Generation of Photo-Realistic Environment using Lidar and Google Earth for AV Simulation

Aravind Chandradoss

Ohio State University

Introduction:

This work is mainly focused on construction of simulation environment for training and testing of Autonomous Vehicles. The intuition behind this approach is that, "Can we combine Lidar and Google Earth images for reconstruction? Coz, as such, if we reconstruct from Google images, the environment would not be "accurate" enough. On the other hand, if we use accurate Lidar data, we would lack texture and photo-realism for the Simulation. Can we combine the benefits of two methods? Yes!"

In this report, the procedure and results for reconstructing Photo-realistic 3D Environment of Linden is discussed. The workflow for creating the simulation environment as shown in Figure 1, and is introduced here in the context of the Linden Residential Area route. The overall aim is to use time-stamped three dimensional (3D) lidar data to reconstruct the surface of the entire environment and then to add texture details from Google Maps (Google Earth Studio [2]) and road details from OpenStreetMaps.

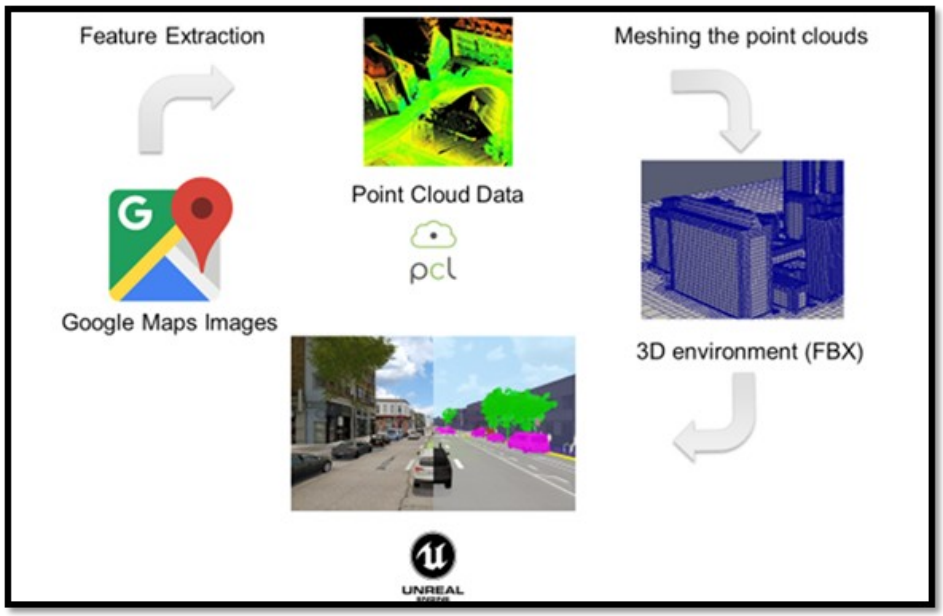

Figure 1. Pipelined Approach Used for Importing Real Maps into Unreal Engine

Pre-processing

In this step, we mainly focus on reducing the redundant point cloud data to boost the processing speed without considerable loss in reconstruction. Due to lidar construction, point cloud density is dependent on measurement distance, i.e., the point cloud will be very dense near the AV and sparse at further distance. Therefore, we use the Poisson Sampling Disk approach to remove highly cluttered point clouds using sampling disks. By using this step, it is possible to reduce up to 40-45% of lidar point cloud data. The recorded lidar data usually contains point clouds corresponding to other vehicles on the road. It is highly recommended to remove any such unwanted point clouds that correspond to objects that are stationary for a significant time during recording, for example, any stationary car waiting for a traffic signal. One can remove such point clouds in the pre-processing step using the SqueezeSeg [3] Convolutional Neural Network (CNN) or other start-of-art methods. During the pre-processing step for the Linden Residential Area recorded data, the freely available and open source MeshLab, and Point Cloud Library (PCL) [4] tools were used.

Figure 2.a. Linden Residential Area AV Route as a Rendered Mesh

Figure 2.b. Linden Residential Area AV Route as a Rendered Mesh

Figure 2.c. Linden Residential Area AV Route as a Rendered Mesh

Mesh Generation

In this step, a mesh is created from the point cloud data and is smoothed using a smoothening filter after which textures is added. Vertex normals to the point cloud are calculated first. The parameters for calculating the normals are selected heuristically based on lidar resolution, size of simulation environment and other similar factors. With the generated normals, the surface is constructed by building a mesh using the point cloud data. The Ball-Pivoting surface reconstruction was seen to give the best results. The parameters for Ball-Pivoting were selected based on point cloud density, size of simulation environment and others factor. In practice, surface reconstruction results in a mesh with voluminous triangles. Quadratic edge decimation was used to reduce the triangle count. Faulty triangles including duplicate edges and single point triangles were removed using PCL and MeshLab tools. A Laplacian filter was used next to smoothen the surface. Texture from Google Maps is added to the mesh in the next step to make the simulation environment look photo-realistic for simulation. The freely available and open source MeshLab, Blender [5], and PCL tools were used for mesh generation for the Linden Residential Area.

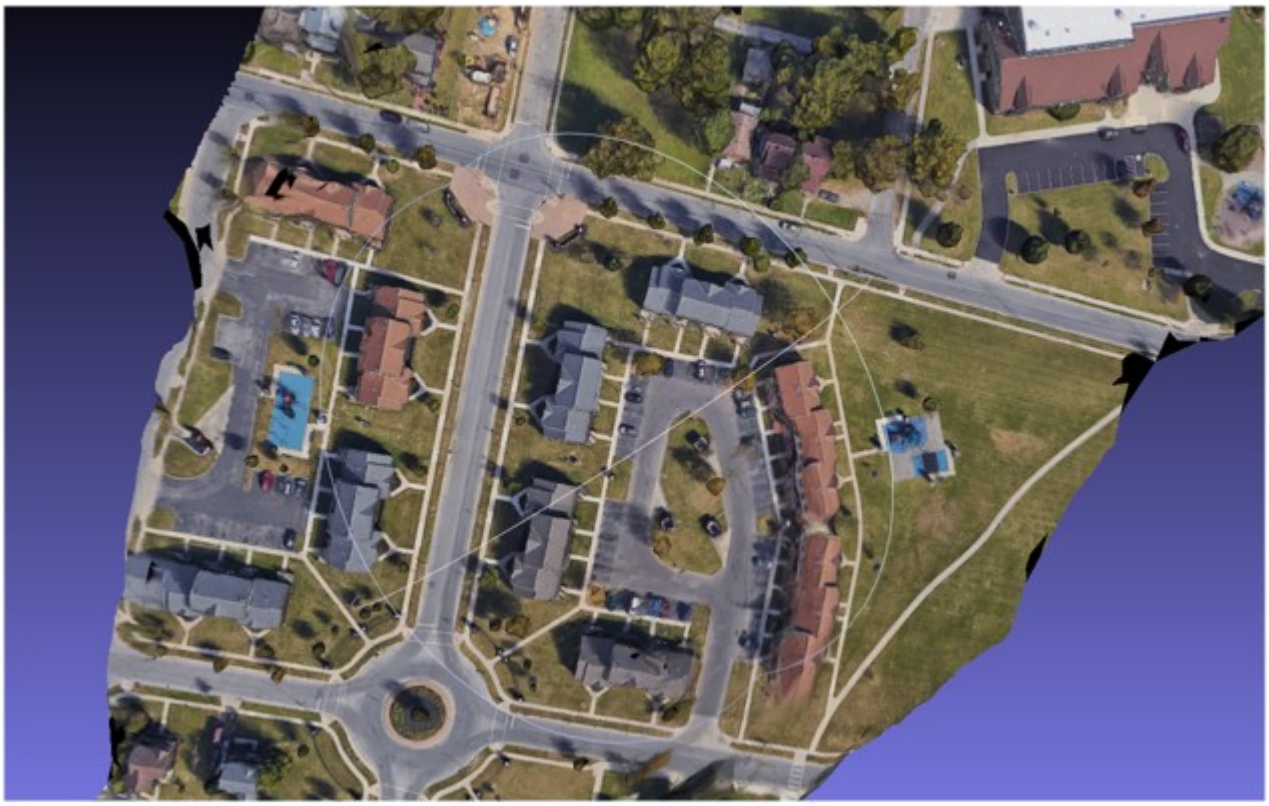

Figure 2.d. Linden Residential Area AV Route as a Rendered Mesh

Nearly 4,000 satellite images of the Linden Residential Area route were retrieved from Google Earth Studio [1] and used for generating mesh texture. It is also possible to

collect data using drones in order to construct a more realistic and up to date replication of the environment. It is not possible to recover texture information in occluded regions such as those under the trees.

In the upcoming steps, only the important tasks required to achieve the desired results are presented while skipping intermediary steps like feature extraction, point-to-point correspondence and feature matching which are easily taken care of using tools from the freely available, open source Meshroom [6] program.

Structure from Motions (SfM) and Depth Map

Using existing photogrammetry tools, with the matched features from previous step, the 3D reconstruction of the environment was achieved using SfM. Meshroom's SfM implementation was used here. The parameters for SfM were selected heuristically. Similar results can also be achieved using the freely available, open sourced VisualSFM tool. The mesh for the Linden Residential Area route was generated from 3D lidar scan data and the texture was only added for a photo realistic look. A depth map can also be generated using the dense point cloud from Google Earth images in order to obtain a better reconstruction of the environment.

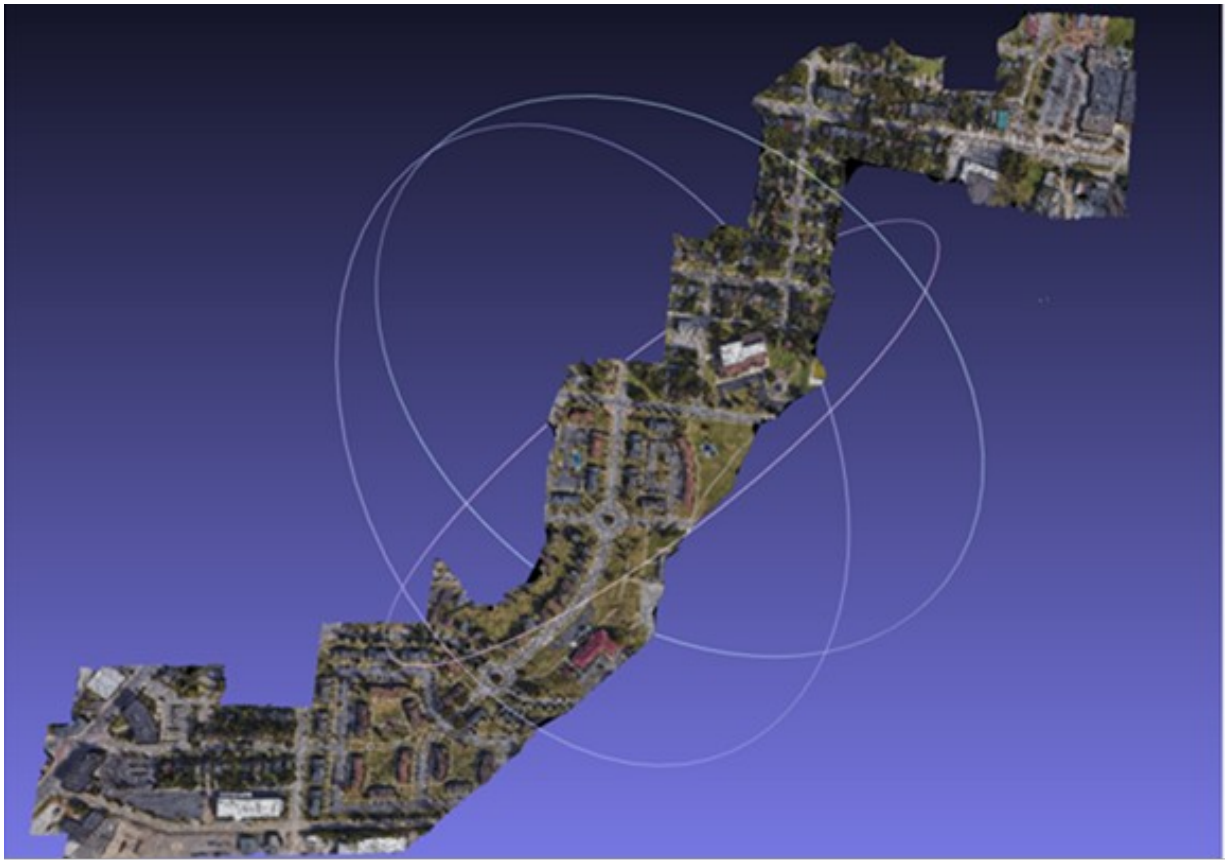

Figure 3. Linden Residential Area AV Route as a Rendered Mesh

Texture Generation

The surface generated from the meshing step is combined with the texture from the dense SfM using Least Square Conformal Maps. Blender was used in order to change the origin before using the mesh for texture generation. Once the textures are added to the surface, Blender and PCL libraries were used to convert the file format to filmbox (.fbx) as it compatible with both the Unreal Engine and Unity platforms.

The resulting rendered mesh with texture for the Linden Residential Area AV route is shown in Figure 2. Figure 3 displays close up views of select parts of that meshed simulation environment.

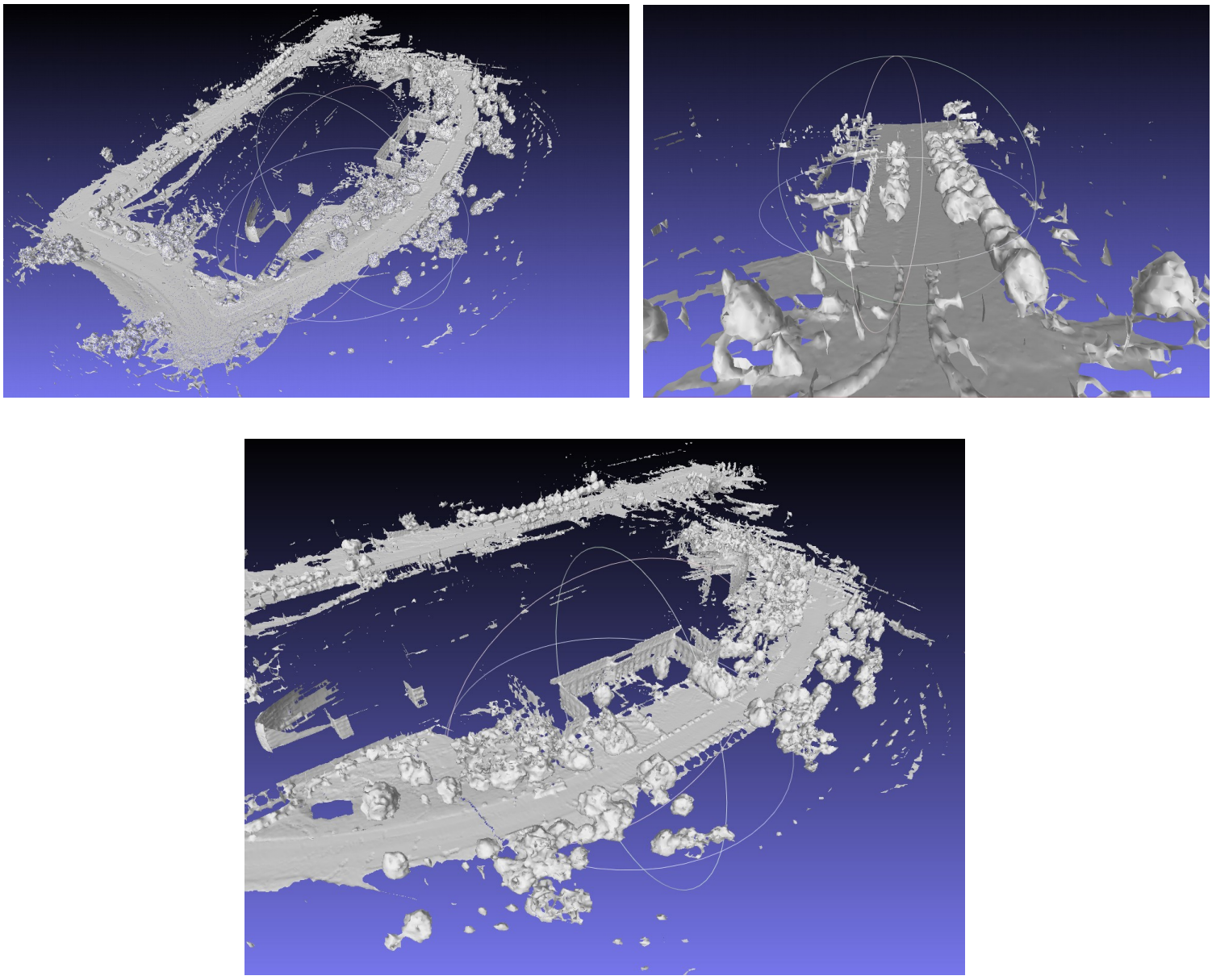

Fig 4. Lidar Mesh of COSI

Reference:

[1] A. Dosovitskiy, R. German, F. Codevilla, A. Lopez and V. Koltun, "CARLA: An open urban driving simulator," arXiv preprint arXiv:1711.03938.

[2] "Google Earth Studio," [Online]. Available: https://www.google.com/earth/studio/.

[3] B. Wu, A. Wan, X. Yue and K. Keutzer, "Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud.," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[4] R. B. Rusu and S. Cousins, "3d is here: Point cloud libary (pcl)," in *IEEE International conference on robotics and automation*, 2011.

[5] "Blender," [Online]. Available: https://www.blender.org/.

[6] A. Vision, "Meshrooom: A 3D reconstruction software.," [Online]. Available: https://github.com/alicevision/meshroom.

[7] S. Kato, T. Eijiro, I. Yoshio, N. Yoshiki, K. Takeda and T. Hamada, "An open approach to autonomous vehicles," *IEEE Micro,* vol. 35, no. 6, pp. 60-68, Nov.-Dec. 2015.

[8] P. Biber and W. Strasser, "The normal distributions transform: a new approach to laser scan matching," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)* , 2003.

[9] P. J. Besl and N. D. Mckay, "A method for registration of 3-D shapes," *IEEE Transactions on Patter Analysis and Machine Intelligence,* vol. 14, no. 2, pp. 239-256, 1992.

[10] R. Coulter, "Implementation of the pure pursuit path tracking algorithm," (No. CMU-RI-TR-92-01). Carnegie-Mellon UNIV Pittsburgh PA Robotics INST..

[11] M. Quigley, C. Ken, B. Gerkey, J. Faust, T. Foot, J. Leibs, R. Wheeler and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, 2009.

[12] "RoadRunner," [Online]. Available: https://www.vectorzero.io/roadrunner.