A NEW METHOD OF STORAGE AND VISUALIZATION FOR MASSIVE POINT CLOUD DATASET

Zhiqiang Du*, Qiaoxiong Li

State Key Laboratory of Information Engineering in Surveying Mapping and Remote Sensing, Wuhan University, 129 Luo Yu Road, Wuhan, Hubei, 430072, P. R. China-
duzhiqiang@lmars.whu.edu.cn, liqiaoxiong1@163.com

KEY WORDS: Massive point cloud, Multi-band image, LOD, Visualization, GPU

ABSTRACT:
Three dimensional data captured by laser scanning is widely used in documentation and proves invaluable to cultural heritage. Detailed and high-precision scanning results in huge collections of scanned points and the size of data file of a single object often exceeds the available computer memory. It is difficult to render the massive point cloud data in real time on a commodity PC because the data amount is much larger than the main memory. In this paper, based on the characteristics of massive point cloud data, a new method is presented to solve this problem. Firstly, this method uses multi-band images to store point cloud data and it can automatically adjust the output size of the point cloud image according to the available computer memory. Simultaneously, multi-dimensional information of point cloud dataset, including position, intensity and color, will be stored into bands of images separately. Secondly, through multi-level image down-sampling, this method creates image pyramid for the point cloud data. This image pyramid represents different levels of detail for the model. During the real-time visualization phase, it selects different LOD models to render according to observation location. Thirdly, this method takes full advantage of the GPU's rendering ability. It binds all the rendering data to the graphic memory, which greatly accelerates the rendering speed. This method has no restriction on the computer's memory capacity to process and store massive point cloud data. It can quickly process point cloud dataset with huge size and visualize the dataset with LOD models in real time.

1. INTRODUCTION

Point cloud is 3dimentional positions, possibly associated with additional information, such as colors and normal, and can be considered a sampling of a continuous surface. This representation is extremely simple and flexible. Moreover, it offers the additional advantage of avoiding connectivity information and topological consistency. The fast growing popularity of laser scanners makes capturing of 3D information simple, efficient and direct.

Detailed and high-precision scanning point cloud is widely used in digital documentation and other areas. However, the number of points in the generated point cloud is in the order of million points. Some scanning even generates billions of points and the resolution can be as high as 1/4 mm (Levoy, 2000). Interactively displaying the scanned point cloud data is necessary for management of the datasets.

Interactively displaying and visualizing large amounts of data has been a challenge in computer graphics since its inception. In many cases, the amount of data that a user wants to visualize exceeds available processing power and memory capacity. Digital computers have natural limits dictated by physics, mathematics and cost considerations. Interactive performance, which forces the computation of new frames at 10 Hz or faster, exacerbates the problem (Kasik, 2008). To interactively view the massive point cloud dataset, a powerful workstation is needed. For commodity PC, the data amount may exceed the available memory and it is impossible to load the point cloud dataset into memory.

In this paper, we propose a new method to visualize massive point cloud dataset directly on commodity PC. We store the massive point cloud dataset in multi-band images and create levels of detail (LOD) for the point cloud dataset. The atomic nature of a point sample gives the representation a built-in LOD. Building levels of detail for the point cloud dataset has two main advantages (Luebke, 2002). First, for the massive point cloud dataset, less detailed data can be loaded into memory and visualized on commodity PC. Second, the massive point cloud dataset is often too dense when viewed from a distance. In this case, less detailed data make the rendering image suffer little loss in detail and can be rendered faster. To visualize the massive point cloud dataset, we propose to switch between the different LOD models of the point cloud dataset.

Our contribution is finding a new method to store the massive point cloud dataset. And based on this storage method, we develop a visualization method that can view the massive point cloud dataset on commodity PC.

The rest of the paper is organized as follows. In section 2, the previous work is reviewed. In section 3, we describe the storage process, and in section 4, the visualization phase is presented. In section 5, an application is introduced, which includes the storage and visualization phase, we can examine the efficiency and effect of this method by the result. In section 6, we conclude our work and present some future work.

* Corresponding author: Zhiqiang Du. (duzhiqiang@lmars.whu.edu.cn)

2. REVIEWS

There has been quite a lot technique, including visibility computation, simplification, levels-of-detail, and cache-coherent data management, in the massive data visualization area (Silva 2002). All of these techniques aim to reduce the data amount required to render, and many integrate with the out-of-core techniques. Out-of-core refers to algorithms which process data that is too large to fit into a computer's main memory at one time. Such algorithms must be optimized to efficiently fetch and access data stored in slow bulk memory such as hard drive or tape drives. Different realization strategies of out-of-core technique include Divide-and-conquer, cache-efficient, external memory and streaming processing (Cox, 2007; Pajarola, 2005).

Researchers have studied the problem of rendering massive dataset at interactive frame rates for many years. Researchers (Correa, 2004) use octree to partition the whole dataset. During preprocess, it use an out-of-core preprocessing algorithm to build an on-disk hierarchical representation for the model. At run time, it uses an out-of-core rendering approach that employs multiple threads to overlap rendering, visibility computation, and disk operations. This method provides accurate rendering images but rely on complex realization.

However, these techniques mainly focus on triangle or polygon models. The conceptually most significant difference between points and triangles is that points--much as voxels or pixels--carry all attributes needed for processing and rendering. There is no distinction between vertex and fragment anymore. Using points as rendering primitives is a topic of ongoing research. However, almost all publications in this area deal with the rendering of geometric surfaces, as can be seen in the other articles of this CG&A issue. The aim of the point-based rendering (Gross, 2007; Levoy, 2000) is to create smooth surfaces without holes based on the point-based models. Our interest is to render the scattered point cloud dataset directly.

Researchers (Hopf, 2004) propose to accelerate the visualization of scattered point data by a hierarchical data structure based on a principal component analysis (PCA) clustering procedure. By traversing this structure for each frame they can trade-off rendering speed vs. image quality, and lower hierarchy levels can be used during interaction. This approach uses complex data structure and need lots of preprocess operations. To render the scattered point cloud data directly, we need a simpler and faster approach.

3. STORAGE

3.1 Why Use Multi-band Image

Point cloud generated from laser scanning has multi-dimensional information. Basically, it has three dimension coordinates. Additionally, it may include light intensity , colour and normal information. Some researchers (Gu, 2002) have already proposed to use images to store geometry. The multi-dimensional characteristic of the point cloud require a flexible and extensible structure to store the data. And multi-band image becomes our option for storing point cloud data. Multi-band image is widely used in remote sensing to store the scanned information. It can include as many bands as required. For point cloud, it has two main advantages to store it in multi-band image.

Firstly, the extensible bands can hold all the information the point cloud dataset contains. Coordinates, light intensity, colour, normal or other information, each band corresponds to one dimension of the information. For a point cloud dataset that has only x y z information, a three-band image is needed to store the data, and for a point cloud dataset that has three-dimensional coordinates and light intensity, a four-band image can be used.

Secondly, storing the point cloud in images can introduce image processing methods into point cloud processing. Image is a two-dimensional vector, and pixel is its basic unit. By storing point cloud in images, one scanned point can be considered as one pixel in the image. Image processing is a mature field that has been studied for many years and there are many existing effective algorithms that can be applied on images. By introducing the image processing methods into point cloud processing area, point cloud processing problems can be solved by image processing methods. As an example, in section 3.2, we introduce how the levels of detail of the point cloud dataset are generated by using image down-sampling method.

3.2 Storage Process

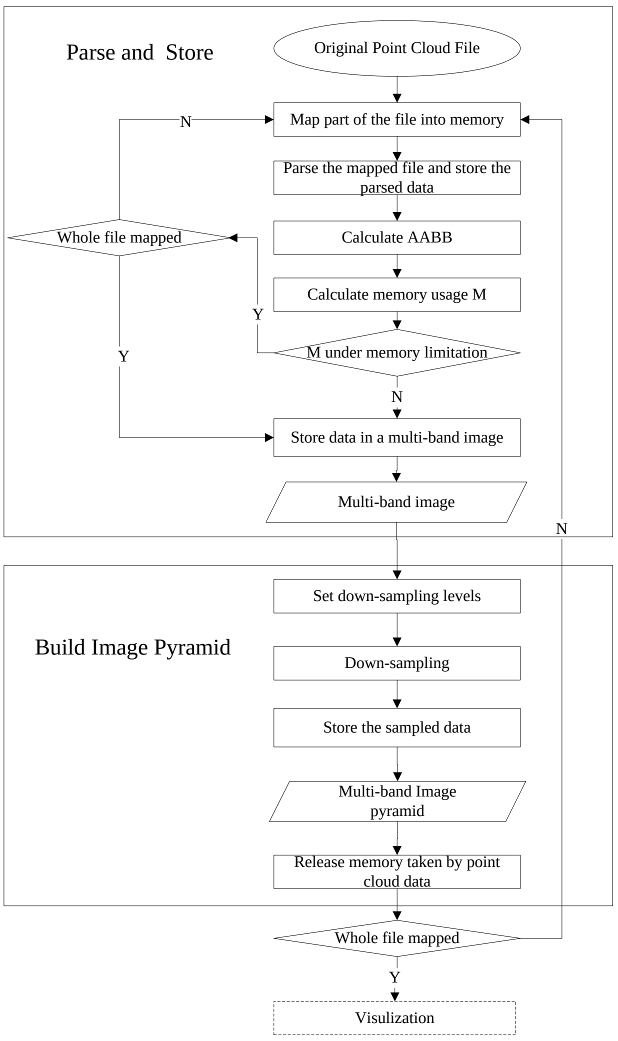


Figure 1. Storage procedure

This section describes the whole process of storing the point cloud dataset in multi-band images and build image pyramid for the all the images. The process is shown in Figure 1.

The aim of storage phase is to parse the original point cloud file and store it in multi-band images. The original point cloud file generated by laser scanning is often larger than the available memory. For files that larger than the memory, it is impossible to load the whole point cloud file into memory at one time. So we use memory file mapping. Every time only part of the file is mapped to memory. The next step after file mapping is to parse the mapped part of the file and calculate the Axis-Aligned Bounding Box (AABB) of the mapped point cloud data. Parsing is the process of getting useful information from the original point cloud file. For a point cloud file that has three-dimensional coordinates and light intensity, according to the requirement, parsing can only get the coordinates information or the coordinates with the light intensity.

AABB is the calculated along with parsing the file. In the visualization phase, AABB is demanded to set the initial viewpoint. When the mapped part of the file is parsed, we have to judge if the memory usage is under the memory limitation. For the parsed data is stored in memory and with the process going, the stored data can exceed the available memory. In case of under limitation, the next part of the file can be mapped and do the same parsing and calculation process. In the other case, the stored data has to be written to a multi-band image. We generate a multi-band image that has the same band count as the information dimension of the stored point cloud data and write each dimension into its correspondent band. The AABB information has to be stored with the point cloud data. In our experiment, we choose GeoTiff as the multi-band image. GeoTiff is widely used and can meet our demand well. We store the point cloud data in its bands and the AABB information in its header.

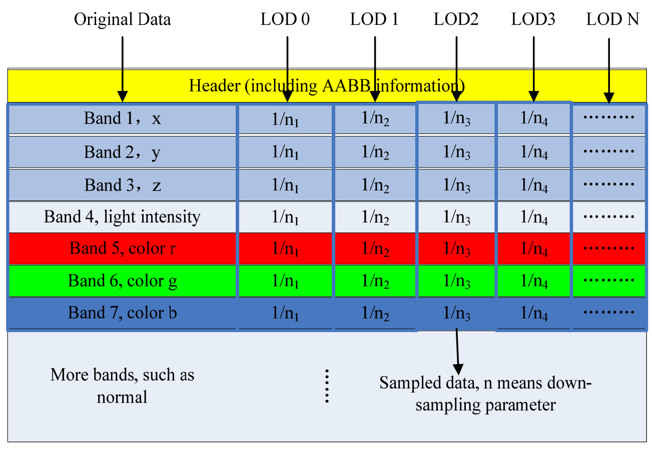


Figure 2. Structure of the stored multi-band image

By building image pyramid for the image file, we create levels of detail (LOD) for the point cloud dataset. Many down-sampling methods can be applied in image processing, including nearest neighbour method, average method, Gaussian method etc. To keep the down-sampling result consistent with the original data, we use the nearest neighbour method. Nearest neighbour down-sampling is fast and simple, which just chooses one pixel from part of the image, and the result is part of the original data. From the point cloud view, down-sampling chooses some points to represent the dataset, and decreases the detail of the original data. To build image pyramid, we set some down-sampling levels, and stored the sampled result in the same file. In this way, each level of the image pyramid corresponds to one level of detail of the point cloud dataset. Figure 2 shows the structure of the multi-band images. Figure 3 shows the image that stores the point cloud data, different colours indicate different point information in the point cloud dataset.

The file mapping and storing is iterated until the whole original point cloud file is processed. The result of the whole process is one or more multi-band images with its image pyramid.

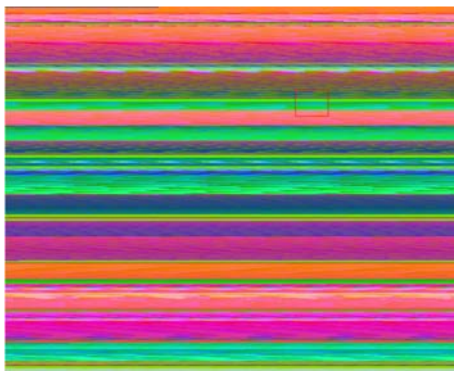


Figure 3. Image that store the point cloud data, viewed with ENVI

The storage phase is direct and simple. Neither time-consuming calculation nor complex structure is needed, all we have to do is to retrieve information from the original point cloud file and store it in multi-band images. By building image pyramid for each image, we generate different LODs for the point cloud dataset. After this phase, the original point cloud dataset is divided and stored in some multi-band images, and the LODs are generated and the bounding boxes are known, it is time to view the point cloud dataset.

4. VISUALIZATION

Point is one of the basic primitive in computer graphic. For point cloud, using point as basic rendering primitive, the rendering process can be simple and fast. The graphic pipeline only need to project the discrete point cloud set to the screen and fill colour in the projected pixels. There is no connectivity information in the point cloud dataset and discontinuous point sampled surface is displayed.

Our visualization phase can be divided into two steps. In the first step, we do all the initialization work, including calculating the initial viewpoint and load the appropriate level of point cloud data. In the second step, with the change of the viewpoint location, it discards the previous data and load new level of the point cloud data. Figure 4 shows the whole visualization phase.

4.1 Initialization Phase

To set the initial viewpoint, the AABB of the whole point cloud dataset has to be calculated. Fortunately, we store AABB of the point cloud data that the image contained in image header. We only need to go through every image to get AABB information contained in each image and calculate the total AABB.

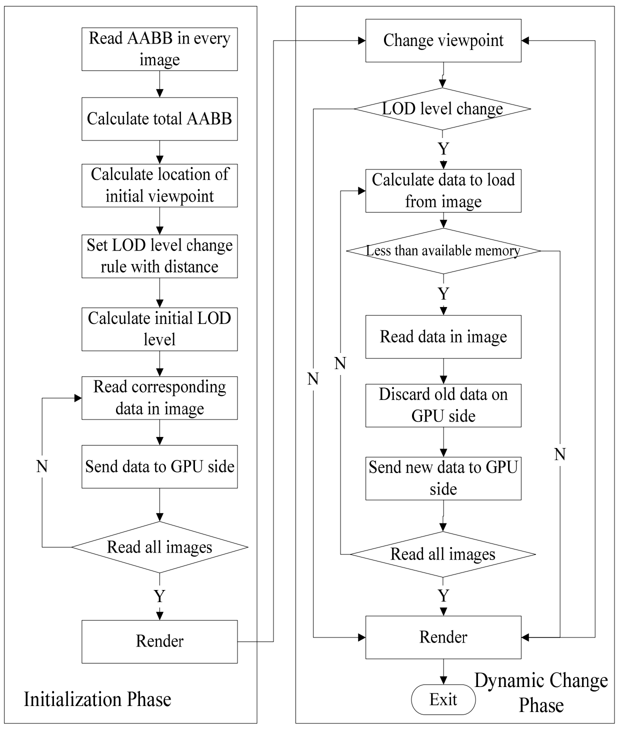

Read AABB in every image
Calculate total AABB
Calculate location of initial viewpoint
Set LOD level change rule with distance
Calculate initial LOD level
Read corresponding data in image
Send data to GPU side
Read all images
N
Y
Render
Initialization Phase
Change viewpoint
LOD level change
Y
Calculate data to load from image
Less than available memory
Y
Read data in image
Discard old data on GPU side
Send new data to GPU side
Read all images
Y
N
N
N
Render
Exit
Dynamic Change Phase

Figure 4. Visualization procedure

With the total AABB of the point cloud dataset, we can decide the viewpoint from where the whole space the point cloud dataset covers can be viewed. After this, the appropriate level of point cloud data is to be load to display. We have to set rules for loading different levels of detail of the point cloud. Distance is a simple but effective rule. Based on the distance between viewpoint and the centre of the AABB, we can decide which level of the data should be loaded. When the distance between them is less than the threshold, more detailed data has to be loaded and when the distance gets larger and greater than the threshold, sparser point cloud data is to be loaded. From the view of stored images, the loading process is to go through each image and load the correspondent level of data from the image pyramid.

To render the point cloud file fast and effectively, it is better to put the rendering data on the GPU side. For in that case, GPU has faster access to the data and it doesn't need to transport the data from memory to GPU side every frame. Modern graphic card support this feature, the data stored in the GPU side is called vertex buffer (Wright, 2007).

When the initialization phase finished, the appropriate level of the point cloud data is loaded into the GPU side and we can see the rendered image of the point cloud.

4.2 Dynamic Change Phase

The displaying level of detail of the point cloud dataset can be changed when the viewpoint location changes. Because we build for the original point cloud static and discrete LODs, in case of the detail level change, old data is out-of-date and should be discarded and new data has to be loaded.

According to the distance rule, when the viewpoint get closer to the centre of the AABB, the more detailed point cloud data is displayed, this process continues until the most detailed data which means the original point cloud data is loaded and displayed. And it loads the less detailed data when the viewpoint get further from the centre of the AABB until the sparsest point cloud data is loaded.

In our visualization, Culling is difficult given only the AABBs. The AABB of Point cloud stored in each image can be the same, for the point cloud in the original file is unordered. It has to load all the same level of the point cloud data stored in every multi-band image. From the view of stored images, the loading process is to go through each image and load the correspondent level of data from the image pyramid. Before loading the data stored in image into memory, it has to judge if the available memory is enough to hold the data in that image, for the point cloud data can be very large and take up a lot of memory. In case of exceeding the available memory, it has to give up loading, otherwise, the point cloud data stored in that image is loaded into memory and transport to the GPU side. When the point cloud data amount exceeds the graphic card memory, it will store the data sent to GPU on main memory, but it still managed by GPU, the penalty is that the access time would increase and rendering speed decreases.

The dynamic change phase switches the rendering data according to the distance between viewpoint and the centre of AABB. We can also consider it as switching between different static LODs of the point cloud dataset.

5. EXPERIMENTS AND ANALYSIS

This process applied in project--Digital Mogao Caves. The original point cloud file is 1.74G, which is stored in text format and contains 47,756,566 points. The point cloud includes three-dimensional coordinates and one-dimensional light intensity. The processing has been evaluated on a PC running Windows XP, with Intel Pentium D CPU, 1GB DDR memory, and a NVDIA GeForce 7300 GT graphic card.

5.1 Storage

We make comparison by storing only three-dimensional coordinates and all of the four dimension information. We set the down-sampling levels to 1/4, 1/16, 1/96 and 1/196. The result is shown in Table 1.

By storing the original text point cloud file into multi-band images, the file amount reduced. The amount of the sampled data is only about 1/3 of the original data and the data amount can be predicted by the sampling parameter. All of the processing finished in few minutes. I/O takes most of the storage time, because there is no complex calculation in the storage phase.

Stored information	3D coordinates	3D coordinates and light intensity
Bands	3	4
Original data amount (MB)	561	760
Sampled data amount(MB)	194	240
Total data amount(MB)	755	1000
Time(s)	211	337

Table 1. Storage result

5.2 Visualization

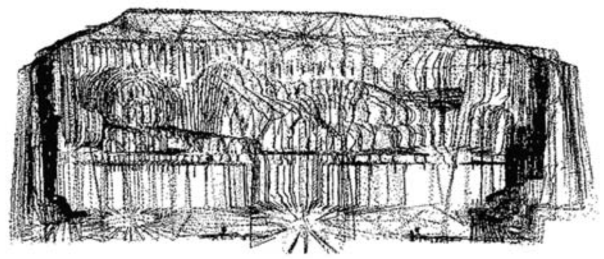
In the storage phase, we build four levels of detail for the original dataset. In table 2, we compare the loading time and rendering speed of different levels of detail. The rendering here only use the coordinates information.

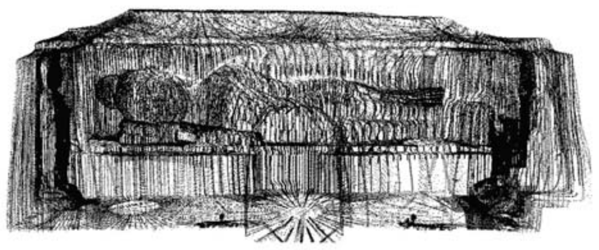
LOD	LOD3	LOD2	LOD1	LOD0	original
Point number	187,810	748,525	2,988,239	11,943,853	47,775,412
Loading time (ms)	129	439	1686	4496	30408
Fps (/s)	60	40	35	12	2

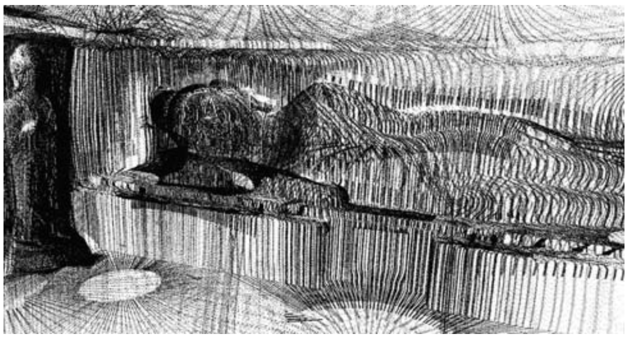
Table 2. Rendering comparison

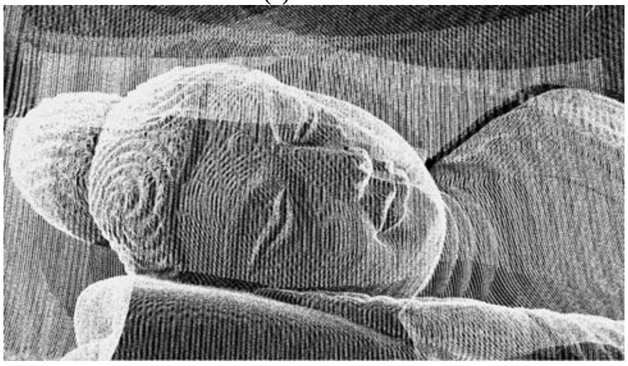
From table 2, we can see that when display the sampled LOD data, the rendering speed fulfils interactivity requirement. When the original data is displayed, the rendering speed decreases to a low degree. Because in our visualization phase, no additional culling is performed, we have to send all the loaded point data to the graphic pipeline. When the data amount increases, the loading time increases but is still acceptable.

Figure 5 shows different visualization effects of the LOD models. We move the viewpoint in the scene, and when the viewpoint gets closer to the point cloud model, more detailed data is displayed.


(a) LOD 3


(b) LOD 2


(c) LOD 1


(d) LOD0

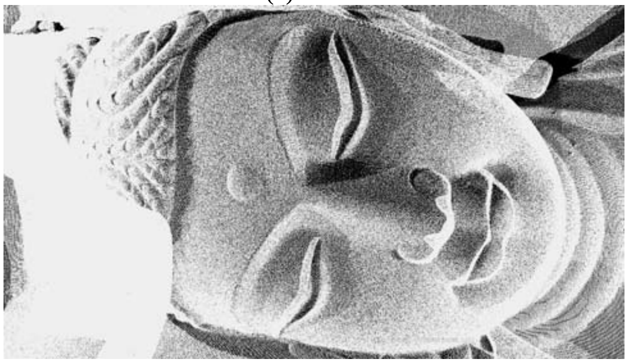

(e) Original point

Figure 5. LOD views of the Point cloud data

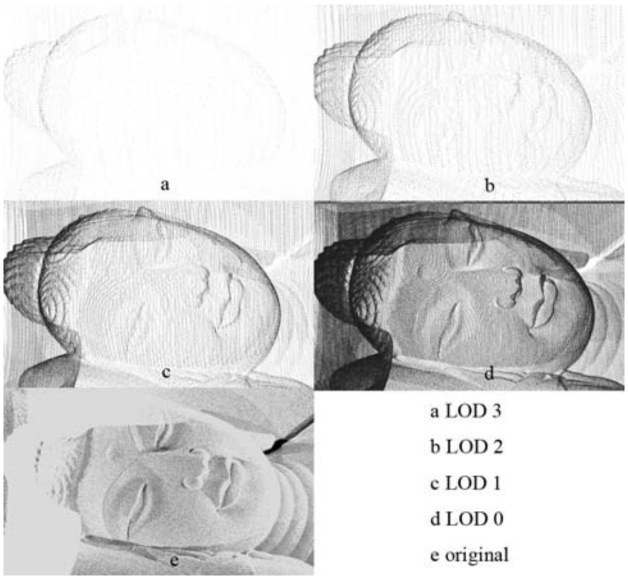


Figure 6. LOD quality

The quality of our LOD models are shown in Figure 6, we fix the viewpoint and load models of different levels, the LOD models created by image down-sampling are quite good.

6. CONCLUSION AND FUTURE WORK

We have presented a simple and fast approach to store and visualize the massive point cloud dataset. Despite it simplicity, using multi-band images to store the point cloud dataset can be a critical point for the future research, for this introduces the image processing approaches into point cloud processing field. For instance, we employ the image down-sampling method to create LODs for the point cloud dataset. In our current approach, the limitation is the LOD model quality can not be promised. In the original point cloud dataset, the points are unstructured and unordered. It is difficult to maintain the geometry characteristic in the LOD model without any other processing. This is what we seek to solve in the future work.

We render the scattered point cloud dataset directly. The current major limitation is in rendering primitive amount and image quality. Without culling, we have to load all the point cloud data at the same detail level. For massive point cloud dataset that is larger than the memory, we can only view its LOD model. And the points on the back face can affect the image quality. Evidently, points cannot occlude one another (unless they accidentally fall along the same ray from the viewpoint), and therefore no point is actually hidden. This affects the rendering effect a lot. To solve the visibility problem of point cloud (Katz, 2007) provides some clues.

References

Correa, T. W., 2004. *New Techniques for Out-Of-Core Visualization of Large Datasets*. Princeton University, USA.

Cox, M., Ellsworth D., 1997. Application-controlled Demand Paging for Out-of-core Visualization. In: *Proceedings of the 8th conference on Visualization '97*, Phoenix, Arizona, United States, pp. 235-244.

Gobbetti, E., Marton, F., 2004. Layered Point Clouds. In: *Eurographics Symposium on Point Based Graphics*, pp. 113-120.

Gross, M., Pfister, H., 2007. *Point-Based Graphics*. Morgan Kaufmann publisher, pp. 1-8.

Gu, X., Gortler, J. S., Hoppe H., 2002. Geometry Images. *ACM Transactions on Graphics*, 21(3), pp. 355-361.

Kasik, D., Stephens, A., 2008. Course Notes: Massive Model Visualization Techniques. In: *International Conference on Computer Graphics and Interactive Techniques*, pp. 1-20.

Katz, P., Tal, A., Basri, P., 2007. Direct Visibility of Point Sets. *International Conference on Computer Graphics and Interactive Techniques*, San Diego, California, pp. 24-35.

Levoy, M., Rusinkiewicz, S., Ginzton M., 2000. The Digital Michelangelo Project: 3D Scanning of Large Statues. *Computer graphics and interactive technique,* pp. 131-144.

Luebke, D., Watson, B., Cohen J., 2002. *Level of Detail for 3D Graphics*. Elsevier Science Inc.

Pajarola, R., 2005. Stream-Processing Points. In: *Visualization 2005*, pp. 239-246.

Rusinkiewicz, S., Levoy M., 2000. QSplat: A Multiresolution Point Rendering System for Large Meshes. In: *International Conference on Computer Graphics and Interactive Techniques,* pp. 343-352.

Silva, C. Chiang,Y., El-Sana, J., Lindstrom, P., 2002. Out-Of-Core Algorithms for Scientific Visualization and Computer Graphics. *IEEE Visualization 2002.* Boston, Massachussetts, United States**.**

Wright, S. R., Lipchak, J. B., Haemel, N., 2007. *OpenGL SuperBible Fourth Edition*. Addison, Wesley, pp. 421-455.