Automatic Data Driven Vegetation Modeling for Lidar Simulation

Jean-Emmanuel Deschaud, David Prasser, M. Freddie Dias, Brett Browning, Peter Rander

***Abstract*— Traditional lidar simulations render surface models to generate simulated range data. For objects with well-defined surfaces, this approach works well, and traditional 3D scene reconstruction algorithms can be employed to automatically generate the surface models. This approach breaks down, though, for many trees, tall grasses, and other objects with fine-scale geometry: surface models do not easily represent the geometry, and automated reconstruction from real data is difficult. In this paper, we introduce a new stochastic volumetric model that better captures the complexities of real lidar data of vegetation and is far better suited for automatic modeling of scenes from field collected lidar data. We also introduce several methods for automatic modeling and for simulating lidar data utilizing the new model. To measure the performance of the stochastic simulation we use histogram comparison metrics to quantify the differences between data produced by the real and simulated lidar. We evaluate our approach on a range of real world datasets and show improved fidelity for simulating geo-specific outdoor, vegetation scenes.**

I. INTRODUCTION

Lidar simulators are regularly used to test UGV autonomy systems, typically using surface models (e.g., triangle meshes) to represent objects in the scene. The simulators use ray tracing, or z-buffering, to approximate the ranging operation of the lidar: determining the range from the simulated sensor to the nearest object (e.g. [1], [2]). Measurement errors are usually modeled as additive noise (e.g., [3]).

The fidelity of this modeling and simulation approach depends greatly on the environment. Indoor, underground, and urban environments often work well because they primarily contain relatively large, solid objects that are well approximated by a polygon mesh with large elements. In contrast, off-road terrain is a greater challenge because it often contains large amounts of vegetation in which the widths of objects (e.g., small tree branches, leaves, and grasses) are often on scale or smaller than the beam width of the lidar (typically $10mm$). This situation is one example of the well known mixed pixel problem [4]: there is more than one "correct" range value because the small geometries reflect only a portion of the beam, allowing the the beam to propagate further and generate additional returns.

This work was sponsored by U.S. Army Engineer Research and Development Center (ERDC) under cooperative agreement "Fundamental Challenges in World and Sensor Modeling for UGV Simulation" (Number W912HZ-09-2-0023). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

All work was done at the National Robotics Engineering Center, Carnegie Mellon. Deschaud is with the Mines ParisTech, CAOR - Robotics Center, Mathematics and System Department, 60 Bd Saint-Michel 75272 Paris, jean-emmanuel.deschaud@mines-paristech.fr.
All other authors are with the National Robotics Engineering Center and Robotics Institute, Carnegie Mellon University, Pittsburgh PA, USA, {dprasser,mfdias,brettb,rander}@rec.ri.cmu.edu.

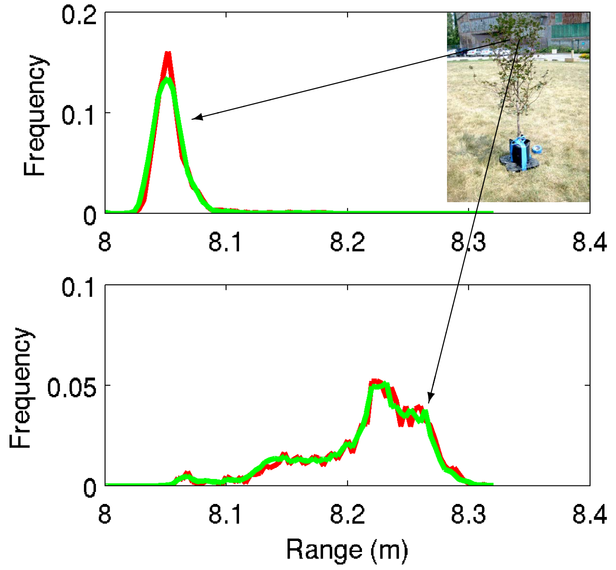

Fig. 1. Normalized histograms of lidar range measurements on two different points of a tree. Approximately 1500 range returns were used to generate each red curve. The histograms in green are produced by our simulation using the whole data to build the volumetric model.

To demonstrate this behavior, we collected more than one thousand range scans of a small tree, shown in Fig. 1 (inset), from a stationary lidar. The red curve in the top graph shows the normalized histogram of returns from a beam repeatedly hitting a relatively small portion of the tree of similar size to the laser spot. The curve shows a single clear peak well approximated by a single surface with additive, approximately Gaussian, noise. The lower graph on the other hand, shows data for a nearby beam angle of the same laser and exhibits dramatically different behavior: it appears multi-modal and asymmetric. Simple additive noise is poorly suited to capturing such large variation in range distributions.

This problem can be overcome by using multi-sample raytracing methods to model the non-zero laser beam width and by building geometric models at the scale of the blades of grass [3]. This generates more realistic behavior in mixed-pixel scenarios and automated tools (e.g., http://www.bionatics.com/) can even simplify the construction process. Of course the models grow dramatically – what might have been just a few polygons in a solid-ground simulation could become millions of polygons per cubic meter – but that may be acceptable if real-time simulation is not needed. The greater challenge is the effort required to make the virtual model match the real world. Many semi-automated techniques can reconstruct large-scale models of building exteriors, roads, and other larger solid objects but these techniques perform poorly when attempting to reconstruct sub-pixel geometries of something as mundane – and common – as knee-high grass. (See results at the end of this paper.)

We propose an alternative model that efficiently accounts for the variability of the scene and which can be automatically estimated from measured lidar data. Our model captures the statistical distribution of the lidar range directly into the model without recovering surfaces. Range noise and mixed pixel effects are both built into the model without need to explicitly recover them, either. This model is substantially simpler to estimate and lends itself to straightforward simulation using ray tracing. The green curves of Fig. 1 are histograms of 150,000 simulated range returns striking the two same parts of the tree used to generate the red histograms, demonstrating that our approach can reproduce the statistics of the real data to a high degree of accuracy.

Section II reviews related efforts in the lidar simulation area. We then describe our model in section III and the corresponding simulation in section IV. Section V shows the experimental system used to acquire data for testing, while section VI investigates the performance of the various models. Section VII concludes the paper and lays out possible directions for future work.

II. Related Work

Lidar range sensing is a topic that has been studied in a number of different areas including robotics, computer vision, graphics, and remote sensing. Lidar has been most heavily studied in the field of remote sensing, with extensive studies on atmospheric propagation, scattering phenomenon, and interaction with vegetation, primarily driven by its usefulness for aerial measurements in remote sensing (e.g. [5]).

Simulating a pulsed, time-of-flight lidar requires modeling the physical generation and propagation of the light pulse out of the sensor and through the atmosphere, the object/terrain interaction leading to reflection or back scatter, the return journey back into the sensor, and the physical mechanism for sampling and reporting the detected range. At every step of this process, un-modeled or poorly calibrated deviations can occur that lead to errors in sensed range. [6] have studied the physical sensor errors, which can be simulated as in [7]. The physics of atmospheric propagation, absorption, and scattering are reasonably well understood [8], and empirical data for some robot sensors exists [9], [10].

In this paper, we focus on the interaction of lidar with the surfaces in the terrain. Ignoring atmospheric effects, for surfaces with approximately Lambertian reflectance and with surface geometries much larger than the lidar beam width, the range error is dominated by limits of the physical sensor and are small and approximately Gaussian. As a result, representing the surface as a triangular mesh and performing ray tracing (or z-buffering), with optional additive Gaussian noise, produces a reasonable approximation of reality. This approach is commonly used in mainstream robot simulation environments [1], [2], [11], and performs well for materials such as bare soil, dry cement, brick walls, and so on. Straight forward extensions to the ray tracing approach can model highly specular (e.g. quartz, metal) or transparent (e.g. water, glass) surfaces. However, the approach fails to capture the intricate behavior of vegetation.

Lidar interaction with vegetation is complicated due to the complex, multi-path reflection, absorption, and transmission that happens within plant cells (e.g. in the leaf cell structure [5]). Furthermore, the dense geometries of plant surfaces create additional complex multi-path reflections. This latter effect can be seen on any surface boundary, as shown by Tuley *et al.* [4] where they demonstrated that lidar beams striking two or more surfaces at different depths can produce one or more returns that in some cases may not correspond to the range of either surface, an effect they called "mixed pixel returns". For vegetation, with its complicated, small surface geometries that are on the scale of lidar beam width, mixed pixel returns can dominate causing non-trivial range distributions as shown in Fig. 1.

Simple approaches to compensating for this behavior, such as modifying the additive noise model, do not work as they do not correctly capture the angular dependence of the returns. Using fine grained triangular meshes combined with dithered multi-ray, ray traced simulations can approximate the behavior [3] but at substantial computational cost. Consider for example, the amount of memory and computing resources required to simulate 100 km^2 of wild grassland.

A second limitation is the generation of the model. For generic, non-geo-specific models, one can use approaches such as the Lindenmayer-systems (L-systems) [12] to generate realistic looking vegetation. However, while there has been work to estimate a tree's large scale structure from lidar data [13], there are no techniques that would allow one to tune an L-system virtual tree to match real data from a specific location (i.e. a geo-specific scene). Secondly, approaches like L-systems do not approach the diversity of leaf structure found in real terrains and cannot capture the complex reflectance properties of real leaves which are often laser wavelength dependent.

In this work, we attempt to fill this niche by learning models for lidar simulation from collected data in a terrain that capture the complexities of vegetation and yet still represent "simple" surfaces well.

III. Volumetric Modeling

In this section, we present the volumetric modeling concept and model estimation techniques.

A. Key Concept

Capturing intricate behavior in a scalable way is challenging. Rather than using a non-scalable approach such as very fine surface modeling, we instead approach this problem by aiming to capture the *statistical* behavior of lidar-world interaction. Thus, the question of modeling changes from how to represent the surfaces in the world, to how to

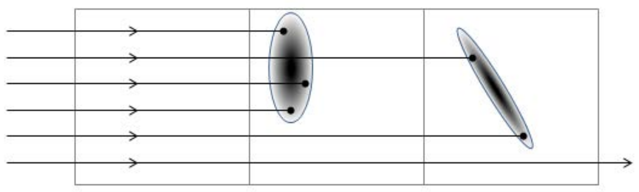

Fig. 2. Two dimensional example of permeability. Of the 4 rays that enter the left ellipse, 3 terminate, and 1 passes through. The left ellipse therefore has permeability of 1/4. The right ellipse has no rays passing through, so it has 0 permeability.

efficiently and accurately capture the statistical properties of lidar-world interaction.

One conceivable approach is to use surfaces and augment them with a stochastic penetration depth model. That is, when lidar strikes a surface, the resulting sensed range is augmented by an additional depth randomly sampled from an appropriate distribution to produce results like that shown in fig. 1. The difficulty with this approach is that for real vegetation, the distribution model is far from simple and, as our later results show, the distribution is dependent upon spatial position, and the incident angle of the ray. Estimating such a complex model would be very difficult.

Instead, we use a volumetric approach which introduces two key ideas. The first is to break space into small volumes, each with its own range return distribution. In this way, we can capture both spatial and incident angle variations. Secondly, we introduce the notion of *permeability* to capture the ability of lidar beams to potentially travel significant distance through material before striking a surface. That is, permeability separates the size of volumes from the maximum penetration depth of lidar beams.

Fig. 2 shows a $2D$ example of this concept with boxes representing volumes, and the ellipsoids representing the range distributions (no ellipsoid means it is free space). Permeability encodes the ability of the lidar ray to pass through each volume. The ellipsoid on the left is formed by 3 points, but one ray passes through without striking, so it is partially permeable. The ellipsoid on the right has no rays passing through it, so it is not permeable.

B. Model Representation

To represent the model, we must decide upon how to partition occupied space into volumes, how to represent range returns, and how to model permeability.

There are many ways to partition occupied space, including tetrahedral meshes (simplicial complexes), voxels, and so on. All of these can be regular in size and axis aligned, or variable. In this work, we focus on regular, axis aligned voxels due to their speed and efficiency. We have also considered tetrahedral volumes, but these offer comparable performance and so are not discussed further here.

To represent range returns, we take the simple approach of representing point cloud distributions inside each volume with a $3D$ Gaussian distribution. That is, the probability of a point occurring at a location $\boldsymbol{q} \in \Re^3$ inside the volume i is simply $\boldsymbol{q} \sim N(\boldsymbol{\mu_i}, \Sigma_i)$, where $N(.)$ is the normal distribution with mean $\boldsymbol{\mu}_i \in \Re^3$ and covariance $\Sigma_i \in S^3$. Finally, we represent permeability simply with a Bernoulli distribution. Hence, the probability of a ray passing through a volume i is given by $\rho \in [0, 1]$. We discuss how to sample from such a model in section IV.

C. Model Estimation

To estimate the model from data, we assume that we have a set of lidar returns with sensor pose information that we can use to calculate a set of lidar beams (origin and ray) and a corresponding 3D point cloud. Lidar beams may of course result in no return, meaning the ray traveled to the limit of the sensor range.

We consider three variations to estimating the volumetric model from the data. In the first model, we divide the world into a regular voxel grid of side s and estimate the point distribution and permeability independently for each occupied volume. In the second model, we follow the top-down clustering approach of [14]. We drop the notion of voxels and instead encode volumes by ellipsoids that are isosurfaces of the fitted point distributions. Finally, we have considered a bottom-up region growing approach following [15]. The algorithm is significantly slower in model construction and generally performs worse than top-down and regular modeling. We do not present the results here due to space constraints.

1) Regular Voxel Estimation: The regular voxel model estimates via Maximum Likelihood (ML) from the data a Gaussian distribution for the point density in each volume. That is, for each regular voxel containing $3D$ points, the sample mean and covariance are estimated and form the model parameters. Although more complex distributions could be learned, for small voxels there is likely insufficient data to estimate these reliably. Indeed, even for $3D$ Gaussian distributions, care must be taken.

One approach to estimating permeability would be to use the ML estimate based on the number of lidar beams that pass through the voxel compared to those that terminate inside the volume. However, we refine this approach further by requiring the lidar beam to not just pass through the voxel but to also pass sufficiently close to the fitted point distribution. We evaluate "closeness" using the Mahalanobis distance to account for the anisotropic shape of the Gaussian distribution. That is, for a lidar beam originating from $\boldsymbol{p_0}$ traveling in the direction of unit vector $\hat{\boldsymbol{r}}$, we find the point on the ray $\boldsymbol{p} = \boldsymbol{p_0} + t\hat{\boldsymbol{r}}$, that has the minimum Mahalanobis distance, $d(t) = \|\boldsymbol{p} - \boldsymbol{\mu}_i\|_{\Sigma_i}$ to the Gaussian with mean $\boldsymbol{\mu}_i$ and covariance Σ_i. This is given by:

$$t = \left[\hat{\boldsymbol{r}}^T \Sigma_i^{-1} \hat{\boldsymbol{r}}\right]^{-1} \hat{\boldsymbol{r}}^T \Sigma_i^{-1} (\boldsymbol{\mu}_i - \boldsymbol{p_0}) \qquad (1)$$

Defining m as the number of points that pass through the volume with distance $t < \tau_m$, for some threshold τ_m, and n as the number of points that terminate within the same threshold, we estimate permeability as $\rho = \frac{m}{m+n}$. Fig 2 shows a simple example of this approach in which the left

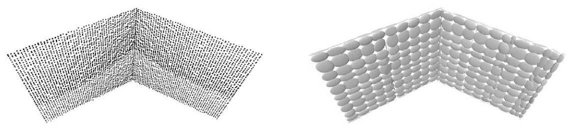

Fig. 3. (Left) A synthetic input point cloud built from two planes with Gaussian range noise. (Right) Volumetric model with isosurfaces drawn at $\tau = 2.0$.

ellipse has $\rho = 0.25$ and the right ellipse has $\rho = 0$. (We use $\tau_m = 2.0$ in the results shown later.)

To demonstrate the model, fig 3 shows an example of a synthetic point cloud from two planes and the resulting regular voxel grid model estimated from the data. Each Gaussian is shown as an isosurface.

2) Top-Down Clustering: One possible problem with the regular voxel method is artifacts caused by arbitrary voxel boundaries. Thus, we explore a top-down clustering method that instead aims to "follow the data". The top-down clustering method we use is based on [14]. It operates by recursively performing K-means clustering, with $K = 2$ on the point cloud data. That is, initially, the entire dataset is split into two via K-means. Each portion is then recursively split until there is insufficient data in each cluster to continue. The flexibility of no longer having regular voxel boundaries comes with a price. As the technique must touch more data early in the process, it is much slower to execute. Additionally, the recursive partitioning is greedy and may force potentially bad cluster boundaries early in the process when the data is very non-Gaussian in appearance. Once the volume distributions are created, permeability is then estimated as above.

We have also considered variations on K, and termination criteria based on how well the cluster represents a "Gaussianess" distribution. Here, for space reasons, we only report on the same implementation as described in [14].

3) Bottom-up Region Growing: The bottom-up method follows [15] and initially splits the data into many small ellipsoids that are merged together subject to a point density constraint. Permeability is then estimated as described above.

IV. SIMULATION

We now introduce the process to draw samples from the model in order to simulate a lidar sensor interacting with the world. There are three parts to the process: beam sampling, permeability sampling, and range sampling.

We use a stochastic ray tracing approach for simulation. Conventional approaches to lidar simulation separate the sensor model from the world model. Our sensor model captures pointing errors due to the angular resolution of the laser scanner angle encoder and errors due to beam divergence and range measurement (e.g. [16]). We first sample a ray from this model, i.e. uniformly sampling a ray from the truncated cone representing the range of possible rays from the lidar. For surface models, this ray is traced (or z-buffered) to find the first intersecting surface. The range to that surface, corrupted by additive noise, is the sampled sensor measurement.

We use a similar concept for volumetric modeling. A ray is sampled from the sensor model to generate the lidar beam. This is ray marched through space to find the first intersecting, non-empty volume element. As with permeability calculations, we perform the intersection with the isosurface of the point distribution. That is, we consider beam-volume intersections when the ray passes within a threshold of the Mahalanobis distance for that volume. We use the same distance threshold $\tau = 2.0$ as for permeability estimation.

Once an intersecting volume is found, we draw a Bernoulli sample to determine if the beam intersects a reflecting surface in that volume (a "hit") or instead propagates through due to its permeability (a "pass through"). If the ray passes through, then we continue tracing the ray until it intersects a volume in a hit or reaches the maximum range of the sensor.

When the ray "hits" a volume, say i, then we draw a range sample. To draw a range sample, we effectively draw a point from the Gaussian distribution $N(\boldsymbol{\mu_i}, \Sigma_i)$ *constrained* to be on the ray $\boldsymbol{p} = \boldsymbol{p_0} + t\hat{\boldsymbol{r}}$. This results in a univariate Gaussian distribution with range given by with $t \sim N(\mu_t, \sigma_t^2)$, where:

$$\mu_t = \hat{\boldsymbol{r}}^T \Sigma_i^{-1} (\boldsymbol{\mu}_i - \boldsymbol{p_0}) \sigma_t^2 \quad (2)$$

$$\sigma_t^2 = \left[\hat{\boldsymbol{r}}^T \Sigma_i^{-1} \hat{\boldsymbol{r}} \right]^{-1} \quad (3)$$

V. EXPERIMENTAL SETUP

Two experiments were conducted: one to quantify the performance of the modeling and simulation method on a variety of representative targets; and the second to demonstrate the system operating as a lidar simulator for off-road terrains. In both cases the volumetric model is first constructed from a training dataset. A second dataset collected in the same area is used to evaluate the simulation quality. The reported sensor poses from the test dataset are used to simulate range returns using the model. These simulated returns are compared against the actual data measured in the test dataset using a variety of metrics. We compare the different models and evaluate their performance along with a more traditional non-permeable surface model.

A. Static Lidar Simulation

To quantify the performance of the non-deterministic simulation a large amount of data is required for a meaningful comparison. The lidar scanner used in these experiments was a Velodyne HDL-64E, a 3D laser scanner which makes range measurements at periodic intervals as it rotates. The sensor makes 1.33 million measurements per second using 64 individual lasers. As the lidar rotates it reports its orientation using a 4,000 count per revolution encoder. For all targets 5 minutes of lidar data was recorded, this means that for each reported orientation of the lidar every individual laser will have recorded approximately 1,500 range measurements. One fifth of each target's dataset was used as training data for model building and the remaining data for testing.

There were six targets in the experiment: a pair of intersecting planes; three trees with low, medium and high density foliage; a sample of tall grass and a sample of short, mowed grass (Fig. 4). Each of these experimental targets was scanned from a distance of approximately $8m$.

B. Mobile Lidar Simulation

To test the volumetric modeling and simulation method at a larger scale, we have collected data with a mobile platform in an outdoor, off-road, lightly forested area using the same lidar. The mobile platform uses a calibrated, high quality RTK GPS/INS and camera system to colorize the lidar points and position them in a global coordinate frame. Two separate runs through the course along an unsealed trail produced two independent datasets with nearby, *but not the exact same*, sensor poses (Fig. 5). One dataset is used for training to build the models. The second dataset is used as the test data set. This experiment provides a qualitative test of the model's suitability for simulating a lidar mounted on a UGV.

C. Comparison Surface Model

The surface model is similar to the conventional polygonal mesh model used in most robotic simulation packages. In our implementation the surface model is a set of triangles, each of which is represented by the 3D coordinates of its vertices. We use the Robust Implicit Moving Least Squares method [17] with marching cubes to estimate the surface mesh over the point cloud. The main parameters to this algorithm control the resolution of the triangle mesh and the level of smoothing of the surface. In our implementation we use a single ray trace operation to determine which triangle the beam intersects and the range at which this intersection occurs. After determining the exact range we add zero mean Gaussian noise to produce the simulated range. This noise was estimated from a large number of range measurements against a planar target as $\sigma = 5mm$. This result is similar to that found previously [18].

VI. Results and Analysis

A. Static Lidar Simulation

We tested the surface modeling and volumetric modeling techniques for datasets collected on the targets outlined in section V.

Fig. 6 shows histograms of the real and simulated data generated using surface and volumetric models for four of the targets. These two dimensional histograms are binned by laser angle (at the resolution of the encoder: $0.09°$ for the Velodyne HDL-64E) and by range (with the same resolution of the lidar's range returns, $2mm$). Effectively each histogram is a planar slice through the point cloud. For these results we used a voxel size of $5cm$ which is larger than the size of the fine features in the targets. The comparison surface model was constructed with RIMLS and marching cubes over voxels of size $1cm$, with a kernel of $4cm$ for a high quality reconstruction.

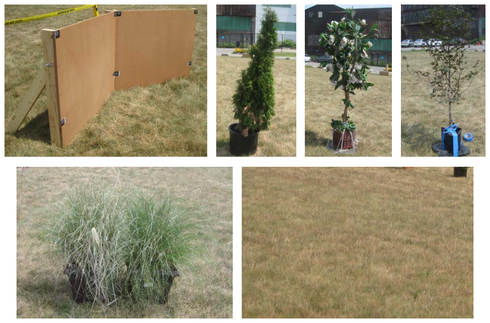
Fig. 4. Top row from left, the planar target and the high, medium and low density trees. Bottom left, the tall grass and right, the short grass.

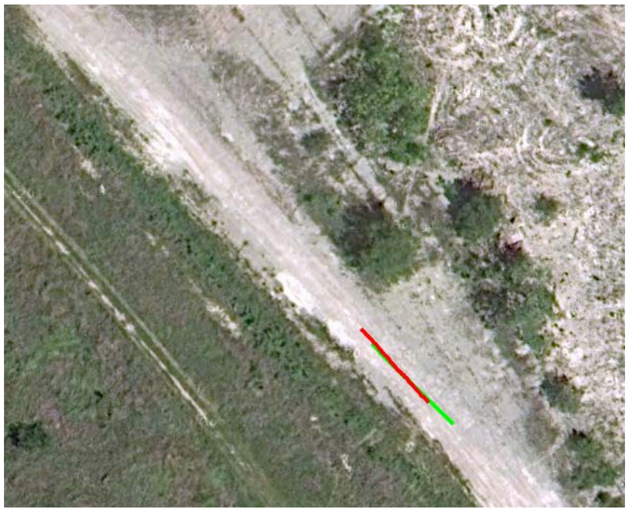
Fig. 5. The trajectory of the mobile platform while acquiring the training data (green) and the test data (red). In both cases the mobile platform was traveling towards the Northwest.

The first two columns of Fig. 6 show that the planar target and the high density tree are simulated satisfactorily by both the surface and volumetric models, although both models do introduce some slight smoothing into the outline of the high density tree. However, for the low density tree and the tall grass the volumetric model produces a better approximation of the real data than the surface model, which filters out some of the variation in the data.

The performance of the two models against the complete set of targets is summarized in Table I. This table shows the Bhattacharyya distance between the $2D$ histograms of the simulated and real data using the same bins as before. An additional range bin for each orientation is used to represent range measurements that pass entirely through the target and generate no return to ensure that the comparison accounts for differences in object permeability produced by the modeling methods. The metric comparisons indicate that, except for the planar target, the volumetric model produces a more faithful simulation. The surface model does not work well on the more permeable targets, although its bad performance on the short grass is likely due to the low grazing angle the grass presents to the lidar.

B. Mobile Data Simulation

Fig. 7 shows the real point cloud and the simulated point clouds generated using the surface and volumetric models

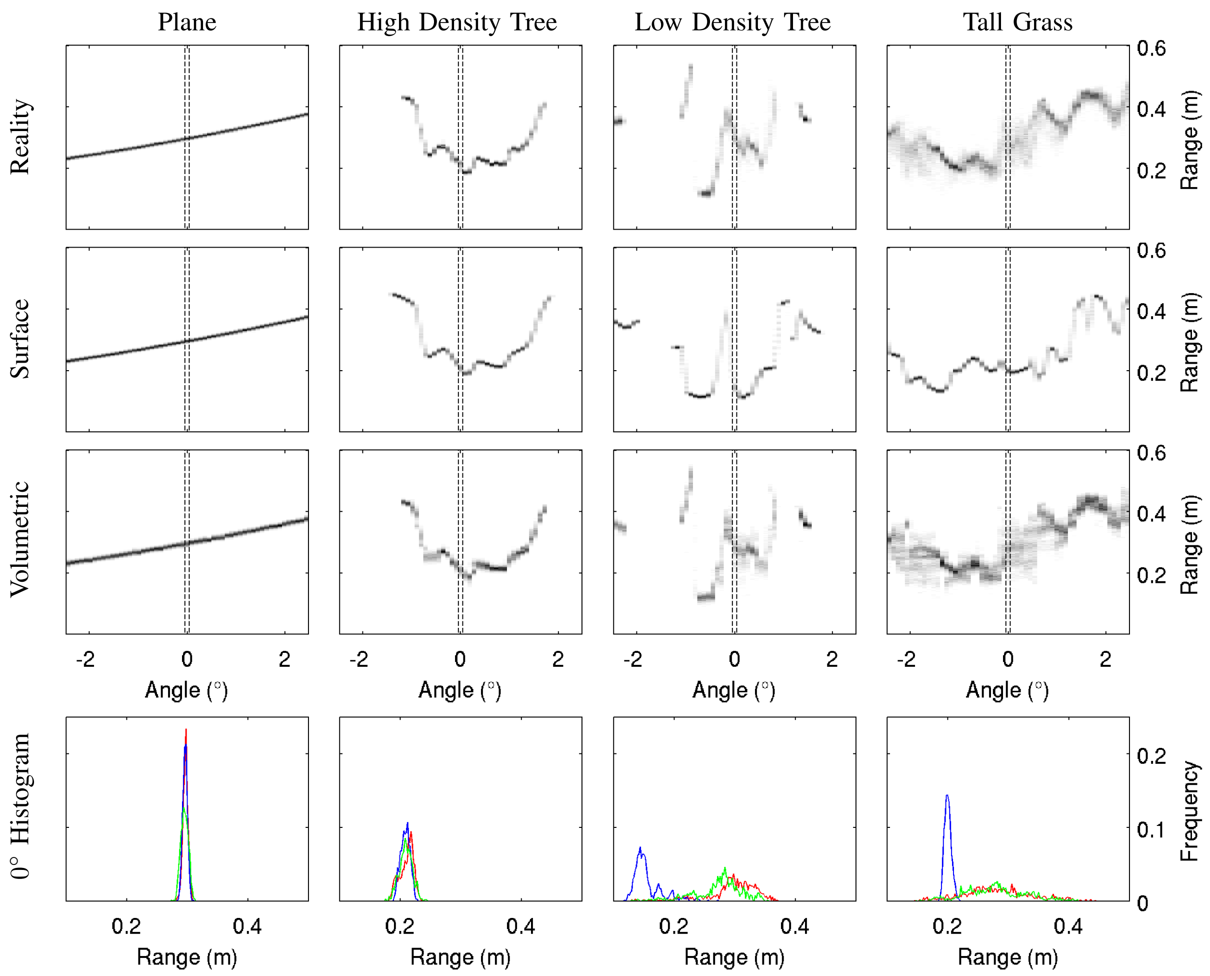

Fig. 6. Real and simulated results for four of the test targets. Each graph is a $2D$ histogram of bearing and range values from an individual laser and shows a cross section of $5°$ of angle and $0.6m$ in range. The bottom row shows the $1D$ range histograms for $0°$ of bearing (a vertical cross section taken along the center of the $2D$ histograms). The real data is in red, the surface model in blue and the volumetric model in green.

TABLE I

BHATTACHARYYA DISTANCE BETWEEN THE REAL AND SIMULATED DATA FOR SURFACE AND VOLUMETRIC MODELING.

Target	Surface	Volumetric
Plane	**0.019**	0.068
High Density Tree	0.111	**0.074**
Medium Density Tree	0.896	**0.125**
Low Density Tree	0.622	**0.054**
Tall Grass	0.888	**0.095**
Short Grass	1.229	**0.091**

for the mobile testing dataset. The volumetric voxel size and surface mesh resolution are $0.3m$ to match the point cloud density from the mobile data (the surface reconstruction kernel was $0.4m$). Since the test data in this case comes from a moving vehicle it is not possible to perform a statistical comparison between the results. It is clear, however, that the volumetric model produces a significantly better simulation of the tree canopy than the surface model.

Also shown in Fig. 7 is the result of using the top-down clustering method. The clustering method appears to work appropriately on surface areas such as the road but makes poor choices for volume location and size in vegetation which reduces the simulation quality. The top-down splitting criteria used [14] was not conceived for permeable objects or significant sensor noise and better clustering criteria is an area of future work. Even with a naive implementation, the voxel-based modeling techniques are near real-time. Initial explorations of GPU implementations show promise for faster than real-time simulation for all methods.

VII. CONCLUSION

We have presented a new automatic method to model complex vegetation from collected data to improve the simulation of a lidar sensor in the same scene from novel poses. The combined permeability and volumetric model capture the distribution of points inside vegetation and show significant improvements over more traditional surface models. Moreover, our volumetric approach requires only modest increases in memory and computation is tractable. Future work will examine whether the approach can *transfer* to simulate a different lidar to the one used for model building.

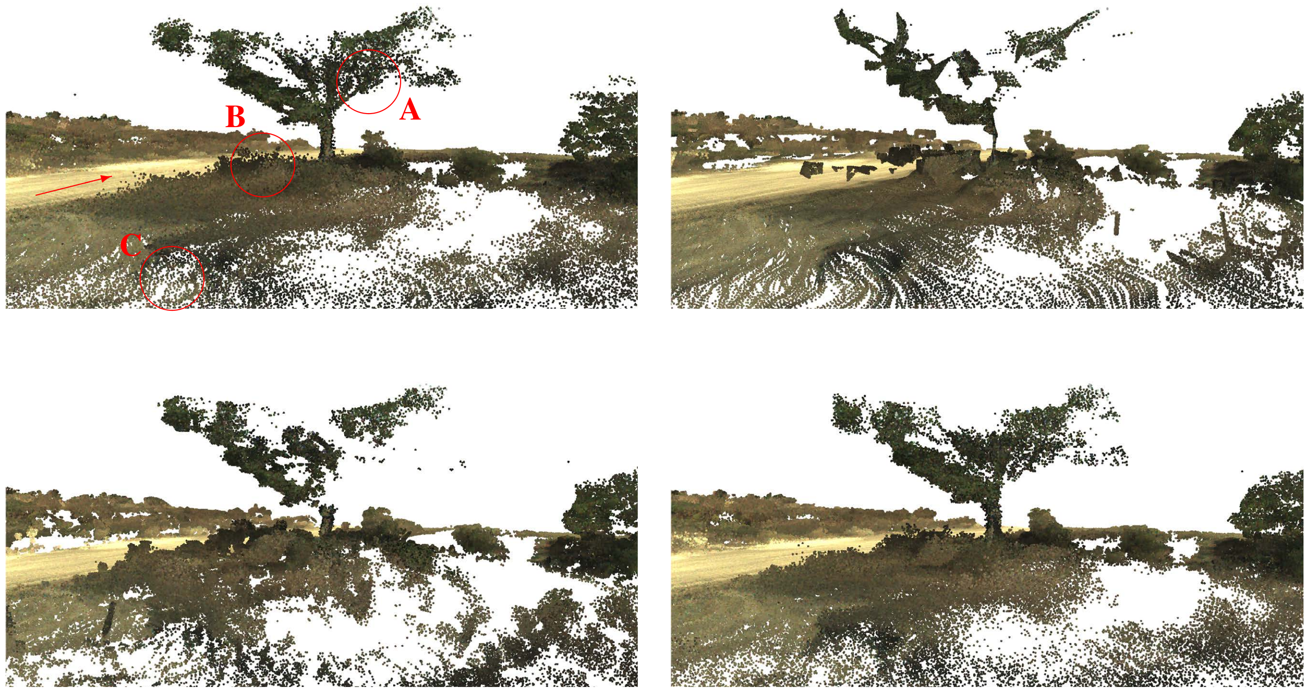

Fig. 7. Top left, the real data with the route taken by the mobile platform indicated by a red arrow. Top right, the simulated point cloud using surface reconstruction. Bottom, the simulated point clouds using our volumetric modeling using top down (left) and voxel (right) segmentation. For visualization the simulated points have been colorized using the real data (the arrow in red shows the path of the mobile platform to collect the data). (A) returns on branches within the tree are too sparse for the surface reconstruction or top-down segmentation to capture the underlying trends; (B) The strong returns on the surface of the bush generate an overly solid surface model; (C) the limited noise in the surface model reveals the scan pattern of the lidar while the volumetric model better preserves the underlying statistical distribution.

ACKNOWLEDGMENTS

The authors thank Tom Pilarski, Scott Perry, and Cliff Olmstead for their generous support in maintaining and deploying the data collection equipment used in this paper.

REFERENCES

[1] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[2] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper, "USARSim: a robot simulator for research and education," in *Proceedings of the International Conference on Robotics and Automation*, 2007.

[3] C. Goodin, R. Kala, A. Carrrillo, and L. Liu, "Sensor modeling for the virtual autonomous navigation environment," in *Sensors, 2009 IEEE*, 2009, pp. 1588–1592.

[4] J. Tuley, N. Vandapel, and M. Hebert, "Analysis and Removal of artifacts in 3-D LADAR Data," in *Proceedings of the International Conference on Robotics and Automation*, 2005, pp. 2203–2210.

[5] H. Jones and R. Vaughan, *Remote sensing of vegetation: principles, techniques, and applications*. Oxford university press, 2010.

[6] C. Mallet and F. Bretar, "Full-waveform topographic lidar: State-of-the-art," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 64, no. 1, pp. 1–16, 2009.

[7] A. Kukko and J. Hyyppa, "Small-footprint Laser Scanning Simulator for System Validation, Error Assessment, and Algorithm Development," *Photogrammetric Engineering and Remote Sensing*, vol. 75, no. 10, pp. 1177–1189, 2009.

[8] H. Hulst, *Light scattering by small particles*. Dover Pubns, 1981.

[9] J. Ryde and N. Hillier, "Performance of laser and radar ranging devices in adverse environmental conditions," *J. Field Robot.*, vol. 26, pp. 712–727, 2009.

[10] C. Dima, C. Wellington, S. Moorehead, L. Lister, J. Campoy, C. Vallespi, B. Jung, M. Kise, and Z. Bonefas, "PVS: A system for large scale outdoor perception performance evaluation," in *Proceedings of the International Conference on Robotics and Automation*, 2011.

[11] J. P. Gonzalez, W. Dodson, R. Dean, G. Kreafle, A. Lacaze, L. Sapronov, and M. Childers, "Using RIVET for parametric analysis of robotic systems," in *Proceedings of the 2009 Ground Vehicle Systems Engineering and Technology Symposium (GVSETS)*, 2009.

[12] A. Lindenmayer, "Mathematical models for cellular interactions in development," *Journal of Theoretical Biology*, vol. 1, pp. 280–315, 1968.

[13] Y. Livny, F. Yan, M. Olson, B. Chen, H. Zhang, and J. El-Sana, "Automatic reconstruction of tree skeletal structures from point clouds," *ACM Transactions on Graphics*, vol. 29, no. 6, Dec. 2010.

[14] A. Kalaiah and A. Varshney, "Statistical geometry representation for efficient transmission and rendering," *ACM Trans. Graph.*, vol. 24, pp. 348–373, 2005.

[15] F. Pauling, M. Bosse, and R. Zlot, "Automatic segmentation of 3d laser point clouds by ellipsoidal region growing," *Australasian Conference on Robotics and Automation (ACRA)*, 2009.

[16] C. Glennie, "Rigorous 3d error analysis of kinematic scanning lidar systems," *Journal of Applied Geodesy*, vol. 1, pp. 147–157, 2007.

[17] A. C. Oztireli, G. G., and G. M., "Feature Preserving Point Set Surfaces based on Non-Linear Kernel Regression," *Proceedings of Eurographics, Computer Graphics Forum*, pp. 493–501, 2009.

[18] C. Glennie and D. D. Lichti, "Static Calibration and Analysis of the Velodyne HDL-64E S2 for High Accuracy Mobile Scanning," *Remote Sensing*, pp. 1610–1624, 2010.