An Automatic Digital Terrain Generation Technique for Terrestrial Sensing and Virtual Reality Applications

Lee Easson, Alireza Tavakkoli(✉), and Jonathan Greenberg

University of Nevada, Reno, NV 89557, USA
leasson@nevada.unr.edu, {tavakkol,jgreenberg}@unr.edu

Abstract. The identification and modeling of the terrain from point cloud data is an important component of Terrestrial Remote Sensing (TRS) applications. The main focus in terrain modeling is capturing details of complex geological features of landforms. Traditional terrain modeling approaches rely on the user to exert control over terrain features. However, relying on the user input to manually develop the digital terrain becomes intractable when considering the amount of data generated by new remote sensing systems capable of producing massive aerial and ground-based point clouds from scanned environments. This article provides a novel terrain modeling technique capable of automatically generating accurate and physically realistic Digital Terrain Models (DTM) from a variety of point cloud data. The proposed method runs efficiently on large-scale point cloud data with real-time performance over large segments of terrestrial landforms. Moreover, generated digital models are designed to effectively render within a Virtual Reality (VR) environment in real time. The paper concludes with an in–depth discussion of possible research directions and outstanding technical and scientific challenges to improve the proposed approach.

Keywords: Digital Terrain Model · Terrestrial Remote Sensing · Geological Landmass Modeling

1 Introduction

Terrains are among the most fundamental features in any virtual application simulating a landmass, ranging from computer games to geological simulations. For example, in an open-world massively multiplayer online role playing game, large-scale natural environments maybe designed for players to explore, where a vast terrain is usually the first part of the authoring pipeline to be subsequently augmented with props that represent rocks, trees, plants and buildings. On the other hand, real-world terrain are usually more complex and varied and may include plains, mountain ranges, and eroded valleys in a single environment. Terrain formation is a combination of long-term and complex geological events with complicated physical and geological interactions amongst different

G. Bebis et al. (Eds.): ISVC 2019, LNCS 11844, pp. 619–630, 2019.
https://doi.org/10.1007/978-3-030-33720-9_48

components comprising the landmass. In addition, different geological features are dominant at difference range scales. These complexities contribute to many unsolved challenges in terrain modeling.

One definition of Digital Terrain Models (DTM), [3], relates to geometrical aspects of the 3D environment acquired from Laser scanning and is a continuous function mapping a 2D position (x, y) to the terrain elevation $z = f(x, y)$. In this definition, the terrain is defined as the boundary between the ground and the air. Yet, there are certain geographical feature, such as overhangs [12], ground vegetation [10], and large man-made structures, that may render the assumptions required for this definition inaccurate [14].

The aforementioned DTM will require utilizing a large amount of data collected by aerial or ground-based Laser scanning technology. This data is generally combined to produce a collection of points referred to as Point Clouds (PC). In essence, each point in a point cloud represented a location in the world from which the light emitted from the scanner is reflected back. The massive amount of data within even a small scanned region makes it necessary to represent the DTM using a more efficient structure.

Several data structures are utilized in the literature that represent DTMs with varied levels of performance [14]. These structures range from pixel-level representation of the elevation data by quantizing the 2D planimetric locations of the point clouds to a hybrid approach by interpolating points on the surface of a grid-mesh structure [1]. To improve the quality of the structure of the DTM, and with the popularity of triangulation techniques in computer graphics, several Triangulated Irregular Networks (TINs) are proposed with the goal of improving storage efficiency of the point cloud representation of the DTM [2], with recent attempts to improve the performance of the triangulation approaches [8,9,11,19]. Most of these methods assume that the terrain is smooth and continuous with a large height difference between neighbouring points on ground and non-ground objects. Therefore, the performance of these methods often decreases through wrongly filtering hilly regions and large buildings.

Because of the simplicity and ease of implementation, morphology-based methods [5–7,13] are mostly used in ground filtering. However, finding the correct structuring element size is a problem in these methods. While a small structuring element is needed for filtering points on vegetation, tree, and cars, a large structuring element should be used for filtering points on buildings.

In this paper we propose an fast ground filtering approach with an efficient DTM representation capable of preserving detailed geological features and applicable to both urban and non-urban landmasses. Unlike most other methods that try to extract ground points via many iterations for DTM generation, the proposed technique extracts all ground points via a series of atomic operations geared towards preserving geological features and eliminating non-ground points. The main hypothesis in the proposed method is that non-ground objects produce sharp variations in elevation within a spacial neighborhood. Hence, we propose using region growing for segmenting non-ground objects. The method is tested on a number of point cloud data sets obtained the United States Geological Survey. The proposed method is also compared with the existing methods.

2 The Proposed Approach

Figure 1 shows an overview of the proposed Point Cloud Filtering and DTM generation pipeline. The proposed architecture is comprised of three components, i.e., preprocessing stage, map generation module, and terrain generation module, shown as the vertical tracks. These components in turn process three different data structures in the form of Point Clouds, Heightmaps, and Landscape mesh.

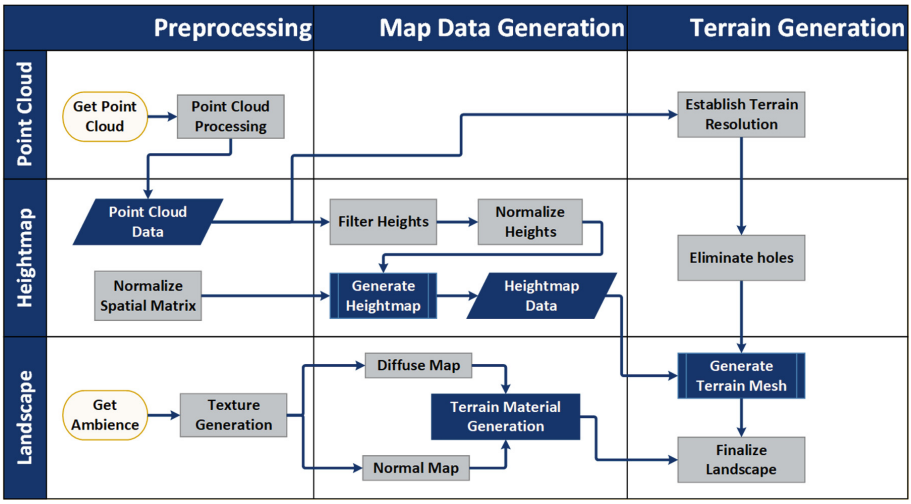


Fig. 1. The proposed processing pipeline.

The first stage of the proposed pipeline is the preprocessing step. In this stage the Lidar point cloud data is processed to represent a gridded topological form. To accomplish this task, we perform the nearest neighbor interpolation in conjunction with a kernel-based statistical outlier removal to generate the raster grid from the Lidar point cloud. In this step, the three data structures representing the point cloud data, the spatial matrix of the map data, and the landscape texture and material data will be established and ready for processing. The texture data is extracted from the point clouds photometric information, if this information is available. The photometric information will be used to produce a diffuse map as well as a normal map for the terrain materials.

The second stage in the pipeline is responsible for generating the heightmap data representing a topological formation of the terrain as well as shader models for rendering a physically realistic view of the material applied on the surface of the terrain. In this stage, the topological spatial matrix is processed in order to accomplish two tasks. First, the overall topological and geological statistics of the terrain is learned by employing Singular Value Decomposition (SVD). Second, the learned statistics of the overall terrain topology is combined with the first

and second order statistics of the point cloud data to eliminate the non-terrain objects while preserving details of the geological features of the terrain.

The last step in the pipeline is the terrain generation stage. In this stage, the resolution of the final landscape is calculated from the overall point cloud data. This information is then used to fill the holes introduced in the topological heightmap as a result of non-terrain object segmentation. Once the overall heightmap of the terrain is established, the terrain mesh generation process will generate an efficient digital mesh model for the terrain represented at various Level of Detail (LoD) information. At this stage the shader models for the terrain materials are also computed and applied to render the terrain.

2.1 The Preprocessing Step

The preprocessing stage of the proposed pipeline, shown in Algorithm 1, is responsible for initializing the gridded and rasterized data structures for the Terrain heightmaps, texture maps, and shader materials.

Algorithm 1: Preprocessing Stage of the DTM Generation Pipeline.

Data: P : Input-Point Cloud. // point $P_i = (x_i, y_i, z_i, r_i, g_i, b_i)$
Result:
L : Landscape Point Cloud Data File.
T : Landscape Texture.
M : Landscape Layered Material.
begin
  $L(x, y) \leftarrow$ Stat_Outlier(P, th) // Eq. (1)
  for *all* P_i **do**
    Find $L_l(x, y, z)$ lowest and $L_h(x, y, z)$ highest Lidar Returns Eq. (2)
  Set texture Coordinates: **begin**
    $T(u, v) = new\_Texture(u, v)$ // coordinate map from Eq. (3)
  Set Shader Material: **begin**
    $M \leftarrow new\_Material(Diff(u, v), Norm(u, v))$

Statistical Outlier Removal: The first step in cleaning out the input Lidar point-cloud data is to eliminate outliers. Outliers include points introduced to the point cloud due to noise or small moving objects, such as airplanes, located at drastically different heights than the terrain, need to be eliminated. In order to perform this task, we first build a non-parametric density estimation of the point cloud in a local spatial neighborhood [17].

Assuming an outlier threshold of th, we eliminate points from the point cloud data whose probability of belonging to the known distribution from which the

Lidar data is generated falls below th. This probability is calculated using the non-parametric kernel density estimation, below:

$$P(z_i|inlier) = \frac{1}{|N_k(z_i)|} \sum_{z_j \in N_k(z_i)} \frac{1}{\sigma\sqrt{(2\pi)}} exp\left(-\frac{\sigma(p_j, p_i)}{2h^2}\right) \tag{1}$$

where z_i is the height value of the ith point p_i in the point cloud data p_j at a spatial neighborhood location of $N_k \in \mathbb{R}^2$.

Top and Bottom Returns: Once the statistical outliers are eliminated, we will need to determine the most likely ground points. In order to accomplish this task, we will set two rasterized data structures for the lowest return and the highest return points at a location (x, y) denoted as $L_l(x, y, z)$ and $L_h(x, y, z)$:

$$\forall P_i \qquad \begin{bmatrix} L_l(x, y, z) \\ L_h(x, y, z) \end{bmatrix} = \begin{bmatrix} \min\limits_{z_i}(P_i : x = x_i \& y = y_i) \\ \max\limits_{z_i}(P_i : x = x_i \& y = y_i) \end{bmatrix} \tag{2}$$

where $x \in [min(x_i), max(x_i)]$ and $y \in [min(y_i), max(y_i)]$.

Shader and Texture Initialization: In order to render physically realistic materials on the surface of the final DTM, we will establish the data structures $T(u, v)$ and M as the texture map and the landscape material, respectively. First, a mapping between the spatial domain of the point cloud $(x, y) \in \mathbb{R}^2$ and the texture-coordinates (u, v) is determined:

$$(u, v)^T = \begin{bmatrix} \Phi_u : (x_{min}, x_{max}) \rightarrow (0, 1) \\ \Phi_v : (y_{min}, y_{max}) \rightarrow (0, 1) \end{bmatrix} \tag{3}$$

Next, the shader material for the landscape is initialized based on the photogrametric information, if this information is included in the Lidar point cloud data. Suppose for each point P_i in the point cloud data, the photogrametric information is given in the form of $C_i = (r_i, g_i, b_i)$ color components. The details about the computation of the diffuse and normal channels of the landscape material are discussed later in the paper in Sect. 2.3.

2.2 The Heightmap Generation Step

Once the point cloud data is refined during the pre-processing step, it is passed through the heightmap generation stage of the algorithm to produce a two-dimensional structure maintaining the overall height associated with the terrain surface. This heightmap object is then utilized to generate a three-dimensional model of the terrain surface as a 3D mesh object. This section discusses the process of generating the terrain heightmap by removing non-terrain objects while preserving significant geological features.

Algorithm 2 shows the overall pipeline of generating the terrain surface heightmap. The process starts by taking the top and bottom point cloud data

Algorithm 2: Heightmap Generation Stage of the DTM Pipeline.

Data:
L_l, L_h : Landscape Top and Bottom Point Cloud Data.
P : Point Cloud Data.
Result:
H : Landscape Heightmap Data File.
begin
  for *all* P_i **do**
    $\hat{L} = L_l \cap L_h$ // Non-ground overhangs
    $L = (L_l \cup L_h) - \hat{L}$ // Potential ground points
    $\hat{H} \leftarrow L.Heights$ // Eq. (4)
    Find $\hat{g} \leftarrow$ S.V.D. $(\hat{H})$ // Eq. (7)
    $g \leftarrow$ **inPaint**$(\hat{g})$ // Fill holes

structures generated from the pre-processing phase to determine the potential ground points and eliminate the over-hangs. Then a polynomial function with sufficient local variance and smooth global consistency is fit onto the data to estimate the overall structure of the terrain ground. This is utilized to computer the ground heightmap values for each point in the landmass.

Terrain Height Estimation. With the Lowest L_l and the highest L_h LiDar returns from the point cloud data, we start modeling the heightmap of the terrain. Each point (x, y, z) in a point cloud belongs to one of two classes, i.e., the ground or the non-ground objects. Both L_l and L_h are quantized in such a way as to represent 2D grids ranging from (x_{min}, y_{min}) to (x_{max}, y_{max}).

It is trivial to eliminate overhangs (or points covering the ground area) if both the ground position and the overhang points are visible within the point cloud data. Points within a spatial location $\mathcal{R}(x, y)$ are considered to belong to the non-ground object covering the surface of the ground if they exist in both L_l and L_h structures. Therefore, the first iteration of the heightmap is generate by interpolating the height values of all points in L_l that do not belong to L_h as:

$$\hat{H} = h(x, y) = \frac{1}{Size(\mathcal{R})} \sum_{(x,y,z)\in\mathcal{R}} \{z|(x, y, z) \in L\} \tag{4}$$

The height of the ground in a landmass may be considered as a low-degree polynomial with the non-ground objects, e.g. shrubbery, biomass, and man-made structures, disrupting the natural curvature and geological features of the terrain. Therefore, we postulate that the heightmap of the terrain is a combination of a ground function $g(x, y)$ and an anomalous function $\mathcal{N}(x, y)$:

$$h(x, y) = g(x, y) + \mathcal{N}(x, y) \tag{5}$$

where h is a heightmap calculated from raw point cloud data (Fig. 2(a)), g is a low-order polynomial function with high degrees of smoothness over a large

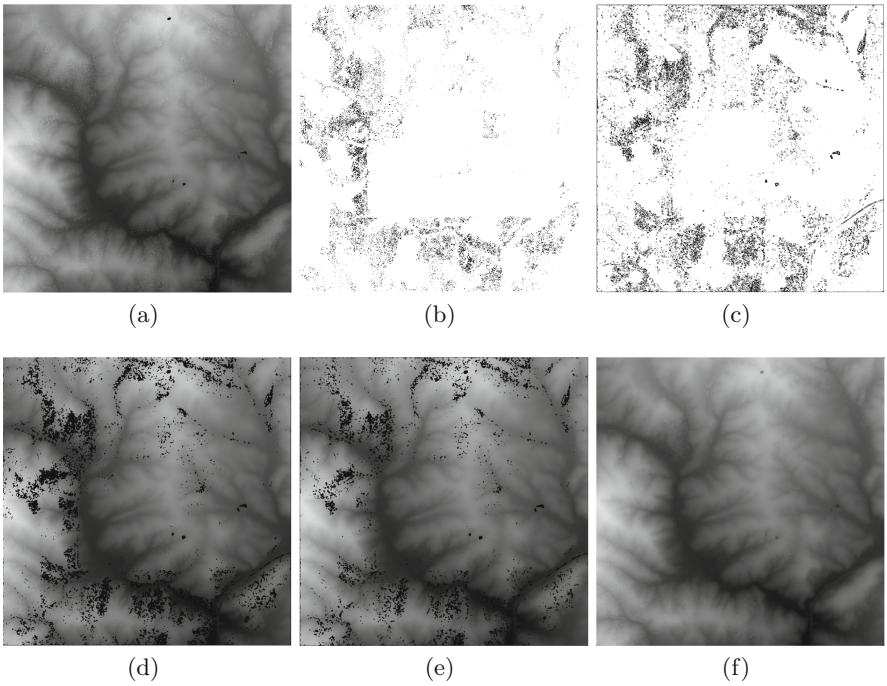


Fig. 2. Terrain modeling heightmap generation step performed on Idaho dataset. (a) Original data with no-terrain elements. (b) Dark areas are non-terrain LiDar returns. (c) Dark areas are non-terrain geological features. (d) Non-terrain geological features are removed. (e) Non-terrain areas are removed. (f) Final terrain heightmap.

spatial area representing the ground heightmap (Fig. 2(f)), and $\mathcal{N}$ represents the non-ground geological and man-made features shown in Fig. 2(b) and (c).

This formulation represents the terrain heightmap modeling as a novelty detection question [15]. Therefore, we represent the ground region of the terrain heightmap data g as a polynomial with degree N of the following form:

$$g(x, y) = \sum_{i,j=0}^{N} a_i b_j (x^i \cdot y^j) \tag{6}$$

Using the above formulation, and given the heightmap from Eq. (4), we need to solve the linear system of equations resulting from all (x, y) values of $\hat{H}$ as follows:

$$[h(x_1, y_1) \cdots h(x_N, y_N)]^T = \begin{bmatrix} a_0 b_0 & a_1 b_0 & \cdots & a_N b_N \\ \vdots & \vdots & \ddots & \vdots \\ a_0 b_0 & a_1 b_0 & \cdots & a_N b_N \end{bmatrix} \begin{bmatrix} 1 & x & y & \cdots & x^N y^N \end{bmatrix} \tag{7}$$

This terrain function may be visualized as the combination of Fig. 2(d) and (e), in which the darker areas represent non-ground objects encoded as $\mathcal{N}$. These

dark areas produce holes in the ground heightmap and are filled using an automatic inpainting algorithm similar to [18]. The final terrain heightmap is shown in Fig. 2(f).

(a)

(b)

Fig. 3. Terrain meshes: (a) Terrain mesh from the original point cloud. (b) Terrain mesh with the proposed heightmap generation technique.

2.3 The Terrain Modeling Step

Digital Terrain Models are employed in a number of applications ranging from geographical analysis, biomass and environmental studies, etc. In order for a DTM to be useful for its intended application, it must be generated in such an efficient manner as to allow for realistic rendering, interactivity, and efficient manipulation. To this end, we propose the use of the Unreal Engine 4's Landscapes [16]. Algorithm 3 provides an overview of this stage of the pipeline responsible for generating the 3D mesh of the terrains as well as shader materials employed for physically realistically rendering of the terrain (Fig. 3).

Terrain Mesh Modeling: The 3D mesh representing the ground surface is generated by applying a polygonal mesh based on the heightmap generated from the previous step. In this stage of the algorithm, a 2D grid is generated for each pair of (x, y) coordinates associated with pixels in the heightmap.

Terrain Texture Modeling: With the mapping between the heightmap data and the point cloud data established, an interpolation technique is used to sample

Algorithm 3: Mesh Generation and Shader Programming of the DTM Pipeline.

Data:
P : Point Cloud Data
H : Heightmap
Result:
M : Terrain Mesh. H : Terrain Heightmap. T : Terrain Texture. Mat : Shader Material.
begin
  Mesh.Vert: M.(vertex.x,vertex.y)$\xleftarrow{\Phi}$ $(H.x, H.y)$
  $M \leftarrow$ Interpolate$[h(\text{Grid}(x, y))]$
  for *all* P_i **do**
    $T \leftarrow$PC2Texture(P, H) // Generate Texture from Point Cloud
  Calculate Terrain Extent
  $(U, V) \leftarrow$ Texture Coordinate Mapping
  $Mat \leftarrow (Diffuse_{uv}, Normal_{uv})$ // Shader Program

(a) (b) (c)

Fig. 4. Terrain texture: (a) Original sparse texture. (b) Dense texture. (c) Modified dense texture.

the color (or intensity) values from point clouds in a neighborhood that map onto the terrain mesh object. Figure 4 shows the results of the interpolation steps taken to generate a photorealistic texture for the terrain from the Point Cloud data.

Terrain Material Modeling: The material applied to the surface of the terrain mesh is comprised of two main channels, a diffuse channel and a normal channel. The diffuse channel of the material utilizes the texture coordinates to map the color (or intensity) values of the terrain texture on the surface of the 3D terrain mesh. The normal channel is computing using a normal map generation technique [4].

Figure 5 shows the final rendering of the Digital Terrain Model with the material applied. As it can be seen, the quality of the rendering is quite realistic. Note the various geological features preserved, while the man-made structures or

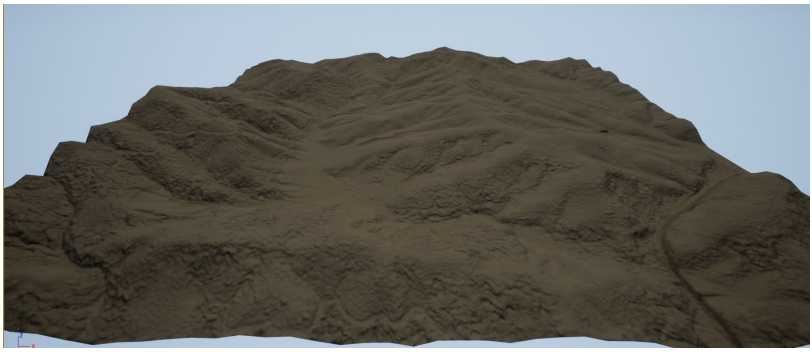
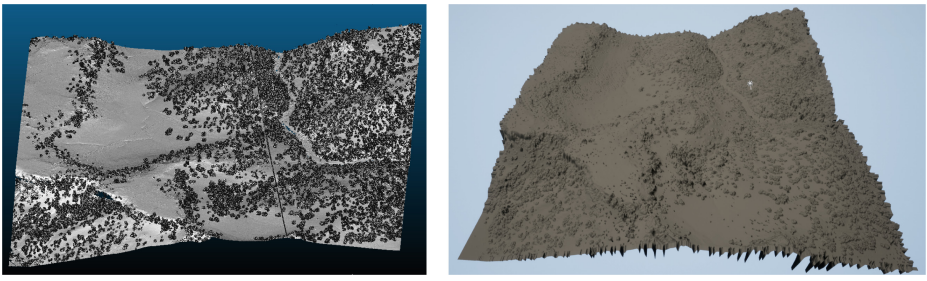


Fig. 5. Final terrain model with shader parameters applied.

non-ground objects are effectively removed from the terrain model. The texture applied on the surface of the terrain in the form of a physically-based material drastically enhances the visualization of the DTM.

3 Experimental Results

This section presents the results of the proposed DTM technique performed on a variety of point cloud data from the USGS datasets. The first set of results (Fig. 6) demonstrates that quality of the modeled DTM compared to the rendering of the point cloud data. The point cloud data rendered in the Cloud Compare software is shown in Fig. 6(a). The main issue is the lack of discrimination between points belonging to the ground surface and other structural elements. The proposed technique has the ability to eliminate the non-terrain elements while preserving significant geological features as evident from Fig. 6(b).



(a) (b)

Fig. 6. The results of the proposed framework. (a) The original point cloud data rendered in Cloud Compare software. (b) The 3D landscape DTM generated by the proposed framework and rendered in Unreal Engine 4.

Figure 7 shows the generated heightmaps (Fig. 7(a)) and the 3D landscape mesh associated with each heightmap (Fig. 7(b)). As seen from the figures, the proposed DTM mesh objects represent the geological features quite accurately.

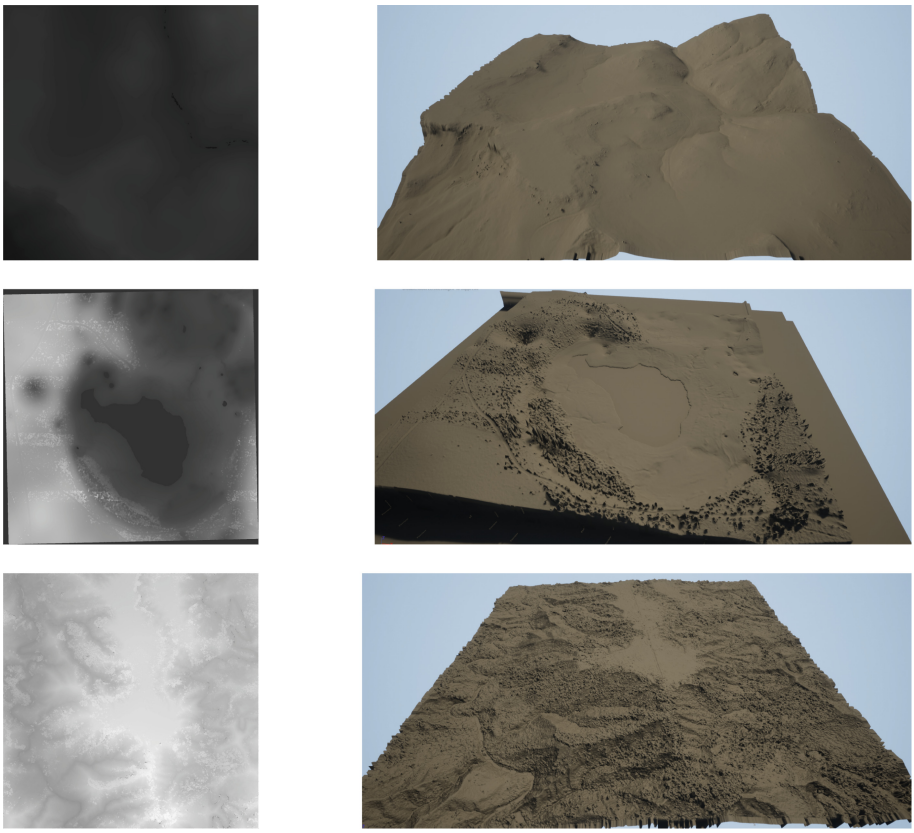


Fig. 7. Heightmaps (a) and their associated terrains (b) generated by the proposed framework. From top: California Calaveras-Tuolumne (CA), Washington County (FL), and Oahu (HI), respectively.

4 Conclusions and Future Work

In this paper we proposed a pipeline for generating Digital Terrain Models (DTM) from a variety of Lidar-based point cloud datasets. The proposed pipeline automatically generates heightmaps by eliminating non-ground points from the point cloud and interpolating the surface height values from the remaining points. The texture and materials are also created to provide photorealistic rendering of the terrain 3D mesh. There are a number of future directions to this work. Performing semantic segmentation on the 3D point cloud data may add higher level information to the data useful for effective generation of heightmaps.

References

1. Ackermann, F.E., Kraus, K.: Grid based digital terrain models. na (2004)
2. Axelsson, P.: Dem generation from laser scanner data using adaptive tin models. Int. Arch. Photogrammetry Remote Sens. **33**(4), 110–117 (2000)
3. El-Sheimy, N., Valeo, C., Habib, A.: Digital Terrain Modeling: Acquisition, Manipulation and Applications (Artech House Remote Sensing Library). Artech House, Norwood (2005)
4. Gimp: Normal map plugin (2019). https://code.google.com/archive/p/gimp-normalmap/
5. Kobler, A., Pfeifer, N., Ogrinc, P., Todorovski, L., Oštir, K., Džeroski, S.: Repetitive interpolation: a robust algorithm for DTM generation from aerial laser scanner data in forested terrain. Remote Sens. Environ. **108**(1), 9–23 (2007)
6. Li, Y., Yong, B., Wu, H., An, R., Xu, H.: An improved top-hat filter with sloped brim for extracting ground points from airborne lidar point clouds. Remote Sens. **6**(12), 12885–12908 (2014)
7. Mongus, D., Lukač, N., Žalik, B.: Ground and building extraction from lidar data based on differential morphological profiles and locally fitted surfaces. ISPRS J. Photogrammetry Remote Sens. **93**, 145–156 (2014)
8. Mongus, D., Žalik, B.: Parameter-free ground filtering of lidar data forautomatic DTM generation. ISPRS J. Photogrammetry Remote Sens. **67**, 1–12 (2012)
9. Mongus, D., Žalik, B.: Computationally efficient method for the generation of a digital terrain model from airborne lidar data using connected operators. IEEE J. Sel. Top. Appl. Earth Observations Remote Sens. **7**(1), 340–351 (2013)
10. Næsset, E.: Vertical height errors in digital terrain models derived from airborne laser scanner data in a boreal-alpine ecotone in Norway. Remote Sens. **7**(4), 4702–4725 (2015)
11. Özcan, A.H., Ünsalan, C.: Lidar data filtering and dtm generation using empirical mode decomposition. IEEE J. Sel. Top. Appl. Earth Observations Remote Sens. **10**(1), 360–371 (2016)
12. Pfeifer, N.: A subdivision algorithm for smooth 3D terrain models. ISPRS J. Photogrammetry Remote Sens. **59**(3), 115–127 (2005)
13. Pingel, T.J., Clarke, K.C., McBride, W.A.: An improved simple morphological filter for the terrain classification of airborne lidar data. ISPRS J. Photogrammetry Remote Sens. **77**, 21–30 (2013)
14. Shan, J., Toth, C.K.: Topographic Laser Ranging and Scanning: Principles and Processing. CRC Press, Boca Raton (2018)
15. Tavakkoli, A.: Novelty detection: an approach to foreground detection in videos. In: Pattern Recognition. IntechOpen (2009)
16. Tavakkoli, A.: Game Development and Simulation with Unreal Technology, 2nd edn. AK Peters/CRC Press, Boca Raton (2018)
17. Tavakkoli, A., Nicolescu, M., Bebis, G.: Automatic robust background modeling using multivariate non-parametric Kernel density estimation for visual surveillance. In: Bebis, G., Boyle, R., Koracin, D., Parvin, B. (eds.) ISVC 2005. LNCS, vol. 3804, pp. 363–370. Springer, Heidelberg (2005). https://doi.org/10.1007/11595755_44
18. Van Sinh, N., Ha, T.M., Thanh, N.T.: Filling holes on the surface of 3D point clouds based on tangent plane of hole boundary points. In: Proceedings of the 7th Symposium on Information and Communication Technology, pp. 331–338 (2016)
19. Zhang, J., Lin, X.: Filtering airborne lidar data by embedding smoothness-constrained segmentation in progressive tin densification. ISPRS J. Photogrammetry Remote Sens. **81**, 44–59 (2013)