Evaluating Surface Reconstruction Meshes Without Ground Truthing

Jacob Fein-Ashley [1]

[1]Advised by Dr. Hua Wang and Hoon Seo

Introduction

Creating surface reconstructions from point clouds is common in computer graphics, vision, and video games. The most common way to evaluate the quality of surface reconstruction is to compare it to a ground truth mesh. However, this is not always possible. A novel method for evaluating surface reconstructions without ground truthing using simple computational geometry will be examined.

(a)

(b)

Figure 1. (a) Common Bunny Mesh Example (b) Points Sampled from $\alpha = 0.05$-Reconstruction of (a)

Sometimes, it is impossible to attain a ground truth mesh to compare a three-dimensional surface reconstruction. Common examples of this occur when a mesh is constructed from a point cloud, and it is desired to evaluate quantitatively the performance of the mesh generator. Suppose that Figure 1a is a mesh generated from a point cloud Figure 1b. The desired metric will compute the distance between points in the point cloud and triangulations of the mesh.

Method and Introductory Definitions

A mesh can be defined as an overlay of discrete triangles on a continuous surface. The surface reconstruction algorithm will create a mesh that approximates the continuous surface. We can measure the quality of the surface reconstruction by comparing the point cloud to the mesh. Define the following qualities of a point cloud P. Let

$$P = \{(x_i, y_i, z_i) \mid i \in n\}$$

where there are n three dimensional points defined in the point cloud. It is also possible to define the qualities of a mesh $\mathcal{M}$. Suppose that $\mathcal{M}$ is constructed from a set of vertices $V \subseteq \mathbb{R}^3$ and triangular faces $F \subseteq \{1, \ldots, |V|\}^3$ where F is constructed from arbitrary three dimensional vertices composed of a set of three vertices, namely, three vertices $v \in V$ such that any arbitrary face f is defined by three vertices $f = [v_1, v_2, v_3]^T$ that form set F (also define $|F| = m$) as:

$$F = \left\{ \begin{bmatrix} v_{\in|V|} \\ v_{\in|V|} \\ v_{\in|V|} \end{bmatrix}_1, \begin{bmatrix} v_{\in|V|} \\ v_{\in|V|} \\ v_{\in|V|} \end{bmatrix}_2, \ldots, \begin{bmatrix} v_{\in|V|} \\ v_{\in|V|} \\ v_{\in|V|} \end{bmatrix}_m \right\}$$

$$= \{f_1, f_2, \ldots, f_n\}$$

where f_i is a face of the mesh $\mathcal{M}$ and $f_i = [v_{i1}, v_{i2}, v_{i3}]^T$ is a face of the mesh $\mathcal{M}$.

Algorithm Explanation

Additionally, define a k-d tree, simply a k dimensional binary tree structure. We compose the k-d tree of each face $f \in F$. The k-d tree sorts each face according to Euclidean distance, defined as ($k = 3$) and

$$\vec{dist}_{f_1,f_2} = \sqrt{\sum_{i=1}^{3}(f_{1_i} - f_{2_i})^2}$$

between each arbitrary face $f \in F$, and we compute this metric between each face vector in F. This structure speeds up computation and allows for querying nearest points in $\mathcal{O}(\log\{|V|\} \cdot n)$ time complexity. Because we are comparing the point cloud to the mesh, we query the k-d tree for the closest 3 points to each point in P. We wish to calculate the distance between each point $p \in P$ and the closest triangle in the mesh. As we defined earlier, each face is composed of $f = [v_1, v_2, v_3]^T$ vectors.

Geometric Exploration

It follows that we must compute a plane spanned by each vector in the KKT condition where α, β, γ are the optimal barycentric points of the nearest triangle to the point $p \in P$:

$$\alpha v_1 + \beta v_2 + \gamma v_3$$

$$\text{s.t. } \alpha + \beta + \gamma = 1$$

and by method of Lagrangians, we define the following function

$$L(\lambda) = (\alpha v_1 + \beta v_2 + \gamma v_3)^2 - \lambda(\alpha + \beta + \gamma)$$

$$\frac{\partial L}{\partial \alpha} \overset{set}{=} 2 \cdot v_1(\alpha \cdot v_1 + \beta \cdot v_2 + \gamma \cdot v_3) - \lambda = 0$$

which brings us to six cases which David Eberly describes in his paper an algorithm for solving the aforementioned Lagrangian [1]. Define a normal vector to each triangle:

$$\vec{n} = (v_2 - v_1) \times (v_3 - v_1)$$

and normalizing $\vec{n}$ yields $\hat{n} = \frac{\vec{n}}{||\vec{n}||}$. Now, we wish to find a point p_0 which is the closest point from p to the normalized plane $\vec{n}$. Define this distance such that $\min\{||p - p_0||\}$ is the closest distance from p to p_0. To determine where p_0 is located on the triangle formed by the vertices in each face $f \in F$, we turn to the algorithm described in Eberly's paper which solves the Langragian mentioned in six cases to determine where p_0 is located in the triangle. Located in Figure 1 of Eberly's paper:

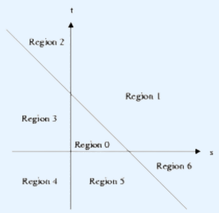

Figure 2. Figure 1 of Eberly's paper [1]

Geometric Exploration Cont.

Solving the Lagrangian yields six cases, which are described in Eberly's paper [1] for α, β, γ. Finally, solving for the point $p_0 = \alpha \cdot v_1 + \beta \cdot v_2 + \gamma \cdot v_3$ must yield values $\alpha, \beta, \gamma \in [0, 1]$ by notions of barycentric coordinates. It is proposed that for each point in the point cloud $p \in P$, we find the closest face $f \in F$ and average the distance to create the average distance of each point to the closest face. This can be defined as $dist_{avg}$ and its correspondent optimality constraint:

$$dist_{avg} = \frac{1}{n} \cdot \sum_{i=1}^{n} \min\left\{||p_i - p_{0_i}||_2\right\} = \frac{1}{n} \min\left\{||p - p_0||_2\right\}^T \mathbb{1}$$

$$dist_{avg} = \sum_{i=1}^{n} \arg\min_{p_{0_i}} \left\{||p_i - p_{0_i}||_2^2\right\}$$

Experimental Results and Conclusion

Using the popular Stanford Bunny Mesh as previously mentioned [2], $dist_{avg} = 1.875 \cdot 10^{-9}$ with the following distance plot, according to each point's distance to the closest face in the mesh:

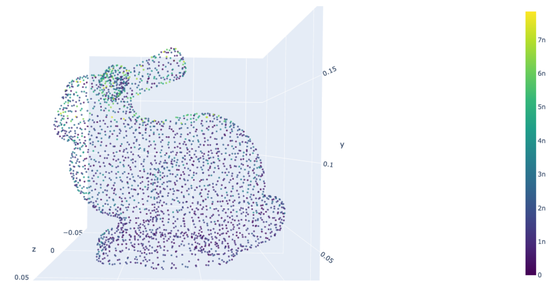

Figure 3. Distance Plot of the Stanford Bunny Mesh

The proposed distance metric uses an ℓ_2 norm to calculate the distance between each point in the point cloud and the closest face in the mesh. Once the distance for each point to its closest face is calculated, one can examine the distribution of the distance array. The proposed metric uses the non-robust ℓ_2 (sample mean) norm to both penalize outliers, and serve as a differentiable function for optimization. The proposed metric can be altered to ℓ_1 (sample median) or ℓ_∞ (spectral norm) norms to serve as a robust distance metric. Further areas of exploration include gathering a set of meshes with varying alpha values and surface reconstruction methods and comparing their respective distance vectors.

References

[1] David Eberly. Distance between point and triangle in 3d - geometric tools, 1999.

[2] Greg Turk Levoy and Marc, Oct 2016.