BOLT ⚡: Bilateral Filtering and Octree Lightweight Technique for Fast and Parameter Free Point Cloud Upscaling

Sharanshangar Muhunthan
shangar.muhunthan@mail.utoronto.ca

Fei Yu Guan
vi.guan@mail.utoronto.ca

March 15, 2024

Abstract

In this paper, we introduce BOLT (Bilateral filtering and Octree Lightweight Technique), a novel fast and parameter free method for upsampling point clouds. We leverage the structural efficiency of the octree data structure and detail preserving properties of the bilateral filter to achieve a fast and parameter free upsampling method. Unlike the current state-of-the-art methods, BOLT does not require any parameters, deep learning, fine tuning or any type of training making it a suitable candidate for real-time applications. BOLT understands the underlying structure of the point cloud by dividing the point cloud into a hierarchical octree structure. Empty children are filled in the octree and outliers are smoothed using a bilateral filter.

1 Introduction

A pointcloud is an unordered 3D representation of a set of points in space. It is commonly used in computer graphics, computer vision, and robotics. Pointclouds are often generated using 3D scanners, LIDAR, and many more 3D applications.

In this work we wish to upsample a pointcloud. Given a set of point clouds, we wish to find a new set of points that are more dense but still represent the same underlying surface. Further, the new points while preserving the underlying structure should not introduce any new artifacts, and should be informative and not clustered around the original points. The unstructured and unordered nature of point clouds makes this a challenging problem. Further, existing methods for point cloud upsampling often are computationally expensive and require extensive training and parameter tuning.

To address the above challenges we present a data-structure-driven method for point cloud upsampling that is fast and parameter free. Our method utilizes an octree data structure without a depth limit to understand the underlying structure and initially add points to the empty children of the octree. The lack of a depth limit allows tighter fitting bounding cubes around points and allows for a more accurate representation of the underlying structure. This representation is often noisy and coarse but captures the underlying structure of the point cloud. To smooth the point cloud, we use a bilateral filter in a point cloud application [5] to smooth the point cloud.

Point cloud upsampling can be used as a downstream task for various applications such as 3D reconstruction, 3D object recognition, and 3D rendering. It can be used to improve the quality of surface reconstruction, enhance object detection, extract features more accurately, and more.

Our method, namely BOLT, learns the geometry and strucutre of the point cloud, upsamples and smooths it without any parameters, deep learning, or fine tuning.

2 Related Work

Many non-deep learning based methods for point cloud upsampling have been proposed in the past such as moving least squares interpolation (MLS interpolation) in 2002 [3], Locally Optimal Projection (LOP) in 2007 [10], Edge Aware Resampling (EAR) in 2013 [8] and graph total variation in 2019 [6].

MLS works by fitting a continuous surface to a set of local points using a weighted least squares fit of a polynomial surface to the points. Points are added by computing the voronoi cells on the local surface and adding points to the vertices of the diagram.

LOP unlike MLS does not require fitting a local surface. Instead, it uses a projection operator to project points onto a surface in a way that minimizes the sum of the weighted distances between the original and projected points. Improvements to LOP such as weighted LOP [7] were proposed that make LOP more robust to noise and outliers.

Both MLS and LOP have demonstrated good results but a common problem with these methods is they don't perform well on sharp edges and corners, as the model often assumes a smooth surface.

EAR was designed to work well on edges [8]. It works by first computing the normals and relateive curvature of each point. Then if the curvature is above a certain threshold, the point is considered to be on an edge, and the point is projected onto the tangent plane of the edge. If a point is considered a surface, the point is projected onto the tangent plane of the surface.

Graph total variation is a method that uses a graph to represent the point cloud. They first construct a triangular mesh, then insert points at the centroids of the triangles. Assuming the point cloud is piecewise smooth, they then minimize a weighted average of the L_1 norms of normals between points.

Many deep learning based approaches also exist, such as PU-Net [12], PU-GAN [9] and PU-GCN [11]. Although these point cloud upsampling methods tackle a different problem, they were still used as a point of comparison. The reason these are solving different problems is because these are large networks trained on large datasets, and require a lot of computational power to train and run. The goal of this paper is to propose a fast and parameter free method for point cloud upsampling.

3 Background

3.1 Octree

An octree is a tree data structure in which each internal node has exactly eight children. Octrees are often used to partition 3D space and are used in various applications such as computer graphics, computer vision, and robotics.

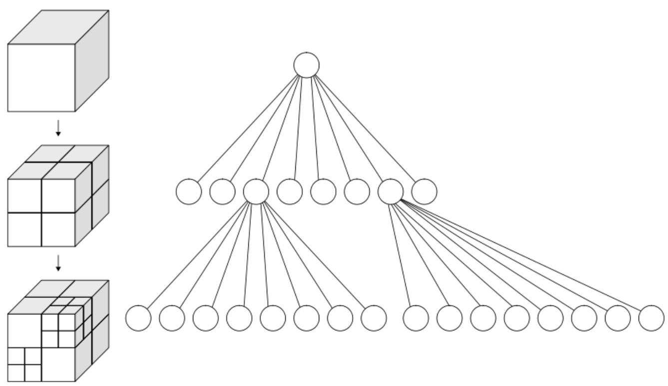


Figure 1: An example of an octree. Each cube gets recursively divided into eight equal octants. The points are stored in the leaf nodes. Image from [1]

In the context of point clouds, an octree is used to partition the 3D space and store the points in the leaf nodes. The octree is a hierarchical data structure that recursively divides 3D space into eight equal octants. Each node in the octree represents a rectangular prism in 3D space with a particular center, width, lenghth and depth. Nodes that aren't leaf nodes have exactly eight children, and leaf nodes store the points in the point cloud. The root node of the octree represents the bounding box of the point cloud. An octree has $\mathcal{O}(\log n)$ complexity for insertion and search operations, where n is the number of points in the point cloud. Bounding cubes nearest points are much finer and tighter fitting than those further away, allowing for a more accurate representation of the underlying structure of the point cloud.

Further, these tighter bounding boxes give hints on where to add new points to the point cloud, since adding points to the tighter bounding boxes will result in points that are more informative and not clustered around the original points, will not introduce any new artifacts, and will preserve the underlying structure of the point cloud.

Most octrees have a depth limit, which means that the octree will not divide the space beyond a certain depth, this is to avoid a problem of infinite recursion. However, in our method, we do not have a depth limit, and we allow the octree to divide the space as much as possible. since the starting point clouds are often sparse and noisy, and the lack of a depth limit allows for a more accurate representation of the underlying structure of the point cloud. Other stops such as checking if an existing close point is already in the node are used to avoid infinite recursion.

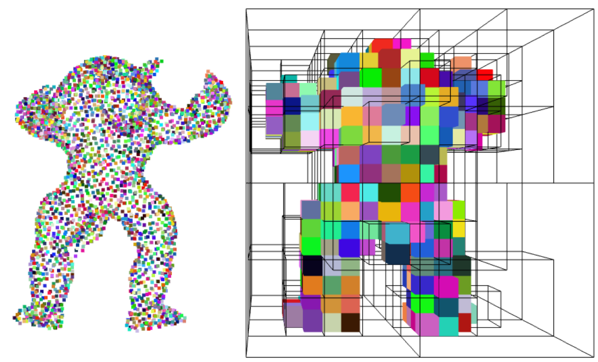


Figure 2: An example of an octree with its point cloud and cubed representation. The figure was from a blog post [2] and was generated using the Open3D library [13] with a max depth of 4.

3.2 Bilateral Filter

The bilateral filter is a non-linear filter that is used to smooth images and reduce noise while preserving edges. It is a generalization of the Gaussian filter, and it is used in various applications such as image processing, computer graphics, and computer vision. It is defined as follows:

$$B(I,x) = \sum_{x_i \in \Omega} I(x_i) f_r(\|x_i - x\|) g_s(\|x_i - x\|)$$

In our paper, we use the bilateral filter to smooth the point cloud. Similar to the image case, the bilateral filter smooths the point cloud while preserving the edges and the underlying structure of the point cloud. It works by shifting points along a normal vector, and the amount of shift is a weighted average distance to its neighbours. We follow Digne et al. [5] and use the following definition of the bilateral filter for point clouds:

$$p' = p + \delta_p \cdot n_p \tag{1}$$

Where n_p is the normal to the regression plane of some k nearest neighbours of p. Also following Dinge et al [5], our implementation computes the normal with PCA. PCA will find the regression plane that fits the data best and the corresponding found eigenvectors will be orthogonal to said plane, and thus is simple to compute compared to an iterative least squares method. δ_p is the displacement of the point p. The displacement is computed as follows, let $\mathcal{N}_p$ be the set of k nearest neighbours of p:

$$\delta_p = \frac{\sum_{q \in \mathcal{N}_p} w_d(\|q-p\|) w_n(\langle n_p, q-p \rangle) \langle n_p, q-p \rangle}{\sum_{q \in \mathcal{N}_p} w_d(\|q-p\|) w_n(\langle n_p, q-p \rangle)} \tag{2}$$

In our implementation, w_d and w_n are the Gaussian functions defined as follows:

$$w_i(x) = \exp\left\{-\frac{x^2}{2\sigma_i^2}\right\} \tag{3}$$

In our implementation we set $\sigma_d = 0.1$ and $\sigma_n = 0.1$.

4 Methodology

Our goal is to upscale and then smooth a sparse point cloud using an octree with a large depth and bilateral filtering on a point cloud. We start with a sparse point cloud $\mathcal{P} = \{p_1, \ldots, p_n\}$, and generate an octree $\mathcal{T}$ by iterating through and inserting one at a time. To generate the initial upsampling of the points, we find the parent of each for point p_i in $\mathcal{T}$, then add a new point to an empty child of the parent in $\mathcal{T}$. One such iteration will double the number of points in the point cloud, then another will quadruple and so on. This process gets repeated the number of times necessary to get the desired final number of points. Then we extract all points from $\mathcal{T}$ to get our new upsampled point cloud $\mathcal{P}'$. We then smooth $\mathcal{P}'$ with bilateral filtering. Bilateral filtering requires hyper parameters σ_d, σ_n and k. k indicates the number of neighbours used to find the normal of the regression plane, and σ_d, σ_n are the standard deviations for the gaussians used in (3).

Algorithm 1 Main upsampling algorithm

Require: sparse point cloud P with n points, n_{up} number of iterations required to get the desired number of points

```
function UPSAMPLE(P)
    T ←CONSTRUCTOCTREE(P)
    for i ≤ n_up do
        for p ∈ P do
            parent ←p.parent
            child ←RANDOMEMPTYCHILD(parent)
            p′ ←RANDOMPOINTINSIDE(child.dimensions)
            INSERT(p′)
        end for
    end for
    P ←CONVERTTOPOINTCLOUD(T)
    BILATERALSMOOTH(P)
end function
```

Algorithm 2 Bilateral smoothing algorithm, borrowed heavily from [5]

Require: point cloud P with n points, k neighbours, σ_d, σ_n

```
function BILATERALSMOOTH(P)
    for p ∈ P do
        N_p ←FINDNEIGHBOURS(P, k)
        n_p ←COMPUTEUNITNORMALTOPLANE(N)
        s_w ←0
        δ_p ←0
        for q ∈ N_p do
            w ←w_d(‖p − q‖) · w_n(⟨n_p, p − q⟩)    ▷ From (3)
            s_w ←s_w + w
            δ_p ←δ_p + w · ⟨n_p, p − q⟩
        end for
        p′ ←p + (δ_p / s_w) · n_p
    end for
end function
```

5 Preliminary Experiments

Experiments were done mostly with the ShapeNet dataset [4], which is a large dataset of 3D models. A random set of either 1024 or 2048 point clouds were sampled and then upscaled to double or quadruple the number of points.

At this stage only qualitative results are shown, and quantitative results will be shown in the final paper. Further, comparisons with other non-deep learning based methods, as well control studies comparing our method of placing with an octree and then smoothing to randomly placing points and then smoothing, will be done in the final paper.

5.1 Evaluation Metrics

We will eventually evaluate our model using the Chamfer distance and Hausdroff distance as they are common metrics used in point cloud upsampling, and try to compare to other parameter free works as well as deep learning based methods. The Chamfer distance is a measure of how different 2 shapes are and is defined as the following:

$$C(P,Q) = \frac{1}{|P|}\sum_{p \in P}\min_{q \in Q}\|p-q\|^2 + \frac{1}{|Q|}\sum_{q \in Q}\min_{p \in P}\|p-q\|^2$$

The Hausdroff distance is a measure of how similar 2 sets are. It is defined as the following:

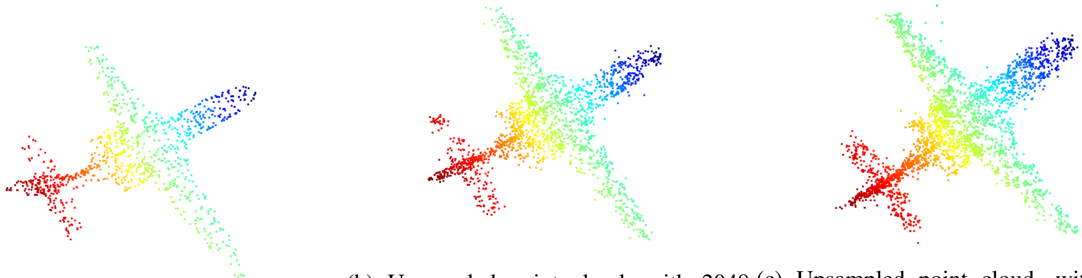
$$H(A,B) = \max(h(A,B), h(B,A))$$

Where:

$$h(A,B) = \max_{a \in A}\min_{b \in B}\|a-b\|$$

5.2 Preliminary Results

In this subsection we qualitatively compare results with octree upsampling only, and no smoothing, as well as both octree sampling and bilateral smoothing.

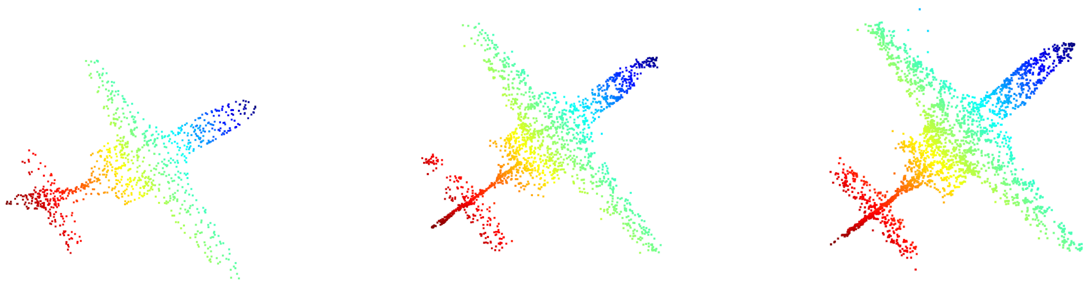


(a) Original point cloud, with 1024 points

(b) Upsampled point cloud, with 2048 points using only octree and no smoothing

(c) Upsampled point cloud, with 4096 points using only octree and no smoothing

Figure 3: A comparison of the original point cloud and the upsampled point cloud using only octree and no smoothing.



(a) Original point cloud, with 1024 points

(b) Upsampled point cloud, with 2048 points using only octree and bilateral smoothing

(c) Upsampled point cloud, with 4096 points using only octree and no smoothing

Figure 4: A comparison of the original point cloud and the upsampled point cloud using only octree and bilateral smoothing.

References

[1] Octree. `https://en.wikipedia.org/wiki/Octree`.

[2] Open3d octree . `https://blog.csdn.net/qq_41068877/article/details/124242265`.

[3] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Cláudio T. Silva. Computing and rendering point set surfaces. *IEEE Trans. Vis. Comput. Graph.*, 9:3–15, 2003.

[4] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository, 2015.

[5] Julie Digne and Carlo de Franchis. The Bilateral Filter for Point Clouds. *Image Processing On Line*, 7:278–287, 2017. `https://doi.org/10.5201/ipol.2017.179`.

[6] Chinthaka Dinesh, Gene Cheung, and Ivan V. Bajic. 3d point cloud super-resolution via graph total variation on surface normals, 2019.

[7] Hui Huang, Dan Li, Hao Zhang, Uri Ascher, and Daniel Cohen-Or. Consolidation of unorganized point clouds for surface reconstruction. *ACM Trans. Graph.*, 28(5):17, dec 2009.

[8] Hui Huang, Shihao Wu, Minglun Gong, Daniel Cohen-Or, Uri Ascher, and Hao (Richard) Zhang. Edge-aware point set resampling. *ACM Trans. Graph.*, 32(1), feb 2013.

[9] Ruihui Li, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-gan: a point cloud upsampling adversarial network, 2019.

[10] Yaron Lipman, Daniel Cohen-Or, David Levin, and Hillel Tal-Ezer. Parameterization-free projection for geometry reconstruction. *ACM Trans. Graph.*, 26(3):22es, jul 2007.

[11] Guocheng Qian, Abdulellah Abualshour, Guohao Li, Ali Thabet, and Bernard Ghanem. Pu-gcn: Point cloud upsampling using graph convolutional networks, 2019.

[12] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-net: Point cloud upsampling network, 2018.

[13] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.