transparency and smooth edges with a minimum of visual artifacts.

Splatting is applied to procedurally generated objects, for example, parametric surfaces and fractal relief.

Implemented the rendering pipeline based on the method of splatting. In addition, we can convert geometric models into point objects in the pre-processing stage. Models have a hierarchical data structure, facilitating multi-resolution and progressive rendering. EWA filtering is used to select image textures on a point

feature.

Figure 2 (pictures a, b) shows a point geometric object. In figure 2 (picture c) terrain-translucent surface.

.

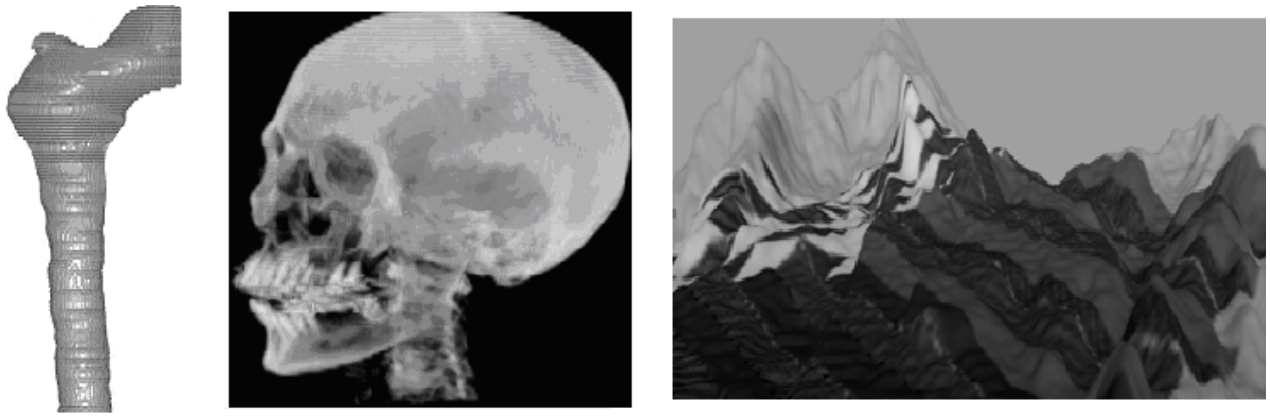

Figure. 2. Examples of 3D images.

Bibliography

1. M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, D. Fulk. The Digital Michelangelo Project: 3D Scanning of Large Statues. In Computer Graphics, SIGGRAPH'2000 Proceedings, Los Angeles, CA, July 2000. P. 131-144.

2. H. Hoppe, T. DeRose, T. Duchampt, J. McDonald, W. Stuetzle. Surface Reconstruction from Unorganized Points. In Computer Graphics, SIGGRAPH'92 Proceedings, Chicago, IL, July 1992. P. 71-78.

3. B. Curless, M. Levoy. A Volumetric Method for Building Complex Models from Range Images. In Computer Graphics, SIGGRAPH'96 Proceedings, New Orleans, LA, August 1996. P. 303-312.

4. J. P. Grossman, W. Dally. Point Sample Rendering. In Rendering Techniques'98, Springer, Wien, Vienna, Austria, July 1998. P. 181-192.

5. W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan. Image-Based Visual Hulls. In Computer Graphics, SIGGRAPH'2000 Proceedings, Los Angeles, CA, July 2000. P 369-374.

6. S. Rusinkiewicz and M. Levoy. QSplat: A Multiresolution Point Rendering System for Large Meshes. In Computer Graphics, SIGGRAPH 2000 Proceedings, Los Angeles, CA, July 2000. P. 343-352.

7. P. Heckbert. Survey of Texture Mapping. IEEE Computer Graphics & Applications, 6(11):56–67, November 1986.

8. H. Pfister, M. Zwicker, J. van Baar, M Gross. Surfels: Surface Elements as Rendering Primitives. In Computer Graphics, SIGGRAPH 2000 Proceedings, Los Angeles, CA, July 2000. P. 335-342.

9. N. Greene, P. Heckbert. Creating Raster Omnimax Images from Multiple Perspective Views Using the Elliptical Weighted Average Filter. IEEE Computer Graphics & Applications, 6(6):21–27, June 1986.

10. P. Heckbert. Fundamentals of Texture Mapping and Image Warping.

Master's thesis, University of California at Berkeley, Department of Electrical Engineering and Computer Science, June 17 1989.

11. N. Jouppi, C. Chang. Z3: An Economical Hardware Technique for High-Quality Antialiasing and Transparency. In Proceedings of the Eurographics/ SIGGRAPH'99 Workshop on Graphics Hardware, Los Angeles, CA, August 1999. P. 85-93.

12. S. Winner, M. Kelley, B. Pease, B. Rivard, and A. Yen. Hardware Accelerated Rendering of Antialiasing Using a Modified A-Buffer Algorithm, August 1997. P. 307-316.

13. L. Carpenter. The A-buffer, an Antialiased Hidden Surface Method. In Computer Graphics, volume 18 of SIGGRAPH'84 Proceedings, July 1984. P. 103-108.