PotreeConverter - Uniform Partitioning of Point Cloud Data into an Octree

Markus Schütz
Student of Visual Computing
Entwurf und Programmierung einer Rendering-Engine
Vienna University of Technology
mschuetz@potree.org

1 Introduction

Potree relies on an octree to store point cloud data and load only those nodes visible from the current point of view. Each node, even internal nodes, stores a subset of the point cloud. The former PotreeConverter, written in Java, assembled the octree by randomly picking points for each node. The new PotreeConverter, written in C++, chooses points uniformly which improves the visual quality on the one hand but also performance since fewer points are needed to achieve a better quality.

2 Partitioning

The new, recursive, approach works as follows:

1. Define a minimum distance between points.
2. Iterate through all points in the unprocessed node d and choose a subset such that the distance between all selected points is bigger than minDistance.
3. Save the selected points in the processed node r.
4. Depending on their position, the remaining points will be split into nodes d0 to d7, which represent the 8 child nodes of d.
5. Repeat the process for each child of d.

A 2-Dimensional example using a quadtree instead of an octree is shown in Figure 1. With each repetition, the minimum distance is cut in half. Nodes with prefix r are finished. Nodes with prefix d are unprocessed and need to be partitioned. The name of each node indicates its location in the octree. Nodes d and r are the roots. Node r07 is the 8th child of the first child of the root node.

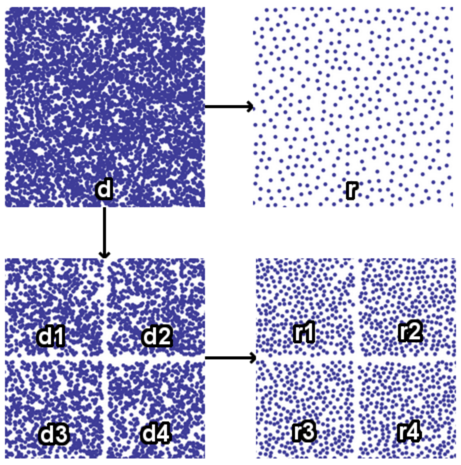


Figure 1: Processing 2 levels

3 Point Selection

Random point selection results in a non-uniform subset. This leads to holes in some areas and clusters of points in other areas, as shown in Figure 3. Selecting points such that the distance between each point is roughly the same improves the quality of the subset. A sparse 3D-Grid is used to do this efficiently. The grid dimension is equal to the Axis Aligned Bounding Box(AABB). The width, height and depth of each cell is equal to the minimum distance between points. Then, the grid cell index of each point is calculated. If the cell and all its neighbours are empty, the point is stored in the cell. If the cell or at least one of the 26 neighbours is not empty, then the distance to the points in this 27 cells area is checked. If the distance to all points is greater than the minimum distance, then the new point is added to the grid. Figure 2 shows how 2 points were successfully added to the grid, but the third point was discarded.

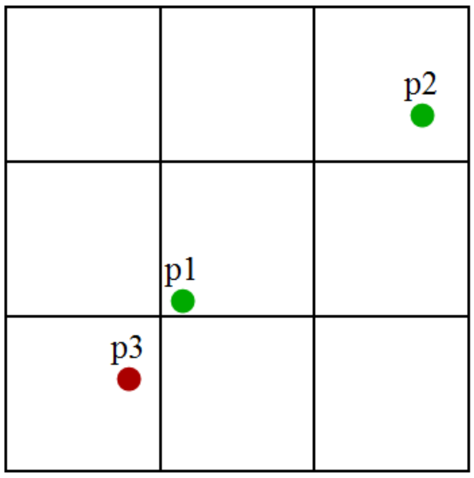
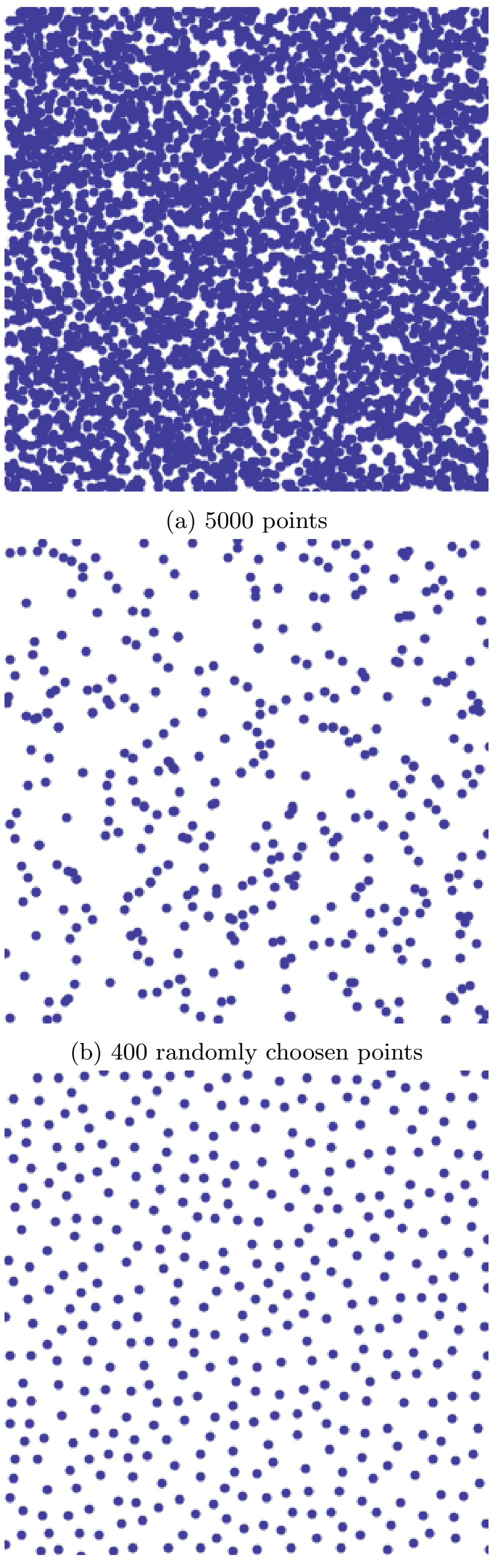


Figure 2: At the beginning, the grid cell and its neighbours are empty. P1 is added to the grid. When p2 is added, the distance between p1 and p2 must be checked. Since it is greater than the required minimum distance, p2 is added to the grid. When p3 is added, the distance between p1 and p3 must be checked. Since it is smaller than the required minimum distance, p3 will not be added to the grid.

The distance between all points inside the grid will be greater than the minimum distance and saved in a node with prefix r. The remaining points will be saved in nodes with prefix d.

This method works for choosing the root node subsample. It is, however, also necessary to ensure that points in descending nodes are not too close to points further up the hierarchy. Therefore, when processing any descendant of the root, d012 for example, all points in its ancestors, r, r0 and r01, must be added to the grid first. Points from ancestor nodes will not be saved to r012, though, since this would result in duplicates. They are only needed for the distance check.



(a) 5000 points

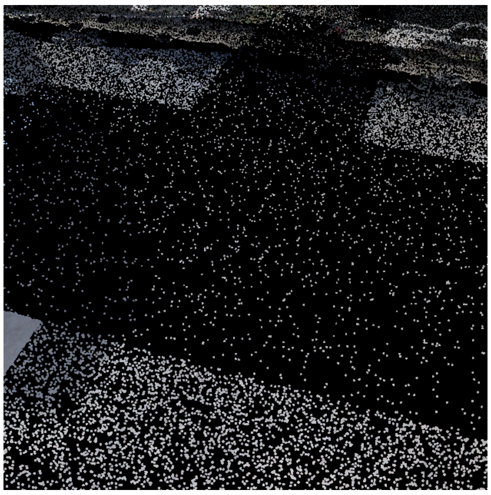
(b) 400 randomly choosen points

(c) 403 points choosen with a minimum distance from each other

Figure 3: 2 different sampling strategies

4 Results

The results of the new PotreeConverter are better in quality but also in performance as fewer points are needed for to achieve a better look. An example is shown in Figure 4.


(a) Pompei point cloud with random sampling.

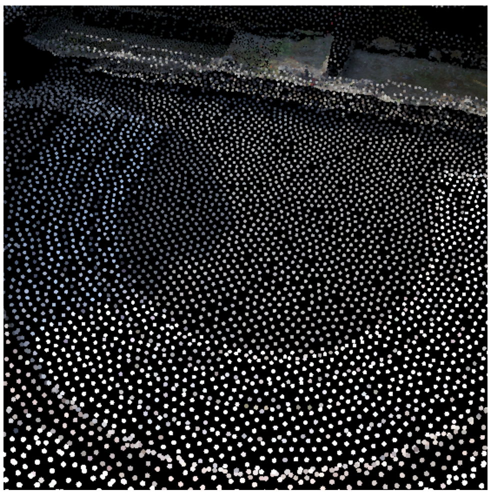

(b) Pompei point cloud with uniform sampling.

Figure 4: Results with a point cloud provided by the CNRS-MAP-Gamsau laboratory