MULTISCALE SURFACE REPRESENTATION AND RENDERING FOR POINT CLOUDS

Sung-Bum Park and Sang-Uk Lee *

School of Electrical Engineering
Seoul National University
Seoul, Korea

Hyeokho Choi†

Department of Electrical and Computer Engineering
Rice University
Houston, Texas, USA

ABSTRACT

We introduce a new multiscale geometry representation based on tree-structured piecewise plane approximation of 3-D surface geometry. Dyadic division and plane approximation of points within each dyadic cube result in a tree-structured multiscale representation of the 3-D surface described by point cloud data. We then perform a complexity-regularized tree pruning to obtain a compact representation of the surface geometry. Based on its adaptivity and multiscale structure, the proposed representation scheme provides a desirable framework for efficient geometry modelling, fast processing, and geometry coding. We apply the proposed geometry models to point cloud rendering to demonstrate supremacy of our efficient geometry representation.

1. INTRODUCTION

Representation and processing of three dimensional (3-D) surface information are important problems in such areas as computer graphics, medical imaging, virtual reality, and 3-D radar/ladar signal processing. With the recent development of 3-D data collection tools, the acquisition of 3-D surface data became practical, and their efficient representation and processing became essential.

Among many different methods used to collect 3-D surface information, widely available laser range scanners allow us to record the coordinates of points on the surface of the scanned object. To represent a complex surface geometry, often millions of points are required for a moderate-sized object, and the prohibitively large data size asks for proper reorganization of information for efficient processing. A typical way of extracting and modelling surface geometry buried in a point data set is to build a triangular mesh by connecting the points into concatenated triangles. Although the regular structure of the mesh provides a good surface representation for such processings as rendering [1] and coding [2, 3], oversampling and inherent scanning noise in the raw data set make the mesh building extremely difficult, often requiring very expensive preprocessing of the original point data.

As an alternative to mesh building, there has been recent work on the direct processing of the point data for applications in computer graphics [4–7]. These point processing algorithms have mostly focused on the organization of data for the fast rendering of point cloud data. Tree based representations such as K-D tree hierarchy [4] or octree hierarchy [5, 7] are widely used in structuring the unorganized points. When these points are rendered, the rendering primitives such as sphere [4] or tangential disk [5,6] enables fast rendering. Because these direct point-based algorithms require very little preprocessing of the raw point data, they are easy to implement and also better represent diverse 3-D objects that are not adequate for mesh representation. In addition, point-based representations do not require connectivity information between points, resulting in much reduced data size and potentially faster processing.

However, to take full advantage of point-based processings, we should overcome several shortcomings of current point-based approaches. First, a huge number of points obtained by scanning an object and their highly irregular distribution are often prohibitive for efficient processing. In addition, the distribution of points in the raw data is often irrelevant to the local smoothness of the described surface, making the data highly redundant. Furthermore, often large measurement noise corrupts the scanned point coordinates, making the extraction of underlying surface geometry very difficult. However, most of the current algorithms developed in the computer graphics community focused only on fast and realistic rendering of large size of point data without much consideration of the underlying geometry information [4–6], and they are not adequate for such applications as geometry coding and 3-D computer vision that require explicit geometry information of the underlying object.

In this paper, we propose a new multiscale geometry extraction and representation algorithm that directly encodes local surface geometry as a framework for general geometry modelling from point cloud data. Inspired by the recently proposed works on multiscale edge and line modelling algorithm in images [8, 9], we successively split the enclosing cube of the given point data set into octants to build an octree with each node representing subset of points in the corresponding dyadic cube at different scales. Since a set of points describing a small local region of a surface can be accurately approximated by a plane, we record the best plane approximation of the points in each dyadic cube. Because the accuracy of this local plane approximation depends on the local smoothness of the underlying surface, we adaptively control the scale of representation in different regions. We can obtain the optimal balance between the approximation error and the complexity of the representation through a tree pruning based on the CART algorithm [10].

The proposed geometry representation framework overcomes the typical shortcomings of point-based algorithms. First, the multiscale tree-structured framework allows direct extraction and explicit representation of surface geometry. In addition, the CART-based tree pruning step [10] enables local adaptation of geometry representation to the local surface smoothness, significantly reducing the redundancy of the raw data set. Furthermore, the local least

*This work was performed when S. Park was visiting Rice University during summer 2003. S. Park and S. Lee were supported by Brain Korea 21 project of Korean government.

†Email: choi@rice.edu. Web: dsp.rice.edu. H. Choi was supported by ONR grant N00014–02–1–0353 and DARPA/AFOSR grants F49620–01–1–0513 and F49620–01–1–0378.

(a) (b) (c) (d)

Fig. 1. Result of geometry extraction and modelling from 'Happy Budda' test data. Each figure shows different balance between model accuracy and data size: model with (b) $252K$ parameters ($63K$ planes), (c) $62K$ parameters ($15.5K$ planes), and (d) $27K$ parameters ($6.6K$ planes). For comparison, the "original" surface rendered based on the original data set ($1,629K$ parameters) was shown in (a). Notice that the original geometry is accurately modelled in (b) with the number of model parameters less than 20% of original.

squares plane fit at multiple scales and the subsequent tree pruning effectively control the noise in the data by accurately extracting the underlying surface geometry. Comparing the results with different pruning levels in Fig. 1, it can be noticed that we can obtain accurate geometry representation both in smooth and detailed regions thanks to the adaptive local accuracy control through the optimal tree pruning even with huge reduction in data size. In Fig. 1 (b), the model effectively represents the original surface geometry very accurately with the size of less than 20% of the original.

2. MULTISCALE GEOMETRY REPRESENTATION FROM POINT CLOUD

To obtain a multiscale approximation of surface geometry, we first hierarchically partition the 3-D domain. Because the accuracy of plane approximation of the surface depends on the local surface smoothness and scale of approximation, this hierarchically nested domain partitioning provides a convenient framework for locally adapting the model accuracy to the local surface properties. As a bonus, this multiscale model allows us to obtain different balance between model complexity and accuracy.

2.1. Dyadic partitioning of domain

Inspired by the recent work in the image processing for edge and curve representation [8–10], we adopt dyadic partitioning and local geometry approximation in each dyadic region for effective multiscale modelling of 3-D geometry. Similar to the 2-D tree-based multiscale algorithms, we can obtain compact geometry representation that can be manipulated using fast algorithms on the tree.

We start by defining an enclosing cube that contains all the points in the raw point cloud data. Then, we dyadically divide the cube into 8 smaller cubes. After each split, a cube may or may not contain points. For those cubes containing points, we repeat dyadic partitioning until each cube contains either 0 or 1 point. The top figure in Fig. 2 illustrates this partitioning process for points on a 2-D domain, and the bottom figure shows the corresponding tree structure of the resulting dyadic cubes. Each node in the octree represents a dyadic cube at the corresponding scale. After finishing, the leaf nodes correspond to dyadic cubes having either 0 or 1 point in each of them.

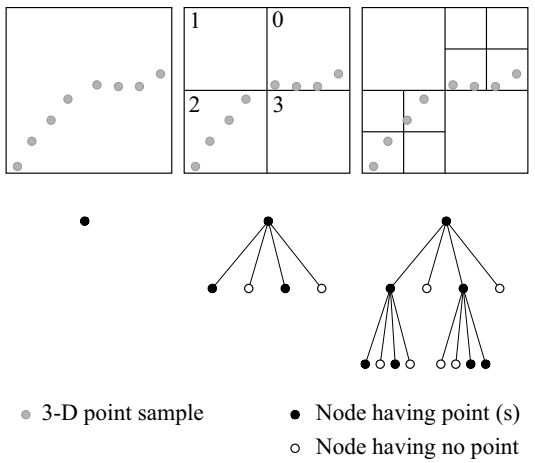

Fig. 2. Dyadic partitioning into cubes and the corresponding tree structure. As an illustration, we show the process for points on a 2-D domain and the resulting quadtree. For 3-D point data, we obtain dyadic cubes in different scales and an octree.

Similar tree organization of point data was adopted in many point-based rendering algorithms [4, 5, 7] to take advantage of the hierarchical processing on the tree for fast rendering of large point data set. In our model, we utilize the obtained hierarchical subsets of points in each dyadic cube to effectively approximate the surface geometry in a multiscale fashion. The tree structure also

leads to fast algorithms for adapting the approximation to different levels of surface smoothness and details in different regions.

2.2. Piecewise plane approximation

Similar to the piecewise linear approximation of 1-D edge contours in images using wedgelets and beamlets [9,10], we can model the 3-D surface described by given point cloud using piecewise plane approximation. If the underlying surface is smooth relative to the size of a dyadic cube, all the points in each dyadic cube are well fitted by a plane. The best fitting plane summarizes all the points within each dyadic cube, approximating local surface geometry described by the points.

We perform the local plane fitting in each dyadic cube, obtaining a piecewise plane approximation of the underlying surface at multiple scales. While error measures such as the Hausdorff distance are more appropriate to quantify the approximation errors in surface approximation, we use simple squared error metric since it has the additivity property that allows fast processing on the tree through efficient algorithms such as the CART-based tree pruning [10]. Later we will experimentally verify that this processing based on the squared error also leads to a reasonable performance behavior in terms of the Hausdorff error measure.

When we have M points located at (x_i, y_i, z_i), $i = 1, \ldots, M$ with respect to a global coordinate system, the plane best fitting the points in the least squares sense is given by solving the following constrained minimization of the squared error as

$$(n_x, n_y, n_z, d) = \arg\min \sum_{i=0}^{M-1} (n_x x_i + n_y y_i + n_z z_i + d)^2, \quad (1)$$

subject to ${n_x}^2 + {n_y}^2 + {n_z}^2 = 1$, where (n_x, n_y, n_z) is the unit normal vector of the plane and d is the orthogonal distance from the origin. We then record the plane parameters at each node of the tree. However, when M is small, such plane is either not uniquely defined or very sensitive to noise. In such cases, we define the planes by simply inheriting the estimated normal from the parent cube. Then, the location of the plane is found by solving the least squares problem in (1) for d with fixed normal vector.

Once we compute all the plane parameters for all dyadic cubes, we have a piecewise plane approximation of the surface at multiple scales. For example, if we truncate the tree at a particular scale and consider the leaf nodes of the truncated tree, we have the surface approximation at that scale. More interestingly, we can prune the tree to control the scale of representation in different regions of the surface. This is particularly useful when the underlying surface has different smoothness so that the accuracy of plane approximation varies widely in regions of different smoothness.

Although the pruning reduces the amount of data we need to retain, it increases the approximation error. It is our interest to prune the tree to reduce the amount of data – proportional to the number of leaf nodes – with minimum increase in the approximation error. Thanks to the additivity of the squared error metric defined in (1), the balance between the tree complexity and the approximation can be achieved through a fast algorithm based on CART [10].

2.3. Complexity-regularized tree pruning and approximation

Smooth regions of a surface can be well approximated by a small number of large planes, while we need many small planes to approximate complicated region with lots of details. In our octree structure of dyadic cubes with best plane fit in each cube, tree pruning method controls the necessary number of planes in different regions to optimize the complexity of the model for the allowed distortion level.

CART-based tree pruning is a simple and fast method to yield the optimally pruned tree having the right balance between the model complexity and the approximation error [10]. Although many different definitions are possible for the model complexity, in its simplest form, we define the complexity R as the number of the planes in our model or equivalently the number of the octree leaf nodes. We can directly obtain the approximation distortion D by summing all the errors of the least squares plane fit in (1) for all the leaf nodes of the tree.

Since the approximation error D is a monotonically decreasing function of the model complexity R, it is our interest to hit a proper balance between D and R to obtain a model with desired complexity that has the smallest possible distortion. In other words, among all pruned trees having R leaf nodes, we want to pick the one having the smallest D.

This problem of the constrained minimization of D can be solved by minimizing the Lagrangian cost C defined as a linear combination of R and D as $C = R + \lambda D$, where λ sets the balance between R and D. For each subtree, we make the decision whether to prune it or keep it by comparing the resulting overall cost C in each case. Before pruning, the contribution of a subtree $\mathcal{T}$ to the cost C is given as

$$C_1 = N_{\mathcal{T}} + \lambda \sum_{i \in \mathcal{L}(\mathcal{T})} D_i, \quad (2)$$

where $N_{\mathcal{T}}$ is the number of leaf nodes in the subtree $\mathcal{T}$ and $\mathcal{L}(\mathcal{T})$ is the set of all the leaf nodes of $\mathcal{T}$ with D_i representing the error in the dyadic cube corresponding to node i. On the other hand, when the subtree is pruned, the contribution to the cost changes to

$$C_2 = 1 + \lambda D_r, \quad (3)$$

where D_r is the error at the root node of $\mathcal{T}$. The number of planes reduces to 1 because we prune all the nodes of $\mathcal{T}$ except for its root. If $C_1 > C_2$, we prune the corresponding subtree to reduce the overall cost C, and otherwise, we keep the subtree $\mathcal{T}$. In the actual implementation of tree pruning, we can simply go up the tree starting from the bottom making decisions to prune or keep each node. After all the nodes are traversed, we obtain the optimal pruned tree.

Although the optimization based on the squared error measure leads to fast and efficient processing, it is often more desirable to quantify the approximation error in terms of the Hausdorff error metric. However, instead of trying to incorporate it in our algorithm, we merely hope that the resulting optimization based on the squared error metric is also meaningful in terms of the Hausdorff measure. To check the behavior of our model in terms of the Hausdorff distance between surfaces, we calculated the Hausdorff distance for models having different complexity using the method in [7]. Fig. 3 shows the changes in the Hausdorff error as we vary the parameter λ in the tree pruning. From the plot, we can see that the proposed method provides optimization results also meaningful in the Hausdorff sense.

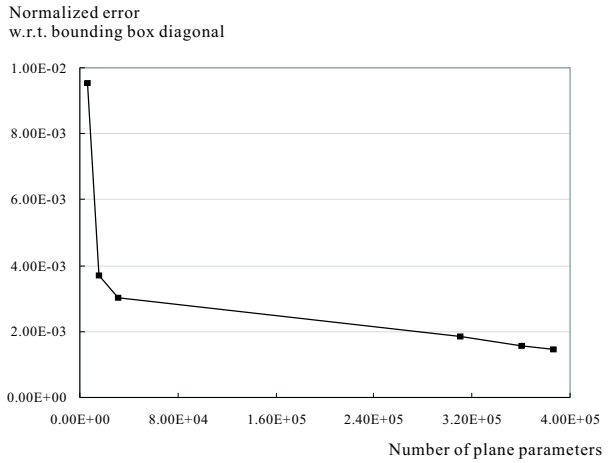

Fig. 3. Changes in the Hausdorff error as we vary the tree pruning. The original Hausdorff distance error is normalized by the diagonal distance of the bounding box.

3. APPLICATION: POINT CLOUD SURFACE RENDERING

Now that we have an efficient multiscale geometry representation for point cloud data, we can easily apply the model in various applications requiring geometry modelling. As a simple application to illustrate the effectiveness of the proposed model, we build multiscale geometry models for a test data set and visualize the accuracy of the described geometry by rendering the modelled surfaces.

3.1. Rendering method

Given the explicit geometry model as a piecewise plane approximation, rendering of the geometry is straightforward. Although many different methods including explicit mesh building can be used for rendering, we adopt a very simple, although very inefficient, method.

From the piecewise plane model, we simply create a new, very dense point cloud by generating points on the plane in each dyadic cube. To easily control the distances between these generated points, we place new points at all the locations where a global, dense discrete grid crosses the plane. Then, the normal vector of each point is set to be the unit normal of the corresponding plane.

While rendering the adjacent planes, discontinuities occasionally occur in the boundary regions of the planes. To remove these "holes," we use a simple way to fill them by allowing volumetric margin or thickness to the plane. The thick plane covers the discontinuities at the boundaries of each dyadic cube and the visual holes are efficiently removed in the rendered result. In our rendering examples, the thickness of each plane is adaptively increased until there are no holes on the boundary region of adjacent blocks.

3.2. Rendering example and analysis

We tested the proposed algorithm on the 'Happy Budda' point cloud data having $543,618$ points. 'Happy Budda' contains both highly detailed region such as silhouettes on the cloth and smooth regions such as the midsection, requiring adaptive control of approximation accuracy in different regions.

Fig. 1 shows the rendering results of the modelled surfaces at different accuracy levels with different values of λ. In Fig. 1(b), we note that the original geometry is very accurately modelled with the number of model parameters less than 20% of the original data set. The adaptivity of the approximation for different surface smoothness is evidently visible when we further reduce the size of the model. For example, when the model parameter size is only 1.6% of the original data in Fig. 1(d), we see that the resulting geometry model effectively adapts to the local smoothness of the surface so that the model uses many small planes to approximate the detailed surface geometry like the silhouette of the cloth, while smooth regions like the midsection are expressed with small numbers of large-sized planes.

4. CONCLUSIONS AND FUTURE RESEARCH

In this work, we introduced a new multiscale geometry representation scheme for unorganized point cloud data. By partitioning the 3-D domain into dyadic cubes and then approximating the local geometry in each cube by a plane, we obtained a multiscale geometry model. The optimal tree pruning effectively controlled the size of dyadic cubes in different regions to adapt the approximation accuracy to the local surface geometry. Surface rendering experiments demonstrated the effectiveness of the proposed modelling framework for compact 3-D geometry modelling and representation.

Based on the proposed modelling framework, we plan to develop an efficient surface coding algorithm through careful encoding of plane parameters. In addition, applying the proposed model to various applications in such areas as computer graphics and computer vision remains as future research.

5. REFERENCES

[1] D. Zorin, P. Schröder, and W. Sweldens, "Interpolating subdivision for meshes with arbitrary topology," *Proc. ACM SIGGRAPH '96*, pp. 189–192, Aug. 1996.

[2] A. Khodakovsky and I. Guskov, "Normal mesh compression," *Geometric modeling for scientific visualization*, Springer, 2002.

[3] S. Lavu, H. Choi, and R. G. Baraniuk, "Geometry compression of normal meshes using rate-distortion algorithms," *Proc. Eurographics/ACM SIGGRAPH symposium on geometry processing 2003*, pp. 52–61, June 2003.

[4] S. Rusinkiewicz and M. Levoy, "QSplat: A multiresolution point rendering system for large meshes," *Proc. ACM SIGGPRAH 2000*, pp. 343–352, July 2000.

[5] H. Pfister, M. Zwicker, J. van Baar, and M. Gross, "Surfels: Surface elements as rendering primitives," *Proc. ACM SIGGRAPH 2000*, pp. 335–342, July 2000.

[6] M. Zwicker, H. Pfister, J. van Baar, and M. Gross, "Surface splatting," *Proc. ACM SIGGRAPH 2001*, pp. 371–378, July 2001.

[7] A. Kalaiah and A. Varshney, "Statistical point geometry," *Proc. Eurographics/ACM SIGGRAPH symposium on geometry processing 2003*, pp. 107–378, June 2003.

[8] M. B. Wakin, J. K. Romberg, H. Choi, and R. G. Baraniuk, "Geometric methods for wavelet-based image compression," *Proc. SPIE Technical conference wavelets X*, Aug. 2003.

[9] D. L. Donoho and X. Huo, "Beamlets and multiscale image analysis," *Multiscale and multiresolution methods, Lecture notes in computational science and engineering*, vol. 20, Springer, 2001.

[10] D. L. Donoho, "Wedgelets: Nearly-minimax estimation of edges," *Annals of Stat.*, vol. 27, pp. 859–897, 1999.