# Generation and VR Visualization of 3D Point Clouds for Drone Target Validation Assisted by an Operator

Louis-Pierre Bergé, Nabil Aouf, Thierry Duval, Gilles Coppin

# Generation and VR Visualization of 3D Point Clouds for Drone Target Validation Assisted by an Operator

Louis-Pierre Bergé*†, Nabil Aouf*, Thierry Duval† and Gilles Coppin†
*Cranfield University, Centre for Electronic Warfare
Shrivenham, Swindon SN6 8LA, UK
Email: n.aouf@cranfield.ac.uk
†UMR CNRS 6285 Lab-STICC - Telecom Bretagne
Technopole Brest-Iroise-CS 83818 29238 Brest Cedex
Email: lp.berge@telecom-bretagne.eu, thierry.duval@telecom-bretagne.eu

***Abstract*—In this paper, we present a process to simulate 3D data; as an unmanned combat aerial vehicle (drone) equipped with a Ladar sensor could do. The data obtained are colored 3D point clouds in where potential targets are highlighted. Our objective with these simulated corpus of 3D point cloud data is to explore the usage of Virtual Reality settings to validate the selection of a target. To do that we detail an implementation of a VR headset to visualize our simulated data. We also introduce a metric to estimate the visibility of the target.**

## I. Introduction

Unmanned Combat Aerial Vehicles (UCAVs) can be positioned in airspace for a long period of time while being able to rapidly attack a target of interest. The use of 3D data provided by modern active sensors such as a Laser Radar (Ladar) on future UCAVs could be a valid option to consider in order to improve target detection, recognition, and identification performances [1]. The availability of such types of 3D data will open new technical challenges, in order to efficiently exploit them in an operator-in-the loop scheme.

3D interaction techniques for manipulating viewpoints [2] and 3D objects [3] are now common and could be used to let the user choose the best possible viewpoint to analyse the data. Moreover, new low-cost Virtual Reality (VR) hardware displays (such as the Oculus Rift or HTC Vive) and input devices (such as the STEM) are now wide spread. These devices are worth using to offer an immersive experience to the end-users of our systems. To do so and in order to develop an efficient 3D user interface, which provides the optimal display options and configurations of the 3D data, a number of human factor experiments and analyses have to be conducted.

Our contributions in this paper include: 1) an innovative method to generate 3D point clouds from 3D scenes composed by meshes, 2) a metric to quantify the difficulty of finding a target inside 3D point clouds and 3) an integration of a VR headset to visualize 3D point clouds by the human operator.

## II. Previous work

Our research is inspired by previous work on the usage of drones and Ladar sensors in military context, on solutions to generate 3D point clouds and visualize them by human operator at the command and control station.

### A. Military UCAV context

In current battlefield, a drone (UCAV) is in few cases positioned in the airspace to attack a target. One operator pilots the drone while a second operator uses sensor feedback for finding the target. Classically, infrared (IR) cameras or 2D radars are commonly available sensors on a drone. Also, to help the operator 2D automatic target recognition (ATR) algorithms are common on drones, for advanced tracking systems of a target [4]. However, these algorithms can present false alarms that preclude the reliable automatic selection of such targets [5], [6]. Operator-in-the-loop, i.e. man designation, is thus required to guarantee successful recognition of a target and effective impact by a missile launched from the drone [1]. Moreover, IR images can easily be noisy because of sensitivity of IR signals to weather changes, internal camera defaults or configuration.

In 2004, the U.S. Army Research wanted to exploit the usage of current and future airborne Laser Radar (Ladar) sensors [1]. Now, this type of sensors is compact enough to be envisaged onboard of a drone. The main advantage of Ladar is that the data acquired are 3D point cloud data. As for IR or 2D radar, ATR algorithms with 3D data have been recently explored [7]. A high recognition performance (more than 90%) is obtained. However, this result confirms that an operator is still required for a fully correct target detection/recognition. Having the operator-in-the-loop means he has to manipulate and visualize the generated 3D point cloud data. Doing that will permit adaptation of the point of view and the zoom factor on the data to correctly detecting the target. Our objective in this work then is related to Human-Computer Interaction and Virtual Reality research domains. To explore different VR solutions of visualization and interaction, we first need to create a database of military 3D point clouds representing what a Ladar embedded on a drone could generate. The process for generating these data is one of our contributions described in this paper.

### B. Generating 3D point clouds

Previous research already explored the simulation of Ladar sensors [8]. A lot of work focuses on the physical modelling and simulation of Ladar [9]–[12] such as the sensor model (pulse energy, beam divergence), the atmospheric model (turbulence, fog) and the terrain model (material, reflection). For the

application of these simulations, Kim et al. [11] investigate the fusion of multiple sensors (GPS, IMU and laser scanner) to precisely acquire 3D point clouds with airborne Ladar dealing with sensor errors. Chevalier et al. [10] provided a comparison between simulated and real Ladar data in order to compare algorithms for detecting a target. Lohani et al. [13] took into account, in their simulation, the flight parameters of the aircraft mounting the Ladar while Peinecke et al. [8] proposed a simulation using the GPU for fast calculations. All this previous work uses the principle of ray casting for simulating the laser beam and calculating the 3D point clouds.

In our research, we propose an easy and fast method for generating 3D point clouds starting from any 3D mesh. We use the principle of ray casting to calculate the position of each 3D point cloud without including physical modeling of the atmosphere and the beam Ladar. Moreover, we choose to model the movement of the UCAV as a circular trajectory around an area simulating a loitering munitions scenario. We develop an innovative data processing on the 3D point clouds as they are acquired in order to color them to highlight potential targets. These choices permit to describe with few parameters our method for generating 3D point clouds and assure a quick and easy process for non-experts in physics with Ladar sensors.

### C. Visualization of 3D point clouds

Visualization of Ladar data, i.e. 3D point clouds, could be done in the same way as 3D Meshes. A lot of tools exist to load and visualize 3D point clouds, with both OpenGL and DirectX 3D API. Among them the Point Cloud Library (PCL) [14] including many C++ implementations of 3D point clouds algorithms for filtering, estimation, reconstruction, registration, model fitting and segmentation. CloudCompare [15] is also a good example of GUI software to visualize and to compare 3D point clouds.

In this work, we want to combine Virtual Reality visualization and 3D point clouds. Burwell et al. [16] showed that users positively evaluate the usage of Head-Mounted Display (HMD) for visualizing 3D point clouds: immersion and real-time manipulation, are considered advantageous for viewing the depth and structure of point clouds. Oliver et al. [17] successfully visualized 3D point clouds in a CAVE setting for several interactive analysis tools of geology data. Bruder et al. [18] used HMD for visualization of culture heritage 3D point clouds. Both examples show that VR visualization and dense 3D point clouds (more than 20 million 3D points) are now possible with high performance, optimal frame rate and user satisfaction. In our case, we use a specific toolkit for the generation of 3D point clouds as the PCL Library. For the visualization, we aim at easily adding or changing VR settings in order to explore many available solutions. Thus, we choose to adopt Unity 3D [19] for its versatility. Moreover, we also provide a metric for quantifying the difficulty to detect the target inside the acquired/processed 3D point clouds. The result of this metric depends on the point of view adopted for the visualization and of the 3D scene, i.e. the shape of the 3D objects that occludes the target.

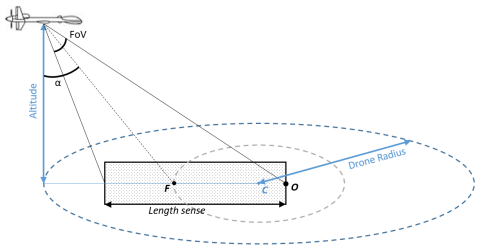



Fig. 1. Drone flight and Ladar sensor parameters.

## III. Our Process

Our full process of 3D points clouds generation from a drone equipped with a Ladar sensor is divided in three main steps: 1) define the parameters of the Ladar sensor and of the drone's flight; 2) generate multiple 3D point clouds and fuse them; 3) add color for segmentation on the 3D points. For each step, we first explain the concept then we show the results of our implementation.

### A. Definition of the parameters

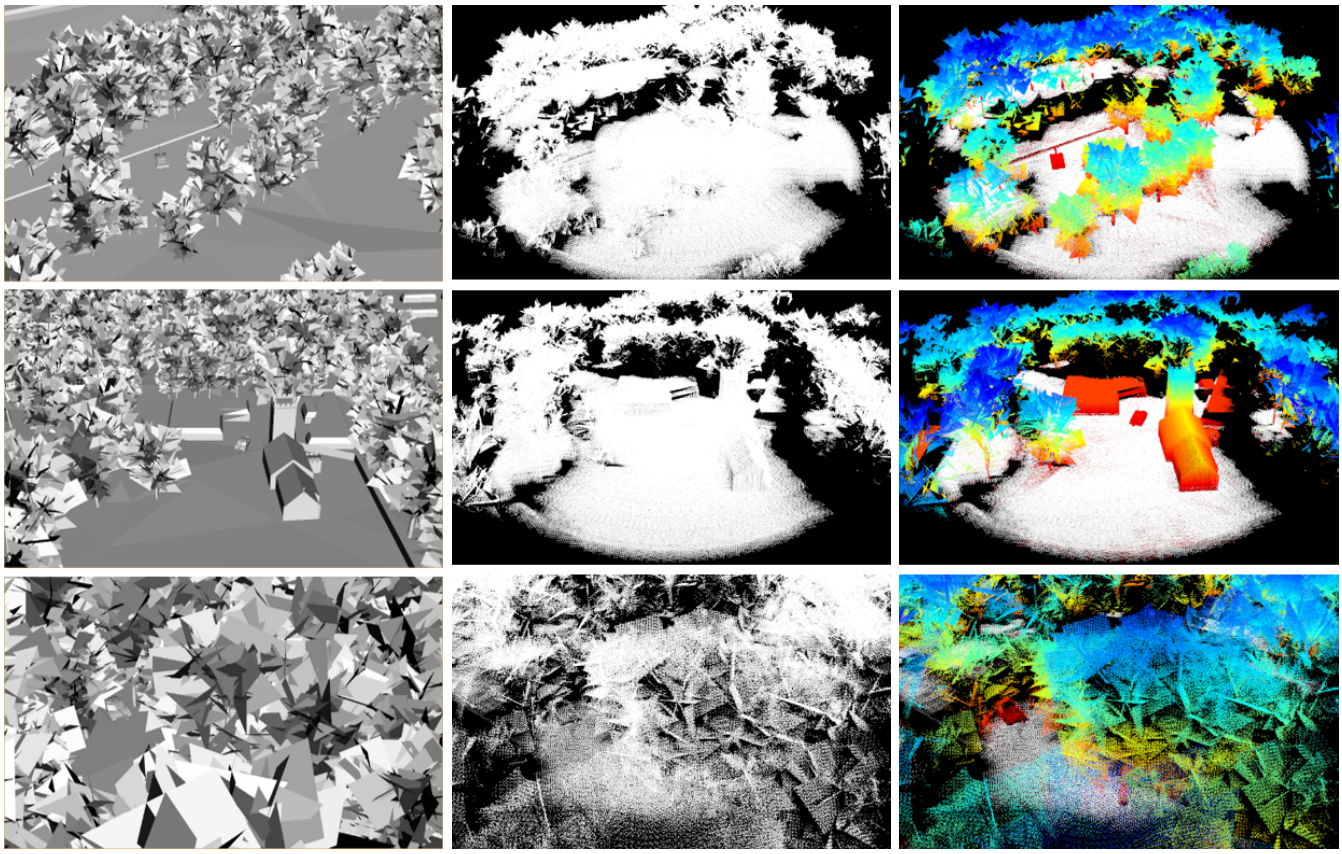
In our military context, we consider that the drone rotates around the area where the target is hiding. For defining the flight parameters of the drone, we need the *Altitude* and the *Drone Radius* (cf. Fig.1). The point *C* defines the center of the circular movement of the drone. For the Ladar sensor, we must define the resolution of the sensor in pixel (*X_res* $*$ *Y_res*), the angle $\alpha$ corresponding to the inclination of the Ladar and the Field of View (*FoV*) of the sensor (cf. Fig.1). The point *P* is corresponding to the point on the ground where the Ladar sensor is focused. The point *O* is the farthest point on the ground visible by the drone. The length *CO* defines the overlap area after the center *C* (cf. Fig.1). This overlap permits you to see high elements (buildings or trees for example) close to the center *C*. If *CO* is null, a hole without 3D points could appear.

With all these rules, the *Altitude*, the overlap size *CO*, the angles $\alpha$ and *FoV* are enough for determining the *Drone Radius* (1) and the size of the length sense on the ground, i.e. *Length sense* (2).

$$DroneRadius = \tan(\alpha + \frac{FoV}{2}) * Altitude - CO \quad (1)$$

$$Length_{sense} = Drone_{Radius} + CO - \tan(\alpha - \frac{FoV}{2}) * Altitude \quad (2)$$

In our implementation, we use the PCL library [14]. PCL uses the Visualization Toolkit library (VTK) [20] for the 3D rendering. We simulate the Ladar sensor with a virtual camera that looks at the point *F* with an angle of view of *FoV*. In all our examples, we use these values for the simulation: *Alitude* = 300m; *OC* = 10m; *FoV* = 6.5°; $\alpha$ = 45°; *X_res* = *Y_res* = 128px. We obtains *DroneRadius* $\approx$ 326.12m and *Lengthsense* $\approx$ 68.36m corresponding to a full circular area covered with a diameter of 125m.



(a) Original Mesh (b) White point clouds (c) Colored point clouds

Fig. 2. Three illustrations of our process.

### B. Generation of the 3D point clouds

In order to generate the full 3D point clouds viewed by the drone, we subdivide the circular movement of the drone by *NoV* (Number of Views) position. For each position of the drone, we generate one 3D point cloud of $X\_res * Y\_res$ points that fits what the Ladar sensor sees. At the end, this step generates *NoV* different 3D point clouds. After that, we fuse all these point clouds into a big 3D point cloud of $NoV * X\_res * Y\_res$ point clouds that represent all the area visible by the drone. In our implementation, we choose to generate 100 different positions of the drone: $NoV = 100$. We iteratively translate the virtual camera simulating the Ladar sensor to each position of the drone. For calculating the position of each point cloud, we use the principle of ray casting. For each pixel ($128 * 128$) of the image of the virtual camera simulating the Ladar sensor, we launch a virtual ray. We then store the position of the point corresponding at the intersection with this ray and the meshes of the 3D scene. For one area, we obtain point clouds of 1,638,400 points. For our tests, we use a full 3D mesh model of our university campus and the closest village. We add some targets, i.e. a tank inside these meshes. Fig.2b shows three examples of 3D point clouds generated by this step. On the Fig.2, the point of view adopted for each example is chosen arbitrarily in order to better understand the 3D scene and see the target. Moreover, the field of view is not the *FoV* of the Ladar sensor; it is the default value of 30°.

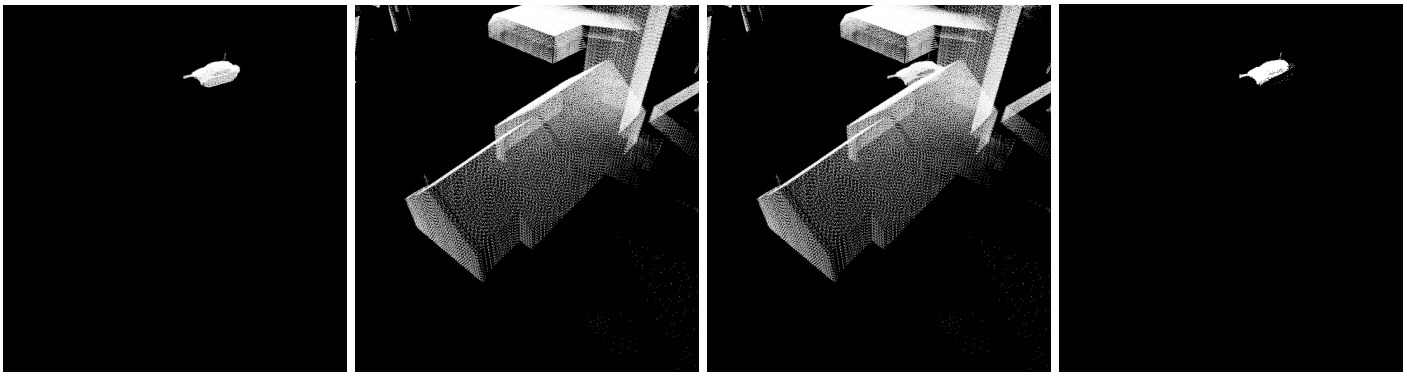
### C. Coloration of the 3D points

At this point, we only obtain white 3D point clouds. We would like to add some color. We use two principles: 1) use a color map in order to highlight point clouds close to the ground, i.e. where the target is; 2) leave the ground in white.

For the color, we use the jet color map. This color map is included for example in Matlab and permits to use every color only once. We interpolate the color between the minimum height (Z coordinate) and the maximum height of the 3D point clouds. We choose a jet color map with 64 values and we put the red color on the point close to the ground (minimum height value) and the blue color on the maximum height value in order to display the target in red color (cf. Fig.2c).

For the white ground color, we use the iterative RANSAC method [21] for detecting planar surfaces inside the 3D point cloud. We use the implementation of this segmentation operation available with PCL [14] as follow. For each *NoV* 3D point cloud generated in the previous step, we apply five steps:

1) Keep the 3D points included between the minimum height value and the minimum height value plus a *threshold* and save it in *PC*.



(a) (b) (c) (d)

Fig. 3. Illustrations of our method for calculating the metric difficulty.

2) Apply the RANSAC algorithm on *PC*.
3) Add in *PG* (points ground) the result of the algorithm if the normal of the plane detected is perpendicular to the ground (*normal(plane).z* $< 0.9$).
4) Remove the *PG* points inside *PC*.
5) Loop to the step 2) until the number of points inside PC is less than 1% of the initial PC number of points.

At the end, we set all the 3D points included in *PG* to white. In step 1), we take the value of the *threshold* as 3 corresponding to search the planar surface between the minimum height value and 3 meters in order to not detect ground on the top of buildings. This *threshold* is one of the changeable parameters to correctly find the ground if the area has some cliffs or hills. In step 3), the normal vector test permits you to remove vertical plane detections corresponding, for example, to building walls, sides of cars, trunks of trees. Fig.2c shows examples of results of our full process with the complete colorization.

## IV. VR VISUALIZATION

After multiple generations of different 3D scenes with our process, we have implemented the visualization of the 3D point clouds using an immersive HMD setup. In this section, we also detail our method for calculating a metric to evaluate the difficulty of finding a target inside the 3D point clouds.

### A. Immersive HMD setup

In our setup we visualize 3D point clouds on the OSVR HDK 1.3 HMD (Open Source Virtual Reality Hacker Development Kit 1.3 [22]). For rendering, we use the official OSVR package for Unity 3D with direct mode rendering and distortion correction. To display the 3D point clouds generated by our process with the PCL library in Unity 3D, we have written an asset in C# to load a "*.PCD*" file according to the format defined by PCL. For the head tracking, we use the data provided by the internal sensor of the HDK 1.3. For the position, we use an external motion capture (OptiTrack system) with six infrared cameras. We have implemented two modes for the interaction with the user:

- An automatic mode, where the point of view of the user rotates around the center of the 3D point clouds. This mode gives the user the same perspective as a drone rotating around the simulated area. The user can only use the head tracking to adapt his/her point of view; he/she cannot change his/her position. When he/she sees a potential target, he/she can press the space bar to stop this mode and enter the second mode.
- In this second mode, the user can walk through the 3D point clouds, move his/her body and head to be close to the potential target.

For rendering, we use a computer with Windows 10, a 3.70GHz Intel Xeon E5-1630 processor, 16GB of main memory and one Nvidia GeForce GTX 960 graphics card. The VR visualization is rendering in full HD resolution with a stereoscopic renderer and no significant lag is seen.

### B. Metric for estimating visibility

We would like to propose a metric that characterizes how difficult it is to see a target inside a 3D point cloud. This value depends on the point of view chosen by the user. The main concept of the metric is to calculate the percentage of visible pixels of the target for each view.

Firstly, we generate three white 3D point clouds with our process detailed in section III-B: *P1* with only the mesh of the target (cf. Fig.3a), *P2* with only the meshes of the environment (buildings, trees) without the ground (cf. Fig.3b) and *P3* the meshes of the target and the environment without the ground (cf. Fig.3c). *P1* corresponds to the theoretical point clouds of the target if there is not environment and *P3* corresponds to the point clouds viewed by the drone with a perfect detection of the ground. Secondly, the metric is calculated for each point of view of the user as follows:

1) Calculate the binary image *I1* (cf. Fig.3a), *I2* (cf. Fig.3b) and *I3* (cf. Fig.3c) from a single point of view of *P1*, *P2* and *P3* respectively.
2) Calculate *I4* = *I2 xor I3*. This image (cf. Fig.3d) corresponds to the visible pixels of the target.
3) The metric is the ratio between the number of pixels in *I4* (target occluded by the environment) and the number of pixels in *I1* (total number of pixels in the target).

From the point of view chosen in the Fig.3, the number of pixels of target visible with the environment is 4733px (Fig.3d) and the number of pixels of the target visible without the environment is 6908px (Fig.3a). The metric is thus 68.5%.

To estimate the global difficulty of an area, we take different point of views on the point clouds. We choose to segment the circular rotation into 100 images, rotating the virtual camera as in the first mode of the immersive visualization. The final metric is the mean metric across all each images. For example, on the three examples of the Fig.2, the metrics are 54.4%, 77.6% and 24.6%. We calculate the final metric on different areas. On easy areas, when the target is almost fully visible in each image, the metric is more than 50%. On difficult areas, the metric is less than 20%.

## V. Conclusion and future work

In this paper, we detail a process to generate 3D point clouds based on 3D meshes. To do that, we simulate the data sensed by a Ladar embedded on a drone flying around a simulated combat area. Our full process generates colored 3D point clouds with the estimated ground in white and potential targets in red. Our quick and easy process permits the generation of many 3D point clouds in order to create a corpus of data, for exploring the usage of Virtual Reality settings to visualize and manipulate them. For this, we provide details about an implementation method for visualization with a VR headset. We also define a method to estimate the difficulty of finding a target inside the 3D point clouds generated.

Our future work is now to explore, in depth, the usage of the VR setting with our corpus. First, we will plan user experiences to evaluate the display and interaction fidelity [23] with a VR headset for searching a target inside 3D point clouds. Secondly, we plan to design and evaluate natural interaction techniques [24] for manipulating the 3D point clouds: for example, techniques for zooming close to the target, for using a cutting plane in order only to display the 3D point clouds close to the target, or for changing the color of the 3D point clouds for improved viewing. Finally, designing how to display and interact with results of a 3D ATR algorithm [7] could be also a good proof of concept, to develop the usage of Virtual Reality in a military context.

## Acknowledgment

We would like to thank Lounis Chermak (Cranfield Universtiy) for his technical support, Jonathan Searle and his team for the 3D meshes kindly provided and the French DGA for their financial support on this project.

## References

[1] J. E. Grobmyer, Jr., T. Lum, R. E. Morris, S. J. Hard, H. L. Pratt, T. Florence, and E. Peddycoart, "Airborne ladar man-in-the-loop operations in tactical environments," *Proc. of SPIE*, vol. 5412, pp. 137–148, 2004.

[2] M. Hachet, F. Decle, S. Knodel, and P. Guitton, "Navidget for 3D interaction: Camera positioning and further uses," *International Journal of Human-Computer Studies*, vol. 67, no. 3, pp. 225–236, 2009.

[3] T. T. H. Nguyen, T. Duval, and C. Pontonnier, "A New Direct Manipulation Technique for Immersive 3D Virtual Environments," in *Proc. of ICAT-EGVE*, 2014, pp. 67–74.

[4] G. J. Gray, N. Aouf, M. Richardson, B. Butters, and R. Walmsley, "Countermeasure effectiveness against an intelligent imaging infrared anti-ship missile," *Optical Engineering*, vol. 52, no. 2, 2013.

[5] G. J. Gray, N. Aouf, M. A. Richardson, B. Butters, R. Walmsley, and E. Nicholls, "Feature-based recognition approaches for infrared anti-ship missile seekers," *The Imaging Science Journal*, vol. 60, no. 6, pp. 305–320, nov 2012.

[6] ——, "Feature-Based Target Recognition in Infrared Images for Future Unmanned Aerial Vehicles," *Journal of Battlefield Technology*, vol. 14, no. 2, pp. 27–36, jul 2011.

[7] O. K. Stamatis and N. Aouf, "Fast 3D object matching with Projection Density Energy," in *23rd Mediterranean Conference on Control and Automation*. IEEE, jun 2015, pp. 752–758.

[8] N. Peinecke, T. Lueken, and B. R. Korn, "Lidar simulation using graphics hardware acceleration," in *Digital Avionics Systems Conference*. IEEE, oct 2008, pp. 4.D.4–1–4.D.4–8.

[9] R. J. Grasso, G. F. Dippel, and L. E. Russo, "A model and simulation to predict 3D imaging LADAR sensor systems performance in real-world type environments," *Proc. of SPIE*, vol. 6303, no. 1, 2006.

[10] T. R. Chevalier and O. K. Steinvall, "Laser radar modeling for simulation and performance evaluation," *Library*, vol. 7482, pp. 1–13, 2009.

[11] S. Kim, S. Min, G. Kim, I. Lee, and C. Jun, "Data Simulation of an Airborne LIDAR System," *Proc. of SPIE*, vol. 7323, pp. 1–10, 2009.

[12] D. D. Blevins and D. D. Blevins, "First-principles-based LIDAR simulation environment for scenes with participating mediums," *Proceedings of SPIE*, vol. 6214, pp. 62 140G–62 140G–12, 2006.

[13] B. Lohani and R. K. Mishra, "Generating lidar data in laboratory: Lidar simulator," in *ISPRS Workshop on Laser Scanning and SilviLaser*, 2007, pp. 264–269.

[14] R. B. Rusu and S. Cousins, "3D is here: point cloud library," *IEEE International Conference on Robotics and Automation*, pp. 1 – 4, 2011. [Online]. Available: http://pointclouds.org/

[15] "CloudCompare - Open Source project." [Online]. Available: http://cloudcompare.org/

[16] C. Burwell, C. Jarvis, and K. Tansey, "The potential for using 3D visualization for data exploration, error correction and analysis of LiDAR point clouds," *Remote Sensing Letters*, vol. 3, no. 6, pp. 481–490, 2012.

[17] K. Oliver, W. B. Gerald, and H. K. Louise, "Immersive Visualization and Analysis of LiDAR Data," *Advances in Visual Computing*, vol. 5358, pp. 846–855, 2008.

[18] G. Bruder, F. Steinicke, and A. Nuchter, "Poster: Immersive point cloud virtual environments," in *3DUI '14*, 2014, pp. 161–162.

[19] "Unity - Game Engine." [Online]. Available: http://unity3d.com/

[20] "VTK - The Visualization Toolkit." [Online]. Available: http://www.vtk.org/

[21] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Com. of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[22] "OSVR - Open-Source Virtual Reality for Gaming." [Online]. Available: http://www.osvr.org/

[23] R. P. McMahan, D. A. Bowman, D. J. Zielinski, and R. B. Brady, "Evaluating display fidelity and interaction fidelity in a virtual reality game." *IEEE transactions on visualization and computer graphics*, vol. 18, no. 4, pp. 626–33, apr 2012.

[24] R. P. McMahan, A. J. D. Alon, S. Lazem, R. J. Beaton, D. Machaj, M. Schaefer, M. G. Silva, A. Leal, R. Hagan, and D. A. Bowman, "Evaluating natural interaction techniques in video games," in *3DUI '10*. IEEE, mar 2010, pp. 11–14.