VISUALIZATION OF POINT CLOUD MODELS IN MOBILE AUGMENTED REALITY USING CONTINUOUS LEVEL OF DETAIL METHOD

L. Zhang[1,*], P. van Oosterom[1], H. Liu[1]

[1] Faculty of Architecture and The Built Environment, TU Delft, the Netherlands - (L.Zhang-13, P.J.M.vanOosterom, H. Liu-6)@tudelft.nl

Commission IV

KEY WORDS: Point Cloud, Augmented Reality, Continuous Level of Detail

ABSTRACT:

Point clouds have become one of the most popular sources of data in geospatial fields due to their availability and flexibility. However, because of the large amount of data and the limited resources of mobile devices, the use of point clouds in mobile Augmented Reality applications is still quite limited. Many current mobile AR applications of point clouds lack fluent interactions with users. In our paper, a cLoD (continuous level-of-detail) method is introduced to filter the number of points to be rendered considerably, together with an adaptive point size rendering strategy, thus improve the rendering performance and remove visual artifacts of mobile AR point cloud applications. Our method uses a cLoD model that has an ideal distribution over LoDs, with which can remove unnecessary points without sudden changes in density as present in the commonly used discrete level-of-detail approaches. Besides, camera position, orientation and distance from the camera to point cloud model is taken into consideration as well. With our method, good interactive visualization of point clouds can be realized in the mobile AR environment, with both nice visual quality and proper resource consumption.

1. INTRODUCTION

Visualizing the point clouds is an integral part of processing point cloud data, which enables users to explore point clouds more intuitively. However, most of the current point cloud renderers are developed for non-immersive environments. In the last few years, virtual reality (VR), Augmented Reality(AR) and Mixed Reality(MR) come into public view. These new technologies introduce some new ways of presenting 3D content. Different from MR and VR, AR applications can be used on mobile devices without specific equipment like helmet and handles. Therefore, there are already many mature applications for mobile AR, such as architecture, industrial design, navigation, advertisement, medicine, gaming, and so on. Nevertheless, the use of point clouds in the mobile AR application is still waiting to be explored. In our paper, we will try to realize the interactive visualization of point clouds in the mobile AR environment.

To develop mobile AR applications, there are two mainstream SDKs. One is ARKit by Apple, focusing on the IOS platform. Another is ARCore by Google, which develops AR applications on the Android platform. In our paper, we will develop our mobile AR point cloud applications with ARCore SDK. AR is a quite complex technique, which involves sophisticated mathematics, physics, and computer science knowledge. Fortunately, ARCore provides plenty of built-in functions which realize most of the basic operations needed by AR applications, such as motion tracking, lightning, feature detection, etc. Together with the Unity game engine and OpenGL, we can develop a good-quality point cloud renderer in mobile AR.

Similar to rendering point clouds on other platforms, the biggest problem of showing point clouds in mobile AR is the large quantity of point cloud datasets. In mobile AR applications, the frame rate needs to reach 30 fps with the limited CPU and GPU resources. In order to reach the relatively high visual quality and performance requirements, a cLoD (continuous level-of-detail) method is introduced to realize the required interactive visualization of point clouds in the mobile AR environment.

[*] Corresponding author

2. METHODOLOGY

2.1 Continuous LoD calculation

In our thesis, a method (Van Oosterom, 2019) will be taken to generate cLoDs. This cLoD model is developed based on the idea of refining ideal discrete level-of-detail and making them be a continuous function. Compared with state-of-art dLoD (discrete level-of-detail) methods, our cLoD model has an ideal distribution over LoDs, which means we can realize a smooth transition in density, avoid density shocks as present in dLoD approaches and keep the desired relative point density as much as possible.

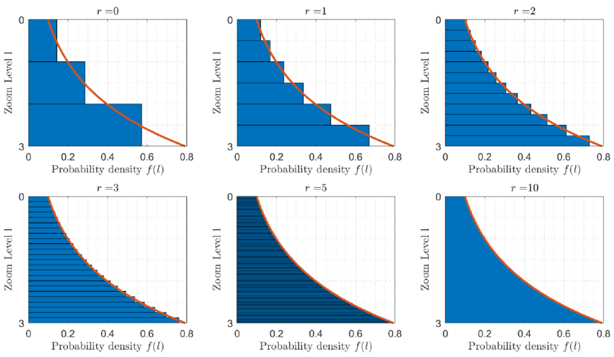

Figure 1. blue bars: refined discrete level-of-detail, red curve: continuous function (Van Oosterom, 2019)

In the following equations, L is the max level, l is a level between 0 and L+1, and n is the number of dimensions. For nD point clouds, there is an ideal continuous distribution function over levels.

$$f(l,n) = \frac{2^{(n-1)l}(n-1)\ln 2}{2^{(n-1)(L+1)}-1} \tag{1}$$

This function has Cumulative Distribution Function (CDF):

$$F(l,n) = \frac{2^{(n-1)l}-1}{2^{(n-1)(L+1)}-1} \tag{2}$$

In our algorithm, random generator U (uniform between 0 and 1) is used to assign cLoD dimension l (value between 0 and L+1) for the next point in nD space:

$$l = F^{-1}(U) = \frac{\ln\left(2^{(n-1)(L+1)}-1\right)U+1}{(n-1)\ln 2} \tag{3}$$

Thus, we can get continuous levels of detail model which accords with an ideal probability distribution.

2.2 Adaptive Point Size

Typically, when showing point clouds, the points will be shown at the same size. However, there are some issues with this traditional rendering strategy.

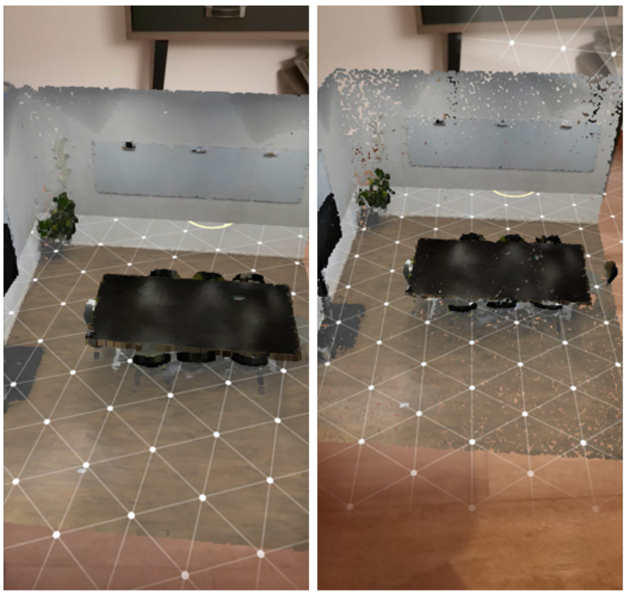

Figure 2. Points with adaptive size(left) and with same size(right)

First, as shown in Figure 2, if the size is too small, then there will be obvious holes between points, which will be more noticeable when zooming in. Second, if the size is too big, the neighboring points will overlap a lot and cause a loss of information. Therefore, in order to get better visual quality, the point size of each point is set as different values in our paper.

Based on the perspective projection matrix, we derive a formula to calculate ideal point sizes at different depth in the viewing z-axis direction. The ideal point sizes are determined by the shape and size of the view frustum, the height of the screen,

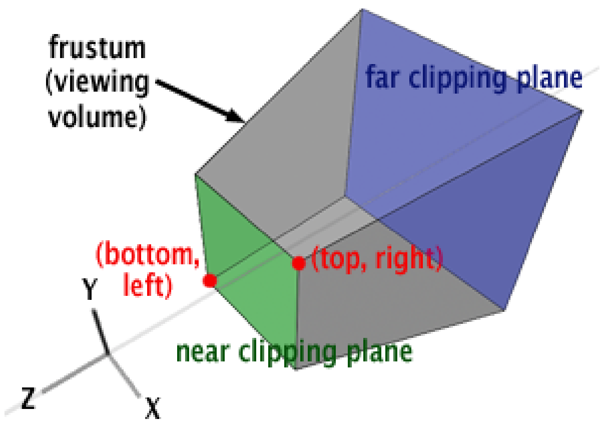

Figure 3. Principle of Perspective Projection (Scratchapixel, 2014)

and the distance from a point to the viewpoint in viewing z-axis direction. As shown in Figure 3, 'r' in the formula is the right coordinate of the near clipping plane, and 'n' is the distance from the viewpoint to the center of the near clipping plane. With this formula, points close to the camera will be larger, while points far away from the camera will be smaller.

$$size = \frac{-0.5 * n * r * screenHeight}{z_{eye} * tan(0.5 * fov)} \tag{4}$$

where
r = right coordinate of near clipping plane
n = near clipping plane distance
fov = field of view
$screenHeight$ = height of device screen
z_{eye} = point depth in local space

These ideal point sizes are not only used while rendering the points, but they will also be used in the next chapter to choose the proper value for parameters of formulas used in the selective query step.

2.3 Selective Query

The biggest issue of conventional dLoD methods is that there will be a sudden change of density at the splice of different levels. Our cLoD method can avoid this kind of artifact since we filter the data in point-wise instead of taking points of the same level as a whole. As we can see in Figure 4, the transition between different levels is more gradual in the cLoD model.

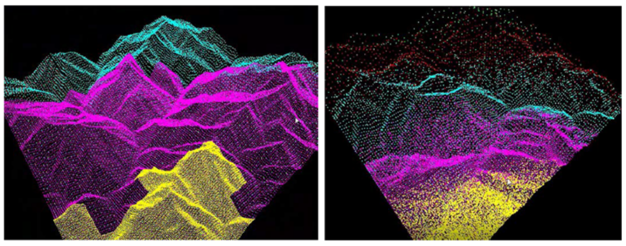

Figure 4. dLoD(left) and cLoD(right)

In our paper, the main idea of filtering the points is to reach an ideal point cloud density. Because of the limited CPU resources of mobile devices, the computation in point-wise, like distance or spacing between points, is too expensive and will cause an extremely low frame rate. Thus we choose to use a simple threshold over cLoD levels to filter the points. First, we want the Cumulative Density (CD) at a certain level. The Cumulative Density (CD) at continuous level l for nD case can be computed from the Cumulative Distribution Function:

$$CD(l, n) = \frac{F(l, n)N}{E^n} = \frac{(2^{(n-1)l} - 1)N}{(2^{(n-1)(L+1)} - 1)E^n} \quad (5)$$

where l = continuous level
N = total number of points in the dataset
n = number of dimensions
E^n = size of spatial domain in nD case

Then we can calculate the particular level l with which we can reach the ideal density. This level l can be derived from the Cumulative Density formula and the wanted ideal density, using the following formula. The value of ideal density is chosen based on the ideal point sizes we computed before. After experiments, we get the recommended value of the ideal density D, which is 100,000 to 200,000 points/m^2. We choose different ideal density value for different datasets is because that our method performs well when working with evenly distributed point clouds. However, when working with point clouds that are not evenly distributed, the value of ideal density needs to be smaller so that we can sufficiently reduce the number of points in the scene. What's more, considering the effect of distance, a logarithm of distance from the center of the point cloud model to the camera is set as the denominator. So that we can get higher density when the model is nearby, and lower density when the model is far away.

$$CD(l, n) = \frac{D}{\ln \sqrt{((x-u)^2 + (y-v)^2 + (z-w)^2) + 1}} \quad (6)$$

where l = continuous level
D = ideal density
x, y, z = coordinates of point cloud center in world space
u, v, w = camera coordinates in world space

Thus we can get a certain level l from the formula above. By only showing points with levels less than l, we can reach the ideal density required by the user. All these selected points will be stored in the vertex buffer and wait to be rendered.

3. IMPLEMENTATION

In order to realize interactive visualization of point clouds in the mobile AR environment, all the methods in our paper are developed on ARCore (version 1.17.0) together with the Unity game engine (version 2018.4.21). The Unity scripts are written using C# and the shader programs are written using HLSL (High-Level Shading Language). The table below shows currently supported functions in our rendering system.

#Number	Functionality
1	Load Points
2	Rendering Points
3	Transform Objects
4	Show Point Count and fps
5	UI to specify settings (point size, lower density, etc)

3.1 Point Cloud Input and Storage

Point cloud files can be divided into non-binary files and binary files. Compared to non-binary files with '.xyz' and '.txt' extensions, binary files like LAS files have more advantages while processing: they are more compact in size; they can carry more standardized information; reading binary files are much quicker than reading non-binary files. Thus, in our point cloud renderer, we take a kind of binary files, LAS files as input. In our tests, compared to reading text files, the speed of reading LAS files is at least two times quicker.

After loading point cloud from the file, we first apply some simple transformation to the point cloud model. First, the point cloud model is translated according to its centroid coordinates so that the referenced origin is located at the center of the point cloud. Second, the point cloud is scaled based on the size of its bounding box.

The transformed point cloud model is stored as Unity's mesh game object. Using mesh game objects is the mainstream method to store and visualize the point clouds in Unity. The parameter called MeshTopology is set as Points when working with point clouds. It can handle a large number of points and improve the visual quality by changing the materials, shaders, lightning, and shadows used to visualize point clouds. When visualizing large scale point clouds, multiple mesh objects might be used to create blocks for point clouds.

3.2 Put Point Cloud in the Scene

When the users move around with their mobile devices, ARCore will use the camera to detect vertical or horizontal surfaces, such as the floor and tables. These detected surfaces are called detected planes, which can be used to anchor virtual objects to the scene. In our paper, point cloud models are put in the scene by hitting on the detected planes.

In order to put 3D virtual objects on 2D planes, ARCore performs a raycast against detected planes. A raycast is a ray that gets sent out from a position in 3D or 2D space and moves in a specific direction. In this case, the direction of the raycast is determined by the touch position on the screen and the camera position. The position where the raycast hits the detected plane will be recorded, and the point cloud model will be transformed based on the hit pose. After the transformation, anchors will be created to store a fixed location and orientation in the real world. With anchors, point cloud models appear to stay in one place in the scene, no matter how the device moves.

3.3 Point Cloud Real-time Update

After putting point cloud models in the scene, the point cloud model will be updated in real-time. The vertex buffer will be updated by doing the selective query. In order to stay at a relatively high frame rate, the selective query step will be implemented once per 10 frames. Meanwhile, point sizes will be recalculated based on the latest position of the camera in the shader program as well.

What's more, ARCore's manipulation system is activated to enable basic interactions, like scaling, rotating, and translating point cloud models. The interactions are realized by the gesture detection together with manipulation system.

4. INITIAL RESULTS

With our methods, the number of points to be rendered is significantly reduced without loss of visual quality due to our cLoD selection from a too large point cloud for basic display. We put the point cloud models into the scene and move the devices from 0.2m to 2m away to the models, and see the changes in the number of points rendered in the scene. As shown in Figure 5, which are results with a dataset of 1498092 points. The number of points is reduced to less than 10% of the original number but still has good visual quality together with the adaptive point size rendering strategy.

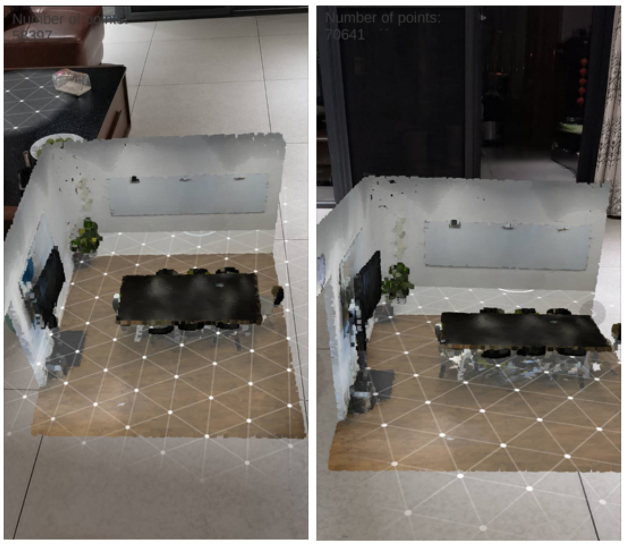

Figure 5. Result with 58,397 points(left) and 70641 points(right)

5. APPLICATIONS AND FUTURE WORK

5.1 Applications

The greatest strength of this method is that we can directly get use of the easily obtained point clouds in the mobile AR environment, without any pre-processing steps like turning them into meshes. There are some potential applications of this efficient interactive visualization of point clouds.

For indoor applications, first, we can put some dense point cloud models of smaller objects, such as tables and chairs, which will be useful for home renovation. Second, large scale point clouds are useful as well. For instance, putting the scanned point clouds of an entire room to simulate different environments, or putting scaled point clouds of an entire building for real estate sales.

As for outdoor applications, our method can be widely used in architecture and industrial design fields. Users can put point cloud models of roads, bridges, or other giant objects into the real world with our method before construction to simulate the final result without spending a lot of time and energy establishing 3D models. Change detection can also be added to our method so that we can highlight the changes between two point clouds in the AR environment.

5.2 Future Work

First, we'll test our method with more datasets and different devices to see its applicability. Second, we only test our method in the indoor environment. In the future, we'll explore more outdoor applications. Third, we will explore the potential of our method to visualize larger point clouds like city or nation wide point clouds, which needs server and caching strategy or information transfer between SSD and CPU. Finally, more interactions will be applied to our methods, like measuring the objects, changing colors of points, and so on.

Work is ongoing, at the time of 3D GeoInfo Conference, the App will be made public together with source code in GitHub.

6. ACKNOWLEDGMENTS

The authors would like to thank Ordnance Survey GB (https://www.ordnancesurvey.co.uk) and 1Spatial (https://1spatial.com/) for sponsoring the publication of this paper.

REFERENCES

Scratchapixel, 2014. The perspective and orthographic projection matrix. https://www.scratchapixel.com/lessons/3d-basic-rendering/perspective-and-orthographic-projection-matrix/opengl-perspective-projection-matrix.

Van Oosterom, P., 2019. From discrete to continuous levels of detail for managing nd-pointclouds. Keynote presentation at ISPRS Geospatial Week.